THE UNIVERSITY OF HULL

Chemometric methods for the analysis of process spectra using minimal reference data

being a Thesis submitted for the Degree of
Doctor of Philosophy

in the University of Hull

by

Nicholas Ian Pedge, BSc (Hons) MRSC CChem

October 2008

Abstract

To construct a spectroscopic multivariate calibration model, a set of representative mixture spectra (independent variables) and the corresponding reference values for the property of interest (dependent variables) must be obtained. For a dynamic system such as a batch or semi-batch chemical reaction, creating such a data set may be very difficult or extremely time consuming. It may not be possible to create synthetic mixtures because reaction between the various reactants may occur. If the reaction proceeds via a reactive intermediate or affords a reactive product, isolated reference standards of those species may not be available. Reactions in industry are often heterogeneous and highly concentrated; sampling the batch throughout the course of the reaction for off-line analysis can be problematic and therefore introduce significant error into measured reference values.

An alternative approach that combined Self-Modelling Curve Resolution (SMCR) methods and Partial Least Squares (PLS) to construct a quantitative model using only minimal reference data was implemented. The objective was to construct a quantitative calibration model to allow real-time *in-situ* UV/ATR measurements to be used to determine the end-point of a chlorination reaction. Difficult reaction sampling conditions and the absence of isolated reference standards for the product and reactive intermediate required the method to be developed using only a few key reference measurements.

Utilising Evolving Factor Analysis and Orthogonal Projection Approach, initial estimates of the concentration and spectral profiles for the intermediate and product were obtained. Further optimisation using Multivariate Curve Resolution-Alternating Least Squares (MCR-ALS) led to refined estimates of the concentration profiles. PLS models were then constructed using the calculated concentration profiles and the pre-processed UV spectra. Using a standard PLS model compatible with the spectrometers standard process software facilitated real-time predictions for new batches.

This method was applied to five 45 L batches in a Large Scale Laboratory facility. The method was used successfully to predict the product concentration of batch 1, but exhibited larger prediction error for subsequent batches. Probe fouling was observed and this resulted in lower measured absorbance values that in turn contributed to larger prediction errors. The largest prediction error was attained during batch 3 (an error of 18.8%). However, the qualitative real-time profiles proved to be extremely useful as they

allowed the end-point to be determined without sampling and performing off-line analysis. Furthermore, the model facilitated real-time visualisation of the intermediate concentration profile which could not be observed using the off-line method. This provided further confidence that the process was approaching the end-point. This work was published in *Applied Spectroscopy* (2007, volume 61, number 9, pp 940-949).

During the manufacture in the LSL, the extent to which the fibre-optic cables would be subjected to movement during normal operation was observed. A custom fibre-optic cable assembly for use with a double beam spectrometer was designed to reduce the effects of movement. A test was devised that allowed reproducible curvature to be introduced into both the standard and custom fibre-optic cables and the effect on the resulting spectra was compared. The tests revealed that although the custom fibre-optic cable assembly did not completely eliminate the effects of movement; the custom fibre reduced the effect of fibre movement by approximately 25% relative to the standard fibre assembly.

The work combining Self-Modelling Curve Resolution (SMCR) methods and PLS demonstrated that a regression model could be constructed from concentration profiles derived directly from the spectral data using SMCR methods. The motivation for this approach was to construct a PLS regression model compatible with standard process spectrometer software that could then be used for the prediction of future batches. The use of SMCR facilitates the estimation of the underlying concentration profiles in the absence of suitable reference measurements.

The premise for SMCR methods to provide the correct concentration profiles is the ability to isolate the correct pure spectral profiles for each component of interest. The concept of utilising a vectorised adaptive Kalman filter for self-modelling curve resolution was investigated. Vectorised linear and adaptive Kalman filters were implemented in Matlab. Using simulated spectral reaction data representing the *N*-benzylation of 1*H*-indole using benzyl bromide, it was demonstrated that a vectorised Kalman filter performs identically to the linear Kalman filter but offered much faster computation times. The standard linear Kalman filter took approximately 1.14 seconds to filter a simulated UV data set comprising 333 spectra acquired at 131 wavelengths variables; the vectorised Kalman filter reduced the execution time to approximately 0.03 seconds (a factor of 38 times faster). The advantage of the vectorised linear Kalman filter became more significant when applied to larger data sets. The standard linear Kalman filter took approximately 70 seconds to filter a data set comprising 427 spectra acquired at 3301 wavenumber variables; the vectorised linear

Kalman filter only took approximately 0.74 seconds (a factor of 95 times faster). The standard adaptive Kalman filter took approximately 400 seconds to filter the same Raman data set whilst the vectorised adaptive Kalman filter took only 1.60 seconds (a factor of 95 times faster). This considerable improvement in execution speed made it feasible to use the vectorised adaptive Kalman filter in a novel SMCR application where it was necessary to re-run the filter several hundred times.

The recursive prediction-correction operations of the vectorised adaptive Kalman filter were then employed in a novel self-modelling curve resolution method called Vectorised Adaptive Kalman Filtering with Iterative Spectral Optimisation. This approach utilised Vertex Vector Sequential Projection analysis to provide initial estimates of the pure component spectra. The transformation matrix of these initial spectra was calculated and iteratively refined using Newton-Gauss-Levenberg / Marquardt non-linear optimisation. During each iterative cycle, new test reference spectra were calculated from the spectral basis vectors by changing the elements of transformation matrix. Each set of test reference spectra are then tested using the vectorised adaptive Kalman filter. The resulting innovations matrix, state-parameter matrix and the diagonal elements of the final state estimate covariance matrix are used to calculate a weighted residual matrix that guides the NGL/M optimisation towards a feasible solution. The performance of this new algorithm was first assessed using simulated UV reaction data by comparing the spectral and concentration profiles with the estimated profiles. The spectral residual sum-of-squares values indicated that the VAKFISO performed as well as the pure variable SMCR method, vertex vector sequential projection. VAKFISO was able to correctly identify the true spectrum of component B, whereas VVSP could only isolate the mixture spectrum corresponding to the maximum concentration of component B. The concentration profiles estimated using VAKFISO exhibited a small amount of rotational ambiguity but the characteristic features of each profile were recovered more successfully than VVSP. When VAKFISO was applied to real data sets, it was observed that initiating the optimisation process using random spectra produced results equivalent or better than those obtained using VVSP initial spectra. One reason for this was that the VVSP initial spectra for the various components were often very similar. Using randomly generated spectra started the optimisation process further away from the final solution and allowed the algorithm to cover a larger search space. When applied to real, highly overlapped data, the VAKFISO method was not able to unambiguously recover the true spectral and concentration profiles. However it did perform as well as existing SMCR methods using

only non-negativity constraints. Specifically, the VAKFISO method was able to identify the most selective regions of each pure spectrum. This information could be useful for further optimisation employing additional constraints.

The base catalysed *N*-benzylation of 1*H*-indole using benzyl bromide was used as test reaction and several reactions using different reactant and base ratios were performed. During each reaction, Raman and UV spectra were acquired and several samples were taken for off-line analysis using HPLC to provide reference measurements. Both the UV and Raman spectra required considerable data preparation and a moving window median filter was written to automate the correction of these data sets. The Raman reaction spectra had a broad, complex baseline contribution and although the median filter provided a good approximation of the underlying baseline spectrum, an automated moving window iterative polynomial baseline fitting method was written. The moving window iterative polynomial baseline fitting method was found to produce better results than median filtering alone and also performed better than the original iterative polynomial baseline subtraction method published by Lieber and Mahadevan-Jansen.

Acknowledgements

I would like to thank my academic supervisor, Dr Anthony Walmsley for his help, guidance and encouragement throughout the duration of the research.

AstraZeneca are gratefully acknowledged for providing the opportunity and funding to undertake part-time study. In particular, I wish to thank my industrial supervisor, Dr Tony Shephard whose experience and advice were invaluable when establishing the project and have continued throughout. I would also like to thank Dr Graham Taylor, Dr Adrian Clarke, Kevin Sutcliffe, Martin Whitehouse and Dr David Ennis for their support. Their consideration and flexibility have allowed me to use the limited time available for study in the most useful and efficient way possible.

Within AstraZeneca, I have also had the privilege to draw upon the knowledge and experience of a number of people. In particular, I would like to thank Dr Steve Eyley, Dr Joël Le Bars, Dr Marijan Štefinović, Glyn Allsop, Dr Eric Merifield and Mike Baker for many useful discussions.

During visits to the University of Hull Chemistry department, I have benefited from spending time with the Chemometrics group. I would like to thank Dr Geir Rune-Flåten, Dr Selena Richards, Dr Ruth Wellock and Dr Tom Dearing for their friendly help and advice during those visits.

I was also fortunate to enjoy support and encouragement from many family and friends. However, I would specifically like to thank my Mother and Father.

Finally but most importantly, I would like to thank my wonderful wife Clare for her unwavering help and support. Undertaking part-time study whilst trying to establish a career in industry does require a significant sacrifice of one's free time, and this obviously affects both the student and his family. The value of Clare's support and patience cannot be overstated. I would also like to dedicate this thesis to my son Daniel; he has brought us so much joy and is a constant reminder of lifes true priorities.

List of Abbreviations

ALS	alternating least squares
API	active pharmaceutical ingredient
ATR	attenuated total reflectance
AU	absorbance units
CALS	constrained alternating least squares
CLS	classical least squares
DAD	diode array detection
EFA	evolving factor analysis
HPLC	high performance liquid chromatography
IPBS	iterative polynomial baseline subtraction
ITTFA	iterative target transformation factor analysis
LSL	large scale laboratory
MCR-ALS	multivariate curve resolution – alternating least squares
NGL/M	Newton-Gauss-Levenberg / Marquardt
OSC	orthogonal signal correction
OPA	orthogonal projection approach
PAC	process analytical chemistry
PAT	process analytical technology
PCA	principal components analysis
PLS	partial least squares or projection to latent structures

RMSEC	root mean square error of calibration
RMSECV	root mean square error of cross validation
RMSEP	root mean square error of prediction
RSD	relative standard deviation
RSSQ	residual sum of squares
SMCR	self modelling curve resolution
SNV	standard normal variate
SVD	singular value decomposition
UV/vis	ultraviolet - visible
VAKFISO	vectorised adaptive Kalman filter with iterative spectral optimisation
VVSP	vertex vector sequential projection
VWD	variable wavelength detector
X-block	matrix of independent variables (*e.g.* spectral data)
Y-block	matrix of independent (*e.g.* spectral) or dependent (*e.g.* concentration) variables.

Algebraic nomenclature

Throughout this thesis, the standard nomenclature for the description of chemometric equations and algorithms was employed. The nomenclature is universal throughout the chemometrics community and is used by the authors of text books and research papers.

Scalars are represented by upper or lower case italics such as a or A. A vector is denoted by a lower case, bold italic letters; for example $\boldsymbol{x}$ or $\boldsymbol{y}$. All vectors are column vectors unless otherwise indicated. Matrices are denoted by upper case, bold italic letters; for example $\boldsymbol{X}$ or $\boldsymbol{Y}$. In general, array dimensions or the upper value of an integer based counter will be denoted by upper case italics; for example the dimensions of a two dimensional array could be defined $\boldsymbol{X}_{JK}$ $(J \times K)$. The transpose operation is denoted by superscript T. The individual elements of a vector or matrix are denoted by subscript indices; for example, the j^{th} element of a $(J \times 1)$ vector $\boldsymbol{x}$ is x_j, whilst the j^{th} and k^{th} elements of a matrix $\boldsymbol{X}$ are denoted x_{jk}. For matrices, individual column or row vectors can be also be referenced; for example, the j^{th} row of $\boldsymbol{X}$ is $\boldsymbol{x}_j^T$ whilst the k^{th} column is denoted $\boldsymbol{x}_k$. The pseudo-inverse of a matrix is often used to calculate the inverse of a collinear (non-invertible) matrix and is denoted by the superscript +. This refers to the matrix operation $\boldsymbol{X}^+ = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$. Predicted values are denoted by the superscript ^ ('hat'), so $\boldsymbol{Y}$ would indicate actual reference values whilst $\hat{\boldsymbol{Y}}$ is a vector or matrix containing the corresponding values predicted using a model.

To describe the Kalman filter or other algorithms that use iterative calculations, the iteration number, time or spectral variable is denoted by a lower case number in parentheses, for example, $\boldsymbol{X}(m)$ is the m^{th} iteration of an algorithm to calculate the matrix $\boldsymbol{X}$. The same nomenclature can be applied to scalars or matrices. Throughout the literature, descriptions of the Kalman filer also employ time-domain terminology to distinguish between estimates made *a priori* or *a posteriori*. The *a priori* estimate of $\boldsymbol{x}$ (conditioned on all prior measurements except the one at time k) is written $\boldsymbol{x}(k|k-1)$. The *a posteriori* estimate (conditioned on all measurements available at time k) is written $\boldsymbol{x}(k|k)$.

Table of Contents

1 INTRODUCTION 1
1.1 Process Analytical Technology (PAT) 1
1.1.1 PAT in the pharmaceutical industry 1
1.1.2 Process analysis of batch processes 1
1.1.3 Process spectroscopy 3
1.1.4 Examples of spectroscopic reaction monitoring 4
1.2 Multivariate data analysis and calibration 7
1.2.1 Taxonomy of data processing methods 7
1.2.2 Univariate calibration 8
1.2.3 Multi linear regression (MLR) 9
1.2.4 Principal component based calibration methods 9
1.3 Calibration free modelling 10
1.4 Self-Modelling Curve Resolution 12
1.4.1 Classification of SMCR methods 12
1.4.2 Relationship between PCA and factor transformation 13
1.4.3 Multivariate Curve Resolution – Alternating Least Squares 15
1.5 Kinetic modelling of reaction spectral data 17
1.6 Band Target Entropy Minimisation 21
1.7 The Kalman filter 24
1.7.1 The Kalman filter in chemistry 24
1.7.2 The adaptive Kalman filter 30
1.7.3 Parallel and multi-model Kalman filters 33
1.8 Model reactions 36
1.8.1 Chlorination of "acetoxyone" using phosphorus oxychloride 37
1.8.2 *N*-benzylation of 1*H*-indole using benzyl bromide 38
1.9 Summary of project aims 39

2 THEORY 40
2.1 Ultraviolet / Visible (UV/Vis) Spectroscopy 40
2.2 Raman spectroscopy 41
2.3 Evolving Factor Analysis (EFA) 44
2.4 Orthogonal Projection Approach (OPA) 45
2.5 Multivariate Curve Resolution – Alternating Least Squares (MCR-ALS) 46
2.6 Partial Least Squares Regression (PLS) 47
2.7 Vertex Vector Sequential Projection (VVSP) 48
2.8 Linear Kalman filter 49
2.9 Adaptive Kalman filter 53
2.10 Vectorised Kalman filters 54
2.11 Vectorised Adaptive Kalman filter with Iterative Spectral Optimisation 58
2.12 Iterative polynomial baseline subtraction 63

3 EXPERIMENTAL 65
3.1 Reagents and Equipment 65
3.1.1 Reagents 65
3.1.2 Spectrometers 66
3.1.3 Chromatographic instruments 69
3.1.4 Reaction vessels 70
3.1.5 Computers and software 73
3.1.6 Visual Basic program used to prepare Cary 50 UV/Vis data 73

3.1.7 Custom Matlab scripts....74
3.2 Combining Self-Modelling Curve Resolution and PLS regression 115
3.2.1 Equipment 115
3.2.2 Laboratory scale development experiments 115
3.2.3 Key reference measurements 116
3.2.4 Overview of the spectral data 117
3.2.5 Derivation of initial concentration profile estimates using EFA 118
3.2.6 Derivation of initial spectral profile estimates using OPA 118
3.2.7 Refinement of profiles using MCR-ALS 119
3.2.8 Calculation of PLS models using refined concentration profiles 119
3.2.9 Comparison of PLS regression coefficients and MCR-ALS spectra 120
3.2.10 Application of the UV method in a Large Scale Laboratory 121
3.2.11 Design of a custom fibre optic cable assembly 122
3.2.12 Testing of a custom fibre optic assembly 125
3.3 Vectorised Kalman filtering for SMCR 131
3.3.1 *N*-benzylation of 1*H*-indole 131
3.3.2 Analysis of reaction samples using HPLC 133
3.3.3 Preparation and analysis of spectroscopic reference standards 135
3.3.4 Preparation and pre-processing of the UV spectral data 136
3.3.5 Optimisation of the IPBS method parameters 137
3.3.6 Preparation and pre-processing of the Raman spectral data 138
3.3.7 Creation of a simulated UV data set for algorithm testing 139
3.3.8 Demonstration of the equivalence of the standard and vectorised linear Kalman filter 141
3.3.9 Application of the adaptive Kalman filter to simulated UV spectra 143
3.3.10 Application of VVSP to simulated UV data 145
3.3.11 Application of VAKFISO to simulated UV data 145
3.3.12 Application of VAKFISO to real UV data 147
3.3.13 Application of VAKFISO to real Raman spectra 148

4 RESULTS AND DISCUSSION 150
4.1 Combining Self-Modelling Curve Resolution and PLS regression 150
4.1.1 Laboratory scale development experiments 150
4.1.2 Overview of the spectral data 151
4.1.3 Derivation of initial concentration profile estimates using EFA 158
4.1.4 Derivation of initial spectral profile estimates using OPA 160
4.1.5 Refinement of profiles using MCR-ALS 163
4.1.6 Calculation of PLS models using refined concentration profiles 165
4.1.7 Comparison of PLS regression coefficients and MCR-ALS spectra 168
4.1.8 Application of the UV method in a Large Scale Laboratory 172
4.1.9 Conclusions: Derivation and implementation of a PLS model using minimal reference data 175
4.1.10 Testing of a custom fibre optic cable assembly 178
4.1.11 Conclusions: Testing of a custom fibre optic cable assembly 189
4.2 Vectorised Kalman filtering for SMCR 190
4.2.1 *N*-benzylation of 1*H*-indole using benzyl bromide 190
4.2.2 Analysis of reaction samples using HPLC 190
4.2.3 Examination of reference standard spectra 193
4.2.4 Preparation and pre-processing of the UV spectral data 195
4.2.5 Optimisation of the IPBS method parameters 198
4.2.6 Preparation and pre-processing of the Raman spectral data 205
4.2.7 Creation of a simulated UV data set for algorithm testing 205

4.2.8 Demonstration of the equivalence of the standard and vectorised linear Kalman filter 207
4.2.9 Application of the adaptive Kalman filter to simulated UV spectra 213
4.2.10 Application of VVSP to simulated UV data 223
4.2.11 Application of VAKFISO to simulated UV data 228
4.2.12 Application of VAKFISO to real UV data 239
4.2.13 Application of VAKFISO to real Raman spectra 247

5 CONCLUSIONS 254
5.1 Background and project aims 254
5.2 Combining SMCR and PLS regression 256
5.3 Kalman filtering for SMCR 260
5.4 Overall conclusions 265
5.5 Further work 267

6 REFERENCES 269

APPENDIX I A1

APPENDIX II A11
ResidualComps.m A11
OPA.m A16
MedianFilter.m A20
LMJ_IPBS.m A26
IPBS.m A30
VVSP.m A37
LinearKF.m A42
AdaptiveKF.m A48
VecLinearKF.m A58
VecAdaptiveKF.m A64
Create_Spectra.m A70
VAKFISO.m A71

1 Introduction

1.1 Process Analytical Technology (PAT)

1.1.1 PAT in the pharmaceutical industry

A number of regulatory organisations such as the United States Food and Drug Administration (FDA) and the European Medicines Agency (EMEA) have implemented initiatives advocating the increased use of Process Analytical Technologies (PAT) during development and manufacture[1]. Historically, a pharmaceutical manufacturing process would be approved based upon a standard operating procedure that had been reproducibly demonstrated to produce product of appropriate quality. If the manufacturer wished to make any changes to the approved process, it was required to submit a large volume of supporting data to the regulatory authority to demonstrate that the product quality would not be affected. The time and cost required to request approval for a process change were sufficient to deter many companies from attempting to implement process improvements. To remove this barrier, regulatory authorities are encouraging a move away from fixed operating procedures to flexible, knowledge-based process development using "quality-by-design" concepts. These concepts combine the use of experimental design, continuous process monitoring and process modelling and control[2, 3]. The goal of employing a "quality-by-design" approach is to study a large experimental space, called a "knowledge space", for a particular product or process. From this "knowledge space", a smaller "process space" that is known to provide product of the desired quality is registered with the regulatory authorities. A manufacturer then has the opportunity to optimise a process as making changes to the process parameters within the "process space" are permitted. An important part of this concept is the use of process analytical technologies to continuously monitor and control processes.

1.1.2 Process analysis of batch processes

Currently, the pharmaceutical and fine chemical industries predominately use batch reactors in their pilot- and full-scale manufacturing plants. The research chemists and engineers responsible for developing a process therefore use small-scale batch reactors to emulate the full-scale equipment for which the process is destined. A batch process is characterised by the introduction of the raw materials into the process in a specific sequence and their conversion into products within a finite duration. The objective of operating a batch process is to achieve reproducibility of those process variables that have

the greatest influence upon the products quality parameters. However, most processes will exhibit some batch-to-batch variation owing to differences in raw materials or performance of the plant equipment. At the end of each batch, a sample is sent to a quality control laboratory to confirm that the process has produced material of suitable quality. If a severe process deviation has occurred, the final product may not meet the required specification and would have to be re-worked or disposed.

Through the use of process analytical technology it is possible to move analytical instrumentation closer to a process. This facilitates analysis at more frequent intervals and measurements can therefore be made throughout the duration of the process. This provides an opportunity for the process operator or control system to detect process deviations much earlier, and if necessary, adjust process parameters to prevent the deviation from affecting the quality of the end product. For this reason, high throughput industries such as the bulk chemical, oil, petroleum and polymer industries have implemented some form of PAT for several decades. Early instrumentation included the measurement of traditional variables such as temperature, pressure and flow; and the use of simple chemical sensors such as pH meters and oxygen sensors[4]. Advanced instrumentation such as on-line chromatography, spectroscopy, NMR and X-ray was also adopted much sooner than other lower throughput industries such as fine chemicals, pharmaceuticals and other speciality products.

The key driver for high throughput industries to move analytical instrumentation closer to the process was to reduce the time required to extract a sample, send it to a remote quality control laboratory and receive the result. If this time was several hours, a large volume of material would have been processed, and may this may need to be disposed or re-worked if it did not meet specification. The considerable cost benefit of diagnosing process deviations earlier was sufficient to justify the investment in PAT.

Guenard and Thurau[4] discussed the differences between the implementation of PAT in the pharmaceutical and bulk chemical industries. One reason for lower exploitation of PAT in the pharmaceutical industry was attributed to the high attrition of chemical entities that reach commercial manufacture. The manufacturing costs of commercial pharmaceutical products are also a lower proportion of the total cost of the product owing to the high development costs and low manufacturing throughput. For regulatory and release purposes, most pharmaceutical processes are devised as a series of discrete batch processes. Historically, the regulatory requirements for product release have also led to

pharmaceutical manufacturing sites building extensive quality control laboratories and the preferential use of off-line testing utilising sophisticated instrumentation.

The availability of process measurements naturally led to the development and implementation of control charts[5]. Statistical Process Control (SPC) charts allow the evolution of a process variable to be compared with statistical limits derived from a number of 'good' batches manufactured using normal operating conditions. These charts are calculated individually for each process variable of interest and do not consider any interaction between the different variables. Multivariate Statistical Process Control is the extension of SPC principles to multivariate data[6-10]. A variety of multivariate projection methods can be used to find linear combinations of process variables and an ideal trajectory can be calculated. The variation of the latent variables can also be correlated to input raw material quality or the final product quality. MSPC models are more robust to variation of individual 'noisy' variables and can account for interactions between multiple variables. These methods also allow spectroscopic variables to be combined with traditional process variables to create complex models that incorporate both physical and chemical parameters.

1.1.3 Process spectroscopy

Process analytical technology encompasses any analytical instrumentation that is used to make timely process measurements in-line, on-line or at-line, and include electrochemical methods, chromatography, spectroscopy, mass spectrometry and even specific sensors such as pH or temperature probes, flow meters and pressure sensors. In contrast to traditional process variables such as temperature, pressure, flow and viscosity; process spectroscopy provides data that is much richer in chemical information[11]. Spectroscopy is non-destructive and can be interfaced with the process to facilitate rapid sampling. These two factors make process spectroscopy a powerful tool as it can be used to rapidly, and frequently determine the chemical composition of a process stream without disturbing the system. The sampling rate of an in-line spectroscopic method is often far greater than other in-line and off-line analytical methods such as chromatography. This allows in-line spectroscopy to provide more detailed trend information that can be extremely useful for gaining understanding about a process, such as the appearance or disappearance of reactants, intermediates and products. The detailed concentration-time profiles extracted from spectroscopic data can then be fitted to a number of different kinetic models to

identify or confirm the reaction mechanism. The empirical kinetic model can then be used to predict the performance of the reaction when it is scaled-up to a larger vessel.

There are a number of different types of spectroscopy that have been employed for process analysis and they are distinguished by the region of the electromagnetic spectrum they use to interact with a sample, and also the physical phenomenon by which they interact with the sample. Some of the most widely used techniques include ultraviolet / visible absorbance spectroscopy, mid-infrared absorbance spectroscopy, near-infrared absorbance spectroscopy, near-infrared diffuse reflectance spectroscopy, Raman spectroscopy, fluorescence spectroscopy, microwave spectroscopy and acoustic spectroscopy[4, 12]. Each of the techniques offers its own set of advantages and disadvantages that must be taken into account when selecting a method. For example, what is the physical nature of the sample and would a transmission, reflection or emission based measurement be most suitable? What are the differences between the molecular structure of the reactants and products in the reaction, and would the change in molecular structure produce observable differences in the measured spectra? Is the method required to accurately quantify a component in a mixture or would a qualitative (relative change) method be suitable to detect steady-state conditions? What is the concentration range of the species to be observed and does the method offer suitable limits of detection? It is usually a combination of all of these factors that ultimately dictate the choice of spectroscopy employed.

1.1.4 Examples of spectroscopic reaction monitoring

A number of different *in-situ* spectroscopic methods have been used to monitor chemical reactions at laboratory and plant scale.

Ampiah-Bonney and Walmsley used Raman spectroscopy to monitor the acid catalysed esterification of ethanol[13]. In this work, molecular fluorescence and other process noise contributed to the measured spectra and obscured the chemical variation of interest. It was observed that PCA of the mean-centred data required only two principal components to model nearly 99.9% of the total variance in the data. The first principal component was found to correspond to the fluorescent baseline dominating the spectra, whilst principal components three and above were modelling measurement noise. Reconstruction of the spectral data using only the second principal component removed the contributions from fluorescence and measurement noise. This allowed the reaction profiles of interest to be plotted using the selective bands of each component. This demonstrated the advantage of

making multivariate measurements, even if the individual chemical components were spectrally resolved.

De Braekeleer *et al.* used *in-situ* mid-infrared spectroscopy and self-modelling curve resolution methods to determine the end-point of an unspecified reaction[14]. The dissimilarity criterion provided by the orthogonal projection approach was used to determine when the reaction had reached steady-state. In this case the reference spectra used to calculate the dissimilarity values were taken from the start of each new region where a different perturbation had occurred, such as a temperature change or the addition of reagent. The spectra from the various steady-state regions were then combined and modelled using multivariate curve resolution-alternating least squares to provide relative quantitative information about the evolution of the reaction. This example also demonstrated that a simple qualitative model could be used to indicate the reaction end-point and reveal the effect of temperature upon the pure component spectra.

Mid-infrared spectroscopy has also been used to monitor a batch polymerisation reaction of methylmethacrylate (MMA) to polymethylmethacrylate (PMMA)[15]. Twelve solutions of known MMA/PMMA composition were used to construct a PLS model that was subsequently used to predict the concentration of MMA in the reaction mixture. The MMA concentrations from eighteen batches operating under normal operating conditions were predicted using PLS and then used to build statistical control charts. Three further batches were used to test the performance of the control chart. The first test batch was operated using normal conditions and the concentration remained well within the 99% control limits of the model. When the second test batch was operated below the normal temperature, the control charts clearly indicated that the conversion was occurring slower than expected. In batch three, a small amount of polystyrene was added and this also seemed to slow down the conversion to product. This example demonstrated that multivariate spectroscopic measurement of a process can serve two purposes: the measurement can be used to predict a property of interest, for example reactant concentration; and the spectra were also sensitive to abnormal physical and chemical perturbations, providing early indication of a process fault or deviation.

Ma *et al.* reported the use of *in-situ* spectroscopy and calorimetry to characterise batch reactions[16]. In this work, the acetylation of salicylic acid was monitored using UV/Visible spectroscopy with simultaneous logging of the reaction temperature and reactor jacket temperature. Several reactions were performed at different temperatures and

concentrations of salicylic acid. The data sets were combined and subjected to multiway self-modelling curve resolution. The reason for combining the data sets was to remove the scaling ambiguity of the concentration profiles that would be observed if the data sets were analysed individually, whilst still allowing the batch-to-batch variation of the pure component spectral profiles to be retained. The relative intensity of the estimated concentration profile for the product showed good correlation with the relative amount of limiting reagent added.

NIR spectroscopy was used to monitor the esterification of isoamyl alcohol using acetic acid[17]. Eleven batches were performed according to an experimental design (3^2 + 2 centre points) which varied the molar ratio of reagents and the amount of catalyst. NIR spectra were acquired at three minute intervals and the resulting three-dimensional array of spectral data (batch × time × wavelength) was decomposed using parallel factor analysis (PARAFAC). PARAFAC was chosen over two-way curve resolution methods because it will give a unique mathematical decomposition for a particular set of data. The experimental design parameters and loadings produced by the PARAFAC model were then fitted to a regression model. In this way, the affect of changing each of the design parameters upon the measured spectra could be interpreted from the loadings vectors and characteristic bands could be identified.

More recently, Garrido *et al.* reported the use of NIR and ^{13}C-NMR spectroscopy to study the reaction between phenylglycidylether and aniline[18]. The reactions were performed in the small liquid cell of a NIR spectrophotometer and spectra were acquired at five minute intervals. An identical reaction was performed simultaneously in a different cell using the same reagent stoichiometries and reaction temperature. From this cell, samples were taken at several time points and their ^{13}C-NMR spectra were acquired. MCR-ALS was applied to the NIR data using six concentration values obtained from the quantitative ^{13}C-NMR measurements as additional constraints. Using this external information reduced the rotational ambiguity of the solution and the concentration profiles estimated by MCR-ALS were verified by off-line HPLC measurements.

1.2 Multivariate data analysis and calibration

1.2.1 Taxonomy of data processing methods

The taxonomy shown in Figure 1.1 illustrates the general classification of the various spectral pre-processing and data analysis methods used or described in this thesis.

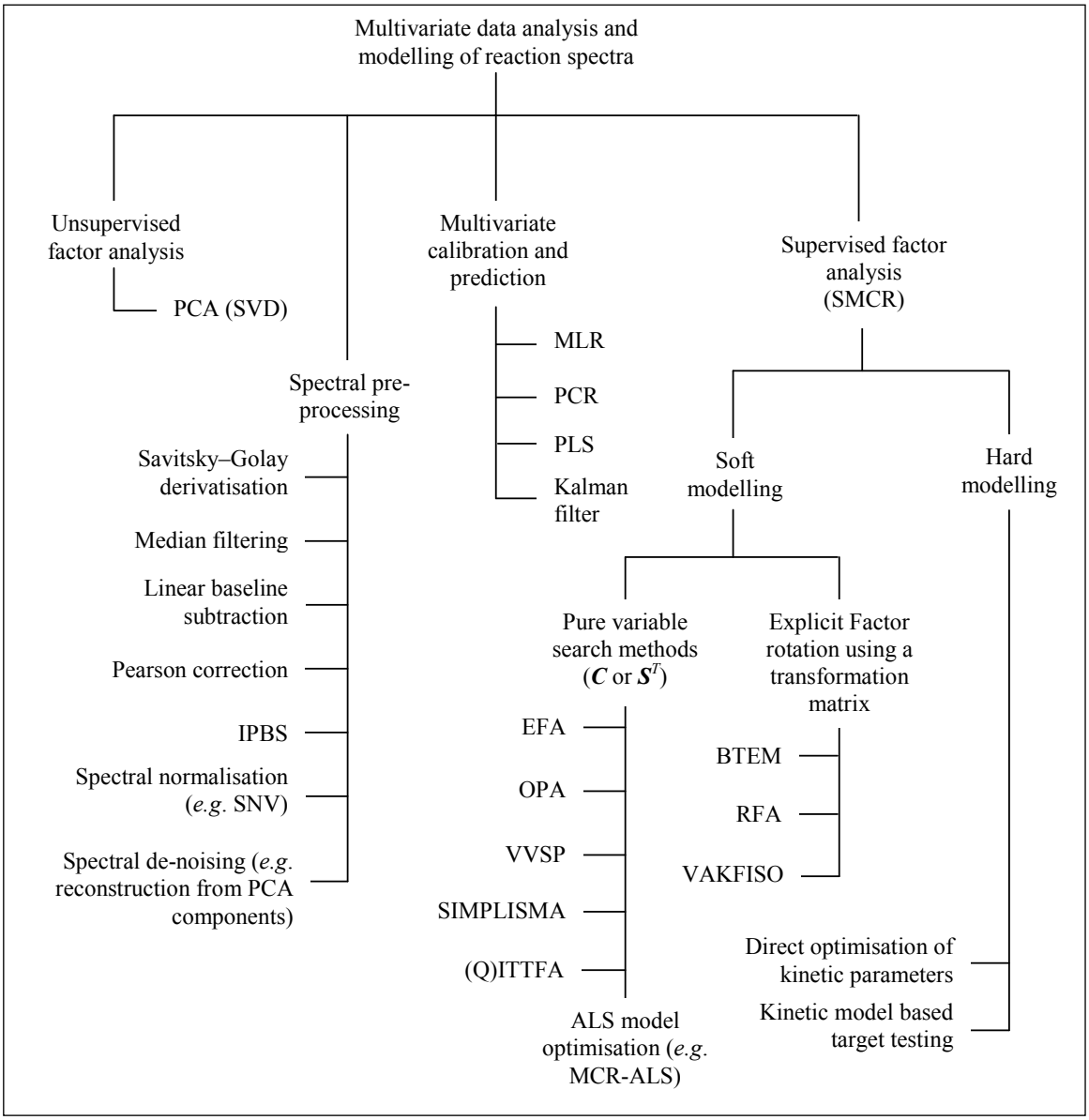


Figure 1.1: Taxonomy of the various spectral pre-processing and data analysis methods used or described in this thesis.

One of the methods most commonly used for the analysis of multivariate data sets such as process spectra is principal components analysis. This is example of an unsupervised method of data analysis as it does not try to fit the data to a known physical model require any prior knowledge. In many cases, process spectra can contain additional unwanted contributions such as multiplicative scattering effects, variable baseline or intensity scaling and additional measurement artefacts. A variety of spectral pre-processing methods are therefore required to remove the additional contributions from the spectra so the true underlying chemical information can be recovered. If the purpose of the application is to predict a chemical property from the measured spectrum, a number of methods such as MLR, PCR and PLS are available. In addition to a set of training spectra, these methods also require a set of reference values for the property of interest in order to construct the calibration model. When reference values are not available, supervised methods of data analysis can be applied. The objective of supervised data analysis methods is to deconvolute a set of spectra to recover the underlying structure of a data set that best represents the true physical process occurring in the system. Hard-modelling methods use a theoretical model of the physical process that the data represents, for example a kinetic model can be used to trial different reaction mechanisms and rate constants to find the best model for the data set. Soft-modelling methods do not implement a theoretical model but many apply a least-square approach to find the best fit of the data set. To reduce the range of feasible solutions obtained, known constraints such as spectral non-negativity can be applied.

1.2.2 Univariate calibration

Univariate calibration is a regression technique commonly used to derive the mathematical expression that relates two single variables to each other[19]. The independent variable, often denoted x is a scalar value and is derived directly from an analytical measurement. For example, x could be the integrated peak area from a chromatographic measurement or the absorbance value at a single wavelength obtained from a spectrophotometric measurement. The dependent variable, often denoted y is the physical or chemical property, such as concentration. For analytical chemists, the aim of employing a univariate calibration is to establish the mathematical relationship between the instrument response (independent variable) and the chemical property of interest (dependent variable) using a set of samples for which both sets of variables are known. The regression coefficients (slope and intercept for a linear calibration) are then used to predict the dependent variable of unknown samples using the measured instrumental response (independent variable).

1.2.3 Multi linear regression (MLR)

It is possible to extend linear regression to include multiple independent variables. This has the advantage of averaging out the noise that may influence individual variables and helps to stabilise the regression coefficients. Using multiple independent variables is also necessary if more than one chemical component is contributing to the measurement[20, 21]. For example, if a sample contains N components, at least N independent variables are required to calculate a regression model. Multi linear regression (MLR) can be applied to simple spectroscopic calibration applications when the concentrations of all the components contributing to the training samples are known. The relationship between the matrix of independent variables (spectra), denoted $\boldsymbol{X}$, and the dependent variables (concentrations), denoted $\boldsymbol{Y}$, is shown in equation 1.1. Once the matrix of regression coefficients have been calculated, the concentrations of new samples can then be predicted from their spectra using equation 1.2.

$$\boldsymbol{B} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y} \qquad \text{(equation 1.1)}$$

$$\hat{\boldsymbol{Y}} = \boldsymbol{XB} \qquad \text{(equation 1.2)}$$

The limitations of the MLR approach are that the matrix inversion step in equation 1.1 can be yield unstable regression coefficients when $(\boldsymbol{X}^T\boldsymbol{X})$ is ill-conditioned (almost singular owing to collinearity between the variables of $\boldsymbol{X}$). If the spectra contain varying amounts of additional, unknown components, MLR will also be unsuitable.

1.2.4 Principal component based calibration methods

Principal components regression (PCR) and partial least squares (PLS) regression are two of the methods most commonly applied to multivariate calibration problems. These methods are suited to spectroscopic calibration because they include data compression steps that help to overcome the problem of collinearity in the matrix of independent variables (X-block).

In principal components regression[12, 21-25], data compression of the spectroscopic data (X-block) is achieved by first applying principal components analysis to the matrix of independent variables ($\boldsymbol{X}$) as shown in equation 1.3. The first A principal components that adequately model the matrix of spectroscopic data are then used to calculate the regression coefficients using the expression shown in equation 1.4.

$$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{P}^T + \boldsymbol{E} \qquad \text{(equation 1.3)}$$

$$\boldsymbol{B} = (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\boldsymbol{Y} \qquad \text{(equation 1.4)}$$

Since the spectroscopic data has been compressed by calculation of the first A principal components, and the columns of the score matrix $\boldsymbol{T}$ are orthogonal, the problem of collinearity in the spectral data is overcome. The inverse of $(\boldsymbol{T}^T\boldsymbol{T})$ is well conditioned and yields stable regression coefficients. PCR can be applied to spectra with many variables and does not require the concentrations of all components to be known so it is useful for complex mixtures. Disadvantages of PCR are that the largest principal components may not be correlated to the property of interest and therefore a larger number of principal components are required. This can make interpretation of the loadings vectors more difficult. PCR also assumes that the vector or matrix of dependent variables is error free.

Partial least squares[12, 20-25] is an alternative approach to PCR and has a number of advantages. The PLS algorithm is described in chapter 2 (section 2.6) so only a brief introduction is given for comparison with PCR. Unlike PCR that applies data compression to the spectral (independent) variables only, PLS decomposes both the independent and dependent variables. The objective of PLS is to find a common set of principal component scores that maximise the amount of variance explained in both the X- and Y-blocks. By modelling both the independent and dependent variables using the same set of scores, not only does PLS achieve data compression, it also calculates factors that maximise the correlation between $\boldsymbol{t}$ and $\boldsymbol{y}$. The additional advantage of PLS over PCR is that the dependent variables are also compressed using a small number of principal components. This helps to remove measurement error from the Y-block and thus explains the structured variance in $\boldsymbol{X}$ and $\boldsymbol{Y}$ using fewer components.

1.3 Calibration free modelling

To construct a spectroscopic calibration model, a common approach is to obtain a data set comprising a series of spectral measurements and the corresponding values for the properties of interest are measured using an external reference method, for example HPLC, GC, NMR, titration, assay *etc.* A mixture design could be used to design a set of non-reacting mixtures with known composition, or process samples could be taken whilst acquiring spectral data. In reactions or processes where it is not possible to prepare mixtures of known composition, perhaps because reference components are unstable or

cannot be isolated, sampling from the process is usually the only practical alternative. However, the lack of reference standard materials for the components that cannot be isolated will still preclude the quantification of that component using an off-line reference method.

The difficulties described above naturally led to the investigation and development of 'calibration free' methods of data analysis and prediction. Kubista *et al.*[26] proposed a method suitable for the calibration of equilibrium systems such as pH-, concentration-, temperature- and ionic strength-titrations. The method was based upon principal components analysis of the spectral data followed by derivation of a rotation matrix that transforms the abstract spectra (principal component loadings) and concentration profiles (principal component scores) into the true spectral and concentration profiles. This approach used the measurement of physical properties such as pH, volume, temperature, pressure, ionic strength *etc.* that affected the concentrations of the components in a predictable way. By fitting the data using a known thermodynamic model, the concentration parameters were derived indirectly. Although this method has been demonstrated for systems with well-understood equilibria, it is not suited to dynamic systems such as reactions.

If a dynamic process can be described by first principle models, the mathematical model could be used to calculate the concentrations directly. A comparison of kinetic and soft modelling for reaction modelling was published by Dyson *et al.*[27]. The catalytic hydrogenation of 1-chloro-2-nitrobenzene was monitored using Raman and IR spectroscopy and the authors used kinetic modelling to calculate the relative concentration profiles. The calculated profiles showed good agreement with the off-line NMR results. A number of self modelling curve resolution methods such as evolving factor analysis (EFA), Window Factor Analysis (WFA) and Iterative Target Transformation Factor Analysis (ITTFA) were used to derive the relative concentration profiles of the major species directly from the spectroscopic data and these too compared well with the off-line NMR results. In this example, the qualitative (relative) concentration profiles were obtained *post-hoc*.

A calibration-free method using NIR reaction spectra acquired during the 4-(dimethylamino)pyridine catalysed esterification of butanol was reported by Gemperline *et al.*[28]. A non-linear fitting routine was used to fit a third order reaction mechanism to five batches simultaneously. In this work, the data from four batches was used to predict the

profiles of the fifth batch. Prediction of new data required the augmentation of the new spectra with spectra acquired in previous runs and fitting a multi-way kinetic model using initial estimates of the kinetic parameters obtained from the previous four reactions.

1.4 Self-Modelling Curve Resolution

Owing to the difficulty, time and cost of constructing valid calibration sets for spectroscopic multivariate calibration models, a number of calibration-free methods based upon factor analysis or alternating least-squares have been devised. These methods derive an empirical model directly from the spectroscopic data and attempt to minimise the sum-of-squares of the residuals between the measured and the reconstructed data.

1.4.1 Classification of SMCR methods

Self Modelling Curve Resolution (SMCR) is the general name given to the family of techniques employed for the mathematical deconvolution of two-way multivariate data obtained from instrumentally unresolved multi-component mixtures[29]. The function of these techniques is to resolve a bilinear data set obtained from a complex multi-component system into factors that describe the instrumental response and relative concentration profile of each chemical species contributing to the unresolved signal.

In a review of SMCR, Jiang *et al.*[29] classified SMCR methods into two types; *unique* resolution and *rational* resolution. Methods that are able to find a unique definition of the factors required to model each chemical species are said to provide unique resolution. These methods often exploit selective regions of the data that only contain information about a specific component. If selective regions are available for most of the components, a unique resolution should be possible. The limitation of unique resolution methods is that in practice, selective regions are often unavailable. The second group of methods are said to provide a rational resolution. These methods use prior knowledge about the expected properties of the factors, such as non-negativity, to produce a feasible solution (or set of solutions) for each species. When the various components exhibit poor resolution, *i.e.* a high degree of correlation between two or more species in the spectral or time dimension of the data, the accuracy of the recovered profiles may be sub-optimal because of rotational or intensity ambiguity.

1.4.2 Relationship between PCA and factor transformation

Principal components analysis is a factor analysis method commonly used to express the variance structure of a two-dimensional array of data using a reduced number of abstract factors (principal components). The history and algorithms of PCA are described in almost all chemometrics text books and a number of review or tutorial papers are also available[20-25, 30-33]. The basic theorem of principal components analysis is that a matrix of data, $\boldsymbol{X}$ with dimensions $(J \times K)$ can be expressed as the product of two smaller matrices as shown in equation 1.5.

$$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{P}^{\boldsymbol{T}} + \boldsymbol{E} \qquad \text{(equation 1.5)}$$

The scores matrix, denoted $\boldsymbol{T}$ has the dimensions $(J \times A)$ where J is the number of objects (samples or spectra) and A is the number of principal components included in the model. The columns of $\boldsymbol{T}$ are orthogonal. The loadings matrix, denoted $\boldsymbol{P}$ has the dimension $(K \times A)$ where K is the number of measurement variables. The columns of $\boldsymbol{P}$ are orthonormal and represent the basis vectors that span the $A-$dimensional subspace of the original measurement data. The residual matrix, denoted $\boldsymbol{E}$ has the same dimensions as $\boldsymbol{X}$ and contains the remaining variance present in $\boldsymbol{X}$ that is not modelled by the first A principal components. Singular value decomposition (SVD) is a computationally efficient way of factorising a data matrix[30, 34] and is commonly used to perform PCA. The SVD factorisation of a matrix is shown in equation 1.6.

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}} \qquad \text{(equation 1.6)}$$

If $J < K,$ which is typical for spectroscopic measurements, $\boldsymbol{U}$ is a $(J \times J)$ matrix of orthonormal left singular vectors that span the column space of the original data. $\boldsymbol{\Sigma}$ is a $(J \times J)$ diagonal matrix whose off-diagonal elements are zero and the diagonal elements contain the singular values. Singular values are equivalent to the square-root of the corresponding eigenvalues and describe the amount of variance captured by each factor. $\boldsymbol{V}$ is a $(K \times J)$ matrix of orthonormal right singular vectors that span the row space of the original data. The matrices obtained by subjecting a matrix to SVD are related to principal component scores and loadings by the equalities shown in equation 1.7.

$$\boldsymbol{T} = \boldsymbol{U}\boldsymbol{\Sigma}, \quad \boldsymbol{P}^{\boldsymbol{T}} = \boldsymbol{V}^{\boldsymbol{T}} \qquad \text{(equation 1.7)}$$

Principal components analysis is a powerful method that allows the mathematical rank of the data to be determined. The mathematical rank is the number of factors (principal components) required to reproduce the measurement matrix such that the residual matrix contains only measurement noise. The chemical rank is the number of chemical species that make independent contributions to the mixture data and is usually a lower number than the mathematical rank owing to additional noise contributions from non-ideal measurements. Determining the correct chemical rank is an important part of SMCR. If the chemical rank of the data is N, the matrices obtained by SVD can be truncated to retain only the first N singular vectors / singular values, denoted $\bar{\boldsymbol{U}}_{(J\times N)}$, $\bar{\boldsymbol{\Sigma}}_{(N\times N)}$ and $\bar{\boldsymbol{V}}_{(K\times N)}$.

The scores and loadings vectors obtained from PCA are often called abstract factors because they are obtained by the mathematical decomposition of the measurement data such that the maximal amount of variance is explained by the minimal number of orthogonal factors. Although the abstract factors (principal component loadings) will possess spectrum like features, they do not represent the true spectral profiles of the individual chemical components. However, the orthonormal loadings vectors do span the same spectral subspace that contains the true spectral profiles and the correct linear combination of these basis vectors will generate the true spectral profiles[31]. Factor transformation can therefore be used to transform abstract factors into physical or chemically meaningful profiles. This transformation is summarised in equation 1.8 where $\boldsymbol{R}$ is an invertible $(N\times N)$ transformation matrix.

$$\boldsymbol{S}^{\boldsymbol{T}} = \boldsymbol{R}^{-1}\bar{\boldsymbol{V}}^{T} \qquad \text{(equation 1.8)}$$

$$\boldsymbol{C} = \bar{\boldsymbol{U}}\boldsymbol{R} \qquad \text{(equation 1.9)}$$

If the correct values for the elements of $\boldsymbol{R}$ can be determined, the orthogonal abstract factor matrices $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{V}}^{T}$ can be transformed into a new set of matrices $\boldsymbol{C}$ and $\boldsymbol{S}^{\boldsymbol{T}}$. The columns of $\boldsymbol{C}$ describe the true (unscaled) concentration profiles of each chemical component and the rows of $\boldsymbol{S}^{\boldsymbol{T}}$ contain the corresponding pure component spectral profiles.

Most SMCR methods do not explicitly use a transformation matrix to produce new estimates of the concentration and spectral profiles. However, they do all share the

common goal of trying to find feasible estimates of the real profiles $\boldsymbol{C}$ and $\boldsymbol{S}^T$. The bilinear model used to express the matrix measurement data as the product of the true concentration profiles and the true spectral profiles of each component contributing to the system is shown in equation 1.10.

$$\boldsymbol{X} = \boldsymbol{C}\boldsymbol{S}^T + \boldsymbol{E} \qquad \text{(equation 1.10)}$$

The objective of curve resolution is to find the best fit of the data (in a least-squares sense) using the above bilinear model such that the residual between the reconstructed and measured data matrix is minimised (equation 1.11).

$$E = \| \boldsymbol{X} - \boldsymbol{C}\boldsymbol{S}^T \|^2 \qquad \text{(equation 1.11)}$$

1.4.3 Multivariate Curve Resolution – Alternating Least Squares

Multivariate Curve Resolution – Alternating Least Squares (MCR-ALS) is a rational resolution technique that has successfully been applied to a wide range of analytical problems[35]. The success of MCR-ALS can be attributed to its flexibility. Any number of chemometric methods can be used to obtain initial estimates of either the concentration or spectral profiles. These can then refined using an alternating least squares approach defined by equation 1.12 and equation 1.13. Since both the concentration and spectral estimates are alternatively re-estimated, neither set of profiles will dominate the final model.

$$\hat{\boldsymbol{S}}^T = (\hat{\boldsymbol{C}}^T\hat{\boldsymbol{C}})^{-1}\hat{\boldsymbol{C}}^T\bar{\boldsymbol{X}} \qquad \text{(equation 1.12)}$$

$$\hat{\boldsymbol{C}} = \bar{\boldsymbol{X}}\hat{\boldsymbol{S}}(\hat{\boldsymbol{S}}^T\hat{\boldsymbol{S}})^{-1} \qquad \text{(equation 1.13)}$$

Solutions obtained from an alternating least squares procedure that only incorporate basic constraints such as non-negativity can suffer from two types of ambiguity. Scaling ambiguity occurs when the relative intensity of a components profile with respect to another component is incorrect. This can be corrected using external information to rescale the profiles. A more severe problem is rotational ambiguity. This is characterised by a difference in the shape of recovered and true profiles and often occurs when the measurement data does not contain selective regions for each of the components. Rotational ambiguity can produce a range of different solutions that are all feasible in a least-squares sense and obey non-negativity constraints but have different spectral and concentration profiles.

What makes MCR-ALS such a powerful technique is that a number of constraints can be applied to either the concentration or spectral matrix during each cycle of the ALS procedure. These constraints force the data to comply with a range of criteria that are appropriate to the data, for example non-negativity, uni-modality, closure (sum of components is a constant) and equality (certain elements of the concentration or spectral profiles match known values). The flexibility with which these constraints can be applied means that any external information that is significant to the data can be incorporated and used to guide the resolution.

Owing to its flexibility, MCR-ALS has become widely used and a considerable number of examples describing its application to a wide variety of analytical systems are published in the chemistry literature. Garrido *et al.*[36] recently published a review describing the use of MCR-ALS applied to spectroscopic data acquired during chemical reactions and contains over one hundred references to examples published between 2000 and 2007. The examples included in this review paper include the application of MCR-ALS to spectroscopic reaction data acquired using UV/Vis, fluorescence, fluorescence energy transfer (FRET), nuclear magnetic resonance, circular dichroism, near-infrared, Raman, and FTIR spectroscopy.

As with all modelling procedures, the quality of the model produced by MCR-ALS is dependent upon the quality of the data and the initial estimates used. For this reason, researchers continue to investigate and develop new methods for calculating initial estimates that can be further resolved using ALS procedures. Richards and Walmsley proposed a novel method called Quantitative Iterative Target Transformation Factor Analysis (QITTFA) that could be used to provide initial spectral estimates for further refinement by MCR-ALS[37]. This method can be summarised by the following steps. A $(c \times c)$ matrix of needle input spectra (a needle spectrum consists of one spectral variable set to one and all other variables set to zero) is first created, where c is the number of spectral variables. The position of the needle is unique to each needle spectrum. The original $(r \times c)$ data matrix is subjected to singular value decomposition and the first nc primary eigenvectors are retained. A matrix of needle output spectra are then created by the sequential projection of each needle input spectrum onto the spectral subspace spanned by the first nc right singular vectors (denoted $\boldsymbol{V}^T$). Iterative target transformation factor analysis is applied to each needle input spectrum until a non-negative spectral profile is obtained. The final spectral profiles output by the ITTFA procedure are collected into a

$(c \times c)$ matrix denoted $\boldsymbol{Z}_s$ and simple-to-use interactive self modelling mixture analysis (SIMPLISMA) is used to select the nc purest spectra. This approach can yield good initial spectral estimates that are well resolved and can be further refined using MCR-ALS to incorporate additional constraints.

Richards *et al.*[38] applied the QITTFA method to the quantification of ethylene, acetic acid, water, vinyl acetate monomer and carbon dioxide from NIR spectra acquired during an industrial vinyl acetate monomer (VAM) process. Utilising spectral non-negativity and concentration non-negativity and correlation constraints, the matrix of measurement data was deconvoluted using MCR-ALS with initial spectral estimates provided by QITTFA. The models obtained using quantitative SMCR compared well with the established PLS calibration but offered a 90% reduction in calibration time owing to the reduced number of training samples used.

To overcome the issue of rank-deficient data obtained from spectroscopic monitoring of chemical reactions, a number of papers describing the simultaneous application of MCR-ALS to multiple data sets have been published. Combining several experiments run using different molar ratios of reactants can overcome rank overlap that occurs when two reactants or products are formed at similar rates[39, 40]. For example, Garrido *et al.*[41] created an augmented measurement matrix of NIR spectra acquired during five runs of a curing reaction using different ratios of phenyl glycidyl ether and aniline. The rank of the individual data matrices was three, which was correct for the underlying reaction scheme. When the data sets were combined, the rank of the augmented matrix matched the total number of absorbing species (four) and confirmed that rank-deficiency had been broken. This approach allowed the spectral and concentration profiles of all four species to be resolved using MCR-ALS.

1.5 Kinetic modelling of reaction spectral data

A significant amount of valuable information can be gained through the acquisition and modelling of reaction spectra. To model a reaction system using first principle kinetic expressions, one must correctly identify the order of the reaction kinetics and obtain the values of the various constants used in the rate constants. In the soft-modelling curve resolution approaches described in the previous sections, the matrix of reaction spectra is factorised into two smaller sub-matrices corresponding to the estimated pure spectral profiles of each active component and their corresponding concentration profiles.

Providing the concentration profiles are correctly resolved and scaled (to describe the concentration of each species in mol.L^{-1} units), the correct rate equation can be selected and the rate constant derived from the concentration profiles. This approach can be considered as the mathematical deconvolution of the mixture data to obtain feasible estimates of the spectral and concentration profiles, followed by derivation of the kinetic parameters from the estimated concentration profiles.

An alternative approach to the soft-modelling approach described above is to calculate the concentration profiles directly using a specific kinetic model. The various kinetic parameters are found using an iterative fitting method to obtain the best fit of the experimental data. Puxty *et al.*[42] and Maeder and Neuhold[22] published excellent reviews of the methods available for the kinetic modelling of multivariate spectroscopic data with non-linear least-squares regression. The basis of these methods is that if the correct kinetic model has been identified, an optimisation method can be used to determine the best estimates of the kinetic parameters. During each iterative cycle of the optimisation process, the concentration profiles are calculated using the current estimates of the kinetic parameters. The matrix of pure component spectral profiles can then be calculated using least-squares owing to the linear relationship between the concentration of a species and its measured spectral response. The residuals between the true and estimated measurement response matrix are used as the objective function to be minimised during the optimisation process. These steps are summarised as follows. The matrix of measured reaction spectra can be expressed as the product of the matrix of concentration profiles ($\boldsymbol{C}$) and the corresponding matrix of pure spectral profiles ($\boldsymbol{S}^T$), plus un-modelled noise as expressed by equation 1.14. The kinetic parameters $\hat{\boldsymbol{k}}$ are estimated using a suitable optimisation method and the concentration matrix $\hat{\boldsymbol{C}}$ is calculated using the appropriate rate equations. The current estimate of the matrix of concentration profiles is used to calculate a matrix of spectral profiles using the explicit least-squares expression shown in equation 1.16. The residual matrix is calculated using equation 1.17. During the optimisation process, some function of the residual matrix $\boldsymbol{E}$ is used as the objective function to be minimised.

$$\boldsymbol{X} = \boldsymbol{C}\boldsymbol{S}^{\boldsymbol{T}} + \boldsymbol{E} \qquad \text{(equation 1.14)}$$

$$\hat{\boldsymbol{C}} = f(\text{model}, \hat{\boldsymbol{k}}) \qquad \text{(equation 1.15)}$$

$$\hat{\boldsymbol{S}}^T = (\hat{\boldsymbol{C}}^T\hat{\boldsymbol{C}})^{-1}\hat{\boldsymbol{C}}^T\boldsymbol{X} \qquad \text{(equation 1.16)}$$

$$\boldsymbol{E} = \boldsymbol{X} - \hat{\boldsymbol{C}}\hat{\boldsymbol{S}}^T \qquad \text{(equation 1.17)}$$

Since the spectroscopic absorbance coefficients (the pure spectral profiles) can be calculated explicitly using the least-squares expression shown in equation 1.16, the number of parameters to be estimated is reduced to a few kinetic parameters only. If the quality of the current estimates can be expressed as a scalar value, for example the sum-of-squares of the residual matrix $\boldsymbol{E}$, simplex optimisation could be applied to find the optimal values of the unknown kinetic parameters. However, Maeder and Neuhold suggest that for approximately ten parameters or more, Newton-Gauss-Levenberg / Marquardt non-linear optimisation is recommended[22]. This is a gradient based method that uses the full residual matrix to find the optimal parameters. It is called a gradient method because the shift parameter used to produce new estimates of the unknown parameters is calculated from the derivative of the residuals with respect to the parameters. This method is particularly robust to divergence and the starting estimates are not required to be as close to the true values as other methods such as simplex optimisation.

Jaumot *et al.*[43] presented a modification of the NGL/M based multivariate non-linear least-squares kinetic fitting method in which a non-negativity constraint was applied to the matrix of spectral profiles before calculating the model residual matrix. This modification ensures that incorrect solutions are eliminated during the optimisation step as the residuals are a function of both the kinetic parameters and the non-negatively constrained spectral profiles.

Furusjö and Danielsson proposed a novel method that utilised kinetic modelling to provide initial concentration profiles that were subsequently refined using a target testing procedure[44]. The motivation for developing this method was to provide a means of testing a number of possible reaction mechanisms using only the measured spectroscopic data. The proposed rate equations were used to create a matrix of concentration profiles corresponding to the measurement times of each spectrum in the data set. This matrix (denoted $\boldsymbol{C}_{sim}$) was then tested by projection onto the subspace spanned by the principal components scores $\boldsymbol{T}$. The difference between the calculated and projected concentration matrix was calculated and the resulting sum-of-squares of this residual matrix was used as an objective function for simplex optimisation to find the optimal values of the rate

constants. Rotation of the loading vectors using the final transformation matrix obtained from an optimal fit would yield the pure spectral profiles for each chemical component.

The limitation of the kinetic modelling methods described here is that the spectroscopic data must resemble close to ideal conditions. Rate constants are temperature dependent so if significant temperature variation occurs during the course of the experiment, the underlying rate constant will also vary. The reactants and products must be observable by the spectroscopic measurement; incomplete dissolution of the reactants or precipitation of the products can alter the initial concentration or cause deviations from the expected concentration profiles predicted using the kinetic model. Additional contributions such as baseline drift or by-products formed during un-modelled side reactions can also introduce errors into the estimated spectral profiles

Hybrid models that combine the hard constraints of kinetic modelling with the soft modelling constraints of curve resolution methods such as MCR-ALS have been developed to overcome some of the issues described above. De Juan *et al.*[45] combined hard-modelling constraints provided by kinetic modelling with the well established soft-modelling method MCR-ALS. The principle of this method is to use MCR-ALS in a similar manner to its soft-modelling use by determining the appropriate number of components and identifying suitable initial estimates to start the ALS routine. The usual soft-modelling constraints such as non-negativity, uni-modality, equality and selectivity constraints can be applied where appropriate. During each ALS cycle, the matrix of estimated concentration profiles is modified before they are used to calculate a new estimate of the spectral matrix. To modify the concentration matrix, the columns corresponding to the chemical components that are to be subjected to kinetic modelling are extracted and used as input vectors that are refined using non-linear multivariate kinetic fitting. The refined concentration profiles then replace the original soft-model estimates and are used to calculate a new spectral matrix. The concentration profiles that are not subjected to kinetic constraints, for example those corresponding to baseline drift or a background interferent are not refined during this step and continue to be refined using the soft-modelling procedure. The selective application of kinetic constraints to selected components allows the combined hard- and soft-modelling approach to reduce rotational ambiguity whilst still modelling non-kinetic contributions to the measurement data.

As with the previous methods, the pre-requisite for the successful implementation of kinetic modelling constraints is that a valid model can be identified and is applicable to the

measurement data. Deviations from non-ideal kinetic behaviour introduce additional complexity or make the kinetic model too cumbersome so that only soft modelling constraints can be applied in practice.

1.6 Band Target Entropy Minimisation

Band target entropy minimisation (BTEM) is a relatively new method for recovery of pure spectral profiles from a set of mixture spectra and was proposed by Chew *et al.*[46, 47] in 2002. The objective of this method is to reconstruct a number of feasible pure spectral profiles from a set of mixture spectra using a combination of singular value decomposition (SVD), entropy minimisation and simulated annealing. Some elements of the BTEM approach were influential to the development of the vectorised adaptive Kalman filter with iterative spectral optimisation (VAKFISO) method reported in this thesis. The algorithm and some recent applications are therefore reviewed. The nomenclature used to describe the VAKFISO algorithm in chapter 2 is used to clarify the similarities and differences between the two algorithms.

The basic principle of the BTEM method is to obtain a set of orthonormal spectral basis vectors by factorisation of the $(J \times K)$ measurement matrix $\boldsymbol{A}$ using singular value decomposition. The matrix of right singular vectors is truncated to retain the first N primary vectors as shown in equation 1.18. In some applications of BTEM such as the recovery of minor components corresponding to catalytic species from FTIR data[48], the number of singular vectors included was often increased to more than 50 so that more information from secondary singular vectors can be included. In this case the transformation matrix is rectangular with dimensions $(N \times Z)$ where Z is the number of singular vectors used.

$$\boldsymbol{A}_{(J\times K)} = \boldsymbol{U}_{(J\times Z)}\boldsymbol{\Sigma}_{(Z\times Z)}\boldsymbol{V}^{T}_{(Z\times K)} \qquad \text{(equation 1.18)}$$

Various linear combinations of these basis vectors are then searched by changing the elements of a $(N \times Z)$ transformation matrix, denoted $\boldsymbol{T}$, to create different estimates of the pure spectral profiles using equation 1.19. In practice, this would make the optimisation very slow as it would be required to optimise a large number of elements in the transformation matrix. The spectra are therefore reconstructed one at a time using a $(1 \times Z)$ transformation vector as shown in equation 1.20.

$$\hat{\boldsymbol{S}}_{(N\times K)} = \boldsymbol{T}_{(N\times Z)}\boldsymbol{V}^{T}_{(Z\times K)} \qquad \text{(equation 1.19)}$$

$$\hat{\boldsymbol{s}}_{(1\times K)} = \boldsymbol{T}_{(1\times Z)}\boldsymbol{V}^{T}_{(Z\times K)} \quad Z \geq \mathrm{N} \qquad \text{(equation 1.20)}$$

Although conventional parameter optimisation methods such as the Nelder-Mead simplex could be applied to optimise the elements of the transformation matrix, for a large number of parameters, the simplex method can become trapped in local minima. To overcome this issue, the authors used simulated annealing (SA) owing to its ability to locate the global minimum and it is also relatively insensitive to the initial parameter estimates.

Since the orthonormal basis vectors stored in $\bar{\boldsymbol{V}}^{T}$ span the subspace of the original data, any linear combination of those basis vectors would produce spectra that also lie within the subspace of the original data. To guide the optimisation process, a penalty function that penalised negativity in both the estimated spectral profiles and the corresponding concentration profiles is employed. The objective of the simulated annealing optimisation is to minimise the penalty function obtained from each estimate of the transformation matrix $\boldsymbol{T}$. The non-negativity penalty function is calculated using equation 1.21 and equation 1.22 where γ is a scaling factor used to give the non-negativity penalty function the appropriate weighting.

$$P = \gamma\left[\sum_{n}\sum_{k} F(\hat{S}_{nk})\hat{S}^{2}_{nk} + \sum_{n}\sum_{j} F(\hat{C}_{nj})\hat{C}^{2}_{nj}\right] \qquad \text{(equation 1.21)}$$

$$F(y) = \begin{cases} 0 & (y \geq 0) \\ 1 & (y \leq 0) \end{cases} \qquad \text{(equation 1.22)}$$

One of the main features of the BTEM method is the use of the Shannon entropy function to measure the degree of simplicity in the estimated spectral profiles. If the linear combination of basis vectors produces realistic spectral features within the band selected, the entropy (calculated using the first derivative spectrum) will be low. Conversely, if the spectral profiles within the selected band are featureless, the entropy of the first derivative spectrum will be large. The Shannon information entropy function is calculated using equation 1.23 where h_{nk} is the normalised absolute value of the derivative spectrum (of degree m).

$$H = -\sum_{n}\sum_{k} h_{nk} \ln(h_{nk}) \qquad \text{(equation 1.23)}$$

$$h_{nk} = \frac{|\hat{S}_{nk}^{m}|}{\sum_{k} |\hat{S}_{nk}^{m}|} \qquad \text{(equation 1.24)}$$

To prevent the BTEM method from producing spectral estimates with very similar or even identical profiles, a spectral dissimilarity function (a measure of the orthogonality between two spectra) was also included. Of the four measures of dissimilarity trialled (angle, Euclidean inner product, determinant of covariance matrix and condition number), the angle constraint gave the best results. However, the authors state that the choice of angle for the angle constraint is arbitrary and will vary from case to case. The inclusion of the dissimilarity function added to the complexity of the overall objective function but was necessary when trying to estimate N spectral profiles simultaneously.

For a synthetic set of FTIR data constructed from seven components and three elementary reactions, comprising 234 spectra measured at 2501 spectral variables, the average time to resolve the spectra was 36.2 hours. However, the method did successfully recover the true spectral profiles from the overlapped mixture spectra using non-negativity, entropy and vector-angle constraints.

One of the limitations of the BTEM method is that a number of band targets containing characteristic spectral features must be selected by the user and the pure component spectral profiles are reconstructed one at a time, each corresponding to a global minimum of the objective function using different band targets. The simulated annealing procedure is repeated several times to identify each new component. A recent modification of BTEM called multi-reconstruction entropy minimisation (MREM) was reported by Zhang *et al.*[49] in 2007. This method retained the penalty functions of BTEM but uses a localised form of the simulated annealing method to identify a number of local minima corresponding to possible pure component spectra. The optimisation process is applied to the full spectrum so the user is not required to specify bands targets. As this method will identify a large set of admissible candidate pure component spectral profiles, the "best" spectral profile for each component is selected based upon the signal entropy. Typically, the spectrum with the lowest signal entropy is selected.

Since 2002, over sixty papers reporting the use of BTEM have been published. The applications of BTEM include the resolution of DRIFT, Raman, UV, NMR, FTIR, fluorescence, mass spectrometry, IR emission and XRD data. One of the applications most similar to the research reported in this thesis was the application of BTEM to a set of Raman spectra acquired during the hydrolysis of acetic anhydride[50]. In this work, BTEM was used to reconstruct spectral profiles from a measurement matrix created by combining data from four reactions using different reactant ratios. The first twelve right singular vectors were retained and three bands regions that contain significant spectral features were selected. Three pure component spectral profiles corresponding to acetic anhydride, acetic acid and white light were recovered using the BTEM method. The corresponding concentration profiles were then calculated using least-squares and the relative differences in initial concentration of acetic anhydride could be observed. Although the band positions of the peaks in recovered spectral profiles were compared with literature values, a direct comparison with the true pure component spectra was not shown.

1.7 The Kalman filter

The Kalman filter is a recursive regression method developed for the least-squares estimation of several parameters (called state parameters). More accurately, the Kalman filter is a way of "estimating the instantaneous "state" of a linear dynamic system perturbed by white noise; by using measurements linearly related to the state but corrupted by white noise"[51].

The first paper describing the system of equations now known as the Kalman filter was published in 1960 by Rudolph Emil Kalman in a paper titled "A New Approach to Linear Filtering and Prediction Problems"[52]. Over the next several decades, many variations of the Kalman filter have been developed and applied in literally hundreds of applications throughout the fields of electrical and mechanical engineering, navigation, astronautical guidance systems, hydrological monitoring and modelling and meterological forecasting to name just a few.

The Kalman filter algorithm is described in the Theory chapter (section 2.8).

1.7.1 The Kalman filter in chemistry

One of the earliest examples describing the application of the Kalman filter to a chemical measurement was published by Seelig and Blount in 1976[53]. In this paper, the authors introduced the theory of the discrete Kalman filter and identified the requirements that

must be met for the filter to be applicable to an analytical measurement. The main requirements identified were that a measurement transformation matrix that relates the state parameters to the measurement is available; a state transition matrix that models the time-dependency of the state parameters is known; the measurement noise is zero-mean and an estimate of the measurement variance can be provided; the variance of the initial state parameter values can be estimated; and an initial estimate of the state parameters can be provided. A mathematical model previously reported in the literature was used to populate the measurement transformation matrix and related the voltammetric measurements to the concentration of the electroactive species. The Kalman filter was applied to synthetic model data corrupted by white noise to estimate a single state (concentration) from the measured current. The authors investigated the affect of varying the ratio of the initial error covariance (denoted p_0) to measurement noise variance (denoted r_0) upon the Kalman gain. It was reported that as r_0 became much larger than p_0, the filter became insensitive to new measurements and the Kalman gain values become very small. Consequently, the Kalman filter returned a final state parameter estimate very similar to the initial estimate provided. When the initial error covariance (p_0) was large relative to the measurement variance (r_0), the innovations were overweighted, causing large fluctuations in the final state estimates. This work illustrated that for very noisy data (S/N = 1.20), significantly decreasing the value of the initial error covariance relative to the measurement variance ($\log(p_0/r_0) = -6$) would produce a very small standard deviation value for the final state estimates, but the state estimates would be in error and close to the initial estimate. As the ratio $\log(p_0/r_0)$ was increased to values in range -3 to 1, the final state estimates became more accurate although the standard deviation of the estimate was increased to reflect the increased uncertainty of the measurements. The effect of varying the initial state estimate over a large range (1.00×10^{-1} to 1.00×10^{-9}) was found to have a negligible affect upon the final estimate when the system and measurement error were well defined. It is the ability of the Kalman filter to produce good results from poor initial estimates and noisy measurements that make it so useful for modelling analytical measurements. Seelig and Blount published two further papers in 1979 describing the application of the Kalman filter to the real time quantification of lead in water samples using voltammetric measurements[54, 55]. The Kalman filter was compared with a non-real time method (multivariate linear regression) and pseudo-real time digital methods that involved various smoothing and automatic peak picking routines to identify

the maximum current used for the univariate calculation of analyte concentration. The correlation coefficient for the estimates obtained using the Kalman filter (real time estimation) and multivariate linear regression (non-real time estimation) were found to be greater than 0.99. The Kalman filter was able to converge to the final state estimates well before the maximum amplitude of the signal was measured, whereas the MLR approach required the complete signal to be measured before calculation of the analyte concentration. However, the precision of the two methods (MLR and KF) was comparable but found to be better than the pseudo-real time algorithms when the S/N < 5.

Throughout the 1980's and early 1990's, the Kalman filter has been applied to a variety of different analytical measurements. There have been several reviews of the development and application of the Kalman filter published in the analytical chemistry literature, for example; Brown (1986)[56], Rutan (1987 and 1990)[57, 58] and Tranter (1990)[59]. The applications covered in these reviews include noise removal (voltammetry, gas chromatography, mass spectrometry), peak resolution (UV spectroscopy, voltammetry, photoacoustic spectroscopy), detection and compensation of instrumental drift and model identification and improvement, determination of kinetic parameters and the removal of variable background responses.

Many of the reported applications describe the use of the discrete Kalman filter to resolve an instrument signal obtained for a multi-component mixture. The real-time capability of the Kalman filter was often mentioned, but was not usually the primary motive for using the Kalman filter. For example, Shi *et al.* described the use of the discrete Kalman filter for the simultaneous spectrophotometric determination of Co(II), Ni(II), Cu(II), Zn(II) and Cd(II) in synthetic and environmental samples[60]. In this application, standard solutions of each analyte were prepared and measured independently over the spectral range 500 to 620 nm to provide the reference measurement functions required by the Kalman filter. The concentrations of the analytes were then quantified for a number of synthetic and environmental samples using the Kalman filter. The Kalman filter successfully estimated the analyte concentrations of the analytes with a recovery of 94% to 107% for the synthetic samples and 91.6% to 108% for the environmental samples. The authors reported that the major advantage of using the Kalman filter was that it allowed the simultaneous quantification of all five analytes. The complexes of the five metals all had significantly overlapped spectra and traditionally sample pre-treatment (precipitation, solvent extraction

and ion exchange) was required to isolate each analyte so it could be quantified independently. The Kalman filter made it possible to quantify the mixtures simultaneously. Although the Kalman filter did converge to the final estimates before the full spectrum had been acquired, the final (most refined) estimate was used so the real-time advantage of the Kalman filter was not required or utilised in this application.

A similar use of the Kalman filter was reported by Volka *et al.*[61]. In this application, the authors used infra-red spectroscopy to quantify the concentration of heptane, hept-1-ene, cis-hept-2-ene and hept-3-ene in synthetic mixtures. The matrix of reference measurement functions was populated with the spectral profiles of each pure solvent acquired independently. The point of interest here was that the authors compared the performance (accuracy) of the conventional and adaptive Kalman filters when applied to the same data over the same spectral range. The adaptive Kalman filter performed slightly better than the conventional Kalman filter; the improved performance was attributed to the ability of the adaptive Kalman filter to adapt the measurement error variance value and therefore heteroscedastic noise was modelled more accurately than using a fixed value.

Obtaining accurate reference measurement functions is an essential requirement for the successful application of the Kalman filter. In 1993, Yongnian *et al.*[62] reported the use of the Kalman filter for the curve resolution of pyrazines measured using differential pulse polarography. Multi-component mixtures containing pyrazine and up to four of its methyl derivatives were measured using differential-pulse polarography. The polarograms of the various components were highly overlapped and previously PLS or PCR had been used to quantify the composition of unknown mixtures. In this work, the authors assessed the performance of the Kalman filter and compared two different methods of deriving the matrix reference measurement functions (denoted $\boldsymbol{H}$). In the first method, the conventional approach of measuring the polarogram of each individual component was employed. The second method employed a least-squares approach to calculate the reference measurement functions from a set of 13 multi-component standards of known composition. The composition of the standards employed a factorial design to ensure orthogonality and prevent collinearity in the matrix of measurements, $\boldsymbol{X}$. The reference measurement functions were calculated using equation 1.26.

$$\boldsymbol{Y} = \boldsymbol{H}^T \boldsymbol{X} \qquad \text{(equation 1.25)}$$

$$\hat{\boldsymbol{H}}^T = \boldsymbol{Y}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T)^{-1} \qquad \text{(equation 1.26)}$$

A blank solution was also measured to obtain a background polarogram that was used to augment the matrix $\hat{\boldsymbol{H}}^T$. Using the two methods of deriving $\boldsymbol{H}$, the authors estimated the concentrations of three components in 22 samples and calculated the mean recovery (%) and relative mean error (%) values. It was reported that using a matrix of reference measurement functions obtained by directly measuring each component independently, the mean recoveries of pyrazine, 2-trimethylpyrazine and 2,2-dimethylpyrazine were 81.8%, 93.1% and 110% respectively, with relative mean errors of 21%, 12% and 9.2% respectively. The results estimated using the Kalman filter improved when the matrix of reference measurement functions calculated using least-squares was used, returning mean recoveries for the same components of 101%, 99.6% and 103%; and relative mean errors of 10%, 9.9% and 3.8% respectively. Small, non-linear perturbations of the pure component polarograms caused by interactions between the various components was attributed to the improvement of the least-squares derived measurement model over the conventional method of measuring each component independently.

An interesting approach using Iterative Target Transformation Factor Analysis (ITTFA) to derive reference measurement functions prior to application of the Kalman filter was reported by Ni *et al.*[63]. The aim of this work was to quantify trace amounts of calcium in rare-earth matrices using ICP-OES. The presence of high background levels of rare-earth elements such Pr, Nd, Tb, Ho and Er caused spectral interferences that would cause large prediction errors if not included in the model. A standards additions approach was employed by taking five aliquots of the same sample solution and adding different amounts of a calcium standard. The emission spectra of each solution were acquired and collated into a matrix for further analysis. To initiate the ITTFA procedure, an initial vector that contained a contribution from the background but a minimal contribution from the analyte of interest was required. The authors used a standard spectrum of calcium to calculate an orthogonal projection matrix using equation 1.27, where $\boldsymbol{I}$ is a $(h \times h)$ identity matrix (h is the number of spectral variables), and $\boldsymbol{k}$ is a spectrum of calcium. The initial vector, $\boldsymbol{k}_b^{(0)}$ was calculated by projection of the sample spectrum (denoted $\boldsymbol{a}$) onto the orthogonal projection matrix $\boldsymbol{G}$.

$$\boldsymbol{G} = \boldsymbol{I} - \boldsymbol{k}\boldsymbol{k}^T \qquad \text{(equation 1.27)}$$

$$\boldsymbol{k}_b^{(0)} = \boldsymbol{Ga} \qquad \text{(equation 1.28)}$$

The ITTFA procedure was then performed to refine the background spectrum for each set of sample spectra. The standard spectrum of calcium and the ITTFA estimated spectrum of the interfering background were then used as the reference measurement functions to estimate the concentration of calcium using the Kalman filter. The results obtained from the ITTFA-KF approach were compared with the routine standard addition method (RSA). When the spectrum of the background interferent was not too severely overlapped with the spectrum of calcium, the results obtained from the two methods were very similar. However, when the spectra of the background interferent and calcium were severely overlapped, the ITTFA-KF method gave better accuracy and precision compared to the RSA method. This application of ITTFA to provide spectral profiles for Kalman filtering demonstrates the promise of combining factor analysis methods with an optimised estimation method such as Kalman filtering.

In the previous examples, the real time filtering ability of the Kalman filter was not exploited and the filter was simply used as a least-squares estimator. In 1993, Pérez-Arribas *et al.*[64] published an interesting study comparing the Kalman filter with classical least squares and pure component calibration methods to quantify quaternary mixtures of chlorophenols using UV spectrophotometry. In this work, an extensive set of standard solutions (40 standards for each component) were prepared at different concentrations in the range 3-15 $mg.L^{-1}$, the UV spectra were obtained and normalised to 1.0 $mg.L^{-1}$ and the mean spectrum was used as the pure component spectrum for that component. Each column of the reference $\boldsymbol{K}$ matrix was used to store the first-derivative mean spectrum for each of the four components. A set of 20 quaternary mixtures was also prepared, their spectra obtained and transformed to their first derivative. A reference matrix $\boldsymbol{K}$ was calculated from those mixture spectra using least-squares. The same set of mixture spectra was also used to estimate the reference matrix $\boldsymbol{K}$ using the Kalman filter. The Kalman filter was used to provide multi-component predictions of two, three and four component mixtures using the reference matrix $\boldsymbol{K}$ obtained from the three methods (pure component, classical least-squares and Kalman filter). With few exceptions, the reference matrix $\boldsymbol{K}$ obtained from pure component spectra returned the largest prediction errors. The reference matrix calculated using the Kalman filter returned prediction errors that were equivalent but often better than the same predictions made using reference matrix obtained using CLS. Unfortunately, the authors did not compare the performance of the three

methods for the prediction step but the results clearly indicated that using a reference matrix derived from pure components with the Kalman filter produced higher prediction errors than using a reference matrix derived using CLS or the Kalman filter. The significance of this work was that using least-squares with Kalman filtering seemed to offer the possibility of a more accurate and robust model than using pure component calibration alone. As many SMCR methods employ an alternating least squares step, the use of the Kalman filter in the derivation or application of curve resolution models could offer great potential.

1.7.2 The adaptive Kalman filter

The adaptive Kalman filter is a development of the discrete Kalman filter that allows the estimation process to adapt to variation in magnitude of the measurement noise or compensate for some model errors during the filtering process. A detailed description of the adaptive Kalman filter is described in section 2.9. The adaptive Kalman filter was first reported in a number of electrical and engineering applications in the late 1960's and early 1970's, and was developed to provide an optimal minimal variance state estimate in the presence of unknown system and measurement noise covariance matrices[65], traditionally denoted $\boldsymbol{Q}$ and $\boldsymbol{R}$.

Rutan and Brown[66] first proposed the idea of utilising the adaptive Kalman filter for analytical chemistry applications in 1984. The discrete Kalman filter will yield optimal results if the measurement model is accurate and the system and measurement noise are well characterised. Since many analytical chemistry applications use a time-invariant system model, the covariance matrix of the system noise ($\boldsymbol{Q}$) is often assumed to be zero. If this assumption is correct, the measurement model is the most probable source of error that can prevent the Kalman filter operating optimally. The measurement model, typically denoted $\boldsymbol{S}^T$ contains the reference measurement functions that are used to relate the measurement to system states. For multi-component spectroscopic measurements, the rows of $\boldsymbol{S}^T$ store the pure spectral profiles of each component contributing to the system. The estimate residuals at each measurement point are stored in a vector called the innovations sequence and can be used to assess the quality of the Kalman filter estimates. If the measurement model is correct and suitable estimates of the measurement noise variance and state parameter error covariance matrix are provided, the Kalman filter will return optimal state estimates and the innovations sequence will resemble zero-mean, white noise. If the measurement model is incomplete, *i.e.* a component contributing to the

measurement is not included in the model; or the measurement model is inaccurate, for example, one or more reference functions contain errors, the resulting innovations sequence will exhibit correlated structure related to the model error.

For the discrete Kalman filter, the measurement noise variance at variable k, denoted $R(k)$, is often set to a constant value R that is applied to all measurement variables. The adaptive Kalman filter uses a smoothing window to adaptively update the estimate of $R(k)$ using the previous m points of the innovations sequence. Over variable regions where the measurement model is correct, the innovations values and the adaptively estimated value of $R(k)$ will be small; the Kalman gain and state estimate update equations will therefore be sensitive to new measurements. When the measurement model is incorrect over several consecutive variables, the innovation and resulting $R(k)$ values will increase; this will reduce the sensitivity of the Kalman gain and state-estimate update calculations to new measurements and changes to the state parameter estimates will be small. Large values of $R(k)$ essentially reduce the sensitivity of the Kalman filter to new measurements until the measurement model is valid again. The fundamental requirement for the adaptive Kalman filter to provide accurate state parameter estimates in the presence of model errors is that the measurement model is not invalid over the entire range of measurement variables. Rutan and Brown also presented a series of simple equations that allow the innovations sequence and the estimated adaptive measurement noise variance values to be used to either modify an existing component in the measurement model, or augment the model with an estimate of an un-modelled component.

Since the introduction of the adaptive Kalman filter to the analytical chemistry community by Rutan and Brown in 1984, a large number of applications were reported throughout the 1980's and 1990's. Many of the applications reported the use of the adaptive Kalman filter to obtain estimates of state-parameters (such as the concentration of a chemical component) in the presence of an unknown background signal or other interferent. For example, Gerow and Rutan[67] used the adaptive Kalman filter to remove background fluorescent signals from silica thin layer chromatography plates. The aim of this work was to quantify various polynuclear aromatic hydrocarbons (PAHs) that were chromatographically unresolved using thin layer chromatography. Emission spectra of the separated components, individual standards and background were acquired from the same plates and the Savitzky-Golay first derivative spectra were calculated. The derivative spectrum of the background was set as the measurement model, the spectrum of the

unknown or standard were set as the response and the adaptive Kalman filter was used to estimate the "concentration" of the background. The calculated value was then used to subtract the correct amount of background from the unknown sample spectrum and also the standard spectra that were eluted in tracks parallel to the unknown. The background corrected spectra of the appropriate standards were set as the measurement model, whilst the background corrected sample spectrum were set as measured response and Kalman filtering was performed a second time to quantify the amount of PAHs in the unknown sample. This approach successfully removed the correct amount of background contribution and the co-eluting PAH components were accurately quantified using the adaptive Kalman filter. When the conventional Kalman filter was used, the prediction errors were much larger because small model errors would cause the state-estimates to diverge. The Kalman filter based background removal method reduced the limit of detection for the various PAH components by a factor of 8.4 compared to simple baseline subtraction.

An interesting extension of the baseline subtraction work described in the previous paragraph was presented by the same authors in which a series of background spectra were acquired and modelled using PCA[68]. The primary loadings vectors were set as reference measurement functions and the weightings required to subtract the appropriate amount of background signal from a sample or standard spectrum was calculated using the adaptive Kalman filter. In this case, the background signal was broader than the analytical signal of interest; consequently there were regions in the measured response where the background signal was the only contribution and this allowed the Kalman filter to accurately estimate the background using an incomplete model.

The accuracy of the state parameter estimates returned by Kalman filtering can be assessed by examination of the diagonal elements of the state estimate error covariance matrix $\boldsymbol{P}$. When the filter has produced optimal results for a specified measurement model, the diagonal elements of $\boldsymbol{P}$ will be minimised. When attempting to estimate state parameters using an incomplete measurement model, the most accurate estimates will be returned by the adaptive filter when the initial state estimates are close to the true values. This would typically require the Kalman filter to be re-run several times using different initial estimates of the state parameters. To automate this process, Rutan and Brown introduced a simplex optimised adaptive Kalman filter[69]. The quality of the state estimates was assessed during the simplex optimisation process by calculating the scalar value $Y = \sum \log \boldsymbol{P}_{ii}$. As the

initial state estimates approached the true values, the adaptive Kalman filter would yield an optimal fit and the diagonal elements, and therefore the trace of $\boldsymbol{P}$ would be minimised. This was an interesting concept as the adaptive Kalman filter was used to assess the quality of initial state estimates using the resulting covariance matrix, but simplex optimisation was used to locate the best values. During the research presented in this thesis, a similar use of the covariance matrix was employed in the VAKFISO method to optimise the measurement model.

1.7.3 Parallel and multi-model Kalman filters

In many of the Kalman filter applications reported in the analytical chemistry literature, the instrument response is a vector comprising K measurement variables such as a spectrum, chromatogram or voltammogram. The recursive Kalman filter calculations are therefore performed using scalar measurement values $z(k)$ to estimate the vector of state parameters for the n components in the measurement model. Parallel Kalman filters have also been applied to a number of applications in various fields of engineering and signal processing. A parallel (or multi-model) Kalman filter allows a measurement datum to be passed through a number of alternative models and the most appropriate model is selected based upon various statistics derived from the innovations and state-estimate error covariance matrix. For example, Sitting and Cheung reported a parallel Kalman filtering algorithm that could detect the occurrence of arrhythmia from the R-R intervals of an electrocardiogram (ECG) signal in real-time[70]. The ECG waveform could exhibit one of four different temporal patterns corresponding to normal fluctuations or various degrees of arrhythmia. Each new data point was input into each of the four models which attempted to fit the data to the various patterns that define each model. The probability of each model was assessed at each time point and could be used to indicate the current behaviour of the patients heartbeat in real-time. In this example, the efficiency of the parallel implementation was 1.77 times faster than the sequential implementation of the same models. The authors did concede that for this application, the additional cost of four separate processors may not be justified by the small gain in execution time, but were able to demonstrate that using a parallel structure to assess multiple models simultaneously can be helpful for variable data streams that cannot be described using a single model.

Parallel Kalman filters have also been employed in analytical chemistry. Wentzell and Vanslyke reported the use of a network of parallel Kalman filter models to determine the accurate value of a pseudo-first order rate constant from kinetic UV absorbance data[71]. A

series of 81 models using fixed values of k covering the range $k_{nom} \pm 40\%$ were used to estimate two state parameters. The parallel Kalman filters were all updated with the new absorbance reading at each time point and the model that returned an innovations sequence with the lowest running sum-of-squares was used to determine the best estimates of ΔA, B and k, corresponding to the change in absorbance contributed by the product at $t = \infty$, the background absorbance and the rate constant respectively. In this example, the Kalman filters were operating independently on the same set of measurement data.

Vanslyke and Wentzell reported the use of a parallel Kalman filter network applied to HPLC-DAD data for real-time peak purity analysis[72, 73]. A subset of n spectral variables measured at each chromatographic retention time were passed through a network of parallel Kalman filters. At each retention time, a set of $(n-1)$ one-dimensional models, and a set of $(n-2)$ two-dimensional models were used to represent the n wavelengths using a linear and planar model respectively. The root-mean-square innovations from each set of models were plotted as a function of retention time and the resulting profiles were used to assess the rank of the data at each time point. If only one component was contributing to the chromatographic data points, both the one-dimensional and two-dimensional models would produce a relatively flat profile with a magnitude equivalent to the measurement noise. If two components were contributing to the chromatographic data points, the RMS innovations values for the two-dimensional model would still resemble a flat line but the innovations of the one dimensional model would begin to increase as the concentration of the un-modelled component increased. By inspection of these plots it was possible to assess the local rank at each time point. The weightings applied to each of the n variables from the one- and two dimensional models were used to form vectors that were comparable to the eigenvectors obtained using principal components analysis. This approach could therefore be considered as a viable alternative to performing principal components analysis in real time. The disadvantage of the parallel Kalman filter approach was the number and complexity of the models required to model more than two dimensions. Although conceptually, the Kalman filters were operating in parallel, in practice each of the $(2n-3)$ models were calculated sequentially at each time point.

Rutan and Brown reported the application and comparison of the Kalman filter to two- and three-dimensional spectroscopic-kinetic data obtained from a first-order reaction[74]. The two-dimensional data comprised of absorbance values measured at a single wavelength

acquired at number of time points; the three-dimensional data comprised of multi-wavelength spectra acquired at a number of time points. For the three-dimensional data, the reference measurement model comprised of the independently measured spectral profiles of the reactant and product. The system model contained the kinetic expression that relates the expected concentration (and mixture signal) to elapsed time. The Kalman filter performed the recursive calculations on each of the j wavelengths of the spectrum measured at time t_k to obtain estimates of the state parameters k (the rate constant) and $[A]_0$. When the last variable in the spectrum was reached, the final estimates of the state parameters were propagated to the next spectrum at time t_{k+1} and used as initial estimates. The recursive filtering calculations were then repeated for next spectrum starting at the first variable. The results obtained by this approach produced estimates of the rate constant and initial substrate concentration that were superior to simplex optimised linear and non-linear least-squares estimation. Although the full set of measurement data was stored in a two dimensional array, the Kalman filter only operated on one scalar value at a time.

Quencer and Crouch reported a similar application of the extended Kalman filter to simulated multi-component kinetic data[75]. They investigated a simulated system in which two reagents would react with a common reagent to form two similar products assuming pseudo-first order kinetics. The j wavelength measurements recorded at time t were used to update the estimates of the initial reagent concentrations and two rate constants (five state parameters in total). The final estimates were then propagated to be used as initial estimates for the Kalman filtering of the next spectrum. The authors noted that it was necessary to follow the reaction to approximately 50-60% of full completion for the filter to yield accurate estimates. An interesting aspect of the work reported in this paper was the investigation of the affect of varying the degree of spectral overlap between the two products. However, the number of wavelengths used was very small (up to six) and the degree of spectral overlap was only classified as "none", "medium" or "high". Providing the spectral profiles were sufficiently different, the filter could accurately estimate the state parameters even when the ratio of rate constants was unity. However, as the difference between the spectral profiles was reduced, it required more wavelengths to return accurate estimates.

1.8 Model reactions

To synthese a new chemical entity, a variety of different chemical transformations may be required to synthesise the starting materials, intermediates and final product. In synthetic organic chemistry, there are many different transformations that are typically used, such as (de-)alkylation, (de-)acylation, (de-)esterification, (de-)halogenation, (de-)hydrolysis, (de-)sulphonation, oxidation, reduction and polymerisation.

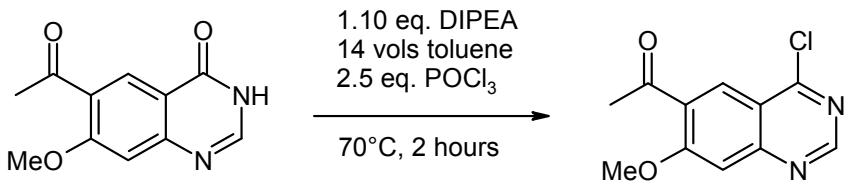
During the development of a process, a research chemist or engineer would need to determine the composition of the reaction mixture at various times throughout the course of a reaction. Sophisticated analytical instrumentation such a nuclear magnetic resonance spectroscopy, high-performance liquid chromatography, gas chromatography and mass spectrometry detection are typically used to analyse samples taken from the reaction mixtures. These analytical techniques provide data that are extremely rich in chemical information and allow the scientist to study the reaction by tracking a number of specific chemical species of interest. During sample preparation, the reaction sample is usually filtered and diluted prior to analysis. The individual components in the mixture can then separated and quantified using the analytical instrumentation. The various components are either physically separated prior to detection (for example, chromatographic separation followed by UV detection) or the instrument has a measurement response resolution that is high enough to identify individual species in an un-separated mixture (for example high resolution NMR or mass spectrometry). The combination of the sample preparation step and instrument resolution means that the physical properties of the reaction mixture, co-varying formation or consumption of products and reagents and molecular structure similarities do not usually complicate the analytical measurement.

The benefits of using *in-situ* spectroscopic measurements during chemical process development were discussed previously in section 1.1.3. However, since the process is measured directly, the physical properties of the process, such as temperature and heterogeneity will influence the spectral measurement. Also, no physical separation of the individual chemical components occurs prior to the spectral measurement so only differences in the measurement response functions can be used to resolve the individual components. For subtle changes in molecular structure, the differences between the spectral profiles of each component in the mixture are often very small. The two reactions studied in this thesis were chosen because they possess the difficulties commonly encountered during industrial process monitoring.

1.8.1 Chlorination of "acetoxyone" using phosphorus oxychloride

Heterocyclic ring structures are useful building blocks in the synthesis of pharmaceutical drugs and are present in many natural products. During the synthesis of organic molecules, alkylation, amination and arylation are common chemical transformations used to couple two or more moieties. Amino-de-halogenation[76] is the coupling of ammonia or a primary or secondary amine (R_3N) to an activated aryl halide (ArX). Halogenation of a heterocyclic ring is therefore an important synthetic step that can be used to activate a heterocyclic ring prior to nucleophilic substitution.

The reaction monitored in this work was the chlorination of 7-methoxy-4-oxo-3,4-dihydroquinazolin-6-yl acetate ("acetoxyone") using phosphorus oxychloride to form 4-chloro-7-methoxyquinazolin-6-yl acetate ("haloacetoxyone"), shown below in Figure 1.2. This reaction and several similar variants have been utilised during the synthesis of a number of AstraZeneca compounds. The chlorination step was originally performed using thionyl chloride ($SOCl_2$) as the chlorinating reagent and solvent, with a small amount of dimethylformamide (Me_2NCHO) present as a catalyst. The thionyl chloride would then be removed by distillation and replaced by toluene. It was necessary to perform a second distillation of the toluene solution to remove the residual thionyl chloride. However, this process was unsuitable for scale-up to a large scale laboratory or pilot plant because the large excess of thionyl chloride would lead to corrosion of the plant equipment.
An alternative process was developed to replace the large excess of thionyl chloride with a much lower excess of phosphorus oxychloride (3.3 to 3.7 equivalents) in toluene. Di-isopropylethylamine (DIPEA) was also required to consume the hydrochloride (HCl) produced during the reaction. In this process, it was not necessary to remove the excess phosphorus oxychloride by distillation as it could be quenched by addition of propan-2-ol. This permitted the active haloacetoxyone product (a chloroimine species) to be kept in solution and telescoped directly into the next step, which was a coupling reaction with an aniline derivative (amino-de-chlorination).



7-methoxy-4-oxo-3,4-dihydroquinazolin-6-yl acetate

"acetoxyone"

4-chloro-7-methoxy quinazolin-6-yl acetate

"haloacetoxyone"

Figure 1.2: Reaction scheme for the chlorination of acetoxyone using phosphorus oxychloride.

1.8.2 *N*-benzylation of 1*H*-indole using benzyl bromide

The alkylation of the basic or substituted indole moiety is a useful chemical transformation in the pharmaceutical industry. In 1973, Heaney and Ley published a simple method for the *N*-benzylation of indole using benzyl bromide[77]. The use of potassium hydroxide combined with a dipolar aprotic solvent (dimethyl sulphoxide) was reported to be important for improving the selectivity of *N*-benzylation over C3-benzylation. The procedure involves pre-formation of the potassium salt of indole, which is then alkylated by addition of 2 equivalents of benzyl bromide. The use of 2 equivalents of benzyl bromide was required to ensure complete conversion of the indole (95% yield) as it was known to react with DMSO. A similar method was reported by Kikugawa and Miyake[78] in which acetone was used in place of dimethyl sulphoxide. Alkylation was achieved by addition of benzyl chloride (2 eq.) but with reported yields of 85%, the method offers no advantage over the use of DMSO.

The use of tetraalkylammonium salt catalysis was first reported for this reaction in 1976[79]. The approach was quite different to those described previously as the indole was added to a two phase system of tetra-*n*-butylammonium hydrogen sulphate (5 mol% with respect to indole) in 50% aqueous sodium hydroxide and benzyl bromide (1.5 eq.) dissolved in benzene. A yield of 93% after 18 hours at 33°C was reported. In the same year, Bocchi *et al.* reported a similar use of a phase transfer catalyst[80]. These authors commented that the organic phase was not necessary and actually slowed down the reaction. Utilising a stoichiometry amount of methyltriotylammonium chloride, they achieved a yield of 93% 1-benzylindole with approximately 5% dibenzylindole detected. The reaction time was not reported.

The indole benzylation reaction was recently employed in an AstraZeneca project. That particular application required the use of non-aqueous reaction conditions to prevent hydrolysis of an ester group substituted at the 2 position of indole. A "soft" inorganic base such as potassium carbonate or cesium carbonate was used in combination with an aprotic solvent such as acetonitrile. The *N*-benzylation of an un-substituted indole using an aprotic solvent and a "soft" inorganic base was selected as a model reaction.

Initial trial experiments for this reaction confirmed that the use of a phase transfer catalyst, tetrabutylammonium bromide (TBAB) improved the reaction selectivity, rate of reaction and extent of conversion with respect to experiments in which a phase transfer catalyst was not used. The reaction scheme and nominal reaction conditions are shown below in Figure 1.3.

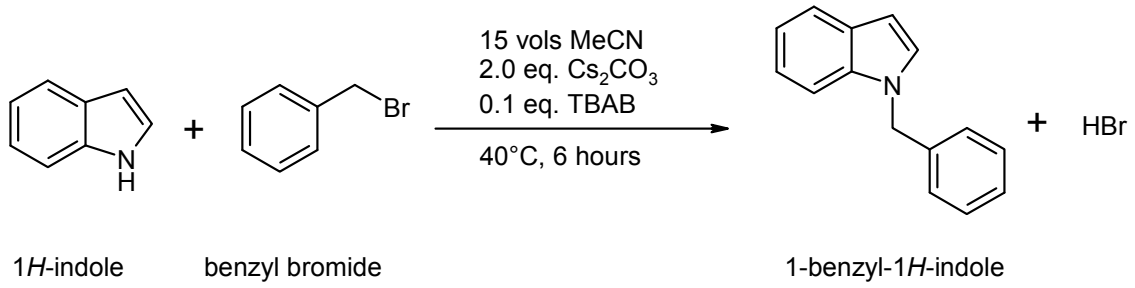


Figure 1.3: Reaction scheme for the *N*-benzylation of 1*H*-indole using benzyl bromide.

1.9 Summary of project aims

The goal of this research was to investigate the use and development of chemometric methods that could be applied to facilitate the recovery and prediction of component concentrations using real process spectra. Of specific interest were methods that could be applied to spectral data sets acquired during processes that resemble typical reactions employed in the fine chemicals and pharmaceutical industries. Industrial reaction processes are often difficult to sample, or the reaction mechanism proceeds via a reactive intermediate that cannot be isolated. An important element of this research was therefore the implementation and development of self modelling curve resolution methods that would allow quantitative or semi-quantitative models to be developed in the absence of external reference data. Process spectra often contain additional contributions that hinder the recovery of the desired chemical information. Improving the robustness of the spectral measurement, and the development of methods that could be used to remove unwanted spectral variation, such as significant baseline contributions were also investigated.

2 Theory

2.1 Ultraviolet / Visible (UV/Vis) Spectroscopy

The ultraviolet and visible regions of the electromagnetic spectrum extend from 10 to 780 nm. However these are arbitrarily split into a number of regions that reflect the various instrumental or experimental distinctions[19]. The region 10 to 200 nm is called the vacuum-ultraviolet region. This is because atmospheric oxygen will absorb strongly below 200 nm so ultraviolet measurements in this region must be performed under vacuum. The visible region of the electromagnetic spectrum extends from 380 to approximately 780 nm and is so called because it is the region that is visible to the human eye. The near-ultraviolet region extends from 200 nm to 380 nm. The near UV and visible regions are the most useful for analytical chemistry applications because they correspond to electronic transitions that are sensitive to changes in molecular structure.

Ultraviolet and visible spectroscopy is often called electronic spectroscopy and refers to the nature of the transitions involved. The absorption of ultraviolet or visible radiation results in the transition of electrons between different electronic states of an atom or molecule[81]. Electronic transitions can include σ, π and *n* electrons, *d* and *f* electrons and charge transfer electrons. Sigma (σ) electrons reside in the bonding molecular orbitals associated with single bonds and transitions to the corresponding anti-bonding molecular orbital are designated $\sigma \rightarrow \sigma^*$. This type of transition requires the largest amount of energy relative to other types of electronic transition, and requires photons with higher energy (shorter wavelength) that are assigned to the vacuum ultraviolet region (typically 100 to 200 nm). Consequently, these transitions cannot be observed in the spectral range that is accessible to ordinary laboratory spectrometers. Transitions involving the promotion of unshared (non-bonding) electron pairs to a sigma anti-bonding orbital are designated $n \rightarrow \sigma^*$. These transitions require less energy than $\sigma \rightarrow \sigma^*$ transitions but as most absorptions occur below 200 nm, they are not observed using ordinary spectrometers. The energy required to effect $n \rightarrow \pi^*$ and $\pi \rightarrow \pi^*$ transitions is lower and results in absorptions occurring in the region 200 to 700 nm. For this reason, they are most convenient for applications using standard UV/vis spectrometers.

The relative intensity of an absorption band at a specific wavelength is defined by its molar absorptivity, ε. The magnitude of this value is proportional to the product of the capture cross section of the species and the probability of an electronic transition occurring on

absorption of a photon at that wavelength. This value has the units $dm^3.mol^{-1}.cm^{-1}$ or $L.mol^{-1}.cm^{-1}$, depending upon which units of molar concentration are used. The value of ε for wavelengths corresponding to $n \rightarrow \pi^*$ and $\pi \rightarrow \pi^*$ transitions are typically 10 to 10^3 and 10^3 to 10^5 $L.mol^{-1}.cm^{-1}$ respectively[82]. The spectra of organic molecules over the wavelength range 200 to 780 nm are therefore dominated by $\pi \rightarrow \pi^*$ transitions, and to a lesser extent $n \rightarrow \pi^*$ transitions.

A molecule must contain absorbing groups called chromophores for its UV/vis spectrum to possess useful features. Most organic molecules will possess conjugated double or triple bonds in either open chains or aromatic ring systems. These result in a large number of delocalised π electrons and such functional groups are strong chromophores with characteristic absorbance bands. An auxochrome is an atom or functional group, such as halogens, hydroxyl groups or amino groups that are not themselves considered chromophores but can affect the absorption of the chromophore to which they are attached. Auxochromes possess *n* electrons that can interact with the π electrons of the chromophore to produce one or more of the following spectral changes. A shift of the absorption maximum to longer wavelengths is called a bathochromic shift; a shift to shorter wavelengths is called a hypsochromic shift. An increase of the molar absorptivity at a specific wavelength is known as hyperchromism and hypochromism is a decrease in the molar absorptivity. These various spectral changes can be quite subtle but often the spectra of different components in a mixture (such as a reaction system) can still be distinguished.

2.2 Raman spectroscopy

Raman spectroscopy is a photon scattering phenomenon that provides information about the quantised vibrational energy levels in a molecule. Raman spectroscopy therefore provides complementary information to mid-infrared spectroscopy over the spectral region 100 to 4000 cm^{-1}. The phenomenon of inelastic photon scattering was postulated by Smekal in 1923 and was first observed experimentally in 1928 by Raman and Krishnan[83].

When a sample is irradiated with monochromatic radiation, the photons may be simply transmitted without any interaction. If the energy of the photon corresponds to the energy difference between two vibrational levels (*i.e.* the energy of the photon is in the range 100 to 4000 cm^{-1}) the photon may be absorbed by the molecule, resulting in a vibrational transition. This is the process studied by mid-infrared absorption spectroscopy. If the energy of the incident photon is far greater than the vibrational energy level spacing, the

photon can still be absorbed but the molecule is raised from the ground electronic state to a virtual state. The virtual state is not a stationary electronic state but a short lived energy level somewhere between the ground and first electronic excited state, caused by the temporary shift of the electron distribution of a covalent bond (molecular orbitals). If the molecule relaxes back to the same initial vibrational level in the ground electron state, a photon is emitted with the same energy as the incident photon. This process is called Rayleigh scattering[82]. The molecule does not experience any net gain or loss of energy during Rayleigh scattering and the process is therefore termed elastic scattering. This process is very efficient and the intensity of a Rayleigh line is several orders of magnitude stronger than the Raman scattering process. If the molecule relaxes from the virtual state to a higher vibrational level in the ground electronic state, the photon emitted will have less energy (longer wavelength) than the incident photon[84]. This process is called Stokes Raman scattering. There is a net gain in vibrational energy to the molecule and the energy difference between the incident and scattered photons is called the Stokes Raman shift. It is also possible for the molecule to relax from the virtual state to a lower vibrational level in the ground electronic state. In this case, the photon emitted will have more energy (shorter wavelength) than the incident photon. This process is called anti-Stokes Raman scattering and there is a net loss of vibrational energy from the molecule.

For typical laboratory conditions, the intensity of Stokes Raman transitions are much more intense than anti-Stokes transitions. This is because anti-Stokes Raman scattering requires that the molecule is initially in a higher vibrational level before the incident photon is absorbed. The Boltzmann distribution can be used to calculate the fraction of molecules in one vibrational level relative to a higher vibrational level for a specific temperature. At room temperature, the fraction of molecules residing in a higher vibrational energy level is much lower than those in a ground vibrational energy level. Throughout this work, all Raman shifts quoted refer to Stokes Raman shifts.

An important property of Raman spectroscopy is that the scattering intensity is proportional to the number of molecules illuminated by the incident radiation[84]. This allows Raman spectroscopy to be used for quantitative analysis. The expression that relates the measured Raman intensity for a specific molecular vibration to the number of molecules (concentration) is shown in equation 2.1. This form is similar to the Beer-Lambert law used in quantitative absorption spectroscopy, shown in equation 2.2.

$$I_R = (I_L \sigma K) cl \qquad \text{(equation 2.1)}$$

$$A_\lambda = (\varepsilon_\lambda) cl \qquad \text{(equation 2.2)}$$

I_R Measured Raman intensity, in photons per second

I_L Laser intensity, in photons per second

σ Absolute Raman cross-section, cm^2 per molecule

K Measurement parameters. Instrument optical parameters such as collection efficiency and transmission are combined in this single constant.

l Sample path-length, in cm

c Concentration, in molecules per cm^3 (equation 2.1) or $mol.L^{-1}$ (equation 2.2)

A_λ Absorbance at wavelength λ

ε_λ Molar absorptivity at wavelength λ, in $L.mol^{-1}.cm^{-1}$

A requirement of the mechanism by which Raman transitions occur is that the vibrational mode is polarisable. This is because Raman scattering involves a transient 'virtual' state created by distortion of the electron distribution of a bond on absorption of a photon. The polarisability of a molecular bond is a measure of the degree to which the bonds electron distribution can be deformed. Bonds that already have a large dipole moment that changes during the vibration have a low polarisability and consequently they are either weak Raman scatterers or are Raman inactive. Bonds that have a symmetrical electron distribution have a large polarisability and are Raman active. Examples of bonds that are Raman active include homonuclear bonds such as C-C, C=C, C≡C, N=N, O=O, heteronuclear bonds with weak dipoles such as C=N, C=S and ring-breathing modes of aromatic or saturated ring systems such as benzene.

2.3 Evolving Factor Analysis (EFA)

Evolving Factor Analysis (EFA) was originally developed by Gampp and Maeder *et al.*[85-88] for the model free resolution of spectrophotometric titration data and was later applied to multi-wavelength chromatographic data.

An $(I \times J)$ matrix of reaction spectra, where I is the number of observations (spectra) and J is the number of variables (wavelengths) can be expressed as the matrix product:

$$\boldsymbol{X} = \boldsymbol{C}\boldsymbol{S}^T + \boldsymbol{E} \qquad \text{(equation 2.3)}$$

Where $\boldsymbol{C}_{(I,K)}$ is the matrix of concentration profiles for K components, $\boldsymbol{S}_{(J,K)}$ is the matrix of spectral profiles for the same k components and $\boldsymbol{E}_{(I,J)}$ is the matrix of residuals (remaining part of $\boldsymbol{X}$ not modelled by $\boldsymbol{C}$ and $\boldsymbol{S}$ that usually accounts for measurement noise). In the absence of measurement noise the expression can be simplified to:

$$\boldsymbol{X} = \boldsymbol{C}\boldsymbol{S}^T \qquad \text{(equation 2.4)}$$

Singular Value Decomposition of $\boldsymbol{X}$ would yield K non-zero singular values equal to the rank of data. The corresponding scores and loadings calculated from the eigenvalues and eigenvectors would loosely resemble concentration and spectral profiles. However, these are abstract solutions and would require additional transformations to obtain the true profiles. Performing principal components analysis (PCA) on the complete data matrix $\boldsymbol{X}$ would reveal the maximum rank of the data but does not provide any information about local rank, *i.e.* how many species are contributing to the signal at each time point.

One advantage of Evolving Factor Analysis is that it utilises the inherent evolutionary structure of the data to determine the local rank at each time point. The basis of EFA is evolving principal components analysis. Starting from the first spectrum, a sub-matrix is created and principal components analysis is performed to calculate the principal components and their corresponding eigenvalues. The analysis is then continued by incrementing the size of the sub-matrix by one spectrum and calculating a new set of eigenvalues. This process is then repeated until the last spectrum in the data matrix is reached. The reverse analysis is performed by starting with the last spectrum and incrementing the window in the opposite direction. The log of the eigenvalues obtained from both forward and backward analysis are then plotted as a function of spectrum number (time). The forward analysis shows the appearance of new components whilst the

backward analysis shows the disappearance of components. The concentration window of each component k can be calculated by combining the k^{th} eigenvalues obtained from the forward EFA with the $(K-k+1)$ set of eigenvalues obtained from the backward EFA $K = \min([I,J])$. The resulting concentration windows are then re-scaled and collected into the matrix $\boldsymbol{C}$. The corresponding spectral profiles can be calculated using least squares using the expression:

$$\boldsymbol{S} = \boldsymbol{X}^T\boldsymbol{C}(\boldsymbol{C}^T\boldsymbol{C})^{-1} \qquad \text{(equation 2.5)}$$

The matrix $\boldsymbol{S}$ calculated above can then be used to calculate a new estimate of $\boldsymbol{C}$ using:

$$\boldsymbol{C} = \boldsymbol{X}\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1} \qquad \text{(equation 2.6)}$$

Alternating Least Squares (ALS) is performed by repeating the calculations shown in equation 2.5 and equation 2.6 to further improve the concentration and spectral profiles utilising basic constraints such as non-negativity. This iterative approach produces physically meaningful solutions that are optimised in a least-squares sense.

2.4 Orthogonal Projection Approach (OPA)

The Orthogonal Projection Approach (OPA) was developed by Sanchez *et al.*[89] for peak purity assessment of HPLC-DAD data. The aim of OPA is to find the purest set of spectra (rows) or variables (columns) in a data matrix and works on the basic assumption that the purest spectra are mutually more dissimilar than the corresponding mixture spectra[25]. The measure of dissimilarity d_i is defined as the determinant of the dispersion (covariance) matrix of $\boldsymbol{Y}_i$, given by:

$$d_i = \det(\boldsymbol{Y}_i^T\boldsymbol{Y}_i) \qquad \text{(equation 2.7)}$$

where $\boldsymbol{Y}_i$ comprises one or more of the reference spectra and the spectrum $\boldsymbol{x}_i$, the i^{th} spectrum from $\boldsymbol{X}$. The determinant of the dispersion matrix $\boldsymbol{Y}_i$ is a measure of the area of the parallelogram defined by the reference spectrum (or spectra) and $\boldsymbol{x}_i$. If the spectrum $\boldsymbol{x}_i$ is very similar to the reference spectra, the angle between the two vectors in the $J-$dimensional space will be small, so the resulting parallelogram will have small area and thus a low dissimilarity value. A spectrum that is quite different to the previously found reference spectra will give a high dissimilarity value.

The routine is usually initiated by selecting the mean spectrum, $\bar{\boldsymbol{x}}$ as the first reference spectrum. The mean-spectrum is normalised to unit length and the dissimilarity of un-normalised spectrum ($\boldsymbol{x}_i$, $i = 1$ to I) is calculated iteratively using equation 2.7. The spectrum with the highest dissimilarity value will replace the mean spectrum as the first reference spectrum and the process will be repeated. The spectrum with the highest dissimilarity value with respect to the first reference spectrum is normalised and added to $\boldsymbol{Y}$. The procedure is continued until the desired number of pure spectra have been identified; the magnitude of the dissimilarity values drop below a threshold value or the dissimilarity profiles resemble noise. Plotting the dissimilarity values for each pure spectrum as a function of spectrum number (time) will often provide a good approximation of its corresponding concentration profile. OPA can be applied in either the row (spectral) direction or the column (variable) direction.

2.5 Multivariate Curve Resolution – Alternating Least Squares (MCR-ALS)

The purpose of Multivariate Curve Resolution-Alternating Least Squares (MCR-ALS)[22, 35, 45, 90-94] is to resolve a multi-component system assumed to have an additive bilinear model as shown in equations equation 2.3 and equation 2.4. MCR-ALS is an iterative method that gives equal priority to the concentration and spectral profile matrices and optimises both $\boldsymbol{C}$ and $\boldsymbol{S}^T$ during each iteration cycle. All components are modelled simultaneously by alternatively calculating $\boldsymbol{C}$ or $\boldsymbol{S}^T$ using equation 2.6 and equation 2.5 respectively.

Methods such as EFA or OPA can be applied to a two-way data matrix to produce initial estimates of the underlying concentration or spectral profiles; Alternating Least-Squares can then be used to refine the estimates using simple constraints such as non-negativity. The true power and flexibility of MCR-ALS lies in its ability to use and combine initial estimates obtained from various methods and apply a number of physically meaningful constraints such as non-negativity, unimodality, and closure. Additional external information such as known concentration or spectral profiles for one or more species can be incorporated and zero-concentration regions can be specified when it is known that a particular species will not be present over a specific time period. These additional options require user input and thus a good understanding of the data and its underlying structure is necessary to prevent the use of invalid assumptions. The advantage of including additional constraints and external information is that it further reduces the range of feasible solutions

obtained ($\boldsymbol{C}$ and $\boldsymbol{S}^T$) and is therefore more likely to produce a solution that resembles the true underlying structure.

2.6 Partial Least Squares Regression (PLS)

Partial Least Squares (also referred to as Projection to Latent Structures)[12, 20-25, 31, 32] is a multivariate regression method commonly used throughout many disciplines such as chemometrics, physical and biological sciences, social sciences, economics *etc.*.

PLS is often used in spectroscopic calibration applications as it can help to overcome the problem of high collinearity between spectral variables that can lead to unstable regression coefficients if traditional multi-linear regression (MLR) were applied. The problem of collinearity is reduced by representing both the $\boldsymbol{X}$ and $\boldsymbol{Y}$ blocks (spectra and concentrations respectively) as linear combinations of the original variables.

$$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{P}^T + \boldsymbol{E} \qquad \text{(equation 2.8)}$$

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{Q}^T + \boldsymbol{F} \qquad \text{(equation 2.9)}$$

The objective of PLS is to find a set of latent variables that maximise the covariance between $\boldsymbol{X}$ and $\boldsymbol{Y}$. The factors are calculated to maximise the correlation between the $\boldsymbol{X}$ and $\boldsymbol{Y}$ blocks whilst also accounting for the maximal amount of structured variance in each block. The number of factors to use in the PLS model is usually chosen by selecting the value that minimises the root mean square error of cross validation (RMSECV) or root mean square error of prediction (RMSEP).

During the calculation of each set of latent variables, the loading vectors $\boldsymbol{p}^T$ and $\boldsymbol{q}^T$ are iteratively improved. Projection of $\boldsymbol{X}$ and $\boldsymbol{Y}$ onto the loading vectors results in a pair of scores vectors $\boldsymbol{t}$ and $\boldsymbol{u}$. The final scores vectors will be found when the difference between two consecutive iterations is below a pre-defined threshold. The inner-relationship that relates the scores matrices $\boldsymbol{T}$ and $\boldsymbol{U}$ ($\boldsymbol{X}$ and $\boldsymbol{Y}$ blocks respectively) is $\boldsymbol{U} = \boldsymbol{T}\boldsymbol{W}$. Although equation 2.8 and equation 2.9 suggest that the loading vectors are calculated for each block independently, this can actually lead to poor correlation between the two sets of scores. In practice, the inner-relationship is improved by exchanging the scores vectors $\boldsymbol{t}$ and $\boldsymbol{u}$ during the iterative cycle. Once all the latent vectors have been found, the regression matrix is calculated using:

$$\hat{\boldsymbol{B}} = \boldsymbol{P}(\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{W}\boldsymbol{Q}^T \qquad \text{(equation 2.10)}$$

Each column of $\hat{\boldsymbol{B}}$ contains the regression spectrum for the corresponding column of $\boldsymbol{Y}$. Future values are predicted from measured spectra using the expression:

$$\hat{\boldsymbol{Y}} = \boldsymbol{X}\hat{\boldsymbol{B}} \qquad \text{(equation 2.11)}$$

2.7 Vertex Vector Sequential Projection (VVSP)

Vertex Vector Sequential Projection (VVSP) is a self-modelling curve resolution method first published by Wang *et al.*[95, 96] in 2006. The method was originally applied to HPLC-DAD data to resolve two artificial data sets and two real data sets into their pure component spectral and chromatographic profiles. When applied in the row (time) direction of a data set, VVSP will identify the spectra that most represent the pure component spectra. These spectra can be used to calculate the corresponding concentration profiles using simple least-squares or constrained alternating least squares (CALS).

As with all SMCR methods, the data is assumed to have a bilinear model of the form shown in equation 2.12: where $\boldsymbol{X}$ is a two-way data matrix with dimensions $(J \times K)$; $\boldsymbol{C}$ is a $(J \times N)$ matrix of pure component concentration profiles; $\boldsymbol{S}$ is a $(K \times N)$ matrix of pure component spectra.

$$\boldsymbol{X} = \boldsymbol{C}\boldsymbol{S}^T \qquad \text{(equation 2.12)}$$

The method is based on the fact that when the spectra are normalised to unit length, they represent data points in a $K-$dimensional space. These points are distributed across a polyhedral hyper-"spherical" surface with the pure component variables located at the vertices. By normalising each spectrum (row) of $\boldsymbol{X}$ to unit *p*-norm, the bilinear model can be written as:

$$\boldsymbol{Y}_s = \boldsymbol{W}_c^p\boldsymbol{S}^T \qquad \text{(equation 2.13)}$$

where $\boldsymbol{Y}_s$ is the normalised two-way data matrix, $\boldsymbol{W}_c^p$ is the normalised weighting (concentration) matrix and $\boldsymbol{S}$ is the matrix of normalised pure spectra. It was proven by the authors that the quadratic function $f(\boldsymbol{w}) = \boldsymbol{w}^T \cdot \boldsymbol{S}^T \cdot \boldsymbol{A} \cdot \boldsymbol{S} \cdot \boldsymbol{w}$ must be maximised at the

pure variable $\boldsymbol{s}_{K+1}$ if it exists, where $\boldsymbol{A}$ is the null matrix of the first K pure spectra collected in the matrix $\boldsymbol{Z}_M = [\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_M](M < N)$.

$$\boldsymbol{A} = \boldsymbol{I}_K - \boldsymbol{Z}_M \boldsymbol{Z}_M^+ \qquad \text{(equation 2.14)}$$

The VVSP algorithm procedure therefore identifies the first N pure spectral profiles by sequentially solving the expression in equation 2.15, using the M previously identified components to calculate the null space as shown in equation 2.14.

$$f(\boldsymbol{w}_j) = \boldsymbol{y}_j^T \cdot \boldsymbol{A} \cdot (\boldsymbol{y}_j^T)^T \qquad \text{(equation 2.15)}$$

When N pure spectral profiles have been identified, the corresponding spectral concentration profiles may be calculated using least-squares:

$$\boldsymbol{C} = \boldsymbol{X}(\boldsymbol{S}^T)^+ \qquad \text{(equation 2.16)}$$

2.8 Linear Kalman filter

The Kalman filter is a set of equations that use a recursive prediction – correction approach to optimise the prediction of state parameters for a given measurement vector. Each element in the measurement vector (such as a spectrum) is used in the recursive calculations and the final estimation of the state parameters is optimal in the sense that they minimise the estimated error covariance. The equations summarising the linear Kalman filter are shown in Table 2.1.

The first equation describes the system dynamics of the model; how the system state parameters change from time $(k-1)$ to k. For a time-variant system model, the equation is $\boldsymbol{x}_j(k) = \boldsymbol{H}(k \mid k-1) \cdot \boldsymbol{x}(k-1) + \boldsymbol{w}(k)$. The matrix $\boldsymbol{H}(k-1)$ is called the state transition matrix and describes how the system state parameters change from time $(k-1)$ to k. It should be noted that throughout the literature, descriptions of the Kalman filter often refer to the index k as time. When applying the Kalman filter to multivariate spectroscopic data to estimate the concentration of each component of interest, the Kalman filter is applied to each element in the spectrum and k therefore denotes spectroscopic variables, not time. When the Kalman filter is applied to multivariate spectroscopic data, the state parameters are invariant with respect to k and the time-invariant system model shown in

equation 2.17 is used. The time-invariant system model substitutes $\boldsymbol{H}(k \mid k-1)$ with a $(N \times N)$ identity matrix, $\boldsymbol{I}$. The variable $\boldsymbol{w}(k)$ is the system noise vector at time (or variable) k and is assumed to be zero-mean white noise.

State estimate extrapolation (equation 2.18) and error covariance extrapolation (equation 2.19) allows the estimated state parameters and the associated error covariance calculated at index $(k-1)$ to be propagated to index k. The $(N \times N)$ matrix $\boldsymbol{Q}(k-1)$ is the covariance matrix of the noise in the system model $\boldsymbol{w}(k)$ and each diagonal element is the variance contribution from each of the N components to the system model. For the time-invariant model used for spectroscopic data, $\boldsymbol{Q}(k)$ and $\boldsymbol{w}(k)$ are assumed to be zero. The variance of the noise contributing to the measurement process, defined by the scalar $R(k)$, is therefore assumed to be the major source of error.

The Kalman gain is calculated to minimise the *a posteriori* error covariance $\boldsymbol{P}$, and is then used to update the state parameter estimates. The Kalman gain is calculated using equation 2.21. The reference measurement function $\boldsymbol{S}$ is a $(N \times K)$ row-oriented matrix of reference spectra (pure spectral profiles) for each component contributing to the system. When the Kalman gains have been calculated, a new estimate of the state parameters using the measurement at index k is calculated using equation 2.22. The error covariance is updated at each iteration using equation 2.23 where the diagonal elements of the $(N \times N)$ matrix $\boldsymbol{P}$ contain the variance of the corresponding state parameters estimates.

The innovation, calculated using equation 2.20, is the residual of the Kalman filter fit at each point of the measurement vector and is the difference between the measured data point at index k and the predicted value from the *a priori* prediction of the state parameters. In the basic linear Kalman filter, the innovation $v_j(k)$ is a scalar but is usually stored in a row-vector to form an innovations sequence. When the Kalman filter is operating optimally, the innovations sequence should resemble zero-mean white noise with variance R. The structure of the innovations sequence is therefore a useful way to assess the performance of the filter.

The linear Kalman filter is suitable for applications where the reference measurement function vectors (pure spectral profiles) for all components contributing to the system are known. Once the Kalman gains and the error covariance matrix have been calculated for one spectrum, they do not need to be recalculated for subsequent measurement vectors if

the variance of the measurement noise is unchanged. The Kalman gains and error covariance matrix can therefore be used to calculate the state parameters directly using equation 2.18, equation 2.20 and equation 2.22 only.

Table 2.1: Linear and Adaptive Kalman filter equations

Time invariant system model

$$\boldsymbol{x}_j = \boldsymbol{I} \cdot \boldsymbol{x}_j(k \mid k-1) + \boldsymbol{w}(k) \qquad \text{(equation 2.17)}$$

State estimate extrapolation

$$\boldsymbol{x}_j(k \mid k-1) = \boldsymbol{x}_j(k-1 \mid k-1) \qquad \text{(equation 2.18)}$$

Error covariance extrapolation

$$\boldsymbol{P}(k \mid k-1) = \boldsymbol{P}(k-1 \mid k-1) + \boldsymbol{Q}(k-1) \qquad \text{(equation 2.19)}$$

Calculate innovations

$$v_j(k) = z_j(k) - \boldsymbol{s}_k^T(k) \cdot \boldsymbol{x}_j(k \mid k-1) \qquad \text{(equation 2.20)}$$

Kalman gain update

$$\boldsymbol{g}(k) = \boldsymbol{P}(k \mid k-1) \cdot \boldsymbol{s}_k(k) \cdot [\boldsymbol{s}_k^T(k) \cdot \boldsymbol{P}(k \mid k-1) \cdot \boldsymbol{s}_k(k) + R(k)]^{-1} \qquad \text{(equation 2.21)}$$

State estimate update

$$\boldsymbol{x}_j(k \mid k-1) = \boldsymbol{x}_j(k \mid k-1) + \boldsymbol{g}(k) \cdot [z_j(k) - \boldsymbol{s}_k^T(k) \cdot \boldsymbol{x}_j(k \mid k-1)] \qquad \text{(equation 2.22)}$$

Error covariance update (Joseph implementation)

$$\boldsymbol{P}(k \mid k) = [\boldsymbol{I} - \boldsymbol{g}(k) \cdot \boldsymbol{s}_k^T(k)] \cdot \boldsymbol{P}(k \mid k-1) \cdot [\boldsymbol{I} - \boldsymbol{g}(k) \cdot \boldsymbol{s}_k^T(k)]^T + \boldsymbol{g}(k) \cdot R(k) \cdot \boldsymbol{g}^T(k)$$

(equation 2.23)

Additional Adaptive Kalman filter equation

Update measurement noise variance estimate

$$R_j(k) = \frac{1}{m}[\sum_{i=1}^{m} v_j(k-i) \cdot v_j(k-i)] - \boldsymbol{s}_k^T(k) \cdot \boldsymbol{P}(k \mid k-1) \cdot \boldsymbol{s}_k(k) \qquad \text{(equation 2.24)}$$

2.9 Adaptive Kalman filter

The adaptive Kalman Filter was developed by Rutan and Brown[66] to allow the Kalman filter to be applied to a system containing measurement model errors. The adaptive Kalman filter uses the standard equations of the linear Kalman filter shown in Table 2.1. The innovation values calculated using equation 2.20 are stored in a vector and the adaptive measurement noise variance $R(k)$ is calculated over a window of size m using the expression shown in equation 2.24. When the model is valid, $R(k)$ is small and the state-estimate update is sensitive to the small changes in the innovation. If there is an error in the model, for example an additional component is contributing to the measurement, the resulting innovation will be large which in turn leads to a larger value of $R(k)$. When $R(k)$ is large, the state parameter update becomes less sensitive to the changes in the innovations sequence and is therefore less affected by the model errors. If the values of $R(k)$ are stored in a vector, they can be used to either augment the reference measurement function matrix with an additional component, or correct an existing measurement function that is suspected of being in error. The equations required to determine the sign of the model error and update or augment the reference measurement function matrix[66] are shown in Table 2.2.

For the adaptive Kalman filter to be successful, each component represented in the matrix of reference measurement functions must have a small region in the variable space where the model is valid and can yield an accurate estimate of its state parameter. This requires that the reference measurement functions are not completely overlapped with each other or any additional un-modelled component that is contributing to the measurement vector. If the reference measurement functions are all highly overlapped with the additional un-modelled component, the calculated innovations sequence will contain large values which in turn will produce relative large values of $R(k)$. A large value of $R(k)$ will decrease the value of the Kalman gain and therefore the sensitivity of the state parameter update to future measurements.

Another point to consider is that because the measurement noise variance $R(k)$ is adaptively updated at each measurement point, the Kalman gain matrix will also change from spectrum to spectrum if it is not fully modelled by the reference measurement function. Only once the system is completely and accurately modelled will the Kalman gain and error covariance matrices be applicable to all spectra in the data set.

Table 2.2: Functions used by adaptive Kalman filter to augment or correct the reference measurement function matrix.

Sign of model error deviations

$$b(k) = 1, \quad \text{for} \sum_{l=1}^{m} v(k-l+m/2)/m > 0 \qquad \text{(equation 2.25)}$$

$$b(k) = -1, \quad \text{for} \sum_{l=1}^{m} v(k-l+m/2)/m < 0 \qquad \text{(equation 2.26)}$$

Augmentation of the reference measurement function with a new component

$$S^*_{N+1}(k) = b(k) \cdot [R(k+m/2)]^{1/2}, \text{ for } b(k) > 0 \qquad \text{(equation 2.27)}$$

$$S^*_{N+1}(k) = 0, \qquad \text{for } b(k) > 0 \qquad \text{(equation 2.28)}$$

Correction of the n^{th} component of the reference measurement function

$$S^*_n(k) = S_n(k) + b(k) \cdot [R(k+m/2)]^{1/2}, \text{ for } S^*_n(k) > 0 \qquad \text{(equation 2.29)}$$

$$S^*_n(k) = 0, \qquad \text{for } S^*_n(k) < 0 \qquad \text{(equation 2.30)}$$

2.10 Vectorised Kalman filters

Vectorised Linear Kalman filter

The Kalman filter algorithms described in sections 2.8 and 2.9 have been implemented to operate recursively on scalar data points z_k, where k is the k^{th} element of the measurement vector $\boldsymbol{z}$. At each iteration, the scalar measurement value z_k is used to update the estimates of the state parameters in a $(N \times 1)$ vector $\boldsymbol{x}$. The Kalman filter algorithms were readily extended to a $(J \times K)$ matrix of measurement vectors by incorporating an additional loop to process each measurement vector $\boldsymbol{z}_j^T$ independently. In the case of the linear Kalman filter, the Kalman gain and state estimate error covariance matrices was calculated for the first measurement vector in the data set and applied to subsequent measurement vectors without recalculation. This did relieve some of the computational burden but each measurement vector was still filtered sequentially. To

increase the computational efficiency of the Kalman filter when applied to large data sets, a vectorised Kalman filter was implemented. The vectorised Kalman filter operates recursively on a $(J \times 1)$ column vector of data points from a $(J \times K)$ matrix $\boldsymbol{Z}$ comprising J measurements acquired at K variables. The modified equations used to describe the vectorised Kalman filters are shown in Table 2.3.

Consistent with the linear Kalman filter, a time invariant system model is used when applied to multivariate spectroscopic measurements. The system model in equation 2.31 now uses a matrix of state-parameters denoted $\boldsymbol{X}$. The state estimate extrapolation equation (equation 2.32) also retains the same form but has been updated to indicate that the operation is performed on a matrix of state parameters. As written for the original linear Kalman filter, the error covariance matrix $\boldsymbol{P}$ is a $(N \times N)$ square matrix and the extrapolation equation (equation 2.33) remains unchanged. The diagonal elements of $\boldsymbol{P}$ still correspond to the variance of the state parameter estimates for each of the N components, but the variance is now based upon simultaneous prediction and correction of J measurement vectors. The Kalman filter is a recursive least-squares estimator that seeks to minimise the error covariance of the state estimates. The vectorised Kalman filter therefore provides a global minimum error covariance by simultaneously updating all elements of $\boldsymbol{X}$ during each iterative cycle.

The innovations calculation was vectorised as shown in equation 2.34. The innovations vector is the residual between the current column of measurement data at variable (iteration) k and the predicted column of measurement data calculated from the product of the current state parameter estimates $(\boldsymbol{X})$ and the reference measurement functions at variable (iteration) k, denoted $\boldsymbol{s}_k(k)$. Although the innovations are written and calculated as a column vector, they are stored in a $(J \times K)$ matrix $\boldsymbol{V}$.

The update of the Kalman gains shown in equation 2.35 is unchanged from the equation used by the original linear Kalman filter. The Kalman gain update is proportional to the magnitude of the elements in the error covariance matrix $\boldsymbol{P}$. As explained previously, $\boldsymbol{P}$ represents the global error covariance for all of the state parameters estimates in $\boldsymbol{X}$. Therefore, simultaneously applying the same Kalman gains vector $\boldsymbol{g}(k)$ to all J measurements in the state estimate update will lead to final state parameter estimates that have a globally minimised error covariance.

The state estimate update shown in equation 2.36 is the main difference between the original and vectorised Kalman filters. The entire $(J \times N)$ matrix of state parameters $\boldsymbol{X}$ is updated during each of recursive Kalman filter iterations. The Kalman gain vector calculated in the previous step is applied to all measurement vectors simultaneously using the column vector of innovations denoted $\boldsymbol{v}_k(k)$.

The error covariance update using the Joseph implementation[51] shown in equation 2.37 is identical to the expression used by the original Kalman filter. This is because the dimensions of the Kalman gain vector $(\boldsymbol{g})$ and error covariance matrix $(\boldsymbol{P})$ remain the same as those used by the original Kalman filter.

Vectorised Adaptive Kalman filter

It was also possible to extend the vectorised Kalman filter equations to the adaptive Kalman filter. The columns of the innovations matrix $\boldsymbol{V}$ are used to calculate a vector of adaptive measurement noise variances $\boldsymbol{r}_k(k)$, as shown in equation 2.39. As explained previously in section 2.9, the purpose of the adaptive Kalman filter is to reduce the sensitivity of the Kalman filer update equations in the regions that possess large model error. For the original adaptive Kalman filter, when the measurement model is valid, $R(k)$ is small and the state-estimate update is sensitive to the small changes in the innovation. If there is an error in the model, the innovations and therefore $R(k)$ are large and the state parameter update becomes less sensitive to the changes in the innovations sequence. This ensures that the state-estimate updates are less affected by the model errors. To extend this principle to the vectorised adaptive Kalman filter, the largest value in the column vector $\boldsymbol{r}_k(k)$ is assigned to $R(k)$. This ensures that the vectorised adaptive Kalman filter will calculate the Kalman gain update, error covariance update and state estimate update using the largest values in the innovations matrix. For those spectra containing contributions from an un-modelled component; the innovations vectors will possess characteristic spectral features of the un-modelled component in the variable regions where it is not overlapped with the known components. If the reference measurement functions accurately model the measurement data, the innovations vectors will resemble zero-mean white noise.

Table 2.3: Vectorised Linear and Adaptive Kalman filter equations

Time invariant system model (system noise assumed to be zero)

$$\boldsymbol{X}(k) = \boldsymbol{I} \cdot \boldsymbol{X}(k \mid k-1) \qquad \text{(equation 2.31)}$$

State estimate extrapolation

$$\boldsymbol{X}(k \mid k-1) = \boldsymbol{X}(k-1 \mid k-1) \qquad \text{(equation 2.32)}$$

Error covariance extrapolation

$$\boldsymbol{P}(k \mid k-1) = \boldsymbol{P}(k-1 \mid k-1) \qquad \text{(equation 2.33)}$$

Calculate innovations

$$\boldsymbol{v}_k(k) = \boldsymbol{z}_k(k) - \boldsymbol{X}(k \mid k-1) \cdot \boldsymbol{s}_k(k) \qquad \text{(equation 2.34)}$$

Kalman gain update

$$\boldsymbol{g}(k) = \boldsymbol{P}(k \mid k-1) \cdot \boldsymbol{s}_k(k) \cdot [\boldsymbol{s}_k^T(k) \cdot \boldsymbol{P}(k \mid k-1) \cdot \boldsymbol{s}_k(k) + R(k)]^{-1} \qquad \text{(equation 2.35)}$$

State estimate update

$$\boldsymbol{X}(k \mid k) = \boldsymbol{X}(k \mid k-1) + [\boldsymbol{z}_k(k) - \boldsymbol{X}(k \mid k-1) \cdot \boldsymbol{s}_k(k)] \cdot \boldsymbol{g}^T(k) \qquad \text{(equation 2.36)}$$

$$\boldsymbol{X}(k \mid k) = \boldsymbol{X}(k \mid k-1) + [\boldsymbol{v}_k(k)] \cdot \boldsymbol{g}^T(k)$$

Error covariance update (Joseph implementation) (equation 2.37)

$$\boldsymbol{P}(k \mid k) = [\boldsymbol{I} - \boldsymbol{g}(k) \cdot \boldsymbol{s}_k^T(k)] \cdot \boldsymbol{P}(k \mid k-1) \cdot [\boldsymbol{I} - \boldsymbol{g}(k) \cdot \boldsymbol{s}_k^T(k)]^T + \boldsymbol{g}(k) \cdot R(k) \cdot \boldsymbol{g}^T(k)$$

Additional Adaptive Kalman filter equations

Update measurement noise variance estimate

$$\boldsymbol{r}_k(k) = \frac{1}{m}\left[\sum_{i=1}^{i=m} \boldsymbol{v}_{k-i}(k-i) \cdot \boldsymbol{v}_{k-i}(k-i)\right] - \boldsymbol{s}_k^T(k) \cdot \boldsymbol{P}(k \mid k-1) \cdot \boldsymbol{s}_k(k) \qquad \text{(equation 2.38)}$$

$$R(k) = \max(\boldsymbol{r}_k(k)) \qquad \text{(equation 2.39)}$$

2.11 Vectorised Adaptive Kalman filter with Iterative Spectral Optimisation

Overview

The motivation for developing the vectorised Kalman filter with iterative spectral optimisation (VAKFISO) was to investigate whether recursive prediction-correction operations of the vectorised Kalman filter could be exploited for self-modelling curve resolution.

When the adaptive Kalman filter was originally developed and applied to data acquired using analytical instrumentation, the measurement vectors were considered as independent first order measurements, for example a single spectrum or voltammogram. When it is appropriate to consider all previously acquired measurement vectors, for example during a reaction monitoring experiment, the resulting data matrix becomes a second order data set. The bilinear nature of the data matrix may then be exploited, as the reference measurement functions of each component contributing to the measurement will lie within a subspace spanned by the primary eigenvectors obtained by singular value decomposition.

The objective of VAKFISO is to find a matrix of reference measurement functions (pure spectral profiles) that minimise a weighted residual matrix when used to calculate the corresponding state parameters using the vectorised adaptive Kalman filter. The advantage of utilising the Kalman filter is that it provides a state-parameter error covariance matrix. The diagonal elements of the state-parameter error covariance matrix, corresponding to the variance of the state estimates will be minimised when the matrix of reference measurement functions accurately models the measurement data. To find the set of reference measurement functions that minimise the diagonal elements of the error covariance matrix, the elements of a transformation matrix $\boldsymbol{T}$ are optimised using Newton-Gauss-Levenberg / Marquardt[22, 42] non-linear optimisation. During each iterative cycle, a new estimate of the optimised transformation matrix is calculated. Each spectrum in the matrix of test reference measurement functions is a linear combination of the primary eigenvectors spanning the spectral space. The transformation matrix $\boldsymbol{T}$ is used to transform the eigenvectors into test reference functions. The vectorised adaptive Kalman filter then allows the state-parameters of all components for all available measurement vectors to be calculated simultaneously; analogous to the way ALS methods such as MCR-ALS estimate the entire concentration matrix $\boldsymbol{C}$ during each iterative cycle.

As the matrix of test reference measurement functions approaches a feasible solution, the diagonal elements of the state-parameter error covariance matrix will be minimised. The innovations vector of each spectrum in the data set will resemble zero-mean, white noise indicating the Kalman filter is operating optimally. Without invoking any penalties, minimisation of the diagonal elements of the error covariance matrix or the residual matrix could correspond to negative spectra and / or negative state parameters. To prevent this, a weighted residual matrix is constructed from the initial innovations matrix, but also includes additional terms to penalise large state-estimate variances as well negativity in the test spectra and estimated state-parameters.

Detailed description of the VAKFISO algorithm

The major steps of the VAKFISO method are described below.

Initiation

The Kalman filter requires a reference measurement function (denoted $\boldsymbol{S}$) containing the pure spectral profile of each component contributing to the measurement data. Any SMCR method that allows the rank of the data to be estimated and provides initial estimates of the pure spectral profiles could be used, although VAKFISO specifically uses VVSP. The first step of VAKFISO is therefore to perform VVSP analysis of the full $(J \times K)$ matrix of data denoted $\boldsymbol{Z}$ to obtain initial, normalised estimates of the pure spectral profiles, denoted $\hat{\boldsymbol{S}}_{\boldsymbol{0}}$. The VVSP method is described in section 2.7.

Singular value decomposition[21, 30, 31, 34] is then applied to the matrix of measurement data to obtain the matrices of singular values and left and right singular vectors (eigenvectors).

$$\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V} \qquad \text{(equation 2.40)}$$

$\boldsymbol{Z}$ is the original or pre-processed data matrix; $\boldsymbol{U}$ is the matrix of left singular vectors that span the column space of $\boldsymbol{Z}$; $\boldsymbol{V}$ is the matrix of right singular vectors that span the row (spectral) space of $\boldsymbol{Z}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix of singular values.

The matrix of right singular vectors $\boldsymbol{V}$ is truncated to only retain the first N primary singular vectors to yield a $(K \times N)$ matrix $\bar{\boldsymbol{V}}$.

The initial $(N \times N)$ transformation matrix, $\boldsymbol{T}_{\boldsymbol{0}}$ is calculated from the initial estimates of the pure spectral profiles using the least-squares expression shown in equation 2.41.

$$\boldsymbol{T_0} = (\bar{\boldsymbol{V}}^T\bar{\boldsymbol{V}})^{-1}\bar{\boldsymbol{V}}^T\hat{\boldsymbol{S}}_0^T \qquad \text{(equation 2.41)}$$

Optimisation of the transformation matrix

The main operation of VAKFISO is to search for a transformation matrix $\boldsymbol{T}$ that minimises a weighted residual matrix $\boldsymbol{E}$. The transformation matrix is used to calculate a $(K \times N)$ matrix of reference measurement functions ($\hat{\boldsymbol{S}}$) that are then tested using the vectorised adaptive Kalman filter. The matrix of reference measurement functions is obtained using equation 2.42. Each column of $\hat{\boldsymbol{S}}$ is normalised to unit length or unit height using equation 2.43.

$$\hat{\boldsymbol{S}} = \bar{\boldsymbol{V}}\boldsymbol{T} \qquad \text{(equation 2.42)}$$

$$\hat{\boldsymbol{s}}_n = \hat{\boldsymbol{s}}_n / \left\|\hat{\boldsymbol{s}}_n\right\|_p \qquad \text{(equation 2.43)}$$

The vectorised adaptive Kalman filter is applied to the matrix of measurement data $(\boldsymbol{Z})$ using the current estimate of the pure spectral profiles to obtain a $(J \times N)$ matrix of estimated state parameters, $\hat{\boldsymbol{X}}$. The outputs of the Kalman filter, $\hat{\boldsymbol{X}}$, $\boldsymbol{V}$ and $\boldsymbol{P}$, along with the current matrix of reference measurement functions $(\hat{\boldsymbol{S}})$ are used to calculate a weighted residual matrix $\boldsymbol{E}$. The elements of $\boldsymbol{T}$ are optimised to minimise the elements of $\boldsymbol{E}$ using Newton-Gauss-Levenberg / Marquardt (NGL/M) non-linear optimisation[22, 42]. This optimisation approach allows an unsupervised search of the $K-$dimensional subspace of $\boldsymbol{Z}$ that is spanned by the matrix of right singular vectors $\bar{\boldsymbol{V}}$. The use of the state parameter error covariance matrix $\boldsymbol{P}$ in the calculation of the weighted residual matrix guides the search towards solutions of $\boldsymbol{T}$ (and therefore $\hat{\boldsymbol{S}}$) that allow the Kalman filter to operate optimally. These solutions should closely resemble the true spectral profiles.

One of the key characteristics of the Kalman filter that make it suitable for this type of optimisation problem are the recursive prediction-estimation nature of the calculations as the filter operates along the variable mode of the data. This allows the Kalman filter to provide an estimate of the errors associated with the final state-parameters in a error variance-covariance matrix denoted $\boldsymbol{P}$. The sum of the diagonal elements of error variance-covariance matrix will only approach a minimum if the product of the state

parameters and the current reference measurement functions completely model the data within the measurement error. This will be characterised by a complete set of innovations vectors that resemble zero-mean white noise. The optimisation of $\boldsymbol{T}$ will terminate when the maximum number of iterations or a convergence tolerance is reached. The advantage of using the adaptive Kalman filter is that the measurement variance R is not fixed but is adapted to the previous innovations values. This means that an accurate estimate of the measurement variance is not required and also makes the Kalman filter more tolerant of heteroscedastic noise.

Definition of the weighted residual matrix

The elements of the transformation matrix $\boldsymbol{T}$ are optimised in the sense that they minimise a weighted residual matrix $\boldsymbol{E}$. During each iterative cycle of the optimisation, the elements of $\boldsymbol{T}$ are changed and a new matrix of reference measurement functions, $\hat{\boldsymbol{S}}$ is calculated. The vector adaptive Kalman filter is applied to the data using the latest estimate of $\hat{\boldsymbol{S}}$ and the estimated state parameters $(\hat{\boldsymbol{X}})$, spectral innovations $(\boldsymbol{V})$, and state estimate error covariance matrix $(\boldsymbol{P})$ are used to formulate the weighted residual matrix $\boldsymbol{E}$. The weighted residual matrix also includes penalty terms that penalise negative values in the reference measurement functions and the estimated state parameters. These penalty terms can be can be used as non-negativity constraints to further guide the optimisation of $\boldsymbol{T}$.

The generalised expression for $\boldsymbol{E}$ is shown in

$$\boldsymbol{E} = \alpha_1 \boldsymbol{V} + (\alpha_2 \Pi \boldsymbol{V}) + (\alpha_3 \Sigma \boldsymbol{V}) + (\alpha_4 \Xi \boldsymbol{V}) \qquad \text{(equation 2.44)}$$

α_1 Weighting coefficient for the matrix of innovations, $\boldsymbol{V}$. Usually used to down-weight this term with respect to the other terms.

α_2 Weighting coefficient for the term that includes the trace of the variance-covariance matrix $\boldsymbol{P}$. Usually set to one or greater.

α_3 Weighting coefficient for the spectral negativity penalty function Π. Usually set to one or greater. Set to zero if spectral negativity is permitted.

α_4 Weighting coefficient for the state parameter negativity penalty function Ξ. Usually set to one or greater. Set to zero if state parameter negativity is permitted

The penalty functions expressed in equation 2.44 are defined as follows.

Π is the parameter related to the error covariance matrix for the final state parameter estimates and corresponds to the sum of the diagonal elements of $\boldsymbol{P}$.

$$\Pi = tr(\boldsymbol{P}) \qquad \text{(equation 2.45)}$$

Σ is the penalty function related to the fraction of negative values in the test reference measurement functions and is calculated as shown in equation 2.46.

$$\Sigma = \frac{1}{N}\sum_{k=1}^{k=K}\sum_{n=1}^{n=N}(\boldsymbol{\Omega}_{k,n}^2 / \hat{\boldsymbol{S}}_{k,n}^2) \qquad \text{(equation 2.46)}$$

$$\boldsymbol{\Omega} = \frac{1}{2}(|\hat{\boldsymbol{S}}| - \hat{\boldsymbol{S}})$$

The penalty function Σ will have a value in the range zero to one. If none of the test reference measurement functions have any negative regions, the value of Σ will be zero. If each reference measurement function is completely negative, the value of Σ will be one.

A similar penalty function Ξ can be applied to the state parameter matrix $\hat{\boldsymbol{X}}$ as shown in equation 2.47.

$$\Xi = \frac{1}{N}\sum_{j=1}^{j=J}\sum_{n=1}^{n=N}(\boldsymbol{\Theta}_{j,n}^2 / \hat{\boldsymbol{X}}_{j,n}^2) \qquad \text{(equation 2.47)}$$

$$\boldsymbol{\Theta} = \frac{1}{2}(|\hat{\boldsymbol{X}}| - \hat{\boldsymbol{X}})$$

The penalty function Ξ will also have a value in the range zero to one.

The coefficients α_1, α_2, α_3 and α_4 may be used to adjust the weighting of each term.

It is important that the expected values of $\boldsymbol{V}$ and Π are considered as they can differ by several orders of magnitude and may need to be weighted to ensure the optimisation does not take a long time to converge.

2.12 Iterative polynomial baseline subtraction

A method for automatically subtracting a broad, complex baseline from Raman or FTIR spectra was developed for this work. This method is based upon the automated method for subtraction of fluorescence from Raman spectra published by Lieber and Mahadevan-Jansen[97], and uses a least-squares based polynomial curve-fitting approach to approximate the underlying baseline. The method published by Lieber and Mahadevan is useful for correcting a relatively small region of interest (covering a range of just a few hundred wavenumbers) if the underlying baseline is a simple, continuous function. However it was found to be less suitable for processing a full spectrum (spectral range of three to four thousand wavenumbers) containing a complex baseline function. The iterative polynomial baseline subtraction algorithm is an extension of the method published in the reference cited above and includes several modifications. An estimate of baseline function is first obtained by applying a moving window median filter to the original, unprocessed spectrum $\boldsymbol{s}^T$. A large window is employed to ensure that none of the Raman peaks of interest are removed by the filter (size of window is 10% to 20% of total number of points in the spectrum). The output of the median filter is the filtered spectrum (desired Raman spectrum) and a residual spectrum that is used as an initial estimate of the underlying baseline spectrum. The spectra are denoted $\boldsymbol{s}_{mf}^T$ and $\boldsymbol{s}_r^T$ respectively. The algorithm then fits a polynomial function of degree n to a sub-window of $\boldsymbol{s}_r^T$. The width of the sub-window, w_p is defined by the user. A polynomial function is sequentially fitted to each sub-window of $\boldsymbol{s}_r^T$ to produce a new estimate of the baseline, denoted $\boldsymbol{s}_b^T$. An overlap parameter can also be specified to prevent discontinuity in the final baseline. If an overlap factor of 0.10 is used, a polynomial function is fitted to a window comprising of the current window, w, concatenated to the final 10% of the fitted points in the previous window, $(w-1)$. During the first iteration, when all W windows have been fitted with a polynomial function, the estimate of the baseline $\boldsymbol{s}_b^T$ is corrected using the following criterion:

If any point in the estimated baseline $\boldsymbol{s}_b^T$ is larger than the corresponding point in the initial estimate of the baseline $\boldsymbol{s}_r^T$, that point is replaced by the original data point.

$$\boldsymbol{s}_b^T(k) = \boldsymbol{s}_r^T(k) \text{ for } \boldsymbol{s}_b^T(k) > \boldsymbol{s}_r^T(k) \qquad \text{(equation 2.48)}$$

If any point in the estimated baseline $\boldsymbol{s}_b^T$ is less than zero, that point is replaced by the corresponding point in the median filter residual spectrum $\boldsymbol{s}_r^T$.

$$\boldsymbol{s}_b^T(k) = \boldsymbol{s}_r^T(k) \text{ for } \boldsymbol{s}_b^T(k) < 0 \qquad \text{(equation 2.49)}$$

After correcting the baseline estimate using the above criterion, the polynomial fitting procedure is repeated using $\boldsymbol{s}_b^T$. When all W windows of $\boldsymbol{s}_b^T$ have been fitted with a polynomial function, the updated estimate of the baseline, $\boldsymbol{s}_b^T$ is corrected using the following criterion:

If any point in the estimated baseline $\boldsymbol{s}_b^T$ is larger than the corresponding point in the original unprocessed spectrum $\boldsymbol{s}^T$, that point is replaced by the original data point.

$$\boldsymbol{s}_b^T(k) = \boldsymbol{s}^T(k) \text{ for } \boldsymbol{s}_b^T(k) > \boldsymbol{s}^T(k) \qquad \text{(equation 2.50)}$$

If any point in the estimated baseline $\boldsymbol{s}_b^T$ is less than zero, that point is replaced by the corresponding point in the original unprocessed spectrum $\boldsymbol{s}^T$.

$$\boldsymbol{s}_b^T(k) = \boldsymbol{s}^T(k) \text{ for } \boldsymbol{s}_b^T(k) < 0 \qquad \text{(equation 2.51)}$$

This process is then repeated until the sum-of-squares residual between two consecutive estimates of $\boldsymbol{s}_b^T$ is less than a predefined convergence tolerance or the maximum number of iterations is reached.

The final estimate of the baseline is then subtracted from the original spectrum to give the baseline subtracted spectrum, $\boldsymbol{s}_c^T$.

$$\boldsymbol{s}_c^T = \boldsymbol{s}^T - \boldsymbol{s}_b^T \qquad \text{(equation 2.52)}$$

The use of an initial estimate of the baseline spectrum obtained by median filtering allows the algorithm to converge must faster than starting from the original spectrum, and produces a smoother, more continuous baseline spectrum.

3 Experimental

3.1 Reagents and Equipment

3.1.1 Reagents

Table 3.1: List of reagents and solvents used.

Reagent	Grade	Manufacturer / Supplier
Acetonitrile	HPLC Grade	Romil, UK
Benzyl bromide	>98%	Fluka Chemie GmbH, Germany
1-benzyl-1*H*-indole	>90%	Synthesis product
Cesium carbonate	99.8%	Chemetall GmbH, Germany
4-chloro-7-methoxyquinazolin-6-yl acetate "Haloacetoxyone"	>98%	Synthesis product
Di-isopropylethylamine	>99%	Acros Organics bvba, Belgium
1*H*-indole	>99%	Fluka Chemie GmbH, Germany
7-methoxy-4-oxo-3,4-dihydroquinazolin-6-yl acetate "Acetoxyone"	>99.0%	Fluka Production GmbH, Switzerland
Phosphorus oxychloride	99%	Acros Organics bvba, Belgium
Tetrabutylammonium bromide	>99%	Sigma Aldrich, UK
Toluene	HPLC grade	Fisher Scientific, UK
Trifluoroacetic acid	HPLC grade	Fisher Scientific, UK

3.1.2 Spectrometers

3.1.2.1 Ultraviolet / Visible diode array spectrophotometers

Description

A Carl Zeiss MCS501 single beam UV/Vis diode array spectrophotometer (Clairet Scientific Ltd, Northampton, UK) was used to acquire data for laboratory scale reactions. This instrument was a single beam configuration employing a deuterium source lamp to provide a working spectral range of 200 to 650 nm. However, the use of fibre-optic cables restricted the spectral range further as the instrument uses a high pass filter to attenuate the output of the source below 220 nm to prevent solarisation of the optical fibres. The detector module comprises an aberration corrected concave grating (248 lines / mm) that directs the wavelength dispersed light onto a 512 pixel diode array. The grating and diode array provide a spectral resolution of 0.8 nm.

A Carl Zeiss MCS601 double beam UV/Vis diode array spectrophotometer (Clairet Scientific Ltd, Northampton, UK) was used to assess and compare the effect of fibre movement with standard and custom fibre-optic cable assemblies. This instrument had two detector modules and was enclosed in a purged, explosion proof cabinet so that it could be used within a manufacturing facility. In its standard configuration, the output of the source lamp was split into two using a short, bifurcated cable. One leg of the bifurcated cable transmitted the source light to a process probe, where it was returned to one of the detector modules via a separate fibre-optic cable. The second leg of the bifurcated cable transmitted the source light through a closed loop to the second detector module. This configuration allowed the spectrometer to compensate for the flash-to-flash intensity variation of a xenon flash lamp or the slow intensity reduction of a deuterium lamp if process monitoring was required over several days.

The probe used with this instrument was a Hellma UV/ATR probe, model number 661.822 (Hellma UK Ltd, Southend-on-Sea, UK). The probe was constructed from Hastelloy C22, and was 300 mm in length with a barrel diameter of 6.35 mm. The ATR sapphire prism was a three-bounce design. The probe was fitted with integral 4.5 m fibre-optic cables with high OH content, solarisation resistant 600 μm quartz glass cores terminated with standard SMA905 connectors. The specified spectral range for the probe was 220 to 1100 nm.

Spectral acquisition parameters

For normal reaction monitoring applications, the instrument parameters were configured as follows: The collection mode was set to absorbance using automatic interpolation over the range 220.0 to 650.0 nm in 1.0 nm intervals. In this mode, the spectrometer would collect a single beam energy spectrum, automatically convert it to an absorbance spectrum using a background spectrum previously acquired in air, and then interpolate the absorbance spectrum at 1.0 nm intervals. The detector exposure time was tuned for each experiment to maximise the signal intensity without saturating the detector, but typical values would range from 40 to 80 ms. The exposure time used would depend on the age of the source lamp and the length of the probe and its fibre-optic cables. Each absorbance spectrum was the average of 100 single beam energy spectra. The software was configured to acquire absorbance spectra at 1 minute intervals and to acquire a new dark spectrum before each measurement.

3.1.2.2 Ultraviolet / Visible Scanning Spectrophotometer

Description

The Varian Cary 50 (Varian Limited, Oxford, UK) was a scanning ultraviolet / visible spectrophotometer. The instrument used a focused high intensity xenon flash lamp operating at 80 Hz to provide a highly collimated beam. An internal beam splitter allowed the instrument to use a simultaneous reference beam to correct for flash-to-flash intensity variation. The fast scanning monochromator grating could operate at a maximum scan rate of 24000 $nm.min^{-1}$ over the spectral range 190 to 1100 nm. A fibre-optic coupler module was fitted in the sample compartment of the instrument to allow external probes to be connected using fibre-optic cables terminated with standard SMA905 connectors.

The probe used with this instrument was a Hellma UV/ATR probe, model number 661.822 (Hellma UK Ltd, Southend-on-Sea, UK). The probe was constructed from Hastelloy C22, and was 400 mm in length with a barrel diameter of 12.7 mm. The ATR sapphire prism was a three-bounce design. The probe was fitted with integral 4.5 m fibre-optic cables with high OH content, solarisation resistant 600 μm quartz glass cores terminated with standard SMA905 connectors. The specified spectral range for the probe was 220 to 1100 nm.

Spectral acquisition parameters

The instrument was controlled using the Cary WinUV suite of software and in this work the Scanning Kinetics application was used. This allowed multiple spectra to be acquired at sampling intervals and spectral ranges defined by the user. The scan parameters used to collect reaction spectra were configured as follows: The scan mode was set to collect absorbance spectra using double beam correction. In this mode, the spectrometer would collect a double beam energy spectrum and automatically convert it to an absorbance spectrum using a background spectrum previously acquired in air. The scan range was 400 to 220 nm at 1.0 nm intervals, using a scan speed of 480 $nm.min^{-1}$. The signal averaging time per wavelength point was set to 0.1250 seconds (equivalent to 10 lamp flashes). The software was configured to acquire absorbance spectra at 1 minute intervals.

3.1.2.3 Raman spectrometer

Description

The instrument used to acquire all Raman spectral data was a Kaiser Optical Systems Inc. Rxn1 Raman System (Clairet Scientific Ltd, Northampton, UK). The system was a dispersive Raman spectrometer that used a 785 nm Invictus NIR diode laser with a maximum power output of 400 mW as the excitation source. The spectrometer was coupled to a probe head using fibre-optic cables terminated with FC connectors. The probe head allowed a variety of probe attachments such as immersion probes or non-contact optics to be used. The probe-head also contained an optical notch filter that reduceed the intensity of Rayleigh scattered light returning to the spectrometer. The Raman scattered light then passed through an f/1.8 holographic imaging spectrograph. This comprised of a holographic notch filter that removed the remaining Rayleigh radiation and dispersed the Raman signal into its component wavelengths. The dispersed light was then directed onto a thermoelectrically cooled (-40ºC) charged coupled device (CCD). The instruments spectral range was +100 to + 3450 cm^{-1} with a spectral resolution of 1 cm^{-1}.

The probes used with this instrument were Raman immersion optics comprising of a Hastelloy C276 body (12.7 mm diameter) and a short focus optic protected by a sapphire window. The two probes used through out this work were identical in construction but had barrel lengths of 300 mm or 450 mm. Both probes were manufactured by Kaiser Optical Systems Inc.

Spectral acquisition parameters

The instrument manufacturer's software, HoloGRAMS™ was used for instrument setup and control. The individual spectra were automatically saved in a proprietary binary format (`.hol`) but were also exported as individual Thermo Scientific GRAMS (`.spc`) format files. The full set of reaction spectra could be loaded into HoloREACT™ and then exported as a single Matlab (`.mat`) file to facilitate further data analysis within Matlab. The general instrument specific parameters used to acquire spectral data were as follows: the exposure time and number of accumulations was set to give total acquisition time of 30 to 45 seconds without saturating the detector. For the *N*-benzylation of 1*H*-indole reactions, the exposure was set to 45 seconds and the number of accumulations was set to 1 (a total exposure of 45 seconds). In all experiments, intensity correction and wavelength calibration were applied to the raw spectra. The cosmic ray filter and automatic acquisition of a new dark spectrum was disabled as this would multiply the time required to acquire each spectrum by a factor of four. The laser power was set to a nominal value of 300 mW and the measured power at the probe focal point was 145 to 150 mW.

3.1.3 Chromatographic instruments

3.1.3.1 High Performance Liquid Chromatography

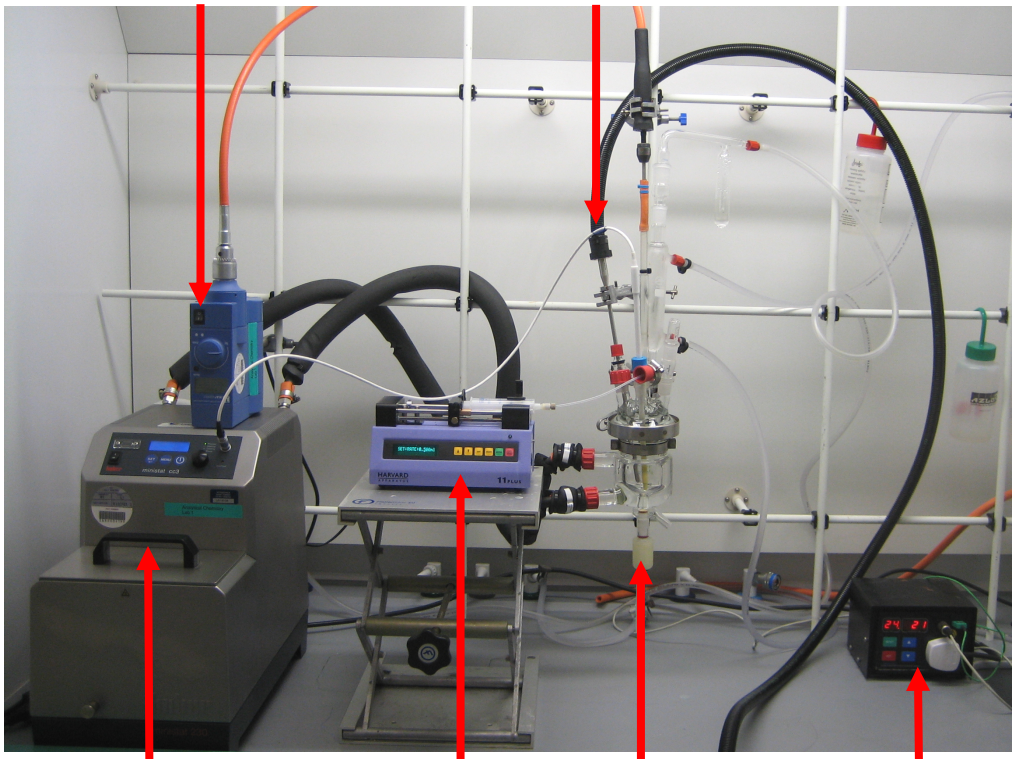
High performance liquid chromatography was performed using an Agilent 1100 Series HPLC system (Agilent Technologies UK Ltd., Wokingham, Berkshire, UK) comprised of a quaternary pump module with degasser, an autosampler module, a heated column compartment module with column switcher and a variable wavelength detector (VWD) module. The analogue output from the variable wavelength detector (VWD) was connected to a LabSystems ChromServer that digitised the analogue signal (0-1000 mV) and transferred the digitised data via Ethernet to a chromatographic data acquisition server (Atlas). The Atlas software could then be used to set peak windows, name and integrate peaks and perform calibrations or quantitative calculations.

The column, mobile phase solutions and method parameters are detailed in section 3.3.2.1.

3.1.4 Reaction vessels

3.1.4.1 Chlorination of acetoxyone using phosphorus oxychloride

The reaction vessel and associated equipment used for small scale laboratory experiments comprised of a 100 mL glass jacketed reaction vessel (Radleys, Saffron Walden, UK) with a Huber Ministat 230 thermoregulator unit (Huber UK, Saffron Walden UK) providing temperature control. The custom glass vessel lid had five Rodavis-type necks that were used to accommodate various items configured as follows: the central B19 port was used to accommodate a 3-blade overhead agitator, the 0º (vertical) B14 port was used to accommodate a Pt100 temperature probe and the remaining B19 ports were used to accommodate a condenser with nitrogen purge, a PTFE addition line introduced through a septum and a 6.35 mm diameter UV-ATR probe. The agitator was coupled to an IKA Werke Eurostar digital overhead stirrer (Fisher Scientific, Loughborough, UK) using a flexible stirrer shaft. A Harvard Apparatus 11 plus syringe driver (Fisher Scientific, Loughborough, UK) was used to deliver liquid reactants such as phosphorus oxychloride at a constant rate. A Contronics 2000M fume cupboard safety monitor was used to monitor the water flow to condenser. This would switch off the Huber unit if the water supply to the condenser failed, preventing evaporation of the reaction mixture. The configuration of the reaction vessel and probes is illustrated in Figure 3.1.



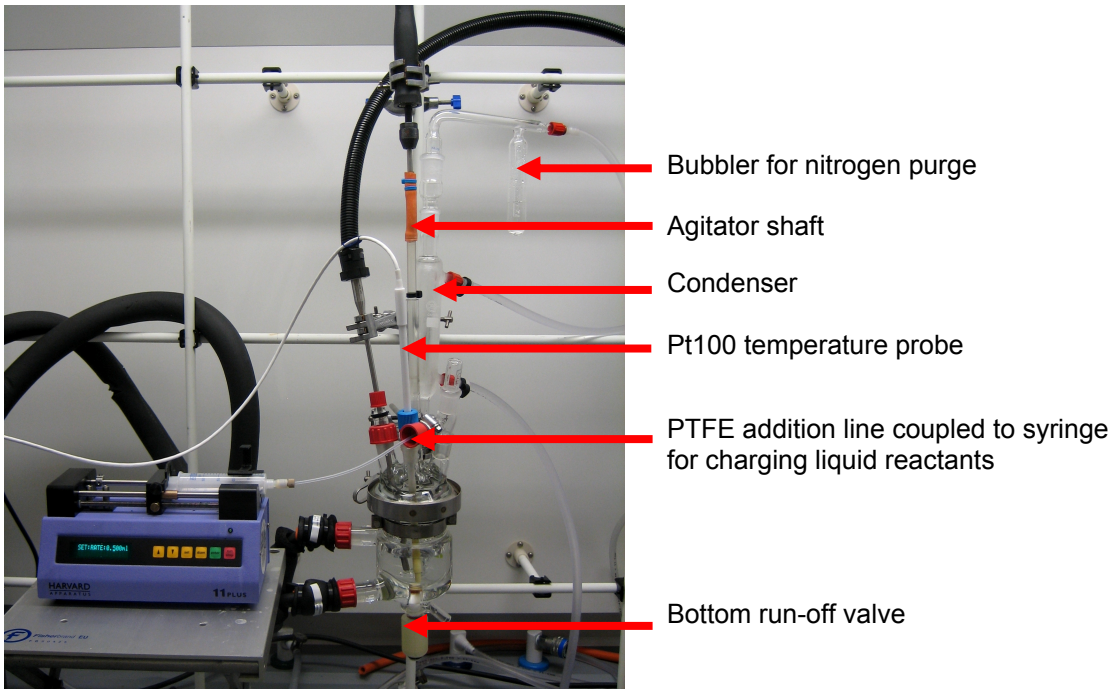


Figure 3.1: Reaction vessel and equipment as configured to perform the laboratory scale reactions for the chlorination of acetoxyone.

3.1.4.2 *N*-benzylation of 1*H*-indole using benzyl bromide

The reactions for these experiments were carried out in a 100 mL glass jacketed reaction vessel described in the previous section. The ports on the vessel head were used to accommodate an overhead agitator, a water-cooled condenser, a 12.7 mm diameter UV/ATR probe and a 12.7 mm diameter Raman immersion probe. The final port was fitted with a PTFE addition line passing through a septum to allow the addition of benzyl bromide via a syringe. The line was also used to take samples for off-line analysis. A Contronics 2010M fume cupboard safety monitor was used to monitor the water flow to condenser. This would switch off the Huber unit if the water supply to the condenser failed. The equipment configuration is shown in Figure 3.2.

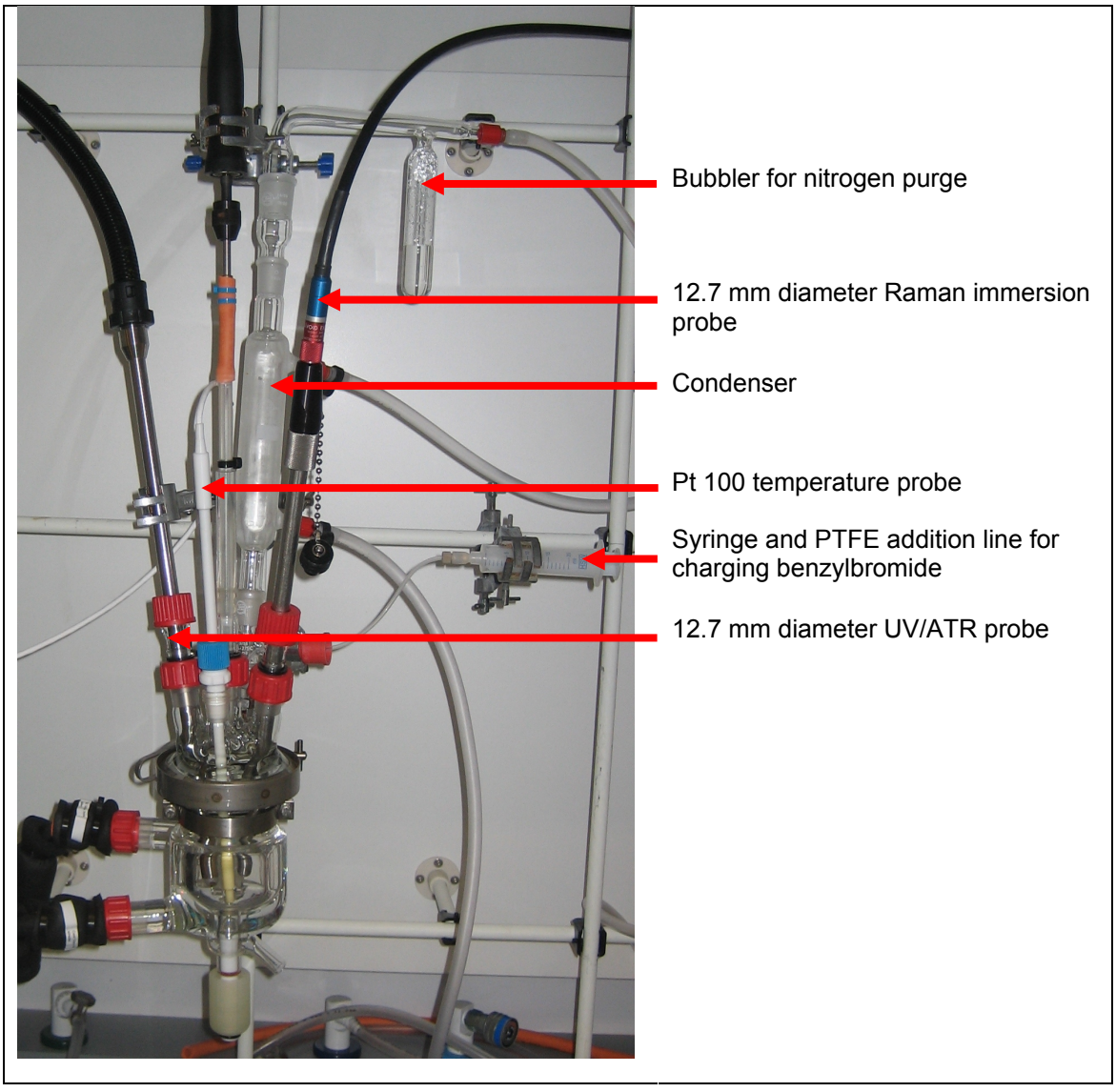


Figure 3.2: Reaction vessel and equipment as configured to perform the *N*-benzylation of 1*H*-indole reactions.

3.1.5 Computers and software

All Matlab scripts and data analysis were performed using Matlab 7.0.1 [R14] (The Mathworks, Natick, MA, USA) running on a Toshiba Satellite Pro 6100 laptop computer (Intel® Pentium ® 4 processor, 2.GHz, 1Gb RAM, Windows XP Professional). The Eigenvector PLS Toolbox (version 3.5, Eigenvector Research, Inc., Washington, USA) was used for data pre-processing and preliminary data analysis. The GRAMS PLSplus/IQ toolbox (Thermo Scientific, Waltham, MA, United States) was also used to transfer and run PLS calibration models on the spectroscopic instruments.

3.1.6 Visual Basic program used to prepare Cary 50 UV/Vis data

The original format of the UV data exported by the Varian Cary 50 software was a comma separated variable (`.csv`) file that stores the wavelength variable number and the corresponding absorbance values in adjacent columns. For a data set of *N* samples, the file would comprise of *N* pairs of columns containing the wavelength variables and the corresponding absorbance values for that sample. Although it was a simple procedure to directly import the data into Matlab and remove every alternate column, there were additional complications. Whilst acquiring data, the instrument software allowed the user to plot a kinetic profile at a single wavelength. Unfortunately, each time a profile was viewed, two additional columns were inserted into the data file. The first column contained the time at which each spectrum was acquired, and the second column contained the corresponding absorbance values at the selected wavelength. Finally, as the instrument used a scanning monochromator, it would occasionally skip a wavelength variable (for example … 284 nm, 283 nm, 282 nm, ( ), 280 nm …). This would cause the columns of data for that spectrum to become misaligned with previous columns.

To overcome these issues, and to provide a solution to other users who did not have access to Matlab, an easy to use Visual Basic 6 program was written to automatically filter and convert the data into a convinient format. The program was compiled into a stand-alone executable file which could be installed onto any Windows based PC. By writing a standalone application to automate this task, the risk of accidentally removing valid data points would be eliminated. Furthermore, other users of the instrument could also use the application to prepare their data for subsequent data analysis without being required to manually format or edit the data. The output of this program is a row-orientated data matrix. The first row of the processed data file contains the wavelength variable labels and the first column contains the sample numbers. All unwanted columns (such as those

generated by time profiles or duplicated wavelength variables) are detected and removed and the data then correctly aligned to compensate for any missed data points. The filtered data file is then saved as a comma separated variable text file with the extension (`.uvd`) that could then be imported into Matlab or Umetrics SIMCA for further analysis.

3.1.7 Custom Matlab scripts

Most multivariate data analysis was performed using Matlab and many custom scripts (functions) were written throughout the course of this research. Some scripts were written to facilitate the automation of simple pre-processing or data formatting tasks, others were written to perform calculations using specific algorithms published in the literature. The Matlab scripts for the various functions referenced in this thesis are included in Appendix II. To aid the reader, the code contains many comment lines describing the variables and calculations used and a detailed overview of each script is provided in the following section. The default command line syntax will be shown for each script. The input / output arguments shown in italic font are optional, whilst those shown in regular font are mandatory.

Table 3.2: Summary of Matlab functions.

Matlab Function name	Description	Section
ResidualComps.m	Visualisation of PCA residual spectra	3.1.7.1
OPA.m	Orthogonal projection approach	3.1.7.2
MedianFilter.m	Moving window median filter	3.1.7.3
IPBS.m	Iterative polynomial baseline subtraction	3.1.7.4
LinearKF.m	Linear Kalman filter (scalar implementation)	3.1.7.5
AdaptiveKF.m	Adaptive Kalman filter (scalar implementation)	3.1.7.6
VecLinearKF.m	Vectorised Kalman filter	3.1.7.7
VecAdaptiveKF.m	Vectorised adaptive Kalman filter	3.1.7.8
VVSP.m	Vertex vector sequential projection	3.1.7.9
VAKFISO.m	Vectorised adaptive Kalman filter with iterative spectral optimisation	3.1.7.10

3.1.7.1 Visualisation of PCA residual spectra (`ResidualComps.m`)

Function

The function of this simple script is to perform PCA (specifically SVD) on a data set. Using the singular values and eigenvectors calculated, the data is re-constructed using an increasing number of components, up to a maximum number of principal components specified by the user. The main benefit of this function is that it will display the re-constructed data alongside the corresponding residual data calculated using a different number of principal components. This allows the contribution of each new principal component to be assessed visually. For each principal component, the residual sum-of-squares are calculated and plotted versus principal component number to provide a numerical indication of the information contributed by each additional component. The user can then specify the number of principal components to use to re-construct the data, and the reduced data is written to the Matlab workspace. This is useful way to filter or clean up data prior to further analysis.

Input arguments

`[Data]` is a 2-dimensional array of data. The dimensions of this data matrix are $(I \times J)$ where I is the number of observations (samples or time points) and J is the number of experimental variables (*e.g.* wavelengths).

`[maxcomps]` is the maximum number of principal components to be used.

`[fig_num]` is the figure filename prefix to store the resulting figures. For example, if the user enters 'figure6', the resulting figure will be saved as 'figure6.fig' and 'figure6.emf'.

`[Xaxis]` is the vector containing the x-axis scale.

`[Xlabel]` is a string variable to used as the x-axis label on the plots generated by the script, *e.g.* 'wavelength (nm)'

`[Ylabel]` is a string variable to used as the y-axis label on the plots generated by the script, *e.g.* 'Absorbance'.

Output arguments

`[RegenData]` is the reduced data set re-constructed using the number of principal components specified by the user during the execution of the function.

Overview of `ResidualComps.m`

The script will perform some simple checks of the input arguments to ensure that string variables are used for the x- and y-axis labels; the size of the x-axis scale matches the dimensions of the data and that the maximum number of components is specified by the user.

Command line syntax

```
[RegenData] = ResidualComps(Data, maxcomps, fig_num, Xaxis, Xlabel, Ylabel)
```

1. Determine number of full figures, **`nfig`**, required to plot re-constructed data based upon number maximum number of principal components specified by user.

The default is 4 subplots (2 PCs) per figure, so 8 PCs would require 4 full figures

2. Determine number of subplots, **`nplotsfinal`**, required for a partially filled figure. For example 9 PCs would require 4 complete figures (with 4 subplots each) and 2 subplots in the final figure.

3. Perform singular value decomposition (SVD) on data. Store **`U`**, **`S`**, **`V`**

4. Set **`figurecounter=1`**

5. Outer loop (increment from **`n=1`** to **`n=nfig`**, step size **`1`**)

`nfig` is the number of figures with 4 subplots required.

- **6.** Open new figure
- **7.** Inner loop (increment from **`i=(n*2)-2+1`** to **`i=(n*2)`**)
  - **8.** Re-construct data using **`i`** principal components

    `X=U*S*V'`, use first **`i`** columns of **`U`** and **`V`** and first **`i`** rows and columns of **`S`**
  - **9.** Calculate residual

    `E=Data-X`
  - **10.** Calculate residual sum-of-squares

    `SSQ(i)=trace(E'*E)`
  - **11.** Plot **`X`** and **`E`**
- **12.** Return to **7**
- **13.** Save figure

14. Return to **5**

15. If **`nplotsfinal > 0`**

loop (increment from **`i=(n*2)-2+1`** to **`i=((n-1)*2)+nplotsfinal`**)

- **16.** Repeat steps **8** to **11** using **`i`** principal components

17. Return to **15**

18. Plot residual sum-of-squares versus principal component number

Save figure

19. User input **`ncomp`**: Number of principal components to use when re-constructing data

20. Re-construct data using **`ncomp`** principal components (calculated as shown in step **8**)

21. Save output variables to workspace

3.1.7.2 Orthogonal Projection Approach (`OPA.m`)

Function

The function of this script is to search a two-dimensional array of data, such as that obtained from a spectroscopic reaction monitoring experiment for estimates of the underlying pure component spectra (or other instrument response profile) using the Orthogonal Projection Approach (described previously in section 2.4)[25, 89, 98-100]. The function does not perform subsequent ALS calculations to optimise the pure component spectra or concentration profiles as this can be achieved using other functions.

Input arguments

`[Data]` is a 2-dimensional array of evolutionary experimental data such as spectra acquired during a reaction monitoring experiment. The dimensions of this data matrix are $(I \times J)$ where I is the number of observations (samples or time points) and J is the number of experimental variables (wavelengths, response variables).

`[N]` is the number of OPA components to calculate.

`[plotting]` allows the user to specify whether results are plotted. Plotting is 'off' when plotting=0 and 'on' when plotting=1.

Output arguments

`[PureSpec]` is a $(J \times N)$ matrix of pure component spectra located using OPA.

`[Dissim]` is an $(I \times N)$ matrix containing the calculated dissimilarity values for each spectrum as it compared with the previously located reference spectra.

`[DW]` is a $(N \times 1)$ column vector containing the Durbin-Watson values calculated for each new OPA component.

`[SI]` is a $(N \times 2)$ matrix. The first column stores the index (spectrum) numbers of the OPA pure component spectra and the second column stores the corresponding dissimilarity values.

Overview of `OPA.m`

Prior to performing the main OPA calculations, the script carries out some simple error checking of the input arguments. A number of appropriately dimensioned storage vectors or matrices with all elements set to zero are then created.

Command line syntax

```
[PureSpec, Dissim, DW, SI] = OPA(Data, N, plotting)
```

1. Calculate the mean spectrum of the input data matrix and assign to variable **`xs`**

2. Normalise **`xs`** to unit length

3. Assign **`xs`** as first column of **`Yi`**

4. Outer loop (increment from **`n=1`** to **`n=N`**, step size **`1`**)

`N` is the number of OPA components to calculate

5. Inner loop (increment from **`i=1`** to **`i=I`**, step size 1)

`I` is the number of observations / spectra

6. If **`n=1`**, set spectrum from row **`i`**, (**`xi`**) as second column of **`Yi`**

If **`n>1`**, set spectrum from row **`i`**, (**`xi`**) as **`n`**-th column of **`Yi`**

7. Calculate the dissimilarity of spectrum **`xi`**

```
Dissim(i,n) = det(Yi'* Yi)
```

8. Return to **5** (counter for **`i`**)

9. Identify the most dissimilar spectrum for component **`n`** from dissimilarity values (**`Dissim`**)

Save index (spectrum) number in storage matrix **`SI(n,1)`**

Save dissimilarity value in storage matrix **`SI(n,2)`**

10. Normalise most dissimilar spectrum identified in step **9** to unit length (**`xs_norm`**)

11. If **`n=1`**, store **`xs_norm`** as first column of **`Yi`**

If **`n>1`**, store **`xs_norm`** as **`n`**-th column of **`Yi`**

12. Calculate Durbin-Watson value from the **`n`**-th column of dissimilarity matrix (**`Dissim`**)

Save value in storage vector **`DW(n)`**

13. Return to 4 (counter for **`n`**)

14. Plot results

15. User input **`ncomp`**: Number of OPA components to retain

16. Truncate output variables to retain first **`ncomp`** OPA components

17. Save variables to workspace

3.1.7.3 Moving Window Median Filter (`MedianFilter.m`)

Function

This script will perform median filtering using a fixed sized window that moves along the variable direction of data vector or matrix. The function is an implementation of the method for the removal of low-frequency background drift based upon a moving window median filter published by Moore and Jorgenson in 1993[101]. Moore and Jorgenson originally developed this method to remove a baseline drift from liquid chromatographic data. The moving median filter is a non-linear filter that does not discriminate by frequency (such as the high-pass or low-pass linear digital filters based upon a fast Fourier transform). The median filter will remove impulses relative to a local background signal. So by selection of an appropriately sized window, both low and high frequency components of a signal are filtered simultaneously.

The function also permits the use of an increasingly larger (or decreasingly smaller) window. This option is for testing purposes and allows the user to directly compare the results obtained using different window sizes. The script will only allow a single spectrum to be used to test a range of window sizes.

An optional non-negativity correction has been included to remove negative values in the filtered data if desired.

Input arguments

`[Data]` is a 1- or 2-dimensional array of data. The dimension of this data matrix are $(J \times K)$, where J is the number of observations (1 for a single spectrum or chromatogram) and K is the number of experimental variables (*e.g.* wavelengths or retention times).

`[WS]` is the initial window size to be used for median filtering. `[WS]` must be odd-numbered. The user can specify a fixed window size by only entering a value for `[WS]`.

`[WE]` is an optional input argument that allows the user to specify a final window size when assessing a range of windows sizes for median filtering. `[WE]` must be odd-numbered. This input argument can be omitted if filtering using fixed window size is required.

`[step]` is an optional input argument used with `[WE]` and allows the user to specify the increment of the window size. This value must be an even-number. If `[WE]` and `[step]` are not provided, a fixed-size window defined by `[WS]` will be used during filtering.

`[NN]` is an optional input argument that allows the user to specify whether a non-negativity correction should be applied to the filtered data. Non-negativity is 'off' when NN=0 and 'on' when NN=1.

`[plotting]` specifies whether the function will display plots during run-time. Plotting is 'off' when plotting=0 and 'on' when plotting=1.

Output arguments

`[Filtered]` is a $(J \times K)$ matrix of signal component subtracted by the process of median filtering. If the original signal is broad and a small window size has been applied to remove high frequency 'spikes', `[Filtered]` will be the high-frequency part of the signal. If the signal contains high frequency peaks, such as FTIR or Raman spectra, and a large window size has been applied to remove a broad baseline contribution, `[Filtered]` will comprise of the desired Raman signal.

`[Residual]` is a $(J \times K)$ matrix of the residual signal component after subtraction of the median filtered signal from the original data. If the original signal is broad and a small window size has been applied to remove high frequency 'spikes', `[Residual]` will be the desired low frequency part of the signal. If the signal contains high frequency peaks such as FTIR or Raman spectra, and a large window size has been applied to remove a broad baseline contribution, `[Residual]` will be the underlying baseline.

During testing `[Filtered]` and `[Residual]` are $(n \times K)$ matrices where n is the number of increments defined by the values of `[WS]`, `[WE]` and `[step]`.

Overview of `MedianFilter.m`

Prior to performing the main median filter calculations, the script carries out some simple error checking of the input arguments. A number of appropriately dimensioned storage vectors or matrices with all elements set to zero are then created. The following description is for normal use when a vector or matrix of data is filtered using a fixed size window.

Command line syntax

```
[Filtered, Residual] = MedianFilter(Data, WS, WE, step, NN, plotting)
```

1. Assign original data matrix to **`X`**

2. Set window size to **`WS`**

3. Calculate the number of points either side of data point using window size **`WS`**

```
X_win = (WS-1)/2
```

4. To allow the filter to operate on the first and last **`x_win`** columns of the data, concatenate the original data matrix with mirror images of the first and last **`x_win`** columns of **`X`**

```
Xf=fliplr(X(:,1:x_win))
Xl=fliplr(X(:,end-x_win+1:end))
Xpad=[Xf, X; Xl]
```

5. Create a zero matrix, **`Residual`**, with dimensions **`J*(K+2*x_win)`** to store the median points

6. Loop (increment from **`k=(1+x_win)`** to **`k=(K+x_win)`**, step size **`1`**)

`K` is the number of variables

7. Copy columns **`k-x_win`** to **`k+x_win`** from **`Xpad`** and assign to variable **`Subset`**

```
Subset = Xpad(:,k - x_win:k + x_win)
```

8. Sort each row of **`Subset`** into ascending order

```
Subset=sort(Subset')
```

9. Assign the column of centre points (median values) from the ordered submatrix **`Subset`** to the matrix **`Residual`**

```
Residual(:,k) = Subset(x_win + 1)'
```

10. Return to **6** (counter for **`k`**)

11. Remove the padded columns from **`Xpad`** and **`Residual`** that were added in step **4**

12. Calculate the filtered data (**`Filtered`**) by subtracting the median values (**`Residual`**) from the original data

```
Filtered = X - Residual;
```

13. Plot results and save variables to workspace

3.1.7.4 Iterative Polynomial Baseline Subtraction (`IPBS.m`)

Function

The function of this script is to automatically subtract a broad, complex baseline from Raman or FTIR spectra. The method (described in section 2.12) is an extension of the automated method for subtraction of fluorescence from Raman spectra published by Lieber and Mahadevan-Jansen[97]. This method calls the Matlab script `MedianFilter.m` to first calculate an initial estimate of the baseline spectrum. The IPBS algorithm then proceeds to iteratively fit and refine this estimate by sequentially fitting a polynomial function to each sub-window of the data. The entire baseline spectrum is corrected during each iteration to preserve the original Raman features and non-negativity.

Input arguments

[D] is a $(J \times K)$ matrix or $(1 \times K)$ vector containing the spectra or spectrum to be processed.

`[n]` is the degree of the polynomial function to be used when fitting the data. The minimum value is `n=1`, the maximum value is `n=10`, default value is `n=4`.

`[Tol]` is the convergence tolerance to be used when iteratively refining the estimates of the polynomial baseline. The default value is `Tol=1E-03`. Convergence is calculated as the sum-of-squares residual between two consecutive estimates of the background function.

`[win]` is the window size to be used when fitting the polynomial baseline. This value may be an odd- or even-number.

`[MFwin]` is the window size to be used when applying median filtering to obtain an initial estimate of the underlying baseline function. This number must be an odd-number.

`[OLP]` is the overlap parameter and defines the fraction of the data in the previous window to be added to the current window. The total size of the window used to fit the polynomial function then becomes `win+(win×OLP)`. The value of `[OLP]` must be in the range 0.00 to 1.00. Note: a value of 1.00 is equivalent to doubling the size of `[win]`. The default value is `OLP=0.20`.

`[PlotOn]` allows the user to specify whether plotting during runtime is 'off' (`PlotOn=0`) or 'on' (`PlotOn=1`).

Output arguments

`[IPBS_OUTPUT]` is a structured array generated by this script to store the various outputs listed below.

`[.CorrSpec]` is a $(J \times K)$ matrix or $(1 \times K)$ vector containing the processed spectra after subtraction of the baseline.

`[.Baseline]` is a $(J \times K)$ matrix or $(1 \times K)$ vector containing the fitted baselines subtracted from the original data.

`[.MF_Residual]` is a $(J \times K)$ matrix or $(1 \times K)$ vector containing the initial estimates of the baselines provided by median filtering of the unprocessed spectra.

`[.MF_Filtered]` is a $(J \times K)$ matrix or $(1 \times K)$ vector containing the processed spectra after median filtering.

`[.MF_CorrSpec]` is a $(J \times K)$ matrix or $(1 \times K)$ vector containing the processed spectra after subtraction of a polynomial baseline estimated during the first iteration. This allows the user to compare the spectra obtained by subtraction of a polynomial baseline with the spectra obtained by median filtering.

`[.MF_Baseline]` is a $(J \times K)$ matrix or $(1 \times K)$ vector containing the polynomial baselines estimated during the first iteration. This allows the user to compare the baselines calculated by polynomial fitting with the baselines obtained by median filtering.

Overview of `IPBS.m`

Prior to performing the iterative polynomial fitting, the script carries out some simple error checking of the input arguments. A number of appropriately dimensioned storage vectors or matrices are then created. If the user has not provided all input arguments, the various optional inputs are set to the default values.

Command line syntax

```
IPBS_OUTPUT = IPBS(D, n, Tol, win, MFwin, OLP, PlotOn);
```

1. First perform moving window median filtering using the window width defined by **`MFwin`**. The Matlab function **`MedianFilter.m`** is called using the following command.

```
[MF_Filtered, MF_Residual]=MedianFilter(D, MFwin,[],[],[],1);
```

2. Outer loop (increment from **`j=1`** to **`j=J`**, step size **`1`**)

`J` is the number of spectra

3. Reset initial convergence for spectrum **`j`**: **`Convergence=1000;`**

Reset iteration counter for spectrum **`j`**: **`Counter=1;`**

Initialise vector to store new estimate of baseline: **`EBG_new=zeros(1,K);`**

Initial estimate of the baseline from MF: **`EBG_old=MF_Residual(j,:);`**

4. Whilst **`Convergence`** is greater than the convergence tolerance,

```
While Convergence > Tol
```

5. Reset counter used to identify first window from subsequent windows

```
WinCounter=1;
```

6. Loop to move sub-window through spectrum with step size **`win`**

(Increment from **`w=1`** to **`w=(nwin-1)*win`**, step size **`win`**)

7. Create variable index vector for sub-window **`w`** and create matrix **`M`** used to calculate polynomial coefficients. **`OL`** is the number of overlap points from previous window to be used.

If **`WinCounter=1`**

```
x=1:1:(1+win-1); x=x';
M=zeros(win,n+1);
```

If **`WinCounter>1`**

```
x=w-OL:1:(w+win-1); x=x';
M=zeros(win+OL,n+1);
```

8. Populate **`M`** with correct values ($1 + x + x^2 + x^3 + x^4 \ldots$). **`n`** is the degree of the polynomial

```
M(:,1)=1;
for p=1:n
    M(:,p+1)=x.^p;
end
```

9. If the iteration counter (**Counter**) is 1, use data from median filter baseline, otherwise use previous estimate of baseline function

if **Counter=1**

```
Dj=MF_Residual(j,:);
```

if **Counter>1**

```
Dj=Baseline(j,:);
```

10. Create subwindow **Dw** and populate with data from current window **w** and overlap points from previous window

if **WinCounter=1**

```
Dw=zeros(1,win);
Dw(1,1:win)=Dj(1,1:win);
Ds=D(j,1:win);
```

if **WinCounter>1**

```
Dw=zeros(1,win+OL);
Dw(1,1+OL:win+OL)=Dj(1, w:w+win-1);
Dw(1,1:OL)=EBG_new(1,w-OL:w-1);
Ds=D(j,x);
```

11. Calculate polynomial coefficients **A** for subwindow **Dw**

```
A=M\Dw';
```

12. Estimate background for window **w**, using polynomial coefficients **A**

```
EBG=(M*A)';
```

13. Correct estimate **EBG** to preserve Raman features and non-negativity.

A difference spectrum is calculated by subtracting **EBG** from the original data **Ds** over the same window,

```
Diff=Ds-EBG;
```

Find vector of indices (**z**) where **Diff** is less than zero

```
z=find(Diff<0);
```

Update those values of **EBG** where **Diff** is less than zero

```
EBG(z)= Ds(z);
```

Find vector of indices (**z**) where **EBG** is less than zero

```
z=find(EBG<0);
```

Update those values of **EBG** where **EBG** is less than zero

```
EBG(z)= Ds(z);
```

14. Copy updated window to vector storing the full spectrum baseline

```
EBG_new(1,x)=EBG;
```

15. Update window counter

```
WinCounter = WinCounter+1
```

16. Return to **6** (counter for **w**)

17. Calculate convergence

18. Update estimates of the full baseline spectrum

```
Baseline(j,:)=EBG_new;
EBG_old=EBG_new;
```

19. Update iteration counter

```
Counter=Counter+1;
```

20. If maximum number of iterations has been reached, exit while loop (step **4**)

21. Return to step **4** (while `Convergence > Tol`)

22. Return to step **2** (counter for `j`)

23. Calculate corrected spectrum by subtracting baseline spectrum from original data

```
CorrSpec=D-Baseline
```

24. Plot results and save results to output variable

3.1.7.5 Linear Kalman filter (`LinearKF.m`)

Function

This function is an implementation of the linear Kalman filter (section 2.8) as described by Rutan *et al.*[57, 66]. This script will run the linear Kalman filter using the first spectrum to calculate the Kalman gain and state parameter error covariance matrices. These are then used directly in the estimation of the state parameters for the remaining measurement vectors. The user can also provide Kalman gain and state parameter error covariance matrices calculated previously using another similar data set. The Kalman filter will also run through each measurement vector (spectrum) twice. The final estimates of the state parameters are propagated from the first pass and the Kalman filter will run through a second time to recalculate the innovations. Although this will not improve the state parameter estimates, the final innovations sequence and innovations based lack-of-fit values will be calculated using optimised state parameters.

Input arguments

`[S]` is a $(N \times K)$ row matrix of reference measurement functions, such as pure component spectra. N is the number of individual components and K is the number of measurement variables.

`[Z]` is a $(J \times K)$ matrix or $(1 \times K)$ vector of process measurement data where J is the number of observations (spectra) and K is the number of measurement variables.

`[KF_options]` is a structured array that can be used to provide additional optional input arguments. If the user does not provide any additional input arguments, the script will call the default values described below.

`[.R]` is an estimate of the measurement noise variance. The default value is 0.0001.

`[.G]` is a $(N \times K)$ matrix of Kalman gain values to be applied during Kalman filtering. The user may provide `[G]` if the Kalman filter has been run previously using the same reference measurement functions. The default value is [ ] (empty field).

`[.Xin]` is a $(J \times N)$ matrix of initial estimates of the state parameters. Enter a row vector for a single measurement vector or a row orientated matrix (J rows) if a number of measurement vectors (spectra) are used. The default value is [ ] (empty field).

`[.EvoluOn]` allows the user to specify whether the input matrix `[Z]` contains evolutionary data (such as HPLC-DAD or reaction spectra). If the data is evolutionary, the values of the state parameters at measurement $(j+1)$ will be similar to the values of the state parameters at measurement j. If `EvoluOn=1`, the final estimated values of the state parameters at sample j, variable $k = K$ are propagated to be the initial estimates for sample $(j+1)$ variable $k = 1$. The default value is `EvoluOn=0`.

`[.Pin]` is a $(N \times N)$ matrix of final state estimate variances to be used by Kalman filter. The default value is `Pin=[]` (empty field).

`[.Plotting]` allows the user to specify whether results are plotted after Kalman filtering has been performed. The default value is `Plotting=1` ('on').

Output arguments

`[KF_output]` is a structured array generated by this script to store the various outputs listed below:

`[.X]` is a $(J \times N)$ matrix or $(1 \times N)$ row vector of calculated state parameters; where J is the number of observations (spectra) and N is the number of state parameters (*e.g.* number chemical components in the system).

`[.G]` is a $(N \times K)$ matrix of Kalman gains calculated for the first measurement vector, $(j = 1)$ and used during Kalman filtering of the remaining $(J - 1)$ measurements.

`[.V]` is a $(J \times K)$ matrix or $(1 \times K)$ row vector of innovation values for each iteration (k) of the Kalman filter.

`[.LOF]` is the average measurement lack-of-fit with respect to the original data for each measurement j. This produces a $(J \times 1)$ column vector. A larger value indicates that the Kalman filter has estimated the original measurement vector with less accuracy and is a good indication of the presence of an un-modelled component.

`[.Pf]` is a $(N \times N)$ matrix that stores the final state parameter error variances (the diagonal elements of the error covariance matrix $\boldsymbol{P}$) at the K^{th} iteration.

Overview of `LinearKF.m`

Prior to performing the main Kalman filter calculations, the script carries out some simple error checking of the input arguments. A number of appropriately dimensioned storage vectors or matrices are then created. If the user has not provided the options structure **`[KF_options]`**, the various optional inputs are set to the default values.

Command line syntax

```
KF_output = LinearKF(S, Z, KF_options)
```

1. If user has not provided **`Pin`**, calculate initial estimate of error covariance matrix from the reference measurement function **`S`**:

```
P = cov(S') * eye(N)* 100
```

If user has provided **`Pin`**:

```
P = Pin
```

2. Outer loop (increment from **`SecondPass=-1`** to **`SecondPass=0`**, step size **1**)

3. Inner loop (increment from **`j=1`** to **`j=J`**, step size **`1`**)

`J` is the number of measurement vectors (spectra) to filter.

4. If **`EvoluOn=1`** and **`j>1`**

propagate state parameter estimates;

```
X(j,:) = X(j-1,:)
```

5. If **`j>1`**

Turn off calculation to update state parameter covariance matrix, **`P`**

```
UpdateP = 0
```

Turn off calculation to recalculate Kalman gains, **`G`**

```
UpdateG = 0
```

6. Inner loop (increment from **`k=1`** to **`k=K`**, step size **`1`**)

`K` is the number of measurement variables.

7. State estimate extrapolation

```
X_old(j,:) = X(j,:)
```

8. Error covariance extrapolation

```
P_old = P
```

9. Calculate innovations

```
V(j,k) = Z(j,k) - (S(:,k)'* X_old(j,:)')
```

10. If **`UpdateG = 1;`**

Calculate Kalman gain for variable **`k`**

```
G(:,k) = (P_old * S(:,k)) *
      inv((S(k,:)'* P_old * S(:,k)) + R)
```

11. Update state parameter estimate, **`X`**

```
g = G(:,k)
X(j,:) = X_old(j,:) + (g * V(j,k))'
```

12. If `UpdateP = 1;`

Update state parameter error covariance matrix, `P`

```
P = (eye(N) - (g * S(k,:)')) * P_old *
(eye(N)-((g * S(k,:)')))' + (g * R * g');
```

else

```
P = P_old
```

13. Return to **6** (counter for `k`)

14. Calculate spectral lack-of-fit from innovations sequence

```
VSSQ = V(j,:) * V(j,:)' / (K);
DATASSQ = Z(j,:) * Z(j,:)' /(K);
LOF(j) = (sqrt(VSSQ / DATASSQ)) * 100;
```

15. Return to **3** (counter for `j`)

16. Return to **2** (counter for `SecondPass`)

17. Plot results and save variables to workspace

3.1.7.6 Adaptive Kalman filter (`AdaptiveKF.m`)

Function

This function is an implementation of the adaptive Kalman filter (section 2.9) as described by Rutan *et al.*[57, 66]. As with the previous script (`LinearKF.m`), this function will run through the data twice. The state parameters are propagated from the first pass and the Kalman filter will run through a second time to recalculate the innovations. This will often yield a small improvement of the state parameter estimates but importantly, the final innovations sequence and innovations based lack-of-fit values will be calculated using optimised state parameters. The adaptive Kalman filter uses a moving window to adapt the measurement variance R to measurement model errors that lead to larger innovations values. This allows the filter to reduce the sensitivity of the state parameter update whilst the measurement model is in error. The user will then have the option to augment the matrix of reference measurement functions with a new component, or update an existing reference measurement function for one of the components. The augmentation or update assumes the reference measurement functions are non-negative as described by Rutan and Brown.

Input arguments

The input arguments `[S]` and `[Z]` are as previously described in section 3.1.7.5.

`[W]` is the window size to be used for adapting the measurement variance estimates, Rk. `W` must be an even-number.

`[KF_options]` is a structured array that can be used to provide additional optional input arguments. `[.Rmin]` is the minimum measurement noise variance permitted. The actual measurement noise variance will be calculated adaptively during the Kalman filtering but the value of `Rk` is not permitted to drop below `Rmin`. This value should be two or three orders of magnitude lower than the expected measurement noise variance. The default value is 1.0×10^{-6}. The other additional input arguments are as described previously in section 3.1.7.5.

Two optional input arguments that are not included in the structured array for `AdaptiveKF.m` are `[.G]` (matrix of Kalman gain values to be applied during Kalman filtering) and `[.Pin]` (matrix of final state estimate variances). This is because both the

Kalman gain and state parameter error covariance update calculations use R, which will be adaptively updated during filtering and does not remain constant.

Output arguments

`[KF_output]` is a structured array generated by this script to store the various outputs described below:

`[.X]` is a $(J \times N)$ matrix or $(1 \times N)$ row vector of calculated state parameters, where J is the number of observations (spectra) and N is the number of state parameters (*e.g.* number chemical components in the system).

`[.G_ALL]` is a $(N \times K \times J)$ array of Kalman gains calculated for each measurement vector.

`[.Pf_ALL]` is a $(J \times N)$ matrix that stores the final state parameter error variances (the diagonal elements of the error covariance matrix P) at the K^{th} iteration for each measurement vector.

`[.V]` is a $(J \times K)$ matrix or $(1 \times K)$ row vector of innovation values for each iteration (k) of the Kalman filter.

`[.Rk]` is a $(J \times K)$ matrix or $(1 \times K)$ row vector of adaptive measurement error values calculated at each iteration.

`[.S_new]` is the reference measurement function `[S]` after augmentation or correction using adaptive variance (Rk) values.

Overview of `AdaptiveKF.m`

Prior to performing the main Kalman filter calculations, the script carries out some simple error checking of the input arguments. A number of appropriately dimensioned storage vectors or matrices are then created. If the user has not provided the options structure **`[KF_options]`**, the various optional inputs are set to the default values.

Command line syntax

```
KF_output = AdaptiveKF(S, Z, KF_options)
```

1. Loop (increment from **`SecondPass=-1`** to **`SecondPass=0`**, step size **`1`**)

2. Loop (increment from **`j=1`** to **`j=J`**, step size **`1`**)

`J` is the number of measurement vectors (spectra) to filter.

3. If **`SecondPass=-1`**,

Calculate initial state parameter error covariance matrix, **`P`**

```
P = cov(S')*eye(N)*100
```

If **`SecondPass=0`**,

Use previously calculated **`P`** stored in 3-dimensional array **`Pf_ALL2`**

```
P = squeeze(Pf_ALL2(j,:,:))
```

4. If **`EvoluOn=1`** and **`j>1`**

propagate state parameter estimates;

```
X(j,:) = X(j-1,:)
```

5. Loop (increment from **`k=2`** to **`k=K-1`**, step size **`1`**)

`K` is the number of measurement variables.

6. State estimate extrapolation

```
X_old(j,:) = X(j,:)
```

7. Error covariance extrapolation

```
P_old = P
```

8. Calculate innovations

```
V(j,k) = Z(j,k) - (S(:,k)'* X_old(j,:)')
```

9. Calculate adaptive measurement variance, **`R`**

```
m=W
if m >= k
m = k-1
```

10. Loop (increment from **`i=1`** to **`i=m`**, step size **`1`**)

11. `V_sum(i)=(V(j,k-i)*V(j,k-i))`

12. Return to **10** (counter for **`i`**)

13. `V_sum=sum(V_sum);`

14. Update estimate of **`R`**, measurement error

```
Rk(j,k)=((inv(m))*(V_sum)) -
                        (S(k,:)'*P_old*S(:,k))
```

15. Limit variance to prevent **`R(j,k)`** approaching zero as that can result in a singular error covariance matrix **`P`**.

if **`Rk(j,k)<Rmin`**

```
Rk(j,k)=Rmin
```

16. Calculate Kalman gain for variable **k**

```
G(:,k) = (P_old * S(:,k)) *
  inv((S(k,:)'* P_old * S(:,k)) + Rk(j,k)
```

17. Update state parameter estimate **X**

```
g = G(:,k)
X(j,:) = X_old(j,:)+(g*V(j,k))'
```

18. Update state parameter error covariance matrix, **P**

```
P = (eye(N)-(g*S(k,:)')) * P_old *
(eye(N)-((g*S(k,:)')))'+(g*Rk(j,k)*g')
```

19. Return to **5** (counter for **k**)

20. Store state parameter error covariance matrix, **P**

```
Pf_ALL1(j,:) = diag(P)
Pf_ALL2(j,:,:) = P
```

21. Calculate spectral lack-of-fit from innovations sequence

```
VSSQ = V(j,:) * V(j,:)' / (K);
DATASSQ = Z(j,:) * Z(j,:)' /(K);
LOF(j) = (sqrt(VSSQ / DATASSQ)) * 100;
```

22. Return to **2** (counter for **j**)

23. Return to **1** (counter for **SecondPass**)

24. User selects from following options

'Press (1) to Augment the Reference Measurement Function' : Calls subfunction **Augment**

'Press (2) to Modify a component of Reference Measurement Function': Calls subfunction **Update**

'Press (3) to make no change to the Reference Measurement Function' : goto **End**

Overview of `Augment` and `update`

These subfunctions are called during **`AdaptiveKF`** function and will augment the reference measurement function with an estimate of the un-modelled component using the adaptive variance values in **`Rk`**, or update an existing estimate. The measurement number with the largest lack-of-fit value, **`LF_Index`** is used for calculations.

Command line syntax

```
[S_new]=augment(V,Rk,S,W,LOF_Index)
[S_new]=update(V,Rk,S,W,LOF_Index)
```

The first loop (steps **1** to **7**) will calculate the value of **`Bk`** for each variable **`k`**. This is used to calculate the average sign of the innovations values.

1. Loop (increment from **`k=2`** to **`k=K-(W/2)-1`**, step size **`1`**)

2. `m=W`

`if m >= k`

`m = k-1`

3. Loop (increment from **`i=1`** to **`i=m`**, step size **`1`**)

4. Calculate average value of **`b`**

`b(i)=V(LOF_Index,k-i+(floor(m/2)))/m`

5. Return to **3** (counter for **`i`**)

6. `b_sum=sum(b);`

if `b_sum > 0;  Bk(k)=1;`

if `b_sum < 0;  Bk(k)=-1;`

7. Return to **1** (counter for **`k`**)

The second loop (steps **8** to **12**) will perform the calculations to augment or update the reference measurement function

8. Loop (increment from **`k=2`** to **`k=K-(W/2)-1`**, step size **`1`**)

9. `m = W`

if `m >= k`

`m = k-1`

10. If subfunction **`augment`** is called

if `Bk(k)=1`

`S_Aug(k)=Bk(k)*(sqrt(Rk(LOF_Index, k+(floor(m/2)))))`

If `Bk(k)~=1`

`S_Aug(k)=0`

If subfunction **`update`** is called, **`n`** is the reference spectrum to update

`S_new=S`

`S_new(n,k)=S(n,k)+Bk(k)*(sqrt(Rk(LOF_index,k+(floor(m/2)))))`

if `S_new(n,k)<0`

`S_new(n,k)=0`

11. Return to **8** (counter for **`k`**)

12. Augment reference measurement function (augment only); **`S_new=[S;S_Aug]`**

3.1.7.7 Vectorised Linear Kalman filter (`VecLinearKF.m`)

Function

This is a vectorised implementation of the linear Kalman filter described in section 2.10. The Kalman filter calculations have been vectorised so if the user specifies a matrix of measurement data, the entire matrix of state parameter estimates will be updated at each iteration (variable k). The vectorised Kalman filter will produce identical results to the standard Kalman filter for the same data set $\boldsymbol{Z}$, reference measurement functions $\boldsymbol{S}$ and measurement noise variance R. However the script will run much faster. The user can also provide Kalman gain and state parameter error covariance matrices calculated previously. The script will run through the data twice. The state parameters are propagated from the first pass and the Kalman filter will run through a second time to recalculate the innovations. This will provide a better estimate of the final innovations (spectral) lack-of-fit calculated using optimised state parameters.

Input arguments

The input arguments `[S]` and `[Z]` have been described previously in sections 3.1.7.5 and 3.1.7.6. `[KF_options]` is a structured array that can be used to provide additional optional input arguments. The optional input arguments `[.R]`, `[.G]`, `[Xin]`, `[.Pin]` and `[.Plotting]` are identical to those described in sections 3.1.7.5 and 3.1.7.6.

Output arguments

`[KF_output]` is a structured array generated that stores the following outputs; `[.X]`, `[.G]`, `[.V]`, `[.Pf]` and `[.LOF]`. The output variables are identical to those described previously in section 3.1.7.5.

Overview of `VecLinearKF.m`

Prior to performing the main Kalman filter calculations, the script carries out some simple error checking of the input arguments. A number of appropriately dimensioned storage vectors or matrices are then created. If the user has not provided the options structure **`[KF_options]`**, the various optional inputs are set to the default values.

Command line syntax

```
KF_output = VecLinearKF(S, Z, KF_options)
```

1. If user has not provided **`Pin`**, allow KF to update **`P`**.

```
UpdateP = 1
```

If user has provided **`Pin`**:

```
UpdateP = 0
```

2. Outer loop (increment from **`SecondPass=-1`** to **`SecondPass=0`**, step size **`1`**)

3. If **`UpdateP = 1`**,

Calculate initial state parameter covariance matrix, **`P`**

```
P = cov(S')*eye(N)*100
```

If **`UpdateP = 0`**,

Use **`Pin`** provided by user

```
P = Pin
```

4. Inner loop (increment from **`k=1`** to **`k=K`**, step size **`1`**)

`K` is the number of measurement variables.

5. State estimate extrapolation (entire matrix is extrapolated)

```
X_old = X
```

6. Error covariance extrapolation

```
P_old = P
```

7. Calculate innovations vector

```
V(:,k)=Z(:,k)-(X_old*S(:,k))
```

8. If **`UpdateG = 1;`**

Calculate Kalman gain for variable **`k`**

```
G(:,k) = (P_old * S(:,k)) * inv((S_trans(k,:)
                * P_old * S(:,k)) + R)
```

9. Update matrix of state parameter estimates, **`X`**

```
g = G(:,k)
X=X_old + (V(:,k)*g')
```

10. If **`UpdateP = 1;`**

Update state parameter error covariance matrix, **`P`**

```
P = (eye(N) - (g * S(k,:)')) * P_old *
(eye(N)-((g * S(k,:)')))' + (g * R * g');
```

If **`UpdateP = 0;`**

```
P = P_old
```

11. Return to **4** (counter for **`k`**)

12. After first pass, turn off update of **`P`** and **`G`**

```
                UpdateP=0
                UpdateG=0
                Store final estimate of error covariance matrix
                Pf=P
```

13. Return to **2** (counter for **SecondPass**)

14. Calculate spectral lack-of-fit from innovations sequence

```
VSSQ=sum((V.^2),2)/(K)
DATASSQ=sum((Z.^2),2)/(K)
LOF = (sqrt(VSSQ ./ DATASSQ))*100
```

15. Plot results and save variables to workspace

3.1.7.8 Vectorised Adaptive Kalman filter (`VecAdaptiveKF.m`)

Function

This function is a vectorised implementation of the adaptive Kalman filter described in section 2.10. The Kalman filter calculations have been vectorised so if the user specifies a matrix of measurement data, the entire matrix of state parameter estimates will be updated during each iteration (variable k). The vectorised adaptive Kalman filter will produce identical results to the original Kalman filter for the same data set $\boldsymbol{Z}$, reference measurement functions $\boldsymbol{S}$ and measurement noise variance R. However the script will run much faster as a consequence of vectorisation. The script will run through the data twice; the state parameters are propagated from the first pass and the Kalman filter will run through a second time to recalculate the innovations. This will provide a better estimate of the final spectral lack-of-fit calculated using optimised state parameters. Unlike the original adaptive Kalman filter '`AdaptiveKF.m`', the user is not given the option to augment the matrix of reference measurement functions with a new component, or update an existing reference measurement for one of the components.

Input arguments

The input arguments `[S]`, `[Z]` and `[W]` have been described previously in section 3.1.7.6. `[KF_options]` is a structured array that can be used to provide additional optional input arguments. The optional input arguments `[.Rmin]`, `[Xin]` and `[.Plotting]` are identical to those described in sections 3.1.7.5 and 3.1.7.6. Two optional input arguments that are not included in the structured array for `VecAdaptiveKF.m` are `[.G]` and `[.Pin]`. This is because both the Kalman gain and state parameter error covariance update calculations use R, which will be adaptively updated during filtering and does not remain constant.

Output arguments

`[KF_output]` is a structured array generated by this script to store various outputs arguments. `[.X]`, `[.V]`, `[.Rk]` and `[.LOF]` are identical to the output arguments described previously in section 3.1.7.6. `[.G]` and `[.Pf]` are identical to the output arguments described in section 3.1.7.7.

Overview of `VecAdaptiveKF.m`

Prior to performing the main Kalman filter calculations, the script carries out some simple error checking of the input arguments. A number of appropriately dimensioned storage vectors or matrices are then created. If the user has not provided the options structure `[KF_options]`, the various optional inputs are set to the default values.

Command line syntax

```
KF_output = VecAdaptiveKF(S, Z, W, KF_options)
```

1. Outer loop (increment from `SecondPass=-1` to `SecondPass=0`, step size `1`)

2. If `SecondPass=-1`,

Calculate initial state parameter covariance matrix, `P`

```
P = cov(S')*eye(N)*100
```

If `SecondPass=0`,

```
P = Pf
```

3. Inner loop (increment from `k=2` to `k=K`, step size `1`)

`K` is the number of measurement variables.

4. State estimate extrapolation (entire matrix is extrapolated)

```
X_old = X
```

5. Error covariance extrapolation

```
P_old = P
```

6. Calculate innovations vector

```
V(:,k)=Z(:,k)-(X_old*S(:,k))
```

7. Calculate adaptive measurement variance, `R`

```
m=W
```

if `m >= k`

```
m = k-1
```

8. Initialise storage matrix `V_sum` to zeros

```
V_sum=zeros(J, m)
```

9. Loop (increment from `i=1` to `i=m`, step size `1`)

10. Square each element in the adaptive variance window

```
V_sum(:,i)=(V(:,k-i).*V(:,k-i))
```

11. Return to **9** (counter for `i`)

12.

```
V_sum2=sum(V_sum, 2);
```

13. Calculate vector of measurement errors `Rk` for variable (index) `k`

```
Rk(:,k)=(inv(m).*V_sum2') -
                          (S_trans(k,:)*P_old*S(:,k))
```

14. Find maximum value of `Rk` to use in update of Kalman gain

```
Rnew=max(Rk(:,k))
```

15. Limit variance to prevent `Rnew` approaching zero as that can result in a singular error covariance matrix `P`.

if `Rnew<Rmin`

```
Rnew=Rmin
```

16. Calculate Kalman gain for variable **k**

```
G(:,k) = (P_old * S(:,k)) * inv((S_trans(k,:)
                * P_old * S(:,k)) + Rnew)
```

17. Update matrix of state parameter estimates, **X**

```
g = G(:,k)
X=X_old + (V(:,k)*g')
```

18. Update state parameter error covariance matrix, **P**

```
P = (eye(N) - (g * S(k,:)')) * P_old *
(eye(N)-((g * S(k,:)')))' + (g * Rnew * g')
```

19. Return to **3** (counter for **k**)

20. Store final estimate of error covariance matrix

```
Pf=P
```

21. Return to **1** (counter for **SecondPass**)

22. Calculate spectral lack-of-fit from innovations sequence

```
VSSQ=sum((V.^2),2)/(K)
DATASSQ=sum((Z.^2),2)/(K)
LOF = (sqrt(VSSQ ./ DATASSQ))*100
```

23. Plot results and save variables to workspace

3.1.7.9 Vertex Vector Sequential Projection (`VVSP.m`)

Function

This function will perform Vertex Vector Sequential Projection analysis on a two-dimensional data array using the algorithm described by Wang *et al.*[95, 96] (section 2.7). The main principle of VVSP is that after p-normalisation (where p>1), all points in a two-way data matrix lie on a polyhedral hyper-"spherical" surface, with the pure variables (spectra) forming the vertices (vertex vectors). A certain quadratic expression $f(\boldsymbol{w}_j) = \boldsymbol{y}_j^T \cdot \boldsymbol{A} \cdot (\boldsymbol{y}_j^T)^T$ is maximised at those spectra that form the vertex vectors, allowing the closest estimates of the pure component spectral profiles to be located. To aid the selection of the number of components to retain the Durbin-Watson values (a measure of autocorrelation) are calculated using the vector of quadratic values $f(\boldsymbol{w})$ used to locate each successive VVSP component. The $\log_{10}$ of projection residuals sum-of-squares is also calculated for each component. The measurement data $\boldsymbol{X}$ is assumed to have a bilinear model $\boldsymbol{X} = \boldsymbol{C}\boldsymbol{S}^T + \boldsymbol{E}$. This function does not apply constrained alternating least squares to refine the initial estimates of $\boldsymbol{C}$ or $\boldsymbol{S}^T$.

Input arguments

`[X]` is the row-orientated 2-way data matrix with dimensions $(J \times K)$ where J is the number of sample or observations and K is the number of variables.

`[NL]` is the number of VVSP pure component profiles to initially locate.

`[NR]` is the number of VVSP pure component profiles to retain.

`[p_norm]` is the type of spectral normalisation to apply during VVSP analysis. Acceptable values are: `[p_norm]=2`, normalise to unit length; `[p_norm]=inf`, normalise to maximum value = 1 (normalise height). `[p_norm]` may also have values of 3, 4, 5 but this type of normalisation is uncommon.

`[plotting]` allows the user to specify whether plotting is 'off' (`[plotting]=0`) or 'on' (`[plotting]=1`).

Output arguments

`[VVSP_output]` is a structured array containing the following output variables:

`[.X]` is the workspace variable name of data matrix that VVSP was applied to.

`[.p_norm]` is the value of p-normalisation applied to the original spectra.

`[.Sopt]` is a $(K \times NR)$ matrix of VVSP pure component spectral profiles.

`[.Copt]` is a $(J \times NR)$ matrix of concentration profiles estimated using least-squares.

`[.SI]` is a $(NR \times 2)$ matrix storing the spectrum number (column 1) and corresponding $f(\boldsymbol{w})$ value (column 2) for each retained VVSP spectrum.

`[.fw]` is a $(J \times NR)$ matrix storing the vector of solutions to the quadratic expression $f(\boldsymbol{w}_j) = \boldsymbol{y}_j^T \cdot \boldsymbol{A} \cdot (\boldsymbol{y}_j^T)^T$, calculated for each VVSP component.

`[.fwNorm]` is a $(J \times NR)$ matrix storing the normalised vector of $f(\boldsymbol{w})$ values.

`[.DW]` is a $(NL \times 1)$ vector storing the Durbin-Watson value for each VVSP component and is calculated from the corresponding vector of $f(\boldsymbol{w})$ values.

`[.PR]` is a $(NL \times 1)$ vector storing the logarithmic SSQ projection residuals calculated for each new VVSP component located.

Overview of `VVSP.m`

Prior to performing the main VVSP calculations, the script carries out some simple error checking of the input arguments. A number of appropriately dimensioned storage vectors or matrices are pre-allocated.

Command line syntax

```
VVSP_output = VVSP(X, NL, NR, p_norm, plotting)
```

1. Loop (increment from `j=1` to `j=J`, step size `1`)

`J` is the number of spectra (row) in the data matrix `X`.

2. First normalise each spectrum in the original data matrix `X` using p-normalisation defined by `p_norm`. Also create a vector of 1-norms

```
u_norm_s(j) = norm(X(j,:),1)
p_norm_s(j) = norm(X(j,:),p_norm)
Y(j,:) = X(j,:)./ p_norm_s(j)
```

3. Return to **1** (counter for `j`)

4. Identify the largest 1-norm value from `u_norm_s` and store spectrum index and 1-norm value in first row of `SI`

```
[SI(1,1), SI(1,2)]=max(u_norm_s)
```

5. Normalise the spectrum with largest 1-norm to give `r`. The spectrum is normalised using the 1-norm

```
r = X(SI(1,2),:) ./ u_norm_s(SI(1,2))
```

6. Find the spectrum z_1 that maximises $\left\| \boldsymbol{r}^T - \boldsymbol{y}_j^T \right\|_2$

Calculate the residual between the spectrum `r` and each p-normalised spectrum in `Y`.

```
R=repmat(r,J,1)
E=R-Y
```

7. Loop (increment from `j=1` to `j=J`, step size `1`)

8. Calculate the p-norm of each spectrum in the residual matrix `E`.

```
ResidNorm(j)=norm(E(j,:),p_norm)
```

9. Return to **7** (counter for `j`)

10. Select the spectrum with largest residual p-norm and store as column vector in `Z`.

```
[SI(1,1), SI(1,2)]=max(ResidNorm);
Z=Y(SI(1,2),:)'
```

11. Now that the first spectrum has been selected, can start the loop to find `NL` VVSP spectra.

Loop (increment from `m=1` to `m=NL`, step size `1`)

12. Calculate null matrix for the vectors in `Z`

```
A=eye(K)-(Z*pinv(Z))
```

13. Use matrix expression to calculate $f(\boldsymbol{w})$ for each normalised spectrum in `Y`

```
fw(:,m) = (diag(Y*A*Y'))
```

Normalise each column of `fw`

```
fwNorm(:,m) = fw(:,m) ./ norm(fw(:,m),2)
```

14. Find maximum value and concatenate to `Z` (unless `m=1`)

`[SI(m,1), SI(m,2)]=max(fw(:,m))`. Store spectrum number and `fw` value

```
Z(:,1)=Y(SI(m,2),:)'        Replace first spectrum if m = 1
Z=[Z, Y(SI(m,2),:)']        Concatenate to Z if m > 1
```

15. Calculate Durbin-Watson value for vector of $f(w)$ values

```
fw_diff = diff(fw(:,m))
fw_diff = [fw_diff; 0]
fw_SSQ  = fw(:,m)' * fw(:,m)
fw_diff_SSQ = fw_diff' * fw_diff
DW(m,1) = fw_diff_SSQ ./ fw_SSQ
```

16. Calculate the $\log_{10}$ of the projection residual sum-of-squares, **PR**

Re-calculate the null matrix **A** using updated **Z**

```
A=eye(K)-(Z*pinv(Z))
```

Calculate the projection residuals using original data

```
XA=X*A
```

Calculate the $\log_{10}$ value of the projection residuals sum-of-squares

```
PR(m,1)=log10(trace(XA'*XA))
```

17. Return to **11** (counter for **m**)

18. If **NR** is not provided as an input argument, ask user to specify the number of VVSP components to retain. Truncate **Z**, **fw**, **fwNorm** and **SI** to retain only first **NR** VVSP components

19. Calculate the concentration profiles using least-squares

```
Sopt=Z
Copt=X*pinv(Sopt')
```

20. Plot results and save variables to workspace

3.1.7.10 Vectorised Kalman Filter with Iterative Spectral Optimisation (`VAKFISO`)

Function

The function of VAKFISO (section 2.11) is to find a matrix of reference measurement functions that minimise a weighted residual matrix when used to calculate the corresponding state parameters using the vectorised adaptive Kalman filter. The diagonal elements of the state-parameter error covariance matrix will be minimised when the matrix of reference measurement functions accurately model the measurement data. To find the set of reference measurement functions that minimise the diagonal elements of the error covariance matrix, the elements of a transformation matrix are optimised using Newton-Gauss-Levenberg / Marquardt non-linear optimisation. During each iterative cycle, a new estimate of the optimised transformation matrix is calculated. Each spectrum in the matrix of test reference measurement functions is a linear combination of the primary eigenvectors spanning the spectral space. The transformation matrix is used to transform the eigenvectors into test reference functions and the vectorised adaptive Kalman filter then allows the state-parameters of all components for all available measurement vectors to be calculated simultaneously. As the matrix of test reference measurement functions approach a feasible solution, the diagonal elements of the state-parameter error covariance matrix will be minimised. Without invoking any penalties, minimisation of the diagonal elements of the error covariance matrix or the residual matrix could correspond to negative spectra and / or negative state parameters. To prevent this, a weighted residual matrix is constructed from the initial innovations matrix but also includes additional terms to penalise large state-estimate variances as well negativity in the test spectra and estimated state-parameters.

Input arguments

`[Z]` is a $(J \times K)$ matrix of process measurement data where J is the number of observations (spectra) and K is the number of measurement variables.

`[N]` is the number of reference measurement functions (pure spectral profiles) to locate and optimise.

`[S]` is a $(K \times N)$ matrix of initial estimates of the reference measurement functions. If `[S]` is not provided, initial estimates would be obtained by applying VVSP to the data set.

`[VAKFISO_options]` is a structured array that can be used to provide additional optional input arguments. If the user does not provide any additional input arguments, the script will call the default values described below. The script '`VAKFISO_SetOptions`' may be run to create an options structure with the default values.

`[.W]` is window size to be used for adapting the measurement variance estimates, $R_k(k)$ during adaptive Kalman filtering. `[.W]` must be even-numbered.

`[.Rmin]` is the minimum measurement noise variance permitted as described previously in sections 3.1.7.6 and 3.1.7.8.

`[.p_norm]` is the type of spectral normalisation to apply during VVSP analysis or when calculating the state parameters using the final estimates of the reference measurement functions.

`[.alpha1]` is a weighting coefficient applied to the original matrix of innovations (spectral residuals) $\boldsymbol{V}$.

`[.alpha2]` is a weighting coefficient applied to penalty term Π. Π corresponds to the sum of the diagonal elements of the state estimate error covariance matrix $\boldsymbol{P}$. The default value is 1. Set this value to 0 if Π should not contribute to the residual matrix used during NGL/M optimisation.

`[.alpha3]` is a weighting coefficient applied to spectral negativity penalty term Σ. The default value is 1. Set this value to 0 if Σ should not contribute to the residual matrix used during NGL/M optimisation.

`[.alpha4]` is a weighting coefficient applied to state-estimate negativity penalty term Ξ. The default value is 1. Set this value to 0 if Ξ should not contribute to the residual matrix used during NGL/M optimisation.

`[.delta]` is the shift to be added to the elements of the transformation matrix $\boldsymbol{T}$ during NGL/M optimisation. A suitable value would be in the range 1.0×10^{-4} to 1.0×10^{-6}.

`[.mp]` is the Marquardt parameter used to prevent divergence during the NGL/M optimisation step. This value is typically set to 1 or a similar value to `[.delta]`.

`[.mu]` is the convergence tolerance limit calculated from the total sum-of-squares of the weighted residual matrix $\boldsymbol{E}$.

`[.MaxIterations]` is the maximum number of iterations permitted. The NGL/M optimisation will terminate if the convergence tolerance value has not been reached and the number of iterations performed is equal to `MaxIterations`.

Output arguments

`[VAKFISO_output]` is a structured array generated by this script to store the various outputs listed below.

`[.S0]` is a $(K \times N)$ matrix containing the initial estimates of the reference measurement functions. These will either be provided by the user or estimated by performing VVSP analysis.

`[.Sf]` is a $(K \times N)$ matrix containing the final, optimised estimates of the reference measurement functions.

`[.X]`, `[.Pf]`, `[.G]`, `[.V]`, `[.Rk]` and `[.LOF]` are the final outputs from the vectorised adaptive Kalman filter obtained by applying the normalised, final estimates of reference measurement functions to $\boldsymbol{Z}$. These output arguments have been described previously in section 3.1.7.8.

`[.T]` is the final optimised $(N \times N)$ transformation matrix used to create `[.Sf]`.

`[.counter]` is the number of iterations reached before the optimisation step was terminated. Termination may have occurred because the convergence tolerance was achieved or because the maximum number of iterations was reached.

`[.convergence]` is the final convergence value when the optimisation step was terminated.

`[.sigma_t]` is column vector containing the standard error for each of the elements in the final transformation matrix `[.T]`.

`[.VVSP_fw]`, `[.VVSP_fwNorm]`, `[.VVSP_SI]`, `[.VVSP_DW]` and `[.VVSP_PR]` are the outputs from the initial VVSP analysis of `Z` and have been described in section 3.1.7.9.

Overview of `VAKFISO.m`

`VAKFISO` will call two other Matlab scripts during execution. These are called using the feval function and function handles are assigned to improve performance during repeated calls. If user has not provided the options structure **`VAKFISO_options`**, the Matlab function **`VAKFISO_SetOptions`** will be called to provide the default parameter values.

Command line syntax

```
VAKFISO_output = VAKFISO(Z, N, S, VAKFISO_options)
```

1. Assign function handles. This improves performance when functions are called repeatedly.

```
fh_VVSP=@VVSP
fh_VAKF=@VecAdaptiveKF
fh_NGLM=@NGLM
```

2. If user has not provided initial estimates of the reference measurement functions, apply VVSP analysis and extract **`S0`**

```
VVSP_output=feval(fh_VVSP, Z, N+2, N, p_norm, 0)
S0=VVSP_output.Sopt
```

If user has provided initial estimates of the reference measurement functions, assign to **`S0`**

```
S0=S
```

3. Obtain right singular vectors by applying Singular Value Decomposition to **`Z`**

```
[U, S, V]=svd(Z, 'econ')
```

Truncate **`V`** to retain first **`N`** primary singular vectors

```
V_bar=V(:,1:N)
```

4. Calculate initial transformation matrix **`T0`** from normalised matrix of reference measurement functions **`S0`**.

```
T0 = pinv(V_bar) * S0
```

Vectorise the transformation matrix

```
t0 = T0(:)
```

5. Call the sub-function **`NGLM`** using the function handle **`fh_NGLM`** to begin optimisation process.

```
[T, Jn, counter, convergence]= feval(fh_NGLM, Z, t0, V_bar,
VAKFISO_options)
```

See separate description of the **`NGLM`** sub-function.

6. Use final estimate of **`T`** to re-estimate **`Sf`**

```
Sf = V_bar * T
```

7. Loop (increment from **`n=1`** to **`n=N`**, step size **`1`**)

8. Normalise the spectra in **`Sf`** using p-normalisation specified in the input argument **`p_norm`**

```
Sf(:,n)=Sf(:,n) ./ norm(Sf(:,n), p_norm)
```

9. Return to **7** (counter for **`n`**)

10. Call the function **`VAKF`** using the function handle **`fh_VAKF`** to perform Kalman filtering using final estimate of **`S`** to obtain final estimate of **`X`**

```
VAKF_output=feval(fh_VAKF, Sf',Z, W, KF_options)
```

11. Calculate standard errors for the transformation parameters.

Vectorise final transformation matrix

```
t=T(:)
```

Calculate the sum-of-squares SSQ of the final innovations matrix.

```
SSQ=trace(V'*V)
```

Calculate the number of degrees of freedom, **nu**

```
nu=(J*K)-length(t)-(N*K)
```

Calculate the standard deviation of the error in the original data

```
sigma_Z = SSQ ./ nu
```

Calculate the standard error for transformation parameters in **T**

```
sigma_t = sigma_Z * sqrt(diag(inv(Jn'*Jn)))
```

12. Write output variables to **VAKFISO_output**

Overview of NGLM

This sub-function is called from the main **VAKFISO** function and performs the main NGL/M calculations as written in the Matlab example provided by Maeder and Neuhold[22]. This function in turn calls a sub-function called **VAKF_opt** that calculates the reference measurement functions from the vectorised transformation matrix **T**, calls **VAKF.m** to perform the Kalman filtering and calls a sub-function called **CWRM** to calculate the weighted residual matrix.

Command line syntax

```
[T, Jn, counter, convergence]=NGLM(Z, t0, V_bar, VAKFISO_options)
```

1. Assign function handle. This improves performance when functions are called repeatedly.

```
fh_VAKF_opt=@VAKF_opt
```

2. Calculate initial sum-of-squares (**ssq_old**) using data matrix **Z**

```
ssq_old=trace(Z'*Z)
```

3. Initialise vector of transformation elements **t**

```
t=t0
```

4. Pre-allocate storage array for Jacobian matrix **Jn**

```
Jn=zeros((J*K),(N*N))
```

5. Initialise iterations counter and Marquardt parameter

```
counter = 0
```

```
mpp=mp
```

6. Loop until break

7. First call of **VAKF_opt** to calculate vector of residuals **re0** using initial values of **t**

```
re0 = feval(fh_VAKF_opt, Z, V_bar, t, VAKFISO_options)
```

8. Calculate sum-of-squares from **re0** and use sum-of-squares to determine convergence

```
ssq_new=sum(re0.*re0)
```

```
convergence=(ssq_old-ssq_new)/ssq_old
```

9. Determine whether convergence tolerance (**mu**) has been reached.

If **abs(convergence) <= mu** and **mp=0**

Convergence tolerance has been reached and Marquardt parameter is zero so break loop

```
break
```

If **`abs(convergence) <= mu`** and **`mp≠0`**

Convergence tolerance has been reached but Marquardt parameter is not zero so set **`mp`** to 0 and confirm convergence by performing another iteration.

```
mp=0
re0_old=re0
```

10. If convergence is greater than convergence tolerance, reduce value of Marquardt parameter and estimate new values of the transformation vector **`t`** (steps **11** to **16**)

If **`convergence > mu`**

```
mp=mp/3
ssq_old=ssq_new
re0_old=re0
```

11. Loop to update each element of the transformation vector **`t`**. This is slice-wise numerical differentiation to create the Jacobian matrix **`Jn`**

Loop (increment from **`q=1`** to **`q=N`**^{2}, step size **`1`**)

12. Add delta to current value of **`t(q)`**

```
t(q)=(1+delta)*t(q)
```

13. Calculate residuals for shifted element **`t(q)`** by calling sub-function **`VAKF`**

```
re=feval(fh_VAKF_opt, Z, V_bar, t, VAKFISO_options)
```

14. Populate the **`q`**-th column of the Jacobian matrix **`Jn`**

```
Jn(:,q) = (re - re0)/(delta*t(q))
```

15. Calculate new shift value for element **`t(q)`**

```
t(q) = t(q)/(1 + delta)
```

16. Return to **11** (counter for **`q`**)

17. If convergence is not detected in steps **9** and **10**, determine whether divergence is occurring.

If **`convergence<-mu`** and **`mp=0`**

Divergence is observed and the Marquardt parameter is set to 0 so set the parameter back to mpp and perform another iteration.

```
mp=mpp
```

If **`convergence<-mu`** and **`mp≠0`**

Divergence is observed but the Marquardt parameter is not set to 0 so increase the value of the parameter and perform an other iteration.

```
mp=mp*5
```

If divergence is observed (irrespective of value of **`mp`**), take back the shifts added previously.

```
t=t-delta_t
```

18. Now Marquardt parameter has been appropriately set according to the observation of convergence or divergence, perform the main calculations to update the elements of **`t`**

Augment Jacobian matrix with diagonal matrix of Marquardt parameters

```
Jn_mp=[Jn; mp*eye(length(t))]
```

19. Augment residual vector with a vector of zeros

```
re0_mp=[re0_old; zeros(size(t))]
```

20. Calculate parameter shifts for every element in the transformation vector **`t`**

```
delta_t=-Jn_mp \ re0_mp
```

21. Add the transformation vector parameter shifts to current estimate of **t**

```
t=t+delta_t
```

22. Increment iteration counter

```
counter = counter + 1
```

23. If **counter** is equal to or greater than the number of iterations, break loop

if **counter >= MaxIterations**

```
break
```

24. When the elements of **t** have been optimised, or the maximum permitted number of iterations is reached, the vector of transformation elements are re-matricised to give **T**

Initialise (**N × N**) matrix of zeros **T**

```
T=zeros(N)
```

25. Loop (increment from **n=1** to **n=N**, step size **1**)

26. `T(:,n) = t((n*N)-(N-1):(n*N))`

26. Return to **25** (counter for **n**)

27. End of sub-function **NGLM**

Overview of `VAKF_opt`

This sub-function is called from the sub-function **`NGLM`** and calculates the reference measurement functions from the vectorised transformation matrix **`T`**, calls **`VAKF.m`** to perform the Kalman filtering and calls a sub-function called **`CWRM`** to calculate the weighted residual matrix **`E`**.

Command line syntax

```
re = VAKF_opt(Z, V_bar, t, VAKFISO_options)
```

1. Assign function handles. This improves performance when functions are called repeatedly.

```
fh_VAKF=@VecAdaptiveKF
fh_CWRM=@CWRM
```

2. Extract relevant options values from options structure **`VAKFISO_options`**

3. Re-matricise the vector of transformation elements to give **`T`**

Initialise (**`N × N`**) matrix of zeros **`T`**

```
T=zeros(N)
```

4. Loop (increment from **`n=1`** to **`n=N`**, step size **`1`**)

5. `T(:,n) = t((n*N)-(N-1):(n*N))`

6. Return to **4** (counter for **`n`**)

7. Use current estimate of transformation matrix to calculate **`S_hat`**

```
S_hat = V_bar * T
```

8. Loop (increment from **`n=1`** to **`n=N`**, step size **`1`**)

9. Normalise the spectra in **`S_hat`** using p-normalisation specified in the input argument **`p_norm`**

```
S_hat(:,n)=S_hat(:,n) ./ norm(S_hat(:,n), p_norm)
```

10. Return to **8** (counter for **`n`**)

11. Apply Vectorised Adaptive Kalman filtering to **`Z`** using current estimate of **`S_hat`**

```
VAKF_output=feval(fh_VAKF, S_hat', Z,  W, KF_options)
```

12. Extract **`X`**, **`V`** and **`Pf`** from the Kalman filter output structured array

```
X=VAKF_output.X
V=VAKF_output.V
Pf=VAKF_output.Pf
```

13. Call the sub-function **`CWRM`** using the function handle **`fh_CWRM`** to calculate the weighted residual matrix **`E`**. The matrix **`E`** is then vectorised (stacked) to produce **`re`**

```
re = feval(fh_CWRM, S_hat, VAKF_output, alpha1, alpha2, alpha3,
                                                        alpha4)
```

14. End of sub-function **`VAKF_opt`**

Overview of `CWRM`

This sub-function is called from the sub-function **`VAKF_opt`** and calculates the weighted residual matrix **`E`**.

Command line syntax

```
re = CWRM(S_hat, VAKF_output, alpha1, alpha2, alpha3, alpha4)
```

1. Assign function handles. This improves performance when functions are called repeatedly.

```
fh_VAKF=@VecAdaptiveKF
fh_CWRM=@CWRM
```

2. Extract relevant options values from Kalman filter output structure **`VAKF_output`**

```
X=VAKF_output.X
V=VAKF_output.V
Pf=VAKF_output.Pf
```

3. Calculate **`PI`** (Π)

```
PI=trace(Pf)
```

4. Calculate **`SIGMA`** (Σ)

```
OMEGA=(abs(S_hat)-S_hat)./2
SIGMA = (trace(OMEGA'*OMEGA))./ (trace(S_hat'*S_hat))
SIGMA = SIGMA ./ N
```

5. Calculate **`XI`** (Ξ)

```
THETA=(abs(X)-X)./2
XI = (trace(THETA'*THETA))./ (trace(X'*X))
XI = XI ./ N
```

6. Calculate weighted residual matrix **`E`**

```
E = (alpha1.*V) + (alpha2.*PI.*V) + (alpha3.*SIGMA.*V) +
                                                (alpha4.*XI.*V)
```

7. Vectorise **`E`** for NGL/M calculations

```
re=E(:)
```

8. End of sub-function **`CWRM`**

3.2 Combining Self-Modelling Curve Resolution and PLS regression

3.2.1 Equipment

The equipment used throughout this work is listed below for convenience and is described in more detail in section 3.1.

Spectrometers and probes

Zeiss MCS501 single-beam UV/Vis diode array spectrophotometer

Zeiss MCS601 double-beam UV/Vis diode array spectrophotometer (used for comparison of standard and custom fibre assemblies).

Hellma UV/Vis ATR Probe (diameter: 12.7 mm; length: 400 mm)

Hellma Process UV / Vis ATR probe (diameter: 25.0 mm; length 850 mm)

Standard fibre-optic cables: Hellma UV/Vis high -OH, solarisation resistant fibre optic cables with 600 micrometer cores. Length: 4.00 m; terminated with standard SMA-905 connectors.

Custom fibre-optic cables: Ocean optics (Ocean Optics B.V., EW Duiven, Netherlands) custom probe design (ZDF-10369).

Figure 3.5 shows the configuration and dimensions of the fibre assembly.

3.2.2 Laboratory scale development experiments

Aim

The purpose of the initial experiments was to assess the feasibility of using UV/ATR spectroscopy to monitor the chlorination reaction described in section 1.8.1. Although other spectroscopic techniques such as FTIR, Raman or NIR were available, there were a number of practical issues that precluded their use for this reaction. Molecular fluorescence prevented the use of Raman spectroscopy and it was not possible to monitor the whole reaction using NIR spectroscopy because the process started as heterogeneous slurry and became a homogenous slurry. This would require both a diffuse reflectance probe and transmission probe combined with a dual channel NIR spectrophotometer (not available). Although it was possible to monitor the laboratory scale reactions using

FTIR/ATR, the corresponding process probe and instrument were not available for use in the LSL.

Experimental

The initial laboratory reactions were performed in a 100 mL jacketed glass vessel. Prior to charging any materials to the reaction vessel, an air background spectrum was acquired at room temperature. The reaction was prepared by charging 5.00 g (1.0 equivalent) of the starting reactant (acetoxyone) to the reaction vessel, followed by 70 mL toluene (14 relative volumes) and di-isopropylethylamine (3.0 g, 1.10 equivalents). The mixture was then heated to 70°C. When the set point temperature had been reached, the reaction mixture was a thick but mobile slurry owing to the very low solubility of the acetoxyone starting material. The reaction was then started by charging phosphorus oxychloride (8.30 g, 2.50 equivalents) using a syringe driver over 20 minutes. At the end of the addition, the reaction mixture was still a heterogeneous slurry owing to un-dissolved starting material. After approximately 2 hours, the starting material had completely dissolved to give a dark homogeneous solution.

Throughout the course of the experiment, several attempts were made to sample the mixture for analysis by HPLC. Unfortunately, the reaction mixture was supersaturated and began to precipitate immediately as the sample cooled in the sampling pipette.

3.2.3 Key reference measurements

Aim

Although initial examination of the UV/ATR reaction data suggested that the reaction progress could monitored spectroscopically, there were a number of practical issues that would preclude attempts to obtain reliable reference measurements for subsequent development of a calibration model. To aid the resolution of the spectral data using SMCR methods, and to provide scaling factors for the recovered profiles, a few experiments were performed to provide key reference measurements.

Experimental

The solubility of the acetoxyone starting material was established by adding small amounts of the material to 100 mL of a toluene / di-isopropylethylamine solution heated to 70°C. When un-dissolved solid was observed in the bottom of the reaction vessel, the

supernatant liquid was sampled and assayed against a standard of known concentration using reversed-phase HPLC.

A 'calibration' reaction was then performed by repeating the experiment described in section 3.2.2. At the end of the reaction, when all starting material had been consumed and the mixture was homogenous, a 10.0 mL sample was extracted using a pipette and diluted immediately for off-line analysis. The concentration of product (haloacetoxyone) was determined by solution assay against a standard of known concentration using reversed-phase HPLC. The extent of reaction based upon the ratio of product to starting material was also determined using HPLC.

3.2.4 Overview of the spectral data

Aim

The structure and features of the spectral data were first examined by visual inspection, simple peak profiling and PCA. The aim was to assess the number of individual components contributing to the data and their corresponding kinetic (time-series) profiles.

Method

The spectra were exported from the Zeiss Aspect Plus software as a comma separated variable (`*.csv`) and imported into Matlab for further data analysis.

The original absorbance spectra were transformed to their second derivative spectra using Savitsky-Golay smoothing and differentiation (Eigenvector PLS toolbox, `savgol` function). The parameters used to calculate the second-order derivative were a 13-point smoothing window and second-order polynomial.

Both the original absorbance spectra and second derivative spectra were analysed using principal components analysis (PCA) to identify the number of significant, independent factors contributing to the data sets. The function `pca` from the Eigenvector PLS toolbox was used.

To examine how the structure of the re-constructed and corresponding residual spectral changes as the data was regenerated using an increasing number of principal components, a simple custom Matlab function (`ResidualComps`) was written (section 3.1.7.1). Visual examination of the residual spectra obtained by reconstructing the data set using different numbers of principal components was a useful way to aid the identification of the correct

number of factors without over fitting. This function was applied to the second derivative data (265 to 350 nm).

3.2.5 Derivation of initial concentration profile estimates using EFA

Aim

The initial analysis of the data acquired from the calibration reaction indicated that there were three to four significant components contributing to the total variance in the data set. Evolving Factor Analysis was applied to provide initial estimates of the true underlying concentration profiles of the major species.

Method

To provide an estimate of the concentration profiles of the spectroscopically active major species, Evolving Factor Analysis was applied to the truncated, second derivative spectra. The spectral data were not mean-centred prior to applying evolving factor analysis.

The combined (forward and backward) EFA estimates of the concentration profiles were calculated for two, three and four components.

3.2.6 Derivation of initial spectral profile estimates using OPA

Aim

In addition to the concentration profile estimates obtained using EFA, the Orthogonal Projection Approach was also applied to second derivative UV data to obtain initial estimates of the pure component spectra for the intermediate and product species.

Method

OPA is a very useful technique for identifying and visualising each different spectrum contributing to a data set. The method is based upon a dissimilarity measurement and simply identifies and stores the spectrum that is most dissimilar to the other reference spectra already identified. This procedure is repeated until the number of components requested by the user is reached.

A custom Matlab function, `OPA`, was written based upon the algorithm published by Sanchez *et al.*[89, 99] and is described in section 3.1.7.2. The Durbin-Watson values used to test for correlation in the dissimilarity vectors for each OPA component were calculated as described by Gourvénec *et al.*[100].

As OPA is not a variance or eigenvector based method, it can be very sensitive to peak shifts that only account for a small fraction of the total variance in the data. In the previous sections, the data were analysed using PCA and EFA and both methods indicated that more components than the expected chemical rank were necessary to model the spectral data. To confirm whether this is a consequence of peak movement, OPA was applied to the truncated (265 to 350 nm), second derivative spectra with a maximum of 8 components sought.

3.2.7 Refinement of profiles using MCR-ALS

Aim

The initial estimates of the concentration and spectral matrices obtained using EFA and OPA respectively, were further refined using MCR-ALS to provide the reference concentration data required to construct a PLS model. MCR-ALS was performed using the MCR-ALS graphical user interface, a Matlab toolbox developed by Jaumot *et al.*[94] that can be freely downloaded from the internet[102].

Method

The concentration profiles were constrained to non-negative solutions and a concentration selectivity matrix was used to include the starting reactant and product concentration information provided by the reference analysis (HPLC solution assay). A concentration selectivity matrix was also used to constrain the concentration of the product to zero during the very early part of the reaction. A spectral selectivity matrix was used to constrain the scaled starting reactant / toluene and product spectra as obtained from OPA. The spectrum for the intermediate species was not initially constrained. In this particular application the truncated second-derivative spectra were used, but in those cases where the original un-transformed spectra are used, the spectral non-negativity constraint could also be applied.

3.2.8 Calculation of PLS models using refined concentration profiles

Aim

Using curve resolution methods, the unit concentration pure component spectra had been estimated and refined. However it was not possible to incorporate these spectra directly

into a ProcessXplorer[1] method as a standard model for predicting the concentrations of future reactions (for example using CLS). The ProcessXplorer software was compatible with PLS or PCR models created using GRAMS PLSplus/IQ chemometric software so the concentration profiles derived using SMCR methods were used to construct a PLS model. Using a standard PLS model allowed predictions to be made in real-time as GRAMS PLSplus/IQ chemometric models could be imported and used by the instrument software.

Method

The original un-processed reaction spectra acquired during the calibration reaction and the corresponding MCR-ALS optimised concentration vectors for the intermediate and haloacetoxyone species were imported into GRAMS PLSplus/IQ chemometric software. The spectra were pre-processed as described previously (265 to 350 nm, transformation to Savitsky-Golay second derivative spectra using 13-point smoothing window). Both the X (spectral) and Y (concentrations) matrices were mean-centred.

A series of PLS1 and PLS2 models were constructed using leave-one-out (LOO) cross validation and also leave out 50 spectra with odd-even split cross-validation. This was equivalent to removing approximately 25% of the spectra prior to calculating the PLS models and therefore produced models with larger cross-validation statistics than leave-one-out cross validation.

The resulting Root Mean Standard Error of Cross Validation (RMSECV) and cumulative variance explained statistics were then examined to select the most suitable model type and number of factors (principal components) to use.

3.2.9 Comparison of PLS regression coefficients and MCR-ALS spectra

Aim

In addition to the RMSECV statistics, the regression spectra $\hat{\boldsymbol{B}}$ of the various PLS models were examined. This was particularly important in this application as the reference data used to construct the PLS models were derived using SMCR methods. The SMCR methods produced not only the concentration profiles but also a set of complimentary pure

[1] ProcessXplorer is the process monitoring software used by Carl Zeiss process spectrometers such as the MCS500 and MCS600 series UV diode array spectrometers used during this work.

component spectra. These pure component spectra revealed the characteristic absorbance features of each constituent. If the PLS model was a good transformation of the pure component spectra into a set of regression spectra, the regression spectrum for each constituent should be similar to the corresponding pure component spectrum.

Method

The vector or matrix of regression coefficients $\hat{\boldsymbol{B}}$ for each PLS1 and PLS2 model were imported into Matlab. The least-squares method described by Trygg[103] for calculating the pure spectral profiles from the PLS regression vectors was employed. These were calculated using the simple expression $\hat{\boldsymbol{K}} = \hat{\boldsymbol{B}}(\hat{\boldsymbol{B}}^T\hat{\boldsymbol{B}})^{-1}$ where $\hat{\boldsymbol{K}}$ is a matrix of pure spectral profile estimates and $\hat{\boldsymbol{B}}$ is the vector or matrix of PLS regression coefficients. The PLS regression vectors and the subsequent pure spectral profiles estimates contained scaling information so they could be directly compared with the MCR-ALS spectra without normalisation. The root mean square sum of errors (RMSE) for the residual spectra were also calculated to quantify the difference between the MCR-ALS and PLS pure spectral profiles.

3.2.10 Application of the UV method in a Large Scale Laboratory

The instrument used during method development was transferred to the large scale laboratory facility and coupled to an 850 mm process UV ATR probe via two 4.00 m lengths of fibre-optic cable terminated with standard SMA905 connectors. The probe was installed using a customised flange fitted to the charging port of the vessel lid. The photograph[2] in Figure 3.3 shows the 50 L reaction vessel and the probe inserted through the man-way used to charge solids to vessel. This configuration was problematic as it was not possible to acquire a background spectrum with the probe installed. As the spectrometer was a single-beam instrument using a deuterium source lamp, a new background spectrum was required prior to starting each batch. For the first batch, it was possible to collect a spectrum with the probe installed on the vessel lid and the ATR crystal positioned in the reactor headspace. In subsequent batches, the background spectra had to be acquired prior to re-installation of the probe.

---

[2] Paul Rowan and Simon Watkins from Process R&D, AstraZeneca R&D Charnwood, Loughborough, UK are gratefully acknowledged for the preparation of the vessel for this photograph.

Figure 3.3: A 50 L reaction vessel a Large Scale Laboratory (LSL) facility. This vessel was used to manufacture five batches of haloacetoxyone.

3.2.11 Design of a custom fibre optic cable assembly

When the UV/ATR probe was coupled to a single beam spectrometer using a pair of standard fibre optic cables, the light from the source lamp passed along one fibre optic cable, through the probe (where it interacted with the sample) and returned along a second fibre optic cable to the detector module. The process spectrometer with a double beam configuration had two detectors. The additional 'bypass' channel was used to actively compensate for flash to flash intensity variation when a xenon flash lamp was used. Figure 3.4 shows the standard configuration of the double beam spectrometer.

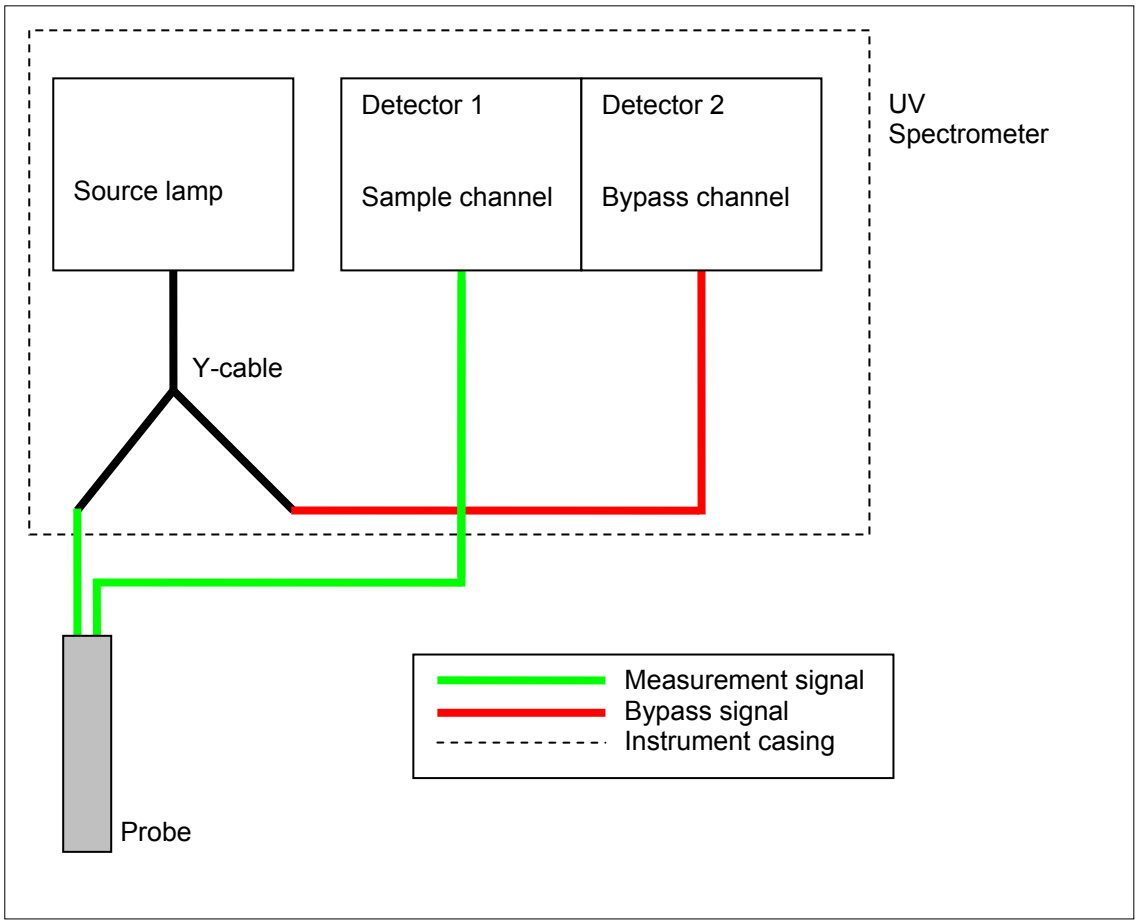


Figure 3.4: Diagram showing standard configuration for a double beam process spectrometer. The light from the source lamp is split using a Y-cable. The fibre optic cable for the sample channel transmits light to the probe. A second cable then transmits the return signal and is connected to detector 1. The fibre optic cable for the bypass channel forms a closed loop and is connected directly to detector 2.

The main consideration when designing the custom fibre assembly was to utilise the second detector channel to correct for the change in transmission through the fibre owing to movement. To achieve this, the fibre that transmitted the signal for the bypass channel was required to follow a path as similar as possible to the sample signal without interacting with the sample. Using a single fibre for the by-pass channel in a looped configuration was not feasible because the minimum bend radius of the fibre would restrict the minimum diameter of the loop to several centimetres. It was not possible to contain this bypass loop within the fibre assembly so it would not be very robust and would also be sensitive to movement. To overcome this issue, a mirror was used to re-direct the by-pass channel. This allowed the light transmitted through the bypass fibre to be reflected 180º in a very short distance without using a loop of fibre. The final design of the fibre assembly constructed is shown in Figure 3.5.

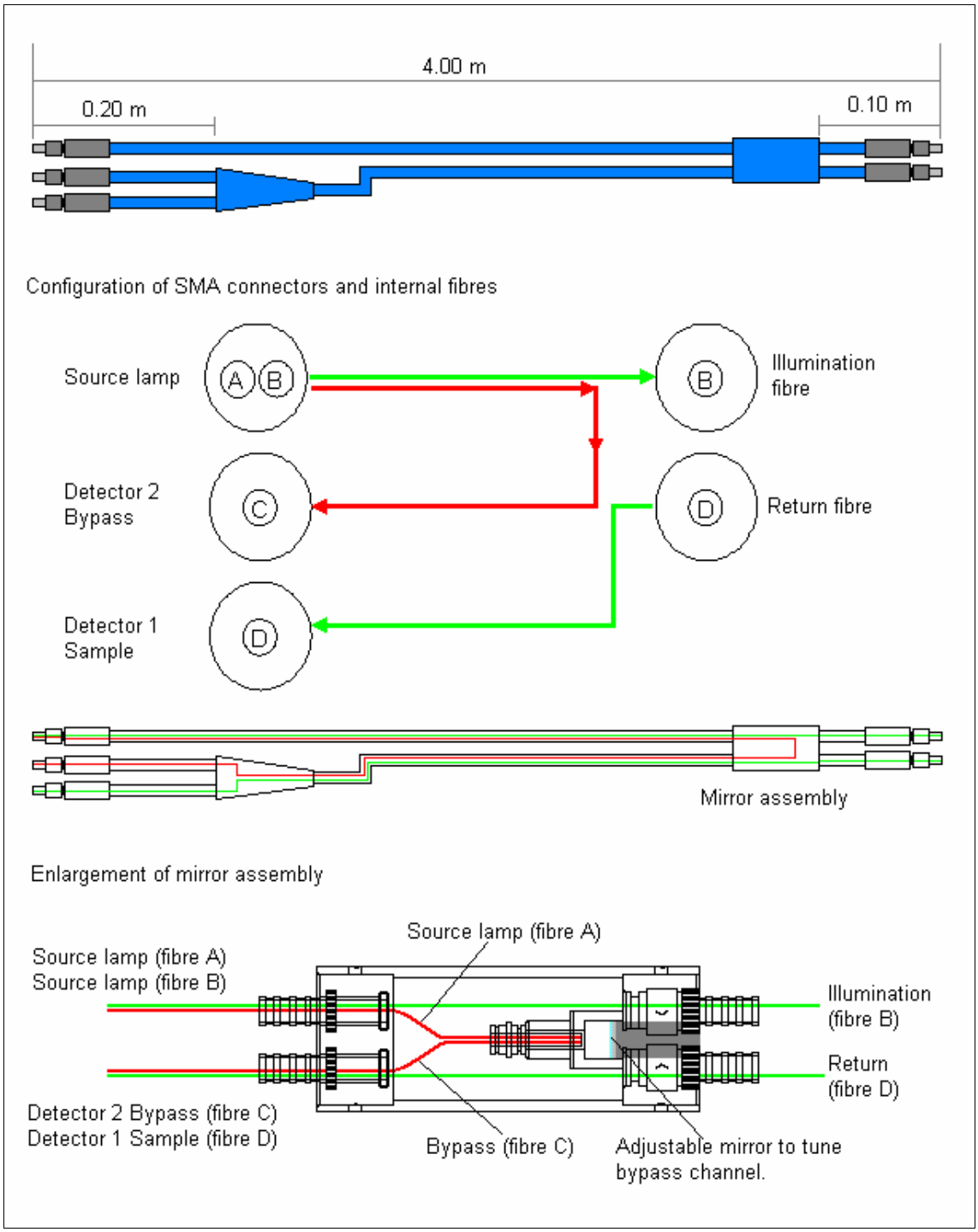


Figure 3.5: Diagram showing the external and internal configurations of the custom fibre assembly.

3.2.12 Testing of a custom fibre optic assembly

3.2.12.1 Comparison of the relative transmission of the standard and custom fibre optic cables

Aim

The purpose of this experiment was to establish whether there was a significant difference in the sensitivity of the two detector modules in the instrument. This was achieved by measuring a single beam energy spectrum through a closed loop using the same fibre optic cable and collection parameters. By repeating the procedure with each fibre leg of the standard and custom cables, the relative transmission of each set of cables was also assessed.

Configuration of instrument and fibres

The output from the deuterium source lamp was passed through a 50% attenuation filter coupled to a short (30 cm) SMA-SMA patch lead. The test fibre was then attached to the attenuation filter, passed over a bracket positioned 1.96 m above the floor and connected to the detector to form a closed loop.

Method

A quick test of each set of fibres established that using the instruments minimum integration time (12 ms) produced spectra with a maximum intensity between 25% and 75% of the detectors working range. Using the same integration time allowed direct comparison of all the measured single beam energy spectra. The single beam spectra of each leg of the standard and custom fibre assemblies were acquired on each detector in turn.

3.2.12.2 Optimisation of the internal reflection mirror position

Aim

The custom fibre assembly used an adjustable mirror to return the source light from the lamp back along the return fibre to bypass channel. Since the bypass fibre and mirror assembly did not have a collimating lens, the intensity of the light returned along the bypass channel varied as the distance of the mirror from the fibres was changed. The aim was to match the throughput of the sample and bypass channels as closely as possible by tuning the position of the mirror.

Configuration of instrument and fibres

As the process ATR probe further attenuated the optical signal transmitted through the sample chain, the best approach was to tune the bypass channel to give a similar throughput to the sample channel. The equipment was configured as follows

The instrument end of the cable containing both the probe illumination fibre A and the bypass fibre B was connected directly to the source lamp (no attenuation filter required). The probe end of fibre B was connected directly to the process ATR probe and the return fibre D connected to the detector 1 (sample). The bypass return fibre C was connected to detector 2 (bypass).

Method

The instrument was configured as a double beam instrument and the energy monitor function was used to tune the energy throughput of the bypass channel by changing the position of the adjustable mirror.

The best achievable throughput of the bypass channel relative to the sample channel passing through the ATR probe was approximately 10%. With a 12 ms exposure, the sample channel maximum signal was approximately 35000 counts and the bypass channel maximum signal was approximately 3000 counts. The robustness tests were therefore performed using unmatched channels.

3.2.12.3 Quantification of fibre transmission as a function of cable displacement

Aim

The aim of this experiment was to apply decreasingly smaller bend radii to the fibre optic cable assemblies by increasing the vertical displacement from the ground at their centre-point. This allowed the change in optical throughput to be measured as a function of the displacement from the nominal starting position for each set of fibres.

Configuration of instrument and fibres: standard fibre assembly

The common leg of a bifurcated cable was connected directly to the source lamp. One leg of the bifurcated cable was passed through a 10% attenuator and then connected to detector 2 (bypass channel) using a 35 cm SMA-SMA patch lead. The second leg of the bifurcated cable was coupled with one leg of the standard fibre using a 35 cm SMA-SMA patch lead and a SMA-SMA union. The remaining leg of the standard fibre pair was

connected to directly to detector 1 (sample channel). The probe ends of the standard fibre assembly were connected to the process ATR probe.

The standard fibre assembly was marked 90 cm from the instrument end terminations. The fibres connected to the instrument were clamped at the 90 cm mark at a height of 8 cm above floor level (the lowest possible height possible with the clamp stand). The instrument was then positioned 50 cm from the clamp (measured from instrument front face to clamp). This relieved the fibres from any tension and prevented this 90 cm section from moving when adjusting the position of the probe.

The fibre was laid across the floor in a straight line and attached to the probe. The probe was clamped in a horizontal position 8 cm above the ground. The length of fibre between the 90 cm marker and the probe was 2.90 m. The centre point of this section of fibre was marked (1.45 m from 90 cm marker). This was the point at which the fibre was raised above the ground to give reproducible displacement.

The throughput through this configuration was slightly lower than observed for the custom fibre so the integration time was increased to give comparable maximum throughput (maximum signal ~53300 counts). This difference was a consequence of using additional attenuators and SMA-SMA patch leads which all introduced additional optical losses. An integration time of 100 ms gave an equivalent maximum throughput.

Configuration of instrument and fibres: custom fibre assembly

The instrument and custom fibre assembly were configured as described in section 3.2.12. The fibres were then marked, clamped and positioned as described in the previous section. The integration time was set to 20 ms. A lower integration time was required for the custom fibre assembly compared to the standard fibre assembly because no attenuation filter was used.

Procedure for the reproducible displacement of the fibres

With the cables set out as described in the previous sections, the energy spectra were measured on both detectors simultaneously. This starting configuration represented the highest transmission of the fibres as they had almost no bends in the fibre optic cable.

The cables were then raised above the ground in 20 cm increments using an adjustable bracket until a maximum displacement of 120 cm was reached. As the fibres were raised, it was necessary to move both the bracket at the centre point and the probe closer to the

instrument to maintain a symmetrical bend around the bracket. At each position, the double beam energy spectra were recorded. When the maximum displacement was reached, the fibres and probes were returned to the starting position and the procedure repeated a further three times to give four replicates at each position.

3.2.12.4 The effect of fibre movement upon the calculated absorbance

Aim

To examine the contribution of baseline variation that solely arose from fibre movement, the double beam energy spectra were transformed into absorbance spectra.

Method

In many double beam spectrometers such as the Zeiss MCS series, the absorbance value is calculated as shown in equation 3.1:

$$\boldsymbol{a}_j = \log_{10}\left(\frac{r_j.\boldsymbol{s}_0}{\boldsymbol{s}_j}\right), \quad r_j = \frac{\|\boldsymbol{b}_j\|_2}{\|\boldsymbol{b}_0\|_2} \qquad \text{(equation 3.1)}$$

$\boldsymbol{a}_j$ is the vector of absorbance values for sample j

r_j is the ratio of the sample and reference bypass energy spectra (using the 2-norm)

$\boldsymbol{b}_0$ is the energy spectrum for the bypass channel of the reference measurement

$\boldsymbol{b}_j$ is the energy spectrum for the bypass channel of the sample measurement j

$\boldsymbol{s}_0$ is the energy spectrum for the sample channel of the reference measurement

$\boldsymbol{s}_j$ is the energy spectrum for the sample channel of the sample measurement j

The 2-norms of the bypass channel spectra are used to calculate a simple scalar ratio to correct for intensity variation. The original reference spectrum $\boldsymbol{s}_0$ is multiplied by this factor prior to calculating absorbance in the usual way. Another approach is to use an element by element division to calculate ratio spectra ($\boldsymbol{s}_0 / \boldsymbol{b}_0$) and ($\boldsymbol{s}_j / \boldsymbol{b}_j$). The absorbance spectrum is then calculated from these ratio spectra. The energy spectra for each set of fibres were converted to absorbance spectra using the first method.

3.2.12.5 Quantification of the effect of fibre movement upon CLS calculations

Aim

In the previous section, the single beam energy spectra acquired using the standard and custom fibre assemblies were transformed into absorbance spectra. The results showed that both cable assemblies had variable baselines as the fibres were displaced but the variation of the custom fibre was lower. To put this difference in variation into context, the aim of this experiment was to examine the effect of the baseline movement upon estimated concentration profiles calculated from a synthetic data set using CLS.

Method

A two-component synthetic reaction data set based upon the chlorination of acetoxyone was created. The concentration vectors for the intermediate and haloacetoxyone species, derived from the laboratory scale experiments using SMCR (described in section 3.2.7) were used as the initial profiles. The concentration profiles were smoothed by applying Savitsky Golay filtering several times (`sgolayfilt.m`, Matlab Signal Processing Toolbox). A first order polynomial and five point window were used. The smoothed concentration profiles were truncated between 20 and 120 minutes to give a (100×2) matrix, $\boldsymbol{C}_{sim}$.

The original (zero-order) spectra at 45 and 170 minutes, the time-points when the intermediate species and product were at their maximum concentrations were used as 'pure' component spectra. The spectra were truncated to 220 to 400nm and scaled to unit molarity by dividing each spectrum by 0.27 mol.L^{-1} to give a (181×2) matrix, $\boldsymbol{S}_{sim}$. The simulated reaction data set was created by calculating the outer product of the concentration and spectral matrices ($\boldsymbol{X}_{sim} = \boldsymbol{C}_{sim} \cdot \boldsymbol{S}_{sim}^{T}$).

The absorbance baselines calculated for each cable assembly in section 3.2.12.4 were replicated to give two, (100×181) matrices, $\boldsymbol{B}_{std}$ and $\boldsymbol{B}_{cust}$. The rows (spectra) of the baseline matrices were randomly re-ordered and then added to the simulated reaction data to produce $\boldsymbol{X}_{std}$ and $\boldsymbol{X}_{cust}$.

The concentration profiles were then estimated from each data set using the least-squares calculation shown in equation 3.2.

$$\hat{\boldsymbol{C}} = \boldsymbol{X} \cdot \boldsymbol{S} \cdot (\boldsymbol{S}^T \cdot \boldsymbol{S})^{-1} \qquad \text{(equation 3.2)}$$

The concentration profiles were also estimated from the second derivative spectra by calculation of the Savitsky-Golay second derivative spectra using the simulated spectral data sets $\boldsymbol{X}_{std}$ and $\boldsymbol{X}_{cust.}$ and the pure component spectra, $\boldsymbol{S}_{sim}$.

3.3 Vectorised Kalman filtering for SMCR

3.3.1 *N*-benzylation of 1*H*-indole

Aim

A series of reactions were performed to provide spectral data for algorithm development and testing. Off-line reference measurements were provided by HPLC analysis of reaction samples extracted during the course the reaction.

Method

The nominal reaction conditions for the *N*-benzylation of 1*H*-indole reaction (described in section 1.8.2) were based upon the material stoichiometries used in a similar reaction developed by AstraZeneca Process R&D and are summarised in Table 3.3.

Table 3.3: Nominal stoichiometry of materials for the *N*-benzylation of 1*H*-indole and amounts required for a 100 mL reaction.

Material	Molecular weight ($g.mol^{-1}$)	Molar equivalent	Relative volumes	Weight (g)	Volume (mL)
1*H*-indole	117.15	1.00	-	6.00	-
Benzyl bromide	171.04	1.00	-	8.76	-
Cesium carbonate	325.82	2.00	-	33.37	-
Tetrabutylammonium bromide	322.28	0.10	-	1.65	-
Acetonitrile	41.05	-	15.0	-	90.0

The experimental parameters for the series of reactions performed are listed in Table 3.4. To aid the reader, each experiment has been given a unique name that comprises of the reaction name and the major parameters that distinguish each reaction. The nomenclature used to name each experiment is BnIndole _Byy_MRzz_D where the prefix BnIndole refers to the series of experiments (the *N*-benzylation of 1*H*-indole); the suffix Byy refers to the amount of base relative to the nominal amount 1*H*-indole (yy denotes the molar equivalent of base); the suffix MRzz is the molar ratio of 1*H*-indole to benzyl bromide (zz is the molar ratio) and the suffix D is used to indicate the data acquired during the experiment (HPLC, Raman or UV).

Table 3.4: List of *N*-benzylation of 1*H*-indole reactions performed.

Experiment number[a]	Experiment name	Amount of base		Amount of 1H-indole and benzyl bromide		
		Molar Eq.	Weight (g)	Molar ratio (1H-indole : Benzyl bromide	Weight 1H-indole (g)	Weight benzyl bromide (g)
1	BnIndole_B2.0_MR0.67	2.00	33.37	0.67	6.00	13.13
2	BnIndole_B2.0_MR1.00	2.00	33.37	1.00	6.00	8.76
3	BnIndole_B2.0_MR1.50	2.00	33.37	1.50	9.00	8.76

The reactions were performed in a 100 mL jacketed reaction vessel described previously in section 3.1. Prior to starting each reaction, the vessel was cleaned and purged overnight with nitrogen to ensure it was dry. The reaction vessel jacket temperature set to 40ºC. A UV background spectrum was acquired with the ATR probe positioned in the nitrogen atmosphere of the vessel. The required amount of 1*H*-indole was accurately weighed into a tared 100 mL amber bottle. To this bottle, 80.0 mL of acetonitrile was charged and the bottle was shaken until the 1*H*-indole was completely dissolved. The required amount of benzyl bromide was accurately weighed into a tared 50 mL amber bottle, followed by 10.0 mL acetonitrile. Cesium carbonate and tetrabutylammonium bromide were accurately weighed into a tared weigh boat and transferred directly to the reaction vessel. The 1*H*-indole solution was quantitatively transferred to the reaction vessel, the overhead agitator set to 600 rpm and spectroscopic data acquisition was started. The mixture was left for sixty minutes to ensure that the tetrabutylammonium bromide was fully dissolved and that the cesium carbonate was completely wetted. After sixty minutes, the benzyl bromide solution was charged rapidly through the PTFE addition line using a syringe. This would initiate the reaction. Reaction samples were extracted 10, 20, 30, 40, 50, 60, 90, 120, 150, 180, 240, 300 and 360 minutes after the addition of the benzyl bromide solution. At the end of the experiment, 90 mL water was charged to quench the reaction.

3.3.2 Analysis of reaction samples using HPLC

3.3.2.1 HPLC solution assay method

A HPLC assay method was developed to allow the concentration of 1*H*-indole, benzyl bromide and 1-benzyl-1*H*-indole to be determined. Reaction samples were analysed off-line and the concentrations determined using HPLC provided useful reference data that were compared with the reaction profiles estimated from spectroscopic data.

An Agilent 1100 Series HPLC system with a quaternary solvent pump and variable wavelength detection was used to analyse the reaction samples. The HPLC method parameters used for the analysis of reaction samples are listed below:

Mobile phase A: 0.10% trifluoroacetic acid (TFA) in water

Mobile phase B: 0.08% trifluoroacetic acid (TFA) in 90/10 v/v acetonitrile /water

Sample diluent: 50/50 v/v acetonitrile / water

Column: Jones Genesis C18 (10 cm × 4.6 mm × 3 μm)

Injection volume: 2 μL

Column compartment oven temperature: 40°C

Flow rate: 0.750 $mL.min^{-1}$

Detector wavelength: 222 nm (bandwidth 7nm)

Mobile phase gradient

Time (minutes)	**%B**
0.00	20.0
2.00	20.0
16.00	100.0
20.0	100.0

3.3.2.2 Preparation of HPLC solution assay standards

To allow the consumption of starting materials and the formation of product to be quantified offline by solution assay, standards of 1*H*-indole, benzyl bromide and 1-benzyl-*H*-indole were prepared using the weights and dilutions summarised in Table 3.5 and Table 3.6. Stock solutions of each compound were prepared in 100 mL volumetric flasks. The diluent was 50:50 v/v acetonitrile / water. The assay standards were then prepared by quantitative dilution of the stock standards, again using 50:50 v/v acetonitrile / water as the diluent.

Table 3.5: Sample weights and dilution volumes used to prepare solution assay stock standards.

Standard	Standard weight (mg)	Dilution volume (mL)	Dilution weight (g)	Stock standard conc. ($mg.mL^{-1}$)	Stock standard conc. ($mg.g^{-1}$)
1*H*-indole	100.5	100.0	78.3557	1.005	1.283
Benzyl bromide	210.3	100.0	78.6731	2.103	2.673
1-benzyl-1*H*- indole	123.6	100.0	78.4308	1.236	1.576

Table 3.6: Sample weights, dilution volumes and final concentrations of solution assay standards.

Standard	Aliquot volume (mL)	Aliquot weight (g)	Dilution volume (mL)	Dilution weight (g)	Assay standard conc. ($mg.mL^{-1}$)	Assay standard conc. ($mg.g^{-1}$)
1*H*-indole	10.0	7.8266	100.0	78.2628	0.1005	0.1283
Benzyl bromide	10.0	7.8574	100.0	78.2628	0.2103	0.2684
1-benzyl-1*H*-indole	10.0	7.8342	100.0	78.2715	0.1026	0.1309

3.3.2.3 Preparation of reaction samples for HPLC analysis

A small aliquot of the reaction mixture (approximately 500 μL) was removed from the reaction vessel at the prescribed sampling time and transferred to a 2 mL screwtop vial. Using an air displacement pipette, 75 μL of the supernatant reaction mixture was then transferred to a tared 50 mL volumetric flask. The aliquot weight was recorded. The sample was diluted to volume with 50:50 v/v acetonitrile / water and the weight of diluent added was recorded. The mixture was well mixed before transferring approximately 1 mL of solution to a HPLC vial for analysis.

3.3.3 Preparation and analysis of spectroscopic reference standards

Aim

To provide reference measurement functions (reference spectra) for Kalman filtering, a number of spectroscopic reference standard solutions of the main reactants and product were prepared at typical reaction concentrations. The spectra were also used to create a simulated data set that was used during the implementation and development of the Kalman filter algorithms.

Method

Spectroscopic reference standard solutions of acetonitrile, 1*H*-indole, benzyl bromide, 1-benzyl-1*H*-indole, cesium carbonate and tetrabutylammonium bromide were prepared at typical reaction concentrations using the weights and dilution volumes summarised in Table 3.7. The instrument parameters used to acquire the UV spectra were previously described in section 3.1. The exposure time for the Raman spectrometer was reduced to 5 seconds to prevent saturation of the detector when acquiring the spectra of clear solutions. This was necessary because the solutions did not contain cesium carbonate (solid) that significantly attenuated the detected Raman signal in the reaction mixtures. The cosmic ray filter was enabled to prevent contamination of the spectra by random, high intensity peaks resulting from cosmic rays striking the detector during acquisition.

The UV spectral data files were then processed as described in section 3.3.4. The Raman spectral data files (`.spc` format) were imported directly into Matlab using the `spcreadr` function from the Eigenvector PLS toolbox.

Table 3.7: Sample weights, dilution volumes and final concentrations of spectroscopic standard solutions.

Standard	Weight of reagent (g)	Volume of solvent (mL)	Molecular weight ($g.mol^{-1}$)	Concentration ($mol.L^{-1}$)
Acetonitrile	Neat	90.0	41.05	32.02
1*H*-indole	3.3319	50.0	117.15	0.569
Benzyl bromide	4.8603	50.0	171.04	0.568
1-benzyl-1*H*-indole	3.4600	25.0	207.28	0.570[a]
Cesium carbonate	18.5590	50.0	325.82	1.139
Tetrabutylammonium bromide	0.9302	50.0	322.28	0.057

3.3.4 Preparation and pre-processing of the UV spectral data

Aim

The spectral data acquired using the Varian Cary 50 UV spectrometer required formatting before they could be imported into Matlab. Once imported into Matlab, the spectra required pre-processing to remove unwanted spectral artefacts ('spikes') and baseline variation prior to Kalman filtering.

Method

The Cary 50 spectral data were exported as comma separated variable (`.csv`) text files. The Visual Basic program described in section 3.1.6 was then used to filter and reformat the data files to remove duplicated wavelength variable columns and re-align rows with missing values. The output of this program was comma separated variable text files with the extension (`.uvd`). The `.uvd` files were then imported into Matlab for further processing.

Many of the data sets contained several spectra that were characterised by a single, high intensity spike. These spikes were believed to be artefacts that resulted from a skipped wavelength as the monochromator scanned across the wavelength range. Although it was possible to remove each spectrum that contained spikes, that approach would have required each affected spectrum to be manually identified and removed. A custom Matlab script, `MedianFilter.m` (section 3.1.7.3) was written to perform moving window

median filtering to remove the spikes. For this application, a window size of three was used. Median filtering was applied to every data set.

The median filtered absorbance spectra were then baseline corrected by subtraction of a linear sloped baseline with the function $f(x) = b_0 + b_1 x$ from each spectrum. The Matlab function `polyfit` was used to calculate the polynomial coefficients b_0 and b_1 for each spectrum over the wavelength range 320 to 370 nm. The coefficients were then used to extrapolate each baseline over the full spectral range 220 to 400 nm using the Matlab function `polyval`. The extrapolated baseline was subtracted from its corresponding spectrum.

3.3.5 Optimisation of the IPBS method parameters

Aim

The custom Matlab function `IPBS.m` described previously in section 3.1.7.4 uses several parameters that can influence the results obtained when applying automated iterative polynomial baseline subtraction. The aim of these experiments was to locate the optimal values of some of the key parameters by performing an exhaustive search over a defined range of values for the degree of the polynomial (n) and the window width (w_p).

Method

To find the optimal values of the key parameters n and w_p, an exhaustive search was performed using a range of values applied to a reaction spectrum from one of the Raman data sets described previously in section 3.3.1. The specific Raman spectrum used to optimise the parameters was acquired at t =30 minutes and corresponded to a spectrum of *1H*-indole, a small amount of tetra butyl ammonium bromide and the un-dissolved inorganic base (cesium carbonate) in acetonitrile. This spectrum was expected to be very similar to the reference spectrum of *1H*-indole in acetonitrile described in section 3.3.3, except that it also had a large baseline contribution from the suspended cesium carbonate. The values of the parameters n and w_p were considered optimal in the sense that they minimised the residual sum-of-squares between the normalised, baseline subtracted test spectrum and the normalised reference spectrum of *1H*-indole in acetonitrile.

Before the search of the parameters n and w_p was started, a suitable window width to use during the median filtering step was chosen. `MedianFilter.m` was applied to the test

spectrum using a window width of 101, 201, 301, 401 and 501 points. Initial experimentation showed that there was no further improvement using a window width greater than 501. A median filter window width of 501 was selected.

The IPBS method was then applied to the test spectrum using the following range of values:

Degree of polynomial: $n = 2$ to $n = 6$ in steps of 1, (5 levels)
Window width: $w_p = 21$ to $w_p = 581$ in steps of 8, (71 levels)
This was repeated three times using overlap values of 0.10, 0.20 and 0.30. The total experiment took approximately 12 hours to complete. The residual sum-of-squares values were stored in three (71×5) matrices.

3.3.6 Preparation and pre-processing of the Raman spectral data

Aim

The Raman spectral data acquired during the *N*-benzylation of 1*H*-indole reactions required pre-processing to remove the complex baseline contributing to each spectrum. Once the variable baseline contribution had been removed, the spectrum to spectrum intensity variation inherent in Raman spectra was eliminated by spectral normalisation.

Method

Each data set was corrected by calculation and subtraction of the baseline contribution using the automated iterative polynomial baseline subtraction method (`IPBS`). The optimal parameters derived previously in section 3.3.5 were used.

Degree of polynomial: $n = 3$
Median filter window width: $w_{MF} = 501$
Polynomial window width: $w_p = 157$
Degree of window overlap: 0.20 (20%)
Convergence tolerance: 1.00×10^{-7}
During data acquisition, the option to automatically filter cosmic rays was disabled in the Raman spectrometer software (HoloGrams) as this would have doubled the amount of time required to acquire each spectrum. To remove the cosmic rays from the data manually, the spectral data sets were reproduced using the first four principal components following singular value decomposition of the un-centred baseline corrected data. After

removing the cosmic rays, the spectra were then normalised to max=1. Examination of the variance spectrum of pre-processed data revealed that the spectra contained no significant structured variance outside the spectral range 100 to 1800 cm^{-1}. The final step of spectral pre-processing was therefore the truncation of the spectra to exclude the data outside this wavenumber range.

3.3.7 Creation of a simulated UV data set for algorithm testing

Aim

A simulated UV data set that represented a set of reaction spectra acquired during the non-aqueous *N*-benzylation of 1*H*-indole using benzyl bromide was created. This data set was used to develop and test the various Kalman filter algorithms using a well defined system with known noise contributions.

Method

A second order reaction with the general expression shown below was simulated using the kinetic parameters listed in Table 3.8.

$$A + B \xrightarrow{k} C$$

The integrated rate equations[42, 104] used to calculate the concentration profiles of a second order reaction are:

$$[A] = \frac{[A]_0([B]_0 - [A]_0)}{[B]_0 e^{([B]_0 - [A]_0)kt} - [A]_0} \qquad \text{(equation 3.3)}$$

$$[B] = \frac{[A]_0([B]_0 - [A]_{0)}}{[B]_0 e^{([B]_0 - [A]_0)kt} - [A]_0} + ([B]_0 - [A]_0) \qquad \text{(equation 3.4)}$$

$$[C] = [A]_0 - [A] \qquad \text{(equation 3.5)}$$

Table 3.8: Parameters used to calculate simulated concentration profiles for the N-benzylation of 1*H*-indole using a second-order kinetic model.

Parameter	Description	Value
$[A]_0$	Initial concentration of species A (1*H*-indole)	0.452 mol.L^{-1}
$[B]_0$	Initial concentration of species B (benzyl bromide)	0.542 mol.L^{-1}
k	Rate constant	1.00×10^{-3} L.mol^{-1}.s^{-1}

After creating the concentration profiles using the integrated rate equations and the parameters above, a 30 minute induction period was inserted. During this period, only species A was present. After 30 minutes, B was then introduced very quickly (within 1 sample time). To make the concentration profiles more realistic, the dilution that would be observed on addition of the benzyl bromide solution was included. The concentration of 1*H*-indole prior to addition of the benzyl bromide solution was 0.532 mol.L^{-1} (equivalent to 6.00 g 1*H*-indole in 96.0 mL of solution comprising 80 mL acetonitrile, 1.65 g TBAB and 33.37 g Cs_2CO_3). The addition of the benzyl bromide solution at 30 minutes resulted in a total reaction volume of 113 mL and reduced the concentration of 1*H*-indole to 0.452 mol.L^{-1} ($[A]_0$). The final concentration profiles were stored in a (333×3) matrix denoted $\boldsymbol{C1}_{sim}$.

The UV spectra of the 1*H*-indole, benzyl bromide and 1-benzyl-1*H*-indole spectroscopic solutions described previously in section 3.3.3 were normalised to unit concentration (1.0 mol.L^{-1}) and collated in single (181×3) matrix denoted $\boldsymbol{S1}_{sim}$. The outer product of the concentration and spectral profiles was calculated using $\boldsymbol{D1}_{sim} = \boldsymbol{C1}_{sim} \cdot \boldsymbol{S1}_{sim}^{T}$ to produce the noise-free reaction spectra in a (333×181) matrix. Homoscedastic noise with zero mean and variance 1.0×10^{-6} was added to the data matrix to give the final simulated data matrix, $\boldsymbol{D2}_{sim}$. The absorbance spectra were transformed to their first derivative form using Savitsky-Golay smoothing and differentiation (Eigenvector PLS toolbox, `savgol` function). The first-derivative spectra were calculated from $\boldsymbol{S1}_{sim}$ and $\boldsymbol{D2}_{sim}$ using a 13-point smoothing window and third-order polynomial, producing $\boldsymbol{S2}_{sim}$ and $\boldsymbol{D3}_{sim}$. The reaction spectra $\boldsymbol{D2}_{sim}$ and $\boldsymbol{D3}_{sim}$, and the reference spectra $\boldsymbol{S1}_{sim}$ and $\boldsymbol{S2}_{sim}$ were then truncated to 220 to 350 nm.

3.3.8 Demonstration of the equivalence of the standard and vectorised linear Kalman filter

Before development of a vectorised version of the adaptive Kalman filter for self-modelling curve resolution of spectral data, it was confirmed that a vectorised linear Kalman filter produced the same results as the standard linear Kalman filter. The two Kalman filter algorithms (standard linear Kalman filter and vectorised linear Kalman filter) were applied to the simulated UV data sets ($\boldsymbol{D2}_{sim}$ and $\boldsymbol{D3}_{sim}$) and the calculated Kalman gain vectors, estimated state parameters and spectral lack-of-fit values were compared.

3.3.8.1 Determination of the measurement noise variance, R

Aim

A critical parameter when applying the linear Kalman filter to a data set with homoscedastic noise is the measurement noise variance, R. If this value is too large, the Kalman filter will not produce accurate results because the data are assumed to have a large measurement error associated with each data point. Using a value of R smaller than the actual measurement variance will generally lead to estimated state parameters and innovations very similar to those that calculated using the actual measurement variance. However, as the elements of the error covariance matrix $\boldsymbol{P}$ are calculated using R, they will also have much lower values. This will give a misleading estimate of the error associated with each state parameter estimate. The optimal value of R is a value equal to the actual measurement noise variance. This will be characterised by the start a minimum in a plot of the innovations sum-of-squares versus R.

Method

Using the last spectrum of the data sets $\boldsymbol{D2}_{sim}$ and $\boldsymbol{D3}_{sim}$, the linear Kalman filter (`linearKF.m`, section 3.1.7.5) was run using a range of values 10^x, for x =-10 to x =-3 using an increment of 0.1. This produced R values from 1.0×10^{-10} through to 1.0×10^{-3} with ten points for each order of magnitude. The spectral root-mean-square lack-of-fit was calculated using the expression shown in equation 3.6. This calculated the average lack-of-fit of the predicted spectral data points as a percentage of the original data.

$$LOF = \sqrt{\left(\frac{1}{K-1}\right) \cdot \frac{\sum_{k=1}^{K} \boldsymbol{e}_k^2}{\sum_{k=1}^{K} \boldsymbol{z}_k^2}} \times 100\% \qquad \text{(equation 3.6)}$$

3.3.8.2 Application of the linear Kalman filter functions to simulated UV data

Aim

The standard linear Kalman filter (`linearKF.m`) and vectorised linear Kalman filter (`VecLinearKF.m`) were to the simulated data sets to obtain the estimated state parameters, Kalman gains and state parameter error covariances. These were then compared directly to confirm that both algorithms produced the same results.

Method

The standard linear Kalman filter (`LinearKF.m`, section 3.1.7.5) and the vectorised linear Kalman filter (`VecLinearKF.m`, section 3.1.7.7) were applied to the simulated data sets $\boldsymbol{D2}_{sim}$ and $\boldsymbol{D3}_{sim}$ using the reference spectra $\boldsymbol{S1}_{sim}$ and $\boldsymbol{S2}_{sim}$ as the reference measurement functions. The estimated measurement noise variance determined previously in section 3.3.8.1 were used ($R = 3.16\times10^{-6}$ for $\boldsymbol{D2}_{sim}$ and $R = 1.00\times10^{-9}$ for $\boldsymbol{D3}_{sim}$). The option to treat the data as evolutionary was not applied for the standard linear Kalman filter as there was not an equivalent option for the vectorised linear Kalman filter.

A simple test was also performed to investigate how the calculated Kalman gain vectors changed as the number of components in the reference measurement function was changed. The linear Kalman filter (`LinearKF.m`) and the vectorised linear Kalman filter (`VecLinearKF.m`) functions were applied to the first spectrum of the simulated data set $\boldsymbol{D2}_{sim}$ using different reference measurement functions. In the first run, the reference measurement function only contained the spectrum of component A; in the second run, the reference measurement function contained the spectra of components A and B; in the third run, the reference measurement function contained the spectra of components A, B and C.

3.3.9 Application of the adaptive Kalman filter to simulated UV spectra

The adaptive Kalman filter based upon the description of algorithm published by Rutan and Brown[66] was implemented as a Matlab function, `AdaptiveKF.m` (section 3.1.7.6). The adaptive Kalman filter was applied to the simulated UV data set $\boldsymbol{D2}_{sim}$ to assess the performance of this method, and to investigate its limitations when applied to highly overlapped spectra.

3.3.9.1 Application of the adaptive Kalman filter using incomplete reference functions

Aim

The purpose of these experiments was to assess the performance of the adaptive Kalman filter when used to predict the concentration of known components in the presence of model errors. The model errors arose from the appearance of a new chemical species not included in the reference measurement function matrix. The adaptive Kalman filter also allowed the reference measurement function to be augmented with an approximation of the unknown component calculated from the vector of innovations. The augmented reference functions were compared with the true spectral profiles and the state parameters compared with the true, scaled concentration profiles.

Method

The adaptive Kalman filter was applied to the simulated UV data set $\boldsymbol{D2}_{sim}$ using incomplete reference measurement functions. To emulate a self-modelling application, the Kalman filter was applied using no prior knowledge of the true spectral profiles. To provide the first reference measurement function corresponding to component A, singular value decomposition was applied to a subset comprising the first three spectra of the data set. The subset was reproduced using the first left and right singular vectors and the first singular value. The mean spectrum from the reproduced subset was calculated and normalised to unit length (2-norm). This de-noised, normalised spectrum was used as the first reference measurement vector corresponding to component A.

The adaptive Kalman filter was applied as explained below using the measurement noise variance ($R = 3.16 \times 10^{-6}$) determined previously in section 3.3.8.1.

Estimation of the state parameters for component A in the presence of component B

In the simulated data $\boldsymbol{D2_{sim}}$, component B appeared quickly at spectrum 32, representing the fast addition of a final reagent to initiate the reaction. Component C also began to appear as the reaction proceeded but over the very early part of the reaction, its concentration was very low. The purpose of this test was to assess the ability of the adaptive Kalman filter to predict the concentration of component A in the presence of component B. The adaptive Kalman filter (`AdaptiveKF.m`) was applied to spectra 1 to 32 of data set $\boldsymbol{D2_{sim}}$ using a window size of 4.

Re-estimation of the state parameters for components A and B using an augmented reference measurement function

After application of the adaptive Kalman filter to the first 32 spectra in the above experiment, the innovations indicated a model error corresponding to the rapid appearance of the unmodelled component B. The reference measurement function matrix was augmented with the estimated spectral profile for component B calculated from the adaptive measurement error values (R_k). The adaptive Kalman filter was then applied to spectra 1 to 32 a second time.

Estimation of the state parameters for components A and B in the presence of component C

The reaction was initiated at spectrum 32 on addition of component B. From spectrum 32 through to spectrum 333, the product (component C) began to increase in concentration as it was slowly formed. Using the augmented matrix of reference measurement functions obtained in the previous step, the adaptive Kalman filter was applied to the complete set of spectra.

Re-estimation of the state parameters for components A, B and C using an augmented reference measurement function

After application of the adaptive Kalman filter to the full data set, the innovations again indicated a model error corresponding to the slow formation of component C. The reference measurement function was augmented with the estimated spectral profile for component C calculated from the adaptive measurement error values (R_k). The adaptive Kalman filter was then applied to the complete set of spectra using the augmented matrix

of reference measurement functions to recalculate the state parameters for all three components.

3.3.10 Application of VVSP to simulated UV data

Aim

The Vertex Vector Sequential Projection method was implemented as a Matlab function (`VVSP.m`, section 3.1.7.9) and applied to the simulated UV data set $\boldsymbol{D2}_{sim}$. The aim of this experiment was investigate the ability of VVSP to detect the number of components contributing to the data set and locate the spectra that best resembled the true, pure spectral profiles.

Method

The Matlab function `VVSP.m` was applied to the simulated data set $\boldsymbol{D2}_{sim}$. The number of components to locate (`NL`) was set to 8 and the type of spectral normalisation was set to `p_norm = 2` (each spectrum normalised to unit length).

3.3.11 Application of VAKFISO to simulated UV data

Aim

The purpose of this set of experiments was to determine how the VAKFISO method (described in section 2.10 and 3.1.7.10) performed when different weighting coefficients (α_1, α_2, α_3 and α_4) were used to calculate the weighted residual matrix $\boldsymbol{E}$ that is used during NGL/M optimisation step.

Identification of suitable method parameters

It was possible to identify suitable starting parameters by assessing the magnitude of the elements in an innovations vector and state-parameter error covariance matrix for a fully modelled system. In the absence of *a priori* information, a fully modelled system may be approximated using the first N primary eigenvectors that span the spectral space of the data set. The primary eigenvectors (denoted $\bar{\boldsymbol{V}}$) obtained by applying singular value decomposition to the data set $\boldsymbol{D2}_{sim}$ were set as the matrix of reference measurement functions, $\boldsymbol{S}$. The vectorised adaptive Kalman filter (`VecAdaptiveKF.m`, section 3.1.7.10) was applied to the data set $\boldsymbol{D2}_{sim}$ using a window size of 4.

Application of VAKFISO using the initial method parameters

Following the initial examination of the data to estimate suitable weighting coefficients, VAKFISO was applied to $\boldsymbol{D2_{sim}}$ using the parameters listed in Table 3.9. The maximum number of iterations was set to 250 to allow the function to break from the optimisation cycle if the convergence tolerance was not achieved.

Table 3.9: List of initial parameters used to test the VAKFISO method. The method was applied to the simulated data set $D2_{sim}$.

Parameter	VAKFISO variable	Value	Comment
α_1	`alpha1`	1.0×10^{-3}	Weighting coefficient for first term ($\alpha_1 \boldsymbol{V}$)
α_2	`alpha2`	1	Weighting coefficient for second term ($\alpha_2 \Pi \boldsymbol{V}$)
α_3	`alpha3`	1	Weighting coefficient for third term ($\alpha_3 \Sigma \boldsymbol{V}$)
α_4	`alpha4`	1	Weighting coefficient for fourth term ($\alpha_4 \Xi \boldsymbol{V}$)
δ	`delta`	1.0×10^{-6}	Shift parameter
mp	`mp`	1.0×10^{-3}	Marquardt parameter
μ	`mu`	1.0×10^{-4}	Convergence tolerance

Application of VAKFISO with exclusion of the state parameter error covariance term

It was possible to eliminate the contribution of state parameter error covariance term from the calculation of the weighted residual matrix by setting the value of α_2 to zero. By setting α_2 to zero, but leaving all other parameters as listed in Table 3.9, the VAKFISO method optimised the model in the sense that it minimised the residual sum-of-squares. The same constraints of spectral and state-parameter non-negativity were applied during the optimisation but the sum of the state estimate variances were not be used in the calculation of the weighted residual matrix. The purpose of this experiment was to demonstrate that using the Kalman filter during the optimisation process provides a unique advantage over simple constrained least squares methods.

3.3.12 Application of VAKFISO to real UV data

Aim

In section 3.3.11, the VAKFISO method was applied to simulated UV data and the results obtained were compared with the known spectral and concentration profiles. To assess how VAKFISO performs on real data, it was applied to the UV/ATR spectra acquired during the *N*-benzylation of 1*H*-indole reactions described previously in section 3.3.1. The estimated pure component reference spectra were compared with the true measured reference spectra and the estimated concentration profiles were compared with the reference data provided by the analysis of reaction samples using HPLC.

3.3.12.1 Preparation and pre-processing of the UV spectral data

The preparation and pre-processing of the UV spectra was described in detail in 3.3.4. In summary, a median filter was applied to each spectrum to remove any spikes (instrument artefacts) that were present. The variable baseline contribution was then removed by fitting a first order polynomial to the 320 to 370 nm region of each spectrum. The calculated intercept and slope were used to subtract the sloped baseline from each spectrum. The first fifty-five spectra were removed from each data set as they corresponded to the region where the mixture of 1*H*-indole, TBAB and cesium carbonate was stirred for one hour to ensure full dissolution of the TBAB prior to addition of benzyl bromide.

3.3.12.2 Examination of the UV spectral data using PCA

To assess the number of independent components contributing to the data, PCA was applied to each data set. Selection of the appropriate number of principal components also allowed the approximate value of measurement noise variance to be estimated from the residual matrix. PCA was applied to both un-centred and mean-centred data to examine the structure of the loadings and scores vectors of the minor principal components before and after subtraction of the mean spectrum.

3.3.12.3 Derivation of initial spectral profile estimates using VVSP

The VAKFISO method uses VVSP to obtain initial estimates of the pure component spectral profiles if none are provided by the user. VVSP was applied to each of the three data sets individually to allow the resulting sets of initial estimates to be compared.

3.3.12.4 Application of VAKFISO using VVSP initial spectral profile estimates

The initial examination of the data using PCA and VVSP revealed that three-components were sufficient to model the data. The measurement noise variance estimated from the PCA residual matrix was approximately 1.0×10^{-6} AU^2 so VAKFISO was applied to each UV data using the parameters described in Table 3.9. VAKFISO automatically performed VVSP to produce initial estimates of the spectral profiles if none were provided by the user.

3.3.12.5 Application of VAKFISO using a random transformation matrix

It is possible to provide initial spectral estimates to the VAKFISO method that will override the default use of VVSP. In the previous section, VAKFISO was applied using VVSP initial estimates. This approach was not successful for the third data set (BnIndole_B2.00_MR1.50) because the initial spectral profiles for the three components were all very similar. To overcome this problem, a set of random spectral profiles were created. This approach is employed by BTEM and MREM and uses a transformation matrix comprising of randomly generated numbers.

A (3×3) matrix of random numbers was created using the Matlab function `randn`. This function creates a matrix of "pseudo-random values drawn from a random distribution with zero mean and standard deviation of one". The resulting matrix was multiplied by the (131×3) matrix of eigenvectors to create the initial spectral estimates using the equation $\hat{\boldsymbol{S}} = \bar{\boldsymbol{V}} \cdot \boldsymbol{T}$. The spectra were normalised to unit length and VAKFISO was then applied to each UV data set using the parameters described in Table 3.9.

3.3.13 Application of VAKFISO to real Raman spectra

Aim

In section 3.3.11 and 3.3.12, the VAKFISO method was applied to simulated and real UV/ATR spectra. The UV spectra were severely overlapped and there were very few selective regions in the concentration mode. To assess how VAKFISO performed on Raman spectra acquired simultaneously during the same reactions, the estimated pure component reference spectra were compared with the true measured reference spectra. The estimated concentration profiles were qualitatively compared with the HPLC reference profiles.

3.3.13.1 Preparation and pre-processing of the Raman data

The preparation and pre-processing of the Raman spectral data was described previously in section 3.3.6. The pre-processing involved the removal of the significant baseline contribution using the custom iterative polynomial baseline subtraction algorithm; removal of random cosmic rays by reducing the dimensionality of the data using SVD, normalisation of the spectra to maximum value equal to one (normalisation to infinity) and truncation of the spectra between 100 to 1800 cm^{-1}.

3.3.13.2 Application of VAKFISO using a random transformation matrix

The ability of VAKFISO to recover estimates of pure spectral profiles starting from random spectra was investigated. A (3×3) matrix of random numbers was created and multiplied by the (131×3) matrix of eigenvectors to create the initial spectral estimates using the equation $\hat{\boldsymbol{S}} = \bar{\boldsymbol{V}} \cdot \boldsymbol{T}$. The spectra were normalised to unit height (normalisation to infinity) and VAKFISO was then applied to each Raman data set using the parameters described in Table 3.9.

4 Results and Discussion

4.1 Combining Self-Modelling Curve Resolution and PLS regression

4.1.1 Laboratory scale development experiments

The aim of this work was to develop a multivariate calibration model to quantitatively monitor a reaction using *in-situ* spectroscopy. Reaction end-point criterions are often defined as an acceptable molar ratio of product to starting material or simply a minimum target concentration of product. To build a Partial Least Squares (PLS) model in the traditional manner would typically require off-line analysis (*e.g.* using HPLC) of many synthetic mixture or actual reaction samples to determine the concentration of each reactant and product of interest.

Observations made during the initial trial experiment revealed that it would not be possible to sample the reaction mixture during the course of the reaction to obtain the necessary concentration reference data owing to the follow difficulties:

(i) The reaction mixture started as a heterogeneous slurry and the starting material (acetoxyone) was only partially soluble so it dissolved slowly during the course of the reaction.

(ii) The reaction solution rapidly cooled on sampling, causing a mixture of acetoxyone / haloacetoxyone to precipitate out of solution; the measured concentrations in a filtered solution was not representative of what the ATR probe measured in the solution phase. This would have led to discrepancies between the *in-situ* spectra and the off-line reference measurements.

(iii) Following complete addition of phosphorus oxychloride, the UV reaction spectra indicated the presence of an intermediate species for which there was no reference material available. Without this reference material, it was not possible to measure a pure UV reference spectrum or prepare a HPLC assay standard of the isolated intermediate species.

A few simple experiments were performed to provide key reference measurements that aided subsequent modelling. The solubility of the starting material in the reaction solvent was determined using HPLC solution assay. Under nominal reaction conditions, the solubility of acetoxyone prior to the addition of phosphorus oxychloride was found to be

2.52×10^{-4} mol.L^{-1}. Examination of the UV/ATR spectra also revealed that owing to the low solubility of the starting material, its UV spectrum could not be distinguished from the solvent. As a consequence, an *in-situ* reaction monitoring method based on the consumption of acetoxyone was not possible.

Upon reaction completion, the reaction mixture was a dark homogeneous solution and although still highly saturated, it was easier to sample than the heterogeneous mixture. The concentration of acetoxyone and haloacetoxyone were measured using HPLC solution assay. The concentration of haloacetoxyone at the end of the calibration reaction was 0.272 mol.L^{-1}. The extent of reaction, based upon a peak area ratio of product to starting material was greater than 98%.

4.1.2 Overview of the spectral data

During the calibration reaction a spectral data matrix comprised of 172 spectra at 1 minute intervals over the wavelength range 220 to 400 nm at 1 nm intervals (resulting in 180 variables) was acquired. Figure 4.1(a) and Figure 4.1(b) show the reaction spectra obtained.

The absorbance values in the original spectra between 220 and 265 nm were almost 3 AU and were not suitable for quantitative modelling as the spectra were optically saturated and contained significant levels of measurement noise. The region between 350 and 400 nm contained no useful spectral features and were also ignored. The data were therefore truncated to only include the region 265 to 350 nm prior to subsequent data analysis. The second derivative spectra support the observations above. The spectral region between 220 and 265 nm contained higher levels of noise relative to the rest of the spectra. The region between 350 and 400 nm had derivative absorbance values of zero, providing confirmation that there were no significant features present in the original absorbance spectra.

The second derivative spectra shown in Figure 4.1(b) reveal a third species in addition to the starting material / solvent and product. The two independent sets of peaks for the intermediate and product can be seen in the spectral region 290 to 340 nm.

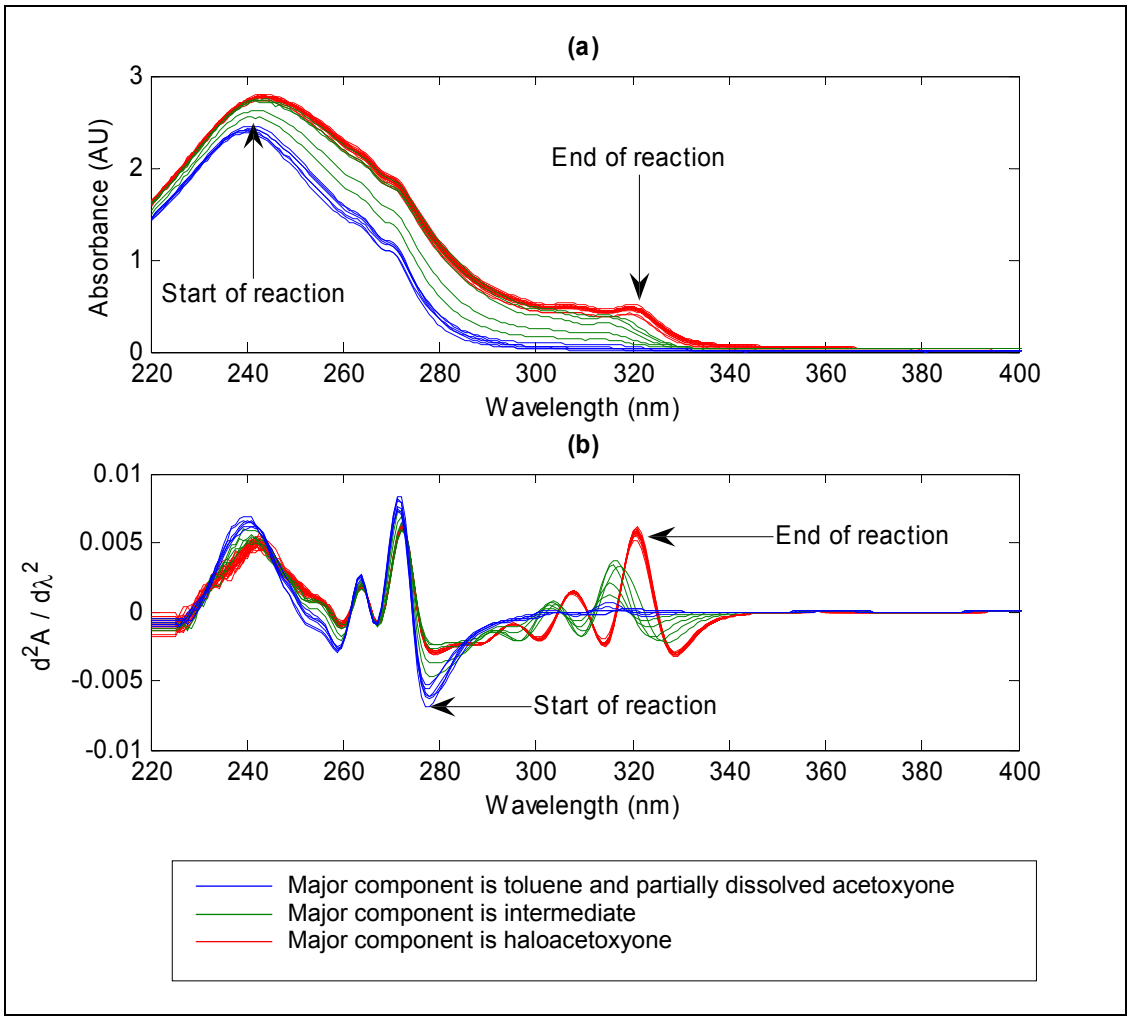


Figure 4.1: (a) UV spectra from laboratory scale reaction. Every fifth spectrum is shown to improve clarity; (b) Second derivative spectra from laboratory scale reaction. Every fifth spectrum is shown to improve clarity

Visual examination and a preliminary examination of the data using principal components analysis also indicated that the region 220 to 265 nm was highly overlapped and was significantly noisier than the region 265 to 400 nm.

Principal components analysis of the mean-centred, second derivative data suggested that two major components could be resolved from spectra. The first two principal components accounted for 98.96% of the total variance whilst the third and fourth components contributed a further 0.51% and 0.37% respectively. The eigenvalues, scores and loadings are shown in Figure 4.2.

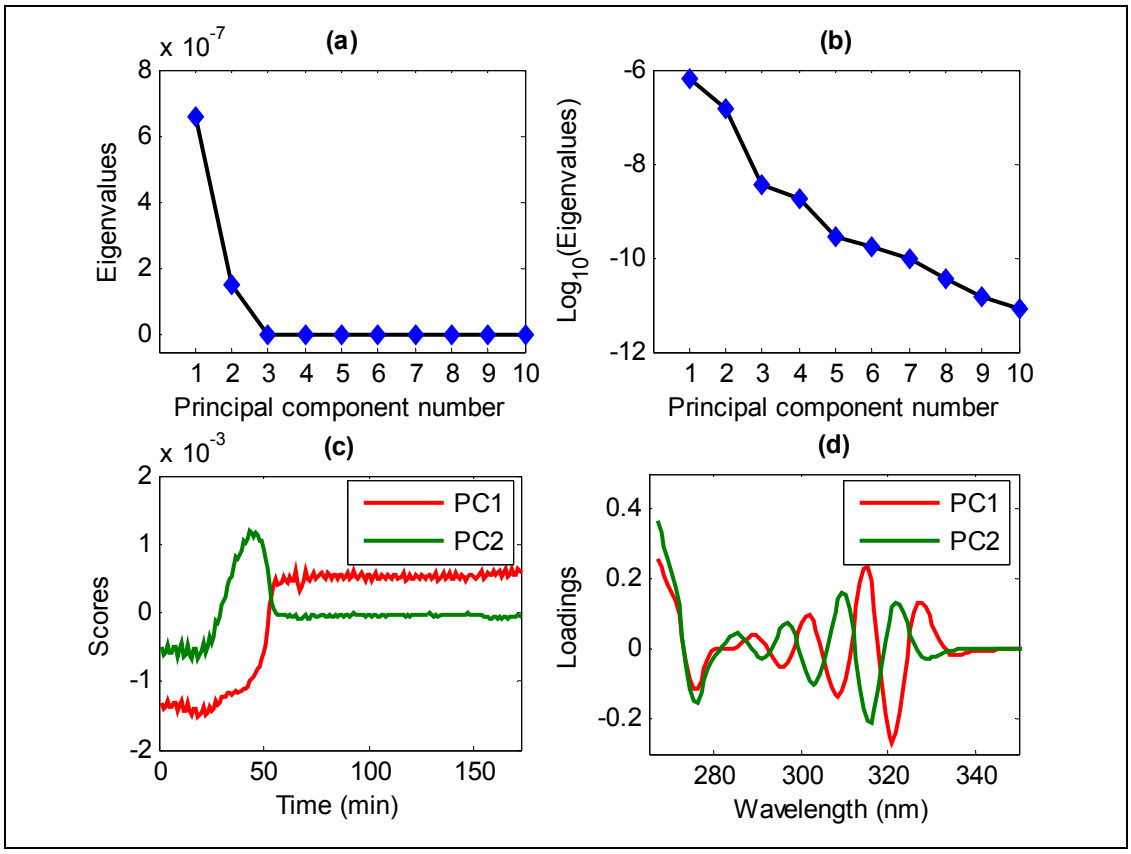


Figure 4.2: Summary plots from PCA of mean centred, second derivative UV spectra acquired during the laboratory scale calibration reaction; (a) plot of eigenvalue versus principal component number; (b) plot of $\log_{10}$(eigenvalue) versus principal component number; (c) plot of scores for first two principal components versus time (min) - Principal component 1 exhibits a profile characteristic of product formation, whilst principal component 2 exhibits a profile characteristic of intermediate formation and subsequent consumption; (d) plot of loading vectors for the first two principal components versus wavelength.

The plot of eigenvalues versus principal component number in Figure 4.2(a) shows that the eigenvalues decreased rapidly over the first three principal components. The eigenvalues then maintained a value close to zero for the remaining principal components. The $\log_{10}$ of the eigenvalues is a more sensitive measure of changes in value over a large range but there was still an obvious break-point at three principal components. This implied that three principal components would be sufficient to model the mean-centred data. As SMCR methods are typically applied to non-centred data, an additional component may be required. This is because when PCA is applied to un-centred data, the first principal component will model the mean and variance about the mean spectrum.

The scores profiles for the first two principal components exhibited features that correspond to possible concentration profiles for an intermediate species and product. It was confirmed using off-line HPLC analysis that the reaction did reach completion (assay 0.27 mol.L^{-1}, conversion >98%) in two hours (reaction sampled at 120 minutes). The

loadings in Figure 4.2(d) clearly show the distinctive peaks for the intermediate and product species. However the loading vectors are not necessarily the true pure component spectra for the intermediate and product. Application of SMCR methods allowed alternative candidates for the true pure component spectra to be calculated.

Another way to confirm the correct number of principal components required to sufficiently model the structured variance in a spectral data set without over-fitting was to examine the residual spectra. A custom Matlab script, `ResidualComps` was written that decomposes the un-centred spectral data using singular value decomposition (SVD). The spectral data are then reconstructed several times using an increasing number of principal components. The corresponding residual matrix is calculated at each principal component number by subtracting the reconstructed data from the original data matrix. The total sum-of-squares (SSQ) is calculated for each residual matrix and plotted against principal component number. Although this plot will have an identical profile to the plot of eigenvalues against principal component number, the ability to directly visualise and compare the reconstructed and residual data sets can also be very useful.

The output plots produced by applying `ResidualComps` to the second derivative spectra acquired during the calibration reaction are shown in Figure 4.3 and Figure 4.4. The total sum-of-squares of each residual matrix, plotted against the number of principal components used to reconstruct the data set are shown in Figure 4.3(a). The line plot shows that the residual sum-of-squares decreased rapidly over the first three principal components as the major contributions to the total variance were modelled. There was a clear break-point at three principal components where the decrease in the residual sum-of-squares was more gradual. This indicated that all subsequent components used to model the data were only capturing small contributions. In spectral data, these components usually account small, non-linear effects such as slight movement of the peak maxima or the peak shapes that cannot be captured by single principal components. The plot of $\log_{10}$ values also suggested that three or four principal components were sufficient to model the useful structured variance in the data.

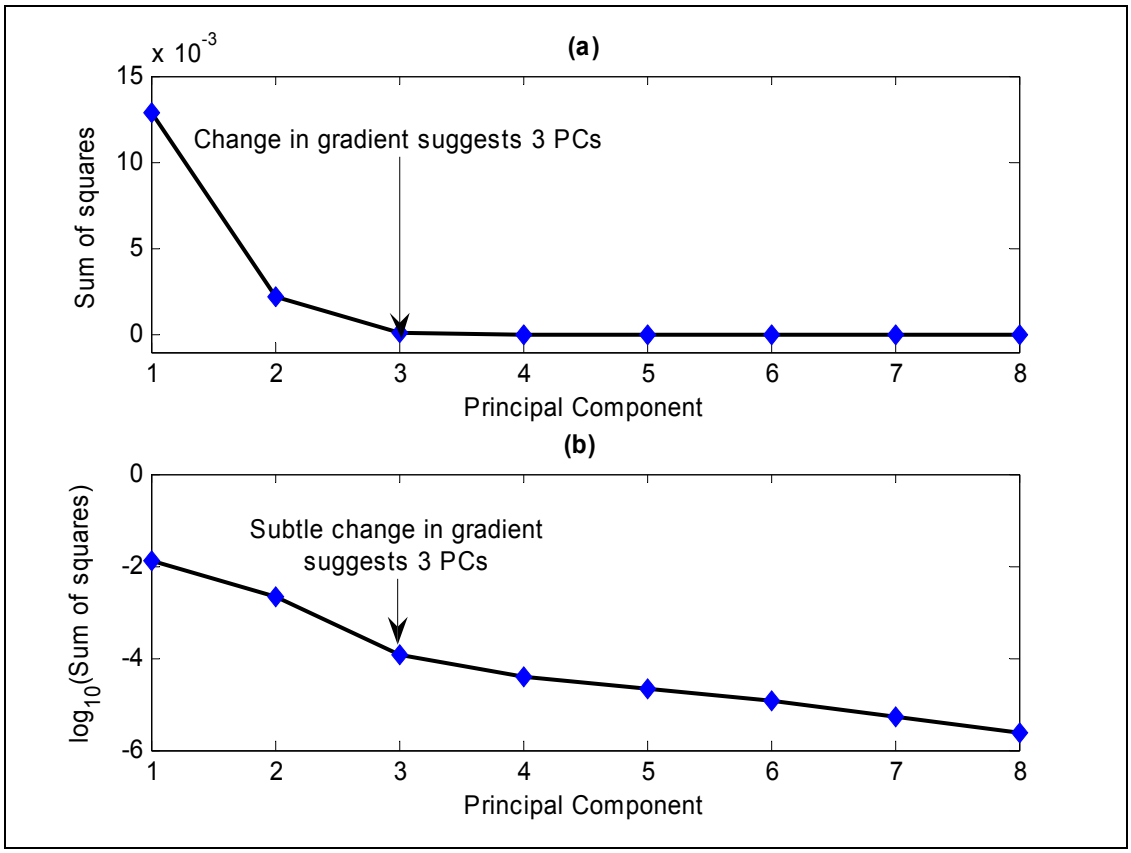


Figure 4.3: (a) total sum-of-squares of each residual matrix plotted against the number of principal components used to reconstruct the data set; (b) $\log_{10}$(sum-of-squares) of each residual matrix plotted against the number of principal components used to reconstruct the data set. The changes in gradient indicated on the plots suggest that three principal components would be adequate to model most of the structured spectral variation.

The reconstructed and residual spectra calculated using one to eight principal components are shown in Figure 4.4. The residual spectra provided a convenient way to examine the remaining structure not modelled by the principal components. For the first three principal components, the additional contributions to regenerated spectra are clearly visible. For principal components four and above, the contribution of the additional components to the regenerated spectra could not be seen. However, the difference to the respective residual spectra was easier to see. When examining the residual spectra, both the structure and the magnitude of the remaining features must be considered. Comparison of the residual spectra after regeneration of the data set using three and four principal components confirmed that the position and profile of the spectral features were very similar. Upon closer inspection, it was observed that the peak maxima in each set of residual spectra were at slightly different wavelengths; indicating that the additional components were modelling subtle peak shifts and changes in peak shape.

The conclusions from these initial examinations of the data were that the original data could be truncated without loss of information and only the spectral region 265 to 350 nm was retained. Application of Savitsky-Golay smoothing and differentiation to calculate the second derivative spectra was sufficient to remove any additive baseline contributions and also enhanced the spectral resolution between the major components of interest. Multivariate analysis using PCA indicated that three principal components were sufficient to model 99.87% of the total variance, and the inclusion of a fourth component accounted for 99.96% of the total variance. Visual examination of the regenerated and residual spectra suggested that three to four components were sufficient to model the data without over-fitting.

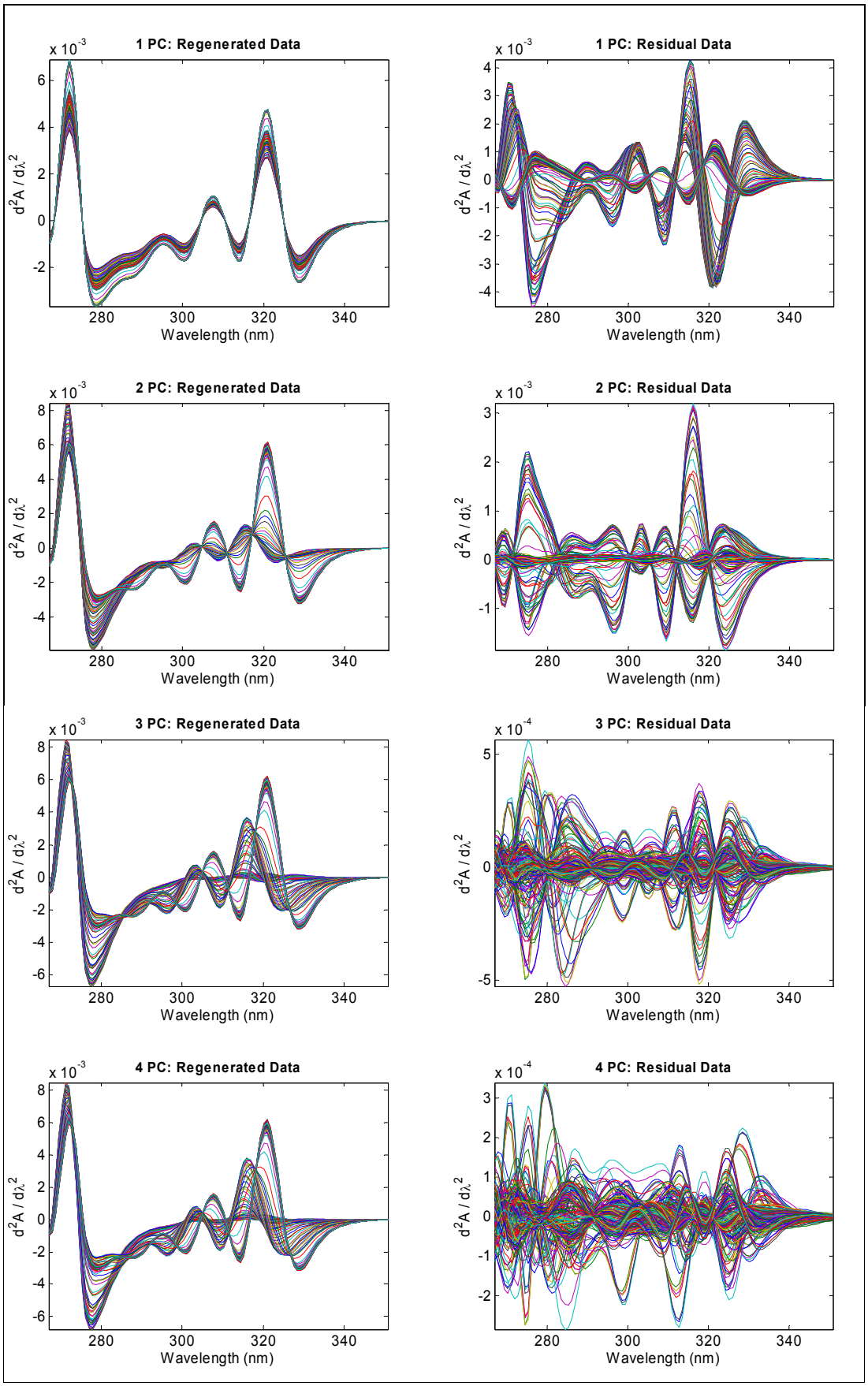


Figure 4.4: The reconstructed and residual spectra obtained using 1 to 4 principal components.

4.1.3 Derivation of initial concentration profile estimates using EFA

The two-, three- and four component EFA profiles obtained by application of EFA to the second derivative UV data are shown below in Figure 4.5.

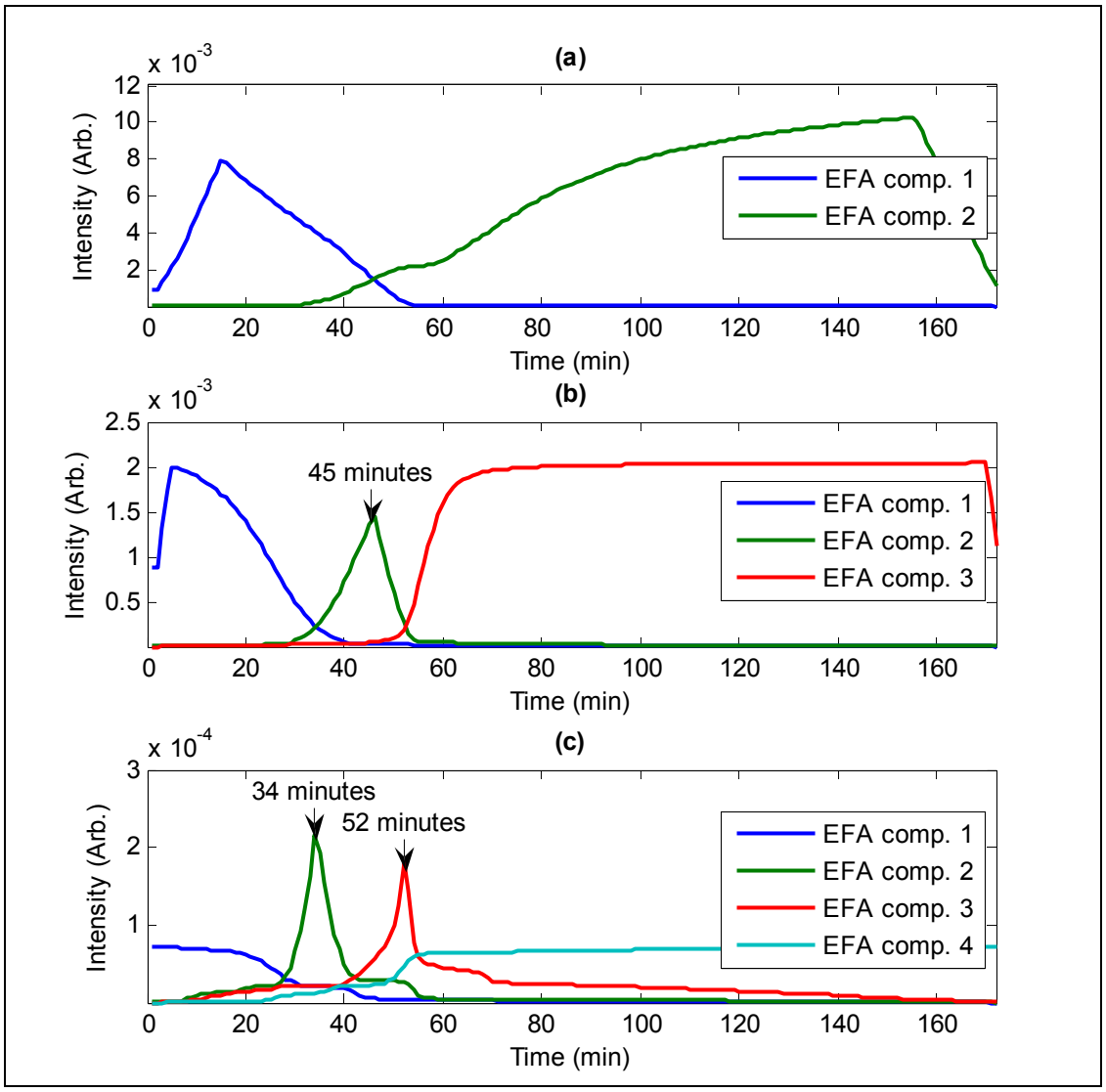


Figure 4.5: Combined forward and backward profiles obtained by applying EFA to the second derivative UV spectra: (a) two EFA components; (b) three EFA components; (c) four EFA components

The two component EFA profiles are shown in Figure 4.5(a). Component 1 corresponds to the starting mixture spectrum (toluene / acetoxyone) and its subsequent dilution on addition of phosphorus oxychloride. It is the ability of EFA to perform a local rank analysis of the first several spectra that allowed this feature to be detected. The intermediate species was not modelled explicitly and was therefore embedded in one or both of the two-component profiles.

The three component EFA profiles shown in Figure 4.5(b) now include a profile that corresponds to the likely concentration window for the intermediate species. The profile reached a maximum at approximately 45 minutes and therefore showed good agreement with the corresponding scores profile obtained using simple PCA. The profile corresponding to product formation also suggested a faster rate of reaction and shows much closer agreement to the time-series profiles obtained using PCA and univariate profiling at selective wavelengths.

The four component EFA profiles shown in Figure 4.5(c) have two profiles that could correspond to intermediate species. The profile corresponding to the formation of product was a feasible estimate as it was similar to the product profile obtained from the three component EFA estimates. However it began earlier in the reaction and was more overlapped with the intermediate profiles. Furthermore, the maxima of the two intermediate profiles were at 34 and 52 minutes (not 45 minutes as expected). The two intermediate profiles obtained from the four component EFA estimates were therefore believed to be modelling subtle spectral peak shifts occurring during the formation and reaction of the intermediate species.

Based upon the observations discussed above, it was concluded that the three component EFA profiles were the best initial estimates of the true concentration profiles. The start and end-points of the EFA profiles were manually corrected to remove the artefacts produced during the EFA analysis. The corrected profiles are shown in Figure 4.6.

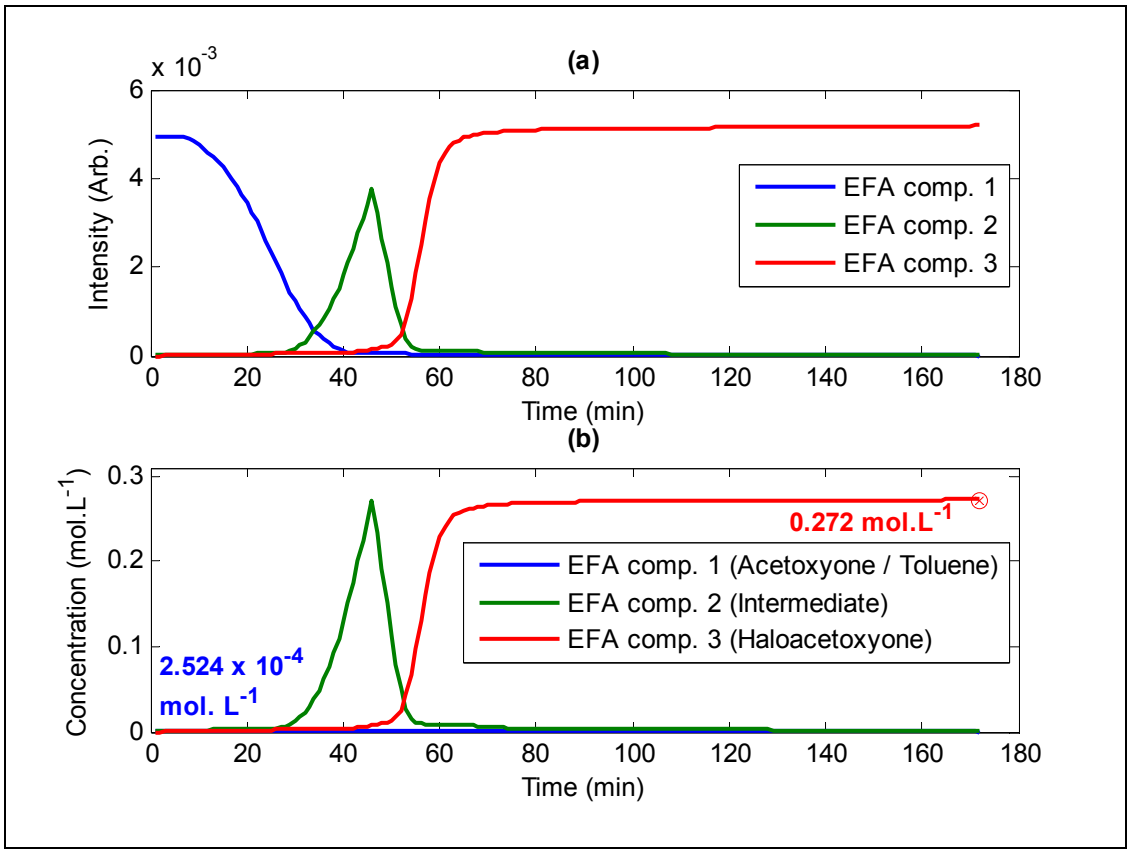


Figure 4.6: (a) Smoothed estimates of concentration profiles obtained by application of Evolving Factor Analysis to the second derivative UV spectra; (b) Scaled initial estimates of the concentration profiles. The profiles were scaled using the HPLC assay concentrations shown in the figure.

Using the experimental values obtained from the HPLC solution assays, the EFA profiles were scaled to provide initial estimates of the underlying concentration profiles in mol.L^{-1} units. In the absence of any concentration reference values for the intermediate species, the profile of the intermediate was scaled to the same maximum value as the product. As the profiles were to be refined using MCR-ALS, it was only necessary to scale the intermediate profile so that it had a comparable magnitude to the product profile; the actual value was not critical. The scaled profiles are shown in Figure 4.6(b).

4.1.4 Derivation of initial spectral profile estimates using OPA

OPA was applied to the truncated (265 to 350 nm), second derivative spectra to confirm whether the additional components identified using PCA and EFA were a consequence of peak shifting in the original data. The output from calculation of the first eight OPA components is shown below in Figure 4.7.

The most dissimilar (pure component) spectra are shown in Figure 4.7(a). The spectra were automatically normalised to unit length by the `OPA` function. The first three components are plotted with bold lines to improve clarity. The first spectrum (OPA comp.

1) corresponds to the starting mixture of toluene, partially dissolved acetoxyone and di-isopropylethylamine. The second spectrum (OPA comp. 2) corresponds to the product, haloacetoxyone and the third component (OPA comp. 3) corresponds to the intermediate. The spectra for OPA components four to eight were also very similar to the first three spectra. The largest differences occur between variables 1 to 20 (265 nm to 282 nm) and variables 50 to 75 (307 nm to 329 nm). This confirmed that the spectra for the minor OPA components were very similar to the major components but were shifted by a few nanometers.

Examination of the dissimilarity plots in Figure 4.7(b) also confirmed that the minor components represent spectral peak shifts. The dissimilarity profiles have all been scaled to max.= 1 to aid comparison but the minor components were much noisier and have less structure (autocorrelation) than the major components.

The Durbin-Watson values calculated for each OPA component's dissimilarity vector are shown in Figure 4.7(c). These values are a measure of the correlation or randomness existing in the dissimilarity profiles. As the Durbin-Watson value tends to zero, there is a strong serial correlation in dissimilarity vector, indicating the OPA component may correspond to true underlying structure in the data. The Durbin-Watson value will increase as the successive values in dissimilarity vector becomes less correlated. The plot in Figure 4.7(c) shows there was a distinct increase in value after 3 components, indicating that the dissimilarity vector for the fourth OPA component was less correlated than the first three components.

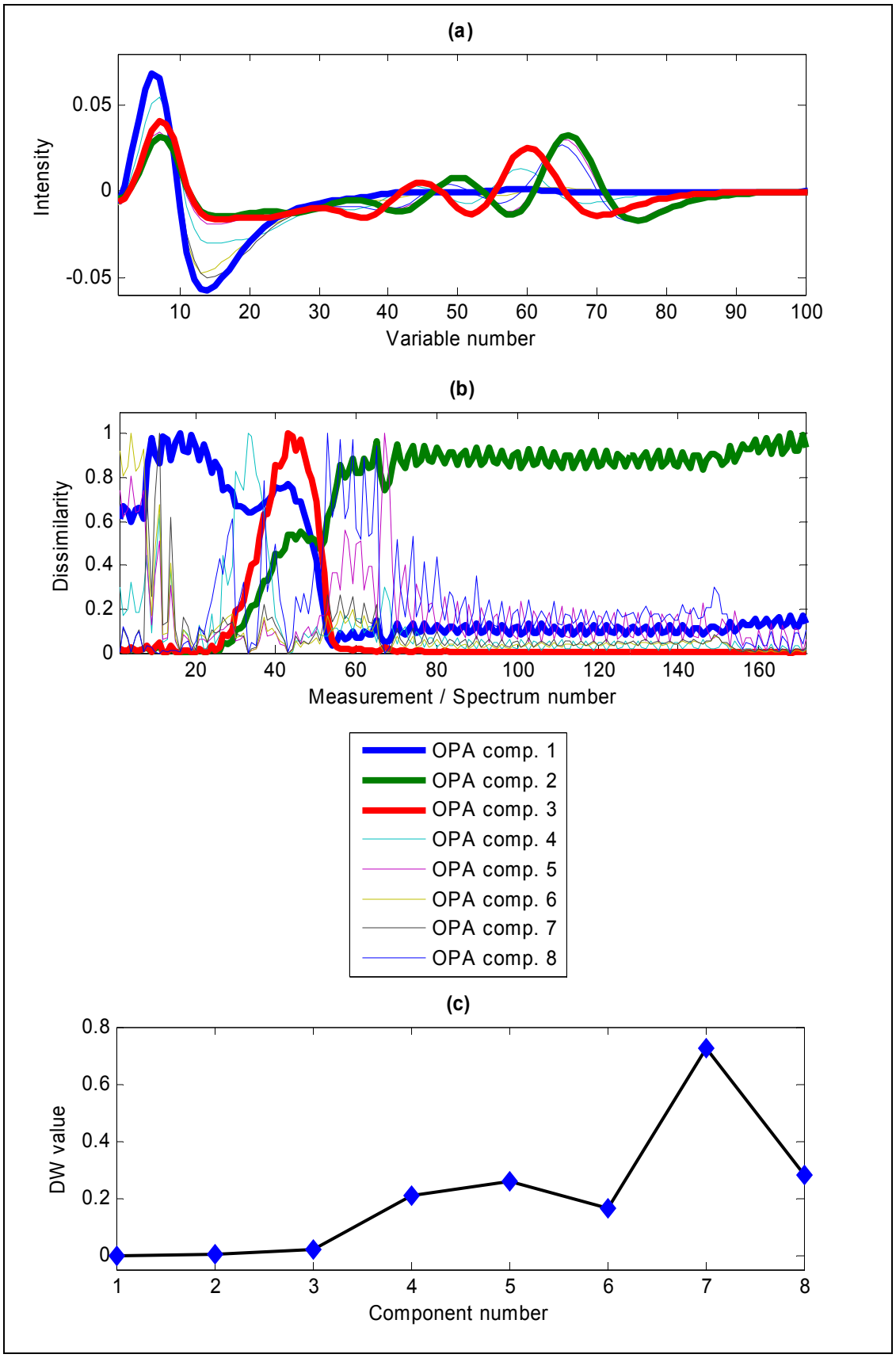


Figure 4.7: Application of OPA to the truncated, second-derivative UV spectra: (a) normalised pure component spectra. The first three OPA components are plotted using bold lines; (b) scaled dissimilarity profiles (scaled to max. = 1). The first three OPA components are plotted using bold lines; (c) Durbin-Watson values calculated using the dissimilarity vectors for each OPA component.

Based upon the EFA profiles, it was expected that the most selective spectra would be observed at approximately 0, 45 and 172 minutes, corresponding to the starting spectrum (predominately toluene), the maximum concentration of the intermediate species and the product at end of reaction respectively. The most dissimilar spectra were identified at 0, 43 and 168 minutes. The un-normalised 'pure' spectra were collected into a matrix $\boldsymbol{S}_{est}$ and scaled to unit concentration (1.0 mol.L^{-1}). The pure spectra corresponding to the starting material and product were scaled using the concentration values obtained by HPLC solution assays at the start and end of the reaction. Since reference standard material for the intermediate species was not available, an assumption regarding its unit concentration intensity was necessary. The spectrum for the intermediate species displays a very similar shape but exhibits a hypsochromic shift with respect to the product spectrum. It was therefore assumed that the 2-norm of the intermediate and product spectra were equivalent. After scaling the product spectrum to unit concentration, the spectrum of the intermediate species was scaled to give the same 2-norm as the unit concentration product spectrum ($\|S_i\|_2 = \|S_p\|_2$). The concentration of acetoxyone at the start of the reaction was 2.524×10^{-4} mol.L^{-1}. This was a factor of 1078 times lower than the concentration of haloacetoxyone at the end of reaction (0.272 mol.L^{-1}). The consequence of this was that when the spectrum of acetoxyone was scaled to unit molarity, its intensity was 1078 times larger than the scaled spectrum of haloacetoxyone. This did not affect subsequent MCR-ALS calculations as its spectrum was constrained. The scaled OPA spectra for the intermediate and haloacetoxyone are shown in Figure 4.8(b).

4.1.5 Refinement of profiles using MCR-ALS

Although EFA identified three profiles, the first component was believed to correspond to the dilution of toluene during the addition of phosphorus oxychloride. PCA of the same data indicated that no significant factors corresponding to this dilution were observed. Since none of the first three principal components modelled this behaviour, the dilution accounted for less than 0.35% of the total variance. Emphasis was therefore placed upon resolution of the profiles for the intermediate and product species only.

To help the MCR-ALS optimisation focus on refinement of the profiles of the intermediate and product species only, the scaled spectrum of acetoxyone / toluene was constrained to prevent it from being updated during each iteration. The scaled spectrum of haloacetoxyone (product) was also constrained. The inclusion of the two known

concentration values for the start and end of reaction in the concentration selectivity matrix ensured that the profiles were forced through two known reference points. In addition, the first several points of the product's concentration vector were set to zero as no product was present before the phosphorus oxychloride was added. The unknown parts of the concentration profiles and the spectrum of the intermediate species were free to be updated during the MCR-ALS optimisation. The refined concentration and spectral profiles are shown in Figure 4.8.

The lack-of-fit of the MCR-ALS optimised factors with respect to the reduced data reconstructed from the first three principal components was 0.0085%. The lack-of-fit of the MCR-ALS optimised factors with respect to the experimental data was 3.55%. This relatively high lack-of-fit value was not surprising as the rank of the data was larger than three owing to non-linear contributions such as peak shifting. The three component MCR-ALS model provided the best least-squares fit of the full-rank data but did not account for the minor contributions that increase the rank of the data above three. The percentage of variance explained by the three MCR-ALS components was 99.47% (*cf.* PCA 99.87%). These statistics revealed that the non-orthogonality of the MCR-ALS pure spectral profiles reduced the amount of variance captured by a three component model relative to the three component PCA model.

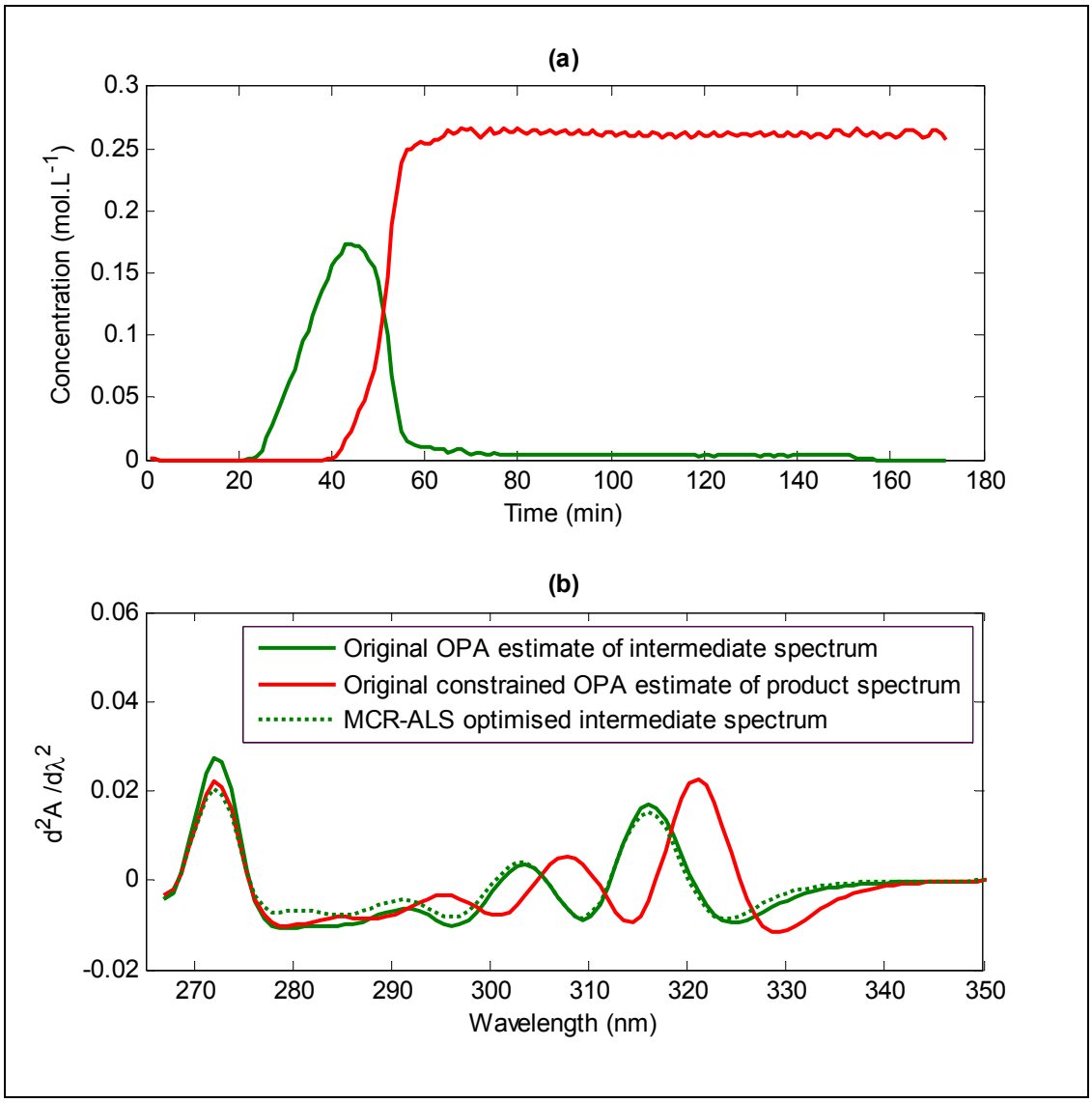


Figure 4.8: (a) MCR-ALS optimised concentration profiles for intermediate and haloacetoxyone; (b) MCR-ALS optimised spectral profiles. Initial estimates of the pure component spectra were obtained using Orthogonal Projection Approach and were scaled to unit concentration. The OPA spectrum for the intermediate species was scaled to have the same 2-norm as the scaled haloacetoxyone spectrum. During the MCR-ALS optimisation, the spectra of the acetoxyone / toluene and haloacetoxyone were constrained to prevent them from being updated. The spectrum of the intermediate species was automatically modified during the MCR-ALS optimisation to give the best least-squares fit of the data.

4.1.6 Calculation of PLS models using refined concentration profiles

A series of PLS1 and PLS2 models were calculated using the concentration profiles derived using MCR-ALS. A PLS model was required to allow the real-time prediction of haloacetoxyone concentration during a manufacturing campaign in a large scale laboratory. The unit molarity pure component spectra were derived using SMCR methods but could not be used directly to predict the concentration of the intermediate and product species.

This was because the standard process monitoring software used by UV spectrometer would not facilitate this approach. The function of the PLS model was to transform the pure component spectra into a set of PLS regression coefficients, $\hat{\boldsymbol{B}}$ via the corresponding concentration profiles. The Root Mean Standard Error of Cross Validation (RMSECV) values for the various PLS models are shown in Table 4.1. It should be noted that although the results from the cross-validation procedure were reported as RMSECV values, they do not correspond to RMSECV values in the traditional sense. Traditionally, RMSECV values indicate the accuracy to which values for a property of interest that have been measured by an off-line reference method can be predicted using a calibration model. In this case, the RMSECV values correspond to the accuracy to which the scaled concentration profiles derived directly from the spectral data using SMCR methods could be predicted using the PLS models.

Table 4.1: Root Mean Standard Error of Cross Validation (RMSECV) values for predicted constituent concentrations using PLS1 and PLS2 models with one to five factors. The values for the three factor model are shown in bold. The units of the RMSECV values are mol.L^{-1}.

		PLS1		PLS2	
Constituent	**Factors**	**Leave-one-out CV**	**Leave-out-fifty CV**	**Leave-one-out CV**	**Leave-out fifty CV**
Acetoxyone	1	3.60×10^{-5}	3.98×10^{-5}	4.78×10^{-5}	5.53×10^{-5}
	2	1.15×10^{-6}	1.21×10^{-6}	1.19×10^{-6}	1.25×10^{-6}
	3	$\mathbf{9.47\times10^{-7}}$	$\mathbf{9.83\times10^{-7}}$	$\mathbf{1.06\times10^{-6}}$	$\mathbf{1.12\times10^{-6}}$
	4	8.04×10^{-7}	8.73×10^{-7}	1.01×10^{-6}	1.05×10^{-6}
	5	7.62×10^{-7}	8.10×10^{-7}	9.70×10^{-7}	1.07×10^{-6}
Intermediate	1	3.16×10^{-2}	3.79×10^{-2}	4.18×10^{-2}	4.69×10^{-2}
	2	3.41×10^{-3}	3.40×10^{-3}	3.43×10^{-3}	3.42×10^{-3}
	3	$\mathbf{2.46\times10^{-3}}$	$\mathbf{2.44\times10^{-3}}$	$\mathbf{2.56\times10^{-3}}$	$\mathbf{2.52\times10^{-3}}$
	4	2.35×10^{-3}	2.42×10^{-3}	2.47×10^{-3}	2.50×10^{-3}
	5	2.31×10^{-3}	2.27×10^{-3}	2.50×10^{-3}	2.53×10^{-3}
Haloacetoxyone	1	9.83×10^{-3}	1.04×10^{-2}	8.67×10^{-3}	9.55×10^{-3}
	2	4.73×10^{-3}	4.64×10^{-3}	4.81×10^{-3}	4.71×10^{-3}
	3	$\mathbf{2.71\times10^{-3}}$	$\mathbf{2.64\times10^{-3}}$	$\mathbf{2.76\times10^{-3}}$	$\mathbf{2.79\times10^{-3}}$
	4	2.28×10^{-3}	2.74×10^{-3}	2.64×10^{-3}	2.63×10^{-3}
	5	1.78×10^{-3}	2.03×10^{-3}	1.90×10^{-3}	2.18×10^{-3}

The two methods of cross-validation did not produce values that were consistently higher or lower than the other method. Generally the leave-out-fifty cross validation returned RMSECV values that were between 5% and 20% higher than leave-one-out cross validation for the first factor. For factors two to four, the leave-out-fifty cross validation returned RMSECV values that were approximately ±5% relative to leave-one-out cross

validation. And for factor five, the values were approximately 10% to 15% higher. As expected, the RMSECV values for acetoxyone were very much lower than those returned for the intermediate or haloacetoxyone (10^{-7} to 10^{-6} mol.L^{-1} and 10^{-3} to 10^{-2} mol.L^{-1} respectively). These values reflect the difference in the magnitude of the concentration profiles. Owing to poor solubility, the concentration of acetoxyone only reaches a maximum concentration of approximately 2.524×10^{-4} mol.L^{-1}, the concentration of haloacetoxyone at the end of reaction was almost 1100 times higher (0.272 mol.L^{-1}).

When deriving the concentration profiles using SMCR methods, it was concluded that the chemical rank (*i.e.* number of independent spectroscopically active species that change concentration during the reaction) was three. The mean[3] RMSECV values returned by the three-factor PLS1 and PLS2 models for product were 2.68×10^{-3} mol.L^{-1} and 2.78×10^{-3} mol.L^{-1} respectively. At the end of the reaction, this corresponded to a prediction error of 0.99% (PLS1) and 1.02% (PLS2). The addition of a fourth or fifth factor only gave a marginal improvement of 0.84% (PLS1) or 0.93% (PLS2). Although including additional factors reduced the RMSECV, a three factor model provided a prediction error equivalent to the HPLC assay method used to derive the key reference measurements.

The choice of whether to use a PLS1 or PLS2 model was considered. The RMSECV values show that a three-factor PLS1 model gave a slightly better prediction than a three-factor PLS2 model, but both were fit for purpose. A PLS2 model was selected as it allows a matrix of dependent variables (Y-block) to be fitted simultaneously within a single model. PLS1 models fit each component of Y independently and although this gave better RMSECV values, it results in a set of scores, weights and loadings matrices for each component of Y. A PLS2 model was chosen because it returned good prediction errors within a single model and like MCR-ALS, it models all components simultaneously.

The main figures of interest for the resulting 3-factor PLS2 model are shown in Figure 4.9.

[3] Mean of the RMSECV values returned by the two methods of cross-validation; leave-one-out and leave-out-fifty.

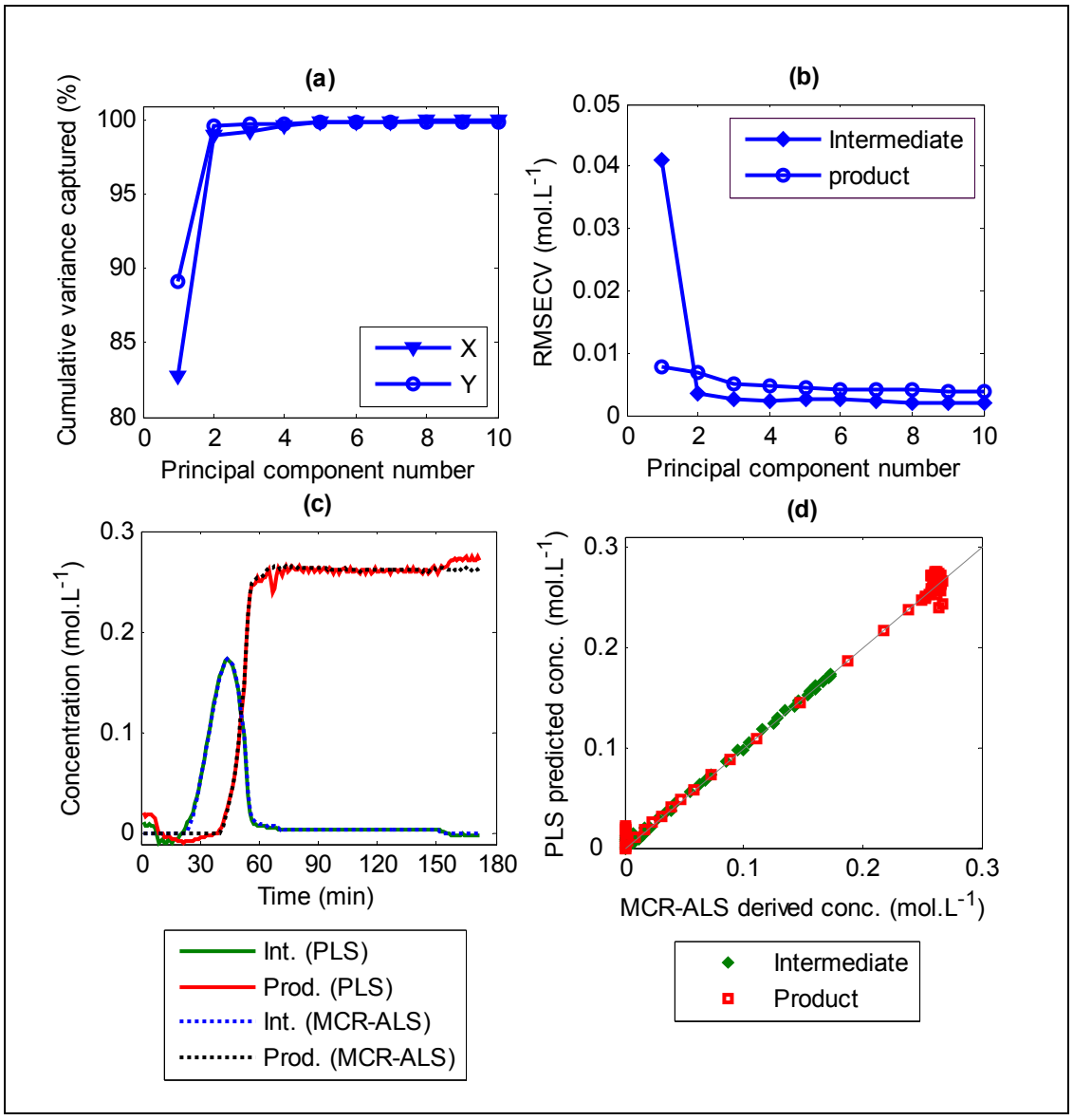


Figure 4.9: (a) Cumulative variance explained, (%) versus principal component number for spectral matrix (X) and concentration matrix (Y); (b) RMSECV versus principal component number; (c) MCR-ALS derived concentrations and PLS predicted concentrations versus time (min); (d) MCR-ALS derived concentrations versus PLS predicted concentrations (analogous to Actual v. Predicted).

4.1.7 Comparison of PLS regression coefficients and MCR-ALS spectra

A least-squares method was used to calculate the pure spectral profiles from each set of PLS1 or PLS2 regression vectors or matrices. The resulting spectral profiles were compared with the MCR-ALS derived pure component spectra for the intermediate and product. This was repeated for each model type (PLS1 or PLS2) and for one to five factors. The overlaid spectra are shown in Figure 4.10.

The plots in the first column show the PLS1 spectral profiles obtained using one to five factors. It is possible to see that the PLS1 spectra showed a strong correlation with the MCR-ALS derived spectra over the region 290 to 350 nm, although there were obvious

differences in the region 265 to 290 nm. It was also possible to discern that the fit between PLS1 and MCR-ALS spectra reached an optimum around three factors. The plots in the second column show the PLS2 spectral profiles obtained using one to five factors. The plots corresponding to the first factor clearly exhibit a huge scaling difference between the PLS2 spectra and the MCR-ALS spectra. The reason for this was that the PLS2 model attempted to fit all three constituents (acetoxyone, intermediate and haloacetoxyone) simultaneously. Consequently, the large difference in the magnitude of the concentration profiles for acetoxyone and the other two constituents meant that the first factor needed to account for the large difference in magnitude between the different constituents. Auto-scaling the Y-block would overcome this issue but unfortunately it was not possible to independently auto-scale the Y-block in the PLSPlus/IQ software. The spectral profiles for the subsequent factors show much better correlation and seemed to reach an optimum at three factors.

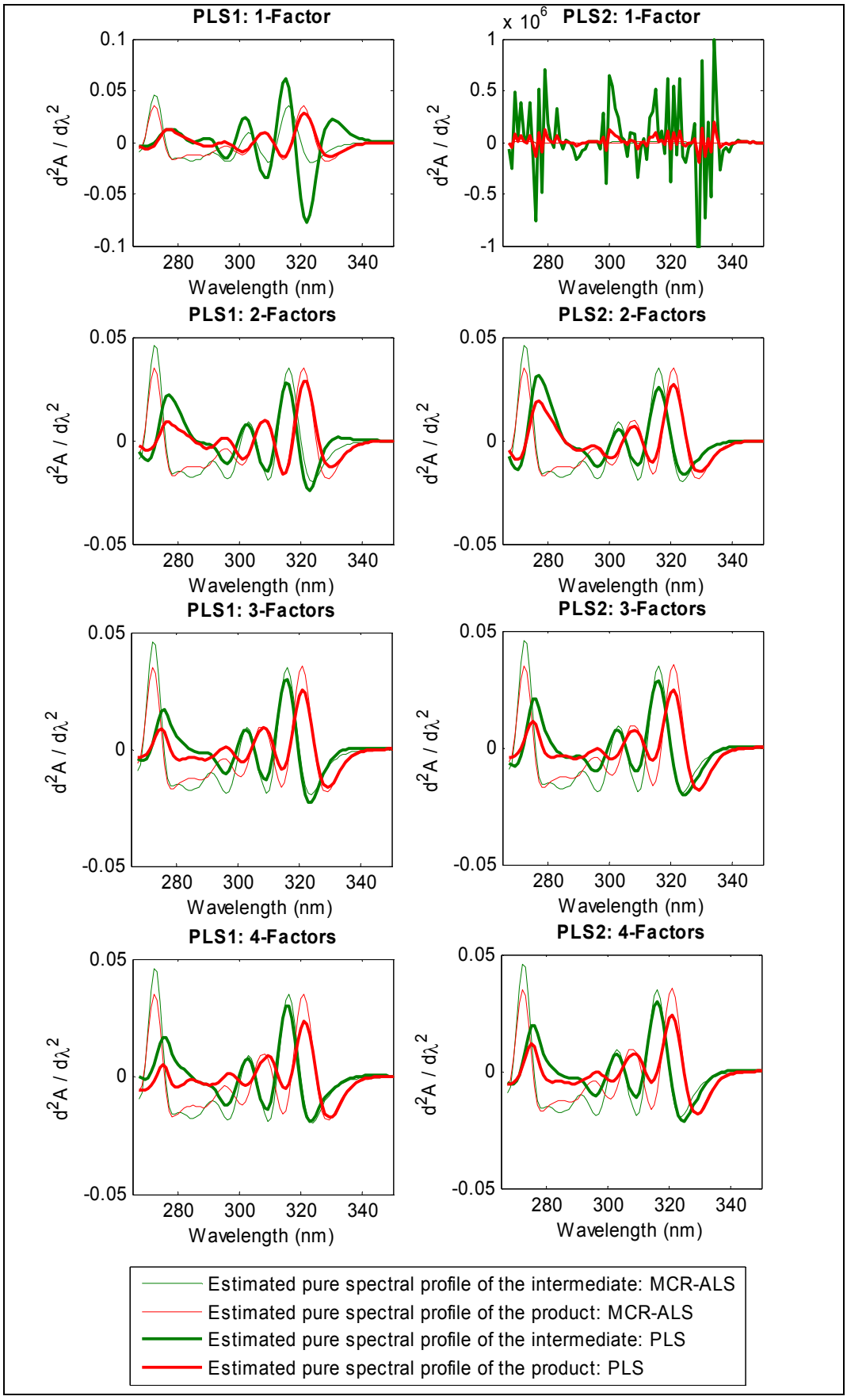


Figure 4.10: Comparison of the pure spectral profiles calculated from the PLS regression vectors (bold lines) and the corresponding pure component spectra derived using SMCR.

To quantify the differences between the PLS and MCR-ALS spectra, the root-mean-square error (RMSE) of the spectral residuals were calculated. These values are shown in Figure 4.11. Although the values obtained from a one-factor PLS2 model were several orders of magnitude larger than a one-factor PLS1 model, the subsequent factors produced more comparable values. The RMSE values for the intermediate approached a minimum starting at three factors and only showed marginal improvement with four or five factor models. The RMSE values for haloacetoxyone were minimised at three factors. It is interesting to note that the RMSE values were almost identical for PLS1 and PLS2 three-factor models. This confirmed that both methods were reproducing the pure component spectra with the same accuracy.

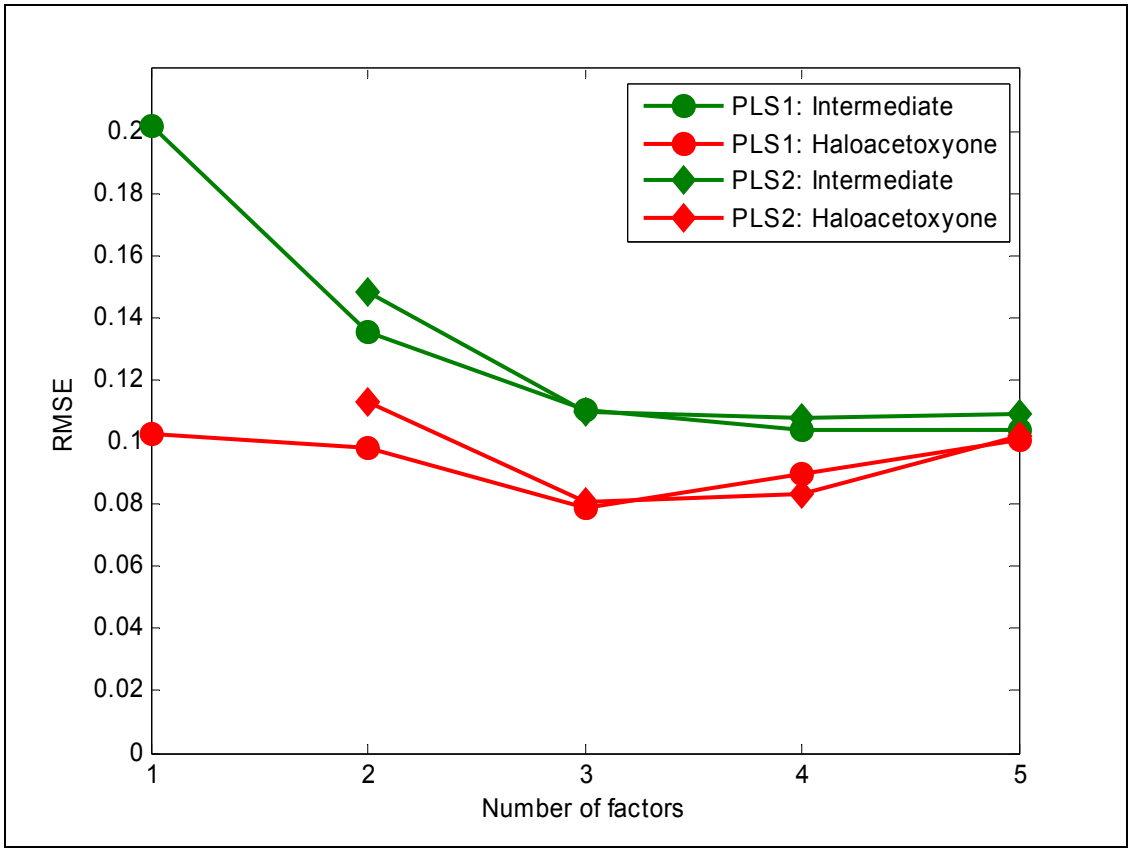


Figure 4.11: RMSE values calculated from the spectral residuals (MCR-ALS - PLS spectrum). The RMSE values obtained for PLS2 1-factor model were too large to show effectively on the same plot. The value for intermediate was 3.07×10^5, the value for haloacetoxyone was 5.57×10^5.

The difference between the MCR-ALS spectra and the pure spectral profiles calculated from the PLS regression vectors may have been a consequence of the way they were derived using each method. Both self-modelling curve resolution and PLS are based upon a bilinear model of the spectral data. The aim of curve resolution is to factorise the spectral data matrix into two appropriately sized matrices representing the pure analyte

concentration and spectral profiles. The factors recovered using curve resolution are optimised in a least-squares sense, *i.e.* they minimise the total sum of squares of the residual un-modelled part of the data. The factors recovered using curve resolution are not constrained to be orthogonal and represent a linear combination of all possible basis vectors spanning the data. If applied to the original data matrix, curve resolution therefore represents a full-rank model. However, PLS will search for a set of orthogonal basis vectors that both maximise the covariance of X / Y and the amount of variance explained in X and Y. These basis vectors form a subspace that span only part of the spectral space. Increasing the number of PLS factors would include additional contributions to the spectrum, subject to the above constraints of maximising the covariance.

4.1.8 Application of the UV method in a Large Scale Laboratory

The PLS model was used to monitor five batches over the period of four weeks. For the first batch, the model successfully produced concentration profiles similar to those observed in the laboratory. More importantly, the predicted end-point concentration showed good agreement with both the expected theoretical value and the actual measured value obtained using HPLC analysis (0.27 $mol.L^{-1}$). The results for batch 1 are shown in Figure 4.12(a). The profiles at the start of the reaction were not zero as expected and clearly show the point at which the addition of phosphorus oxychloride was added at 39 minutes.

The product concentration prediction results obtained from the subsequent four batches were not so accurate. Figure 4.12(b) shows the overlaid predicted product concentration profiles for all five batches. The batches were aligned by setting the time at which phosphorus oxychloride addition was started to t_0. The five overlaid batches all exhibit very similar profiles but the true concentrations were not correctly predicted. The largest prediction error attained was during batch three for which a final concentration of 0.22 $mol.L^{-1}$ was predicted; whilst the true measured value was 0.271 $mol.L^{-1}$ (an error of 18.8%).

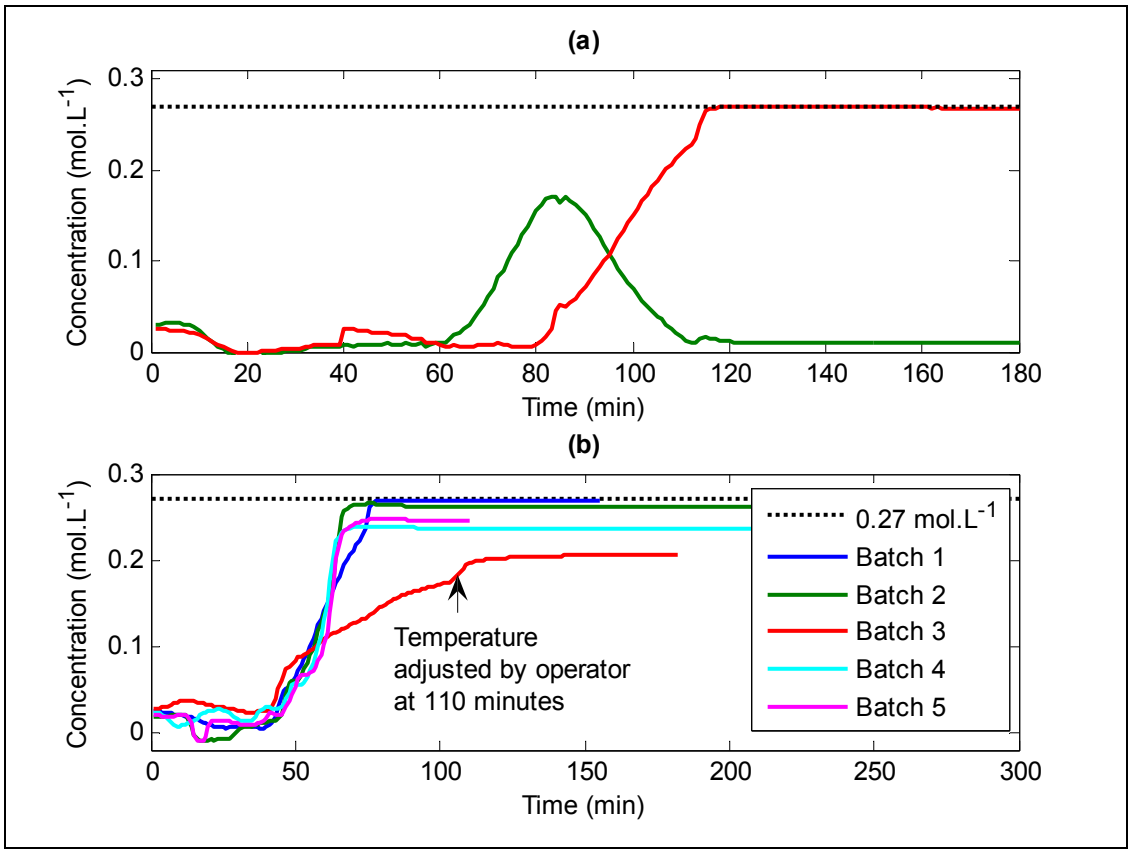


Figure 4.12: (a) Predicted concentrations for batch 1, manufactured in a Large Scale Laboratory; (b) Comparison of predicted product concentrations for 5 batches manufactured in a Large Scale Laboratory. The dashed line shows the theoretical end of reaction limit (0.27 mol.L^{-1}). The batches were aligned by setting the start of the phosphorus oxychloride addition to t_0. During the manufacture of batch 3, it was noted that the product (haloacetoxyone) profile was increasing slower than observed for previous batches. The batch temperature was checked and found to be 10°C lower than the process set-point. The temperature was set to 70°C at 110 minutes as indicated on the plot.

There were two potential experimental factors contributing to the observed prediction errors. The first factor was a consequence of the way in which the background spectrum was acquired for batches 2 to 5. In batch 1, the background spectrum was acquired with the probe installed on the vessel lid but not fixed to the full immersion depth, this allowed the tip to be positioned in the headspace above the charged reactants. Therefore both the probe and the fibre-optic cables were very close to their final orientation when the background spectrum was acquired. Unfortunately, this approach could not be adopted in subsequent batches so the reference spectra were acquired prior to installation of the probe. The change in the optical transmissitivity of the fibres when the probe was installed resulted in subsequent single beam spectra with higher or lower intensity than the background spectrum. Upon conversion to an absorbance spectrum, the difference between the intensities of the background and sample single beam spectra was observed as an increase or decrease in calculated absorbance values. This was usually observed as a baseline shift and could be removed by zero-minimum offset correction or transformation

of the spectra to their first or second derivative form. As the predictions were based upon second derivative spectra, the effect of fibre movement should have been minimised.

The major factor that contributed to the observed prediction error was the magnitude of the absorbance values of the measured spectra and their subsequent derivative spectra. Examination of the second derivative spectra indicated that the magnitude of the absorbance values for the final four batches were lower than those for the development reaction and batch 1. The peak positions and peak shapes for the spectra acquired in all batches showed excellent correlation but clearly displayed a scaling discrepancy. A variable baseline offset was present in all six data sets but was more severe in the LSL batches. Although this could have been caused by movement of fibres as discussed above, it could also suggest that probe fouling was occurring. A potential consequence of probe fouling is that the adsorbed solid can reduce absorbance by the continuous phase owing to a reduced penetration depth and available measurement area on the ATR crystal. The reduced absorbance values were believed to be the reason the values predicted using the PLS model exhibited the correct reaction profiles but the predicted concentrations were low.

Despite the problems with the quantitative predictions described above, the real-time data still proved to be very useful during manufacture. With the exception of batch 3 (the profile in Figure 4.12(b) marked with a dotted line), the qualitative shape of both the intermediate and product profiles for batches 2 to 5 showed good agreement with those observed in the laboratory and batch 1 in the large scale reactor. It was noted at the time of manufacture that the profile for batch 3 suggested that the reaction was proceeding more slowly than was observed for previous batches. This prompted the operators to check the process conditions and identify that temperature set-point was 10°C lower than required. The inflection point in the profile at approximately 110 minutes corresponds to the time at which the operators corrected the temperature set-point. The appropriate time at which to take a sample for off-line analysis to confirm the end-point was also correctly identified from the reaction profiles. Further confidence was taken from the ability to see the appearance and disappearance of the intermediate species as well as the appearance of the product.

4.1.9 Conclusions: Derivation and implementation of a PLS model using minimal reference data

An alternative approach for the derivation and implementation of a PLS model for *in-situ* reaction monitoring using UV spectroscopy was investigated. The reaction studied was the chlorination of acetoxyone using phosphorus oxychloride in toluene. The initial trial experiments showed that the UV spectroscopy was suitable for monitoring this reaction and additional experiments were performed to provide the necessary reference measurements. The solubility limit of acetoxyone at the start of the reaction under nominal reaction conditions was established by solution assay of the supernatant liquid. The concentration of haloacetoxyone (product) at the reaction end-point was measured using HPLC solution assay (0.27 $mol.L^{-1}$). The reaction was then repeated using the nominal process conditions to provide spectra that were subsequently used to construct a PLS model for process monitoring in a large scale laboratory facility.

Savitsky-Golay smoothing and differentiation (second derivative, 13-point smoothing) of the spectral data was adequate to remove baseline off-sets and also enhanced the differences between the spectra of the major constituents. PCA of the data revealed that the first two principal components accounted for 98.96% of the total variance. The scores profiles for the first two principal components provided a useful indication of the underlying concentration profiles of the intermediate and product. A custom Matlab function called `ResidualComps` was also written and applied to the second derivative spectra. The results confirmed that three principal components were sufficient to model the major contributions to the data. Additional components were simply modelling the minor contributions that arose from spectral peaks shifts and other non-linear effects.

Evolving factor analysis (EFA) was applied to the data and the results were used to confirm the rank of the data, and also to provide initial estimates of the concentration profiles. The output from the two-, three and four-component EFA models were compared and it was judged that the three-component model was most suitable. The EFA profiles were smoothed and scaled using the HPLC concentration data to provide the initial estimates of the concentration profiles for acetoxyone, intermediate and haloacetoxyone.

A Matlab script (`OPA`) was written to perform analysis using the orthogonal projection approach. OPA is not an eigenvalue based approach but identifies each possible pure component spectrum based upon it dissimilarity. This was found to be very sensitive to

the subtle peak shifts that were occurring. The direction of the peak shifting could be visualised by overlaying the OPA spectra. The significance of each OPA spectrum was assessed by examination of its corresponding dissimilarity profile. As the significance of each OPA spectrum decreased, it dissimilarity profile exhibited more noise. The Durban-Watson values are a measure of the autocorrelation in the dissimilarity profiles and indicated that three OPA components were appropriate. The OPA spectra corresponding to acetoxyone, intermediate and haloacetoxyone were scaled using the HPLC reference measurements. OPA was found to be a very powerful method because it allowed the effect of peak shifting to be visualised.

Multivariate Curve Resolution – Alternating Least Square (MCR-ALS) was used to combine the initial estimates derived using EFA and OPA. Although MCR-ALS could have been executed using either initial estimate of the concentration or the spectral matrix; using both sets of estimates with equality constraints helped to further constrain the final solutions produced. The final concentration and spectral profiles produced by the optimised MCR-ALS model could not be used directly for the real-time prediction of haloacetoxyone concentration from new reaction spectra. This was because the process monitoring software used by the UV spectrometer did not support the use of the MCR-ALS pure component spectral matrix for the prediction of new spectra. To overcome this restriction, the pure component spectra were converted to a set of PLS regression spectra using the corresponding matrix of concentration profiles from the MCR-ALS model to calculate the PLS model.

The performance of PLS1 versus PLS2 models using different numbers of factor were compared. Based upon the root mean square error of cross validation (RMSECV) values, a three-factor PLS1 model for the prediction of haloacetoxyone gave a RMSECV of 2.68×10^{-3} mol.L^{-1}, whilst the three-factor PLS2 model gave a RMSECV of 2.78×10^{-3} mol.L^{-1}. This corresponded to a prediction error of 0.99% (PLS1) or 1.02% (PLS2) at the reaction end-point. Although lower RMSECV values were achieved using four or five factors, the effect upon the corresponding PLS regression spectra was detrimental.

The purpose of calculating a PLS model was to transform the MCR-ALS pure component spectra into a set of PLS regression coefficients. The PLS pure spectral profiles were calculated using the PLS regression coefficients and compared with the MCR-ALS pure component spectra. Visually, it could be seen that the PLS pure spectral profiles correlated well with the MCR-ALS spectra and a three-factor PLS model offered the best fit. The

root mean square errors of the residual spectra for the intermediate and haloacetoxyone constituents reached a minimum at three factors and confirmed the above observations. The RMSE values for the haloacetoxyone residual spectra were almost equivalent for the PLS1 and PLS2 three-factor models. This confirmed that both PLS1 and PLS2 reproduced the pure component spectra with equal accuracy.

The outcome of using the derived PLS model for predicting new batches was a mixed success. The predicted concentrations from batch 1 showed good agreement with the expected values and the measured reference value provided by HPLC. The predicted concentrations for the remaining batches were less accurate. However, the resulting qualitative reaction profiles all displayed the same characteristic shape and successfully indicated the reaction end-point. The ability to observe the appearance and disappearance of the intermediate species provided further confidence when interpreting the qualitative profiles to judge the end-point. The apparent rate of intermediate and product formation also allowed the low set point temperature of batch 3 to be identified. The temperature was corrected and the batch successfully reached completion in the expected time.

The major cause of inaccuracy in the predictions was a consequence of probe fouling. Improved agitation of the reaction mixture would help to minimise probe fouling, but ultimately it is an engineering problem and new probe designs are required that offer some form of in-process cleaning.

4.1.10 Testing of a custom fibre optic cable assembly

4.1.10.1 Comparison of the relative transmission of the standard and custom fibre optic cables

To assess the relative sensitivity of each detector, the 2-norm of each energy spectrum was calculated. The ratio of the 2-norms calculated for the sample and by-pass channels were a measure of the relative sensitivity of each detector.

Table 4.2: Table showing the 2-norm values calculated from the single beam energy spectra acquired using the standard and custom fibre assemblies. The ratio of the 2-norm values were a measure of the relative sensitivity of detector 1 to detector 2

Cable (fibre)	Detector	2-norm	Ratio of 2-norm (Sample:Bypass)
Standard (A)	Sample	345728.0	0.869
Standard (A)	Bypass	398069.9	
Standard (B)	Sample	341499.3	0.829
Standard (B)	Bypass	411841.0	
Custom (B)	Sample	127923.3	0.871
Custom (B)	Bypass	146841.0	
Custom (D)	Sample	378147.1	0.849
Custom (D)	Bypass	445653.4	

The table above shows that for the same integration time and fibre assembly, the relative response of the sample channel to the response of the bypass channel was approximately 0.85 (mean, $n = 4$). The results also showed that the throughput of standard fibres A and B were well matched but the throughput of custom fibre B (illumination) was significantly lower than custom fibre D (return). The relative response of the custom fibre B to the response of custom fibre D was 0.334 (mean, $n = 2$). This difference in throughput may have been a consequence of the illumination cable containing two fibres (A and B) which were positioned side-by-side. As the two fibres were not exactly in the centre of the cable, the focused light from the source lamp may not have been illuminating the fibres as efficiently as a single fibre core. In a future design a single bifurcated core (splitting into A and B) could be used as this would allow the single core to be positioned in the centre of the SMA connector and would improve the illumination efficiency

4.1.10.2 Optimisation of the internal reflection mirror position

The energy spectra of the sample and bypass channels for the standard fibre and custom fibre configurations are shown in Figure 4.13. The spectra acquired using the standard fibre assembly were well matched by using a 10% attenuation filter on the bypass channel to match the attenuation of the process ATR probe on the sample channel. The integration time used to acquire the spectra with the standard fibre assembly was 100 ms.

The initial fibre / instrument configuration for tuning the position of the mirror in the custom fibre assembly did not include an attenuation filter on either the sample or bypass channels. The justification for this was that the initial position of the mirror was not optimised and that tuning the position of the mirror would be sufficient to match the throughput of the sample and bypass channels. Figure 4.13(b) shows that with the tuning mirror in the optimum position, the throughput of the sample channel was still significantly lower than the bypass channel. This showed that the reduction in transmission through the bypass fibres and mirror assembly was even greater than when the sample channel included a process ATR probe. It was believed that the main cause of the reduced transmission through the bypass fibres was that the fibres in the mirror assembly were not terminated with collimating lenses. This in turn resulted in a poor collection efficiency of the return fibre C as the light was emitted from fibre A and reflected from the mirror. In a future design, the efficiency could be improved by incorporating collimating lenses as used in UV and NIR transflectance probes.

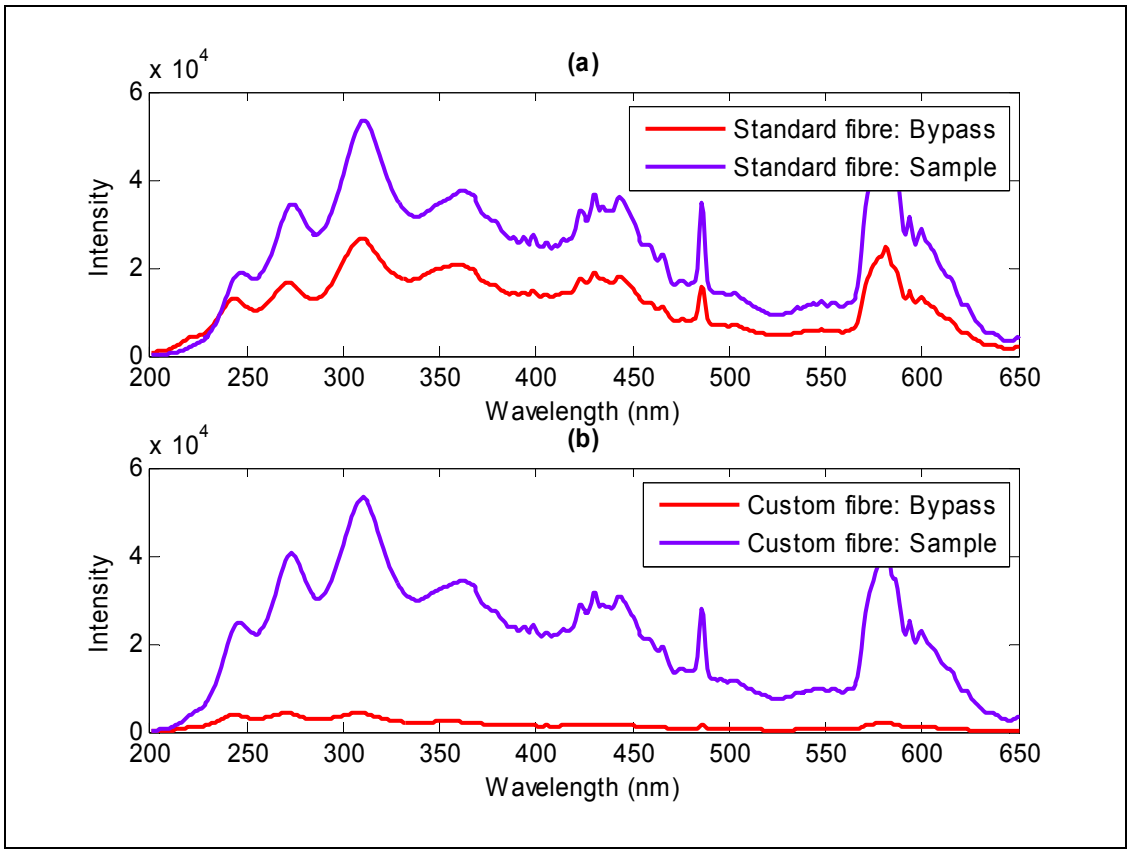


Figure 4.13: Comparison of the energy spectra acquired for the sample and bypass channels using the standard (a) and custom (b) fibre assemblies. The spectra were acquired with the fibres coupled directly to a process ATR probe.

4.1.10.3 Quantification of fibre transmission as function of cable displacement

The 2-norms were calculated for each spectrum acquired and the mean 2-norm value determined for each fibre / displacement / detector combination from the corresponding set of replicates (n = 4). The results are shown in Figure 4.14.

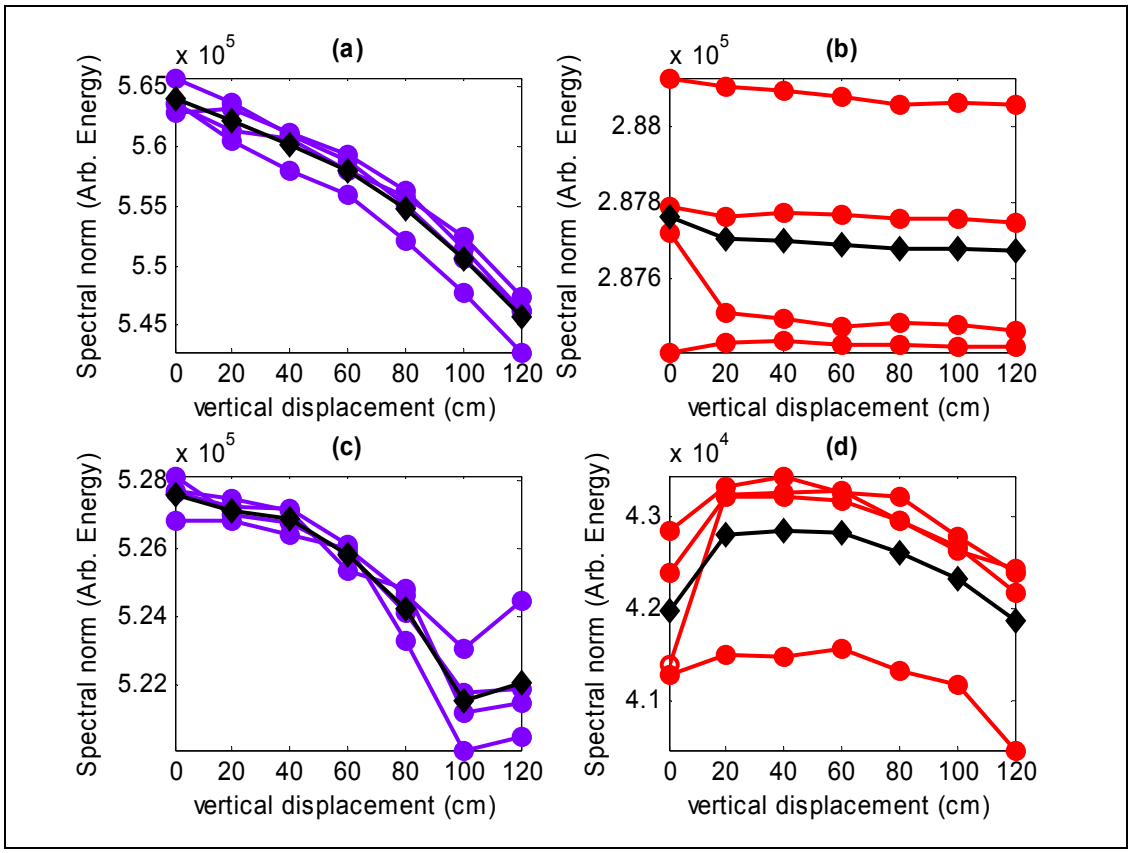


Figure 4.14: The effect of vertical displacement on the transmission of the standard and custom fibre assemblies: (a) 2-norm values for the replicate measurements acquired using standard fibres – sample channel; (b) 2-norm values for the replicate measurements acquired using the standard fibres – bypass channel; (c) 2-norm values for the replicate measurements acquired using the custom fibre assembly – sample channel; (d) 2-norm values for the replicate measurements acquired using the custom fibre assembly – bypass channel. In each plot the line drawn using diamond markers (◊) shows the corresponding mean values

The 2-norm values obtained from the standard fibre assemblies are shown in plots (a) and (b) and exhibit the anticipated profiles. In plot (a), the overall spectral intensity for the sample channel decreased as the displacement of the fibre was increased. This clearly confirmed that as the radius of the bend in the fibres was increased, the transmission decreased. The profiles for the bypass channel are shown in plot (b). As expected, the profiles are almost horizontal lines and confirm that displacing the fibres connected to the probe did not have any effect upon the throughput of the bypass channel. This was expected as the bypass fibre was contained within the instrument cabinet and should not have been sensitive to movement of the sample fibre.

The 2-norm values obtained from the custom fibre assemblies are shown in plots (c) and (d). In plot (c) the spectral intensity for the sample channel decreased as the displacement of the fibre was increased and reflects the behaviour observed for the standard fibre assembly in plot (a). However, the profiles of the bypass channel for the custom fibre

assembly shown in plot (d) are not horizontal lines but show behaviour similar to the sample channel. As the transmission of the bypass channel should also be affected by displacement of the fibres, the profiles of the lines in plots (c) and (d) should be very similar and this was observed by the curvature of the profiles in plot (d)

Using the mean 2-norm values, the ratio of the sample to bypass channel values were calculated for each fibre / displacement. The results are plotted below in Figure 4.15.

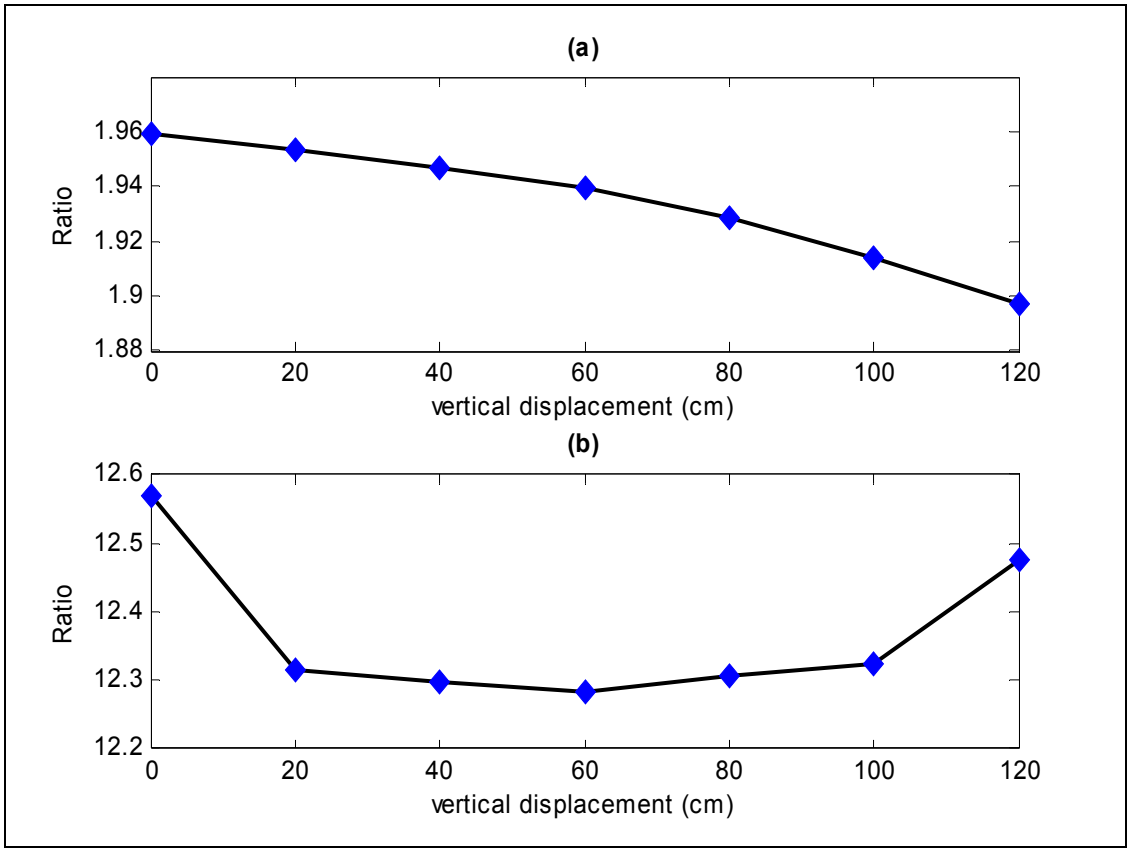


Figure 4.15: The effect of vertical displacement on the mean ratio of spectral intensity (sample to bypass) for: (a) Standard fibre assembly; (b) Custom fibre assembly.

Plot (a) shows the mean ratio values for the standard fibres and again exhibited the familiar decrease in transmission of the sample channel relative to the bypass channel as the radius of the bend in fibres was increased. The mean ratio values for the custom fibre assembly are shown in plot (b). If the fibre assembly was completely eliminating all effects of movement, the plot would be a horizontal line with an intercept at 12.56 (the ratio of the sample to bypass channel at 0 cm displacement).

The scale on the ordinate axes of plots (a) and (b) reflect the difference in the intensity of the sample and bypass spectra measured for the standard and custom fibre assemblies.

The intensity of sample and bypass spectra were more closely matched for the standard fibres so the ratio values are smaller than those for the custom fibres.

As the ratio values were different for the two sets of fibres, the relative standard deviation was used to express the relative amount of variation across the two profiles.

Cable	Standard deviation (n = 7)	Mean (n = 7)	%RSD
Standard	0.0226	1.9341	1.169
Custom	0.1094	12.3676	0.885

The table shows that the relative standard deviation across the range 0 to 120 cm for the custom fibre assembly was lower than for the standard fibre assembly. The increase in robustness of the custom fibre relative to the standard fibre was 24.3%.

The result indicates that the custom fibre assembly did not completely eliminate the effect of movement upon the relative transmission of the sample and bypass fibres. However, the custom fibre still offered a 24.3% improvement in the robustness to movement relative to a standard fibre assembly.

4.1.10.4 The effect of fibre movement upon the calculated absorbance spectra

The energy spectra acquired in the previous experiments were used to calculate the absorbance spectra. These are shown in Figure 4.16. Plot (a) shows that as the vertical displacement was increased, the absorbance spectrum displayed a positive translation along the ordinate axis. This was confirmed by plot (c) which shows the mean value of each spectrum plotted against displacement. The spectra in plot (b) were acquired using the custom fibre assembly and show much smaller translations along the ordinate axis. This was again confirmed by the mean values shown in plot (d). With the exception of the point at 120 cm, the mean values are much closer to a horizontal line and indicate lower sensitivity to movement of the fibres. The shape of the spectral profiles in plots (a) and (b) should also be considered. In plot (a) the slope of the spectrum over region 220 to 350 nm increased as the displacement was increased, resulting in a change of the spectral profile. However, the spectral profiles shown in plot (b) are all very similar; indicating that movement of the fibres did not affect the resulting absorbance spectra as much.

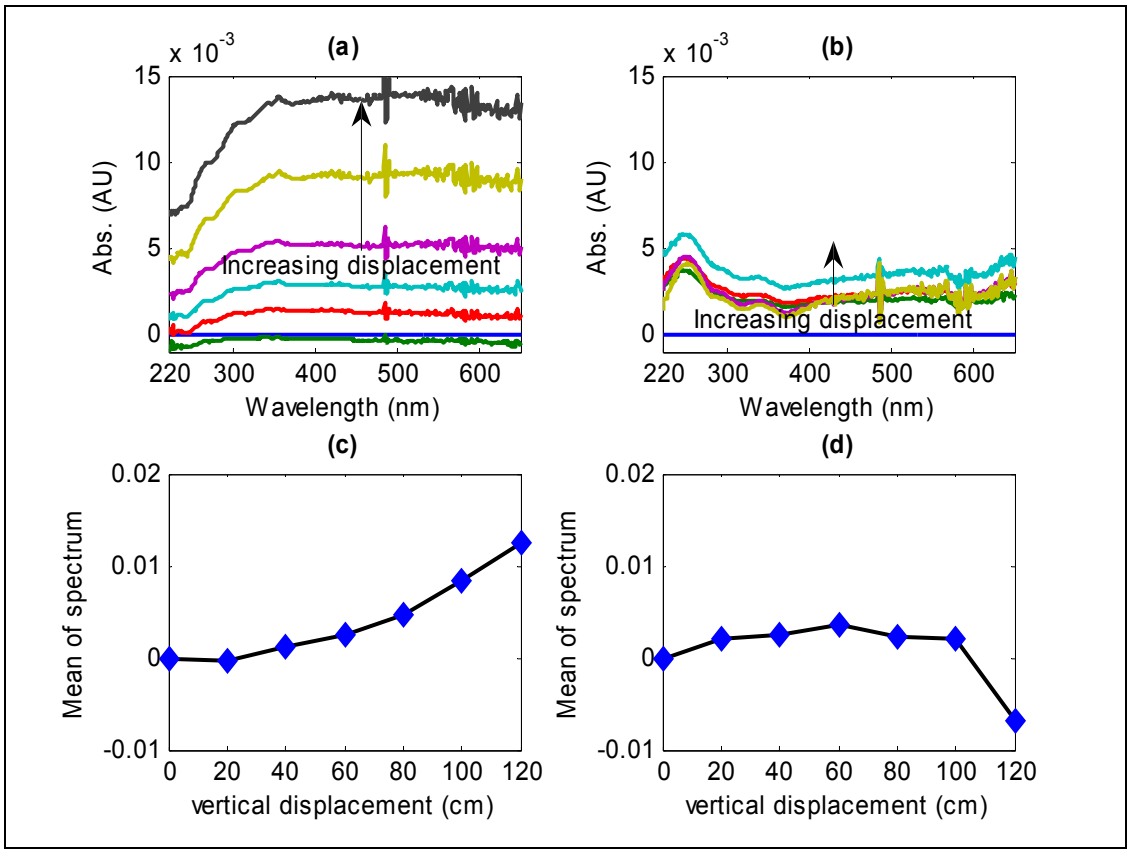


Figure 4.16: (a) Absorbance spectra at various displacements acquired using standard fibres; (b) Absorbance spectra at various displacements acquired using custom fibre assembly; (c) Mean value of each spectrum acquired using the standard fibres plotted against the displacement; (d) Mean value of each spectrum acquired using the custom fibre assembly plotted against the displacement.

4.1.10.5 Quantification of the effect of fibre movement upon CLS calculations

The aim of this simulation experiment was examine the effect of the baseline movement upon estimated concentration profiles calculated from a synthetic data set using CLS. A two-component synthetic reaction data set based upon the chlorination of acetoxyone was created. The smoothed concentration vectors are shown in Figure 4.17(a). The original, zero-order spectra used to represent pure component spectra for the intermediate and haloacetoxyone species are shown in Figure 4.17(b). The spectra were scaled to unit molarity. The synthetic reaction data set was generated by calculating the outer product of the concentration and spectral matrices. The resulting spectral data matrix contained the spectral variation contributed solely by the absorbing species in the absence of any additional system or measurement noise. This matrix therefore represented the underlying chemical spectra and was independent of additional baseline or instrument noise artefacts. The resulting reaction spectra are shown in Figure 4.18(a). Every tenth spectrum is shown to improve clarity.

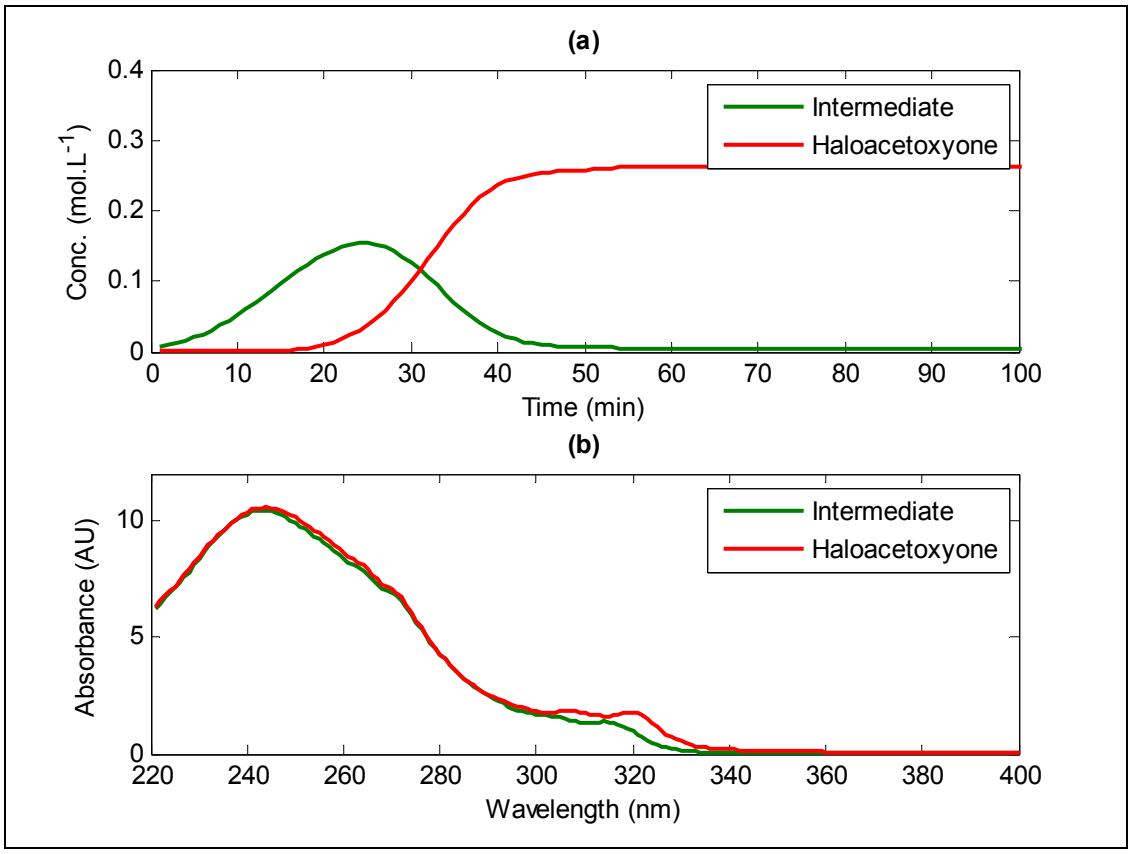


Figure 4.17: (a) Smoothed concentration vectors used to synthesise a two-component UV reaction data set; (b) unit molarity absorbance spectra for intermediate and haloacetoxyone used to synthesise a two-component UV reaction data set.

A matrix of randomly ordered baseline spectra from each set of the fibres was added to the synthetic reaction spectra $\boldsymbol{X}_{sim}$ to give the final data matrices $\boldsymbol{X}_{std}$ and $\boldsymbol{X}_{cust}$. The baseline variation exhibited by each set of fibres is shown in Figure 4.18(b). As demonstrated in the previous sections, the variation in baseline offset exhibited by the standard fibres was greater than the custom fibres. The maximum baseline offset was 15 mAU for the standard fibres and 5 mAU for the custom fibres. The baseline contribution from the standard fibre therefore contributed up to ~25% of the signal intensity for the low intensity spectra (when maximum absorbance was ~ 0.06 AU) at the start of the reaction, and up to ~2.5% of the signal intensity for the high intensity spectra (when maximum absorbance was ~ 0.60 AU) at the end of the reaction. The custom fibres exhibited less variation and contributed up to ~8.3% of the signal intensity for the low intensity spectra and up to ~0.083% of the signal intensity for the high intensity spectra.

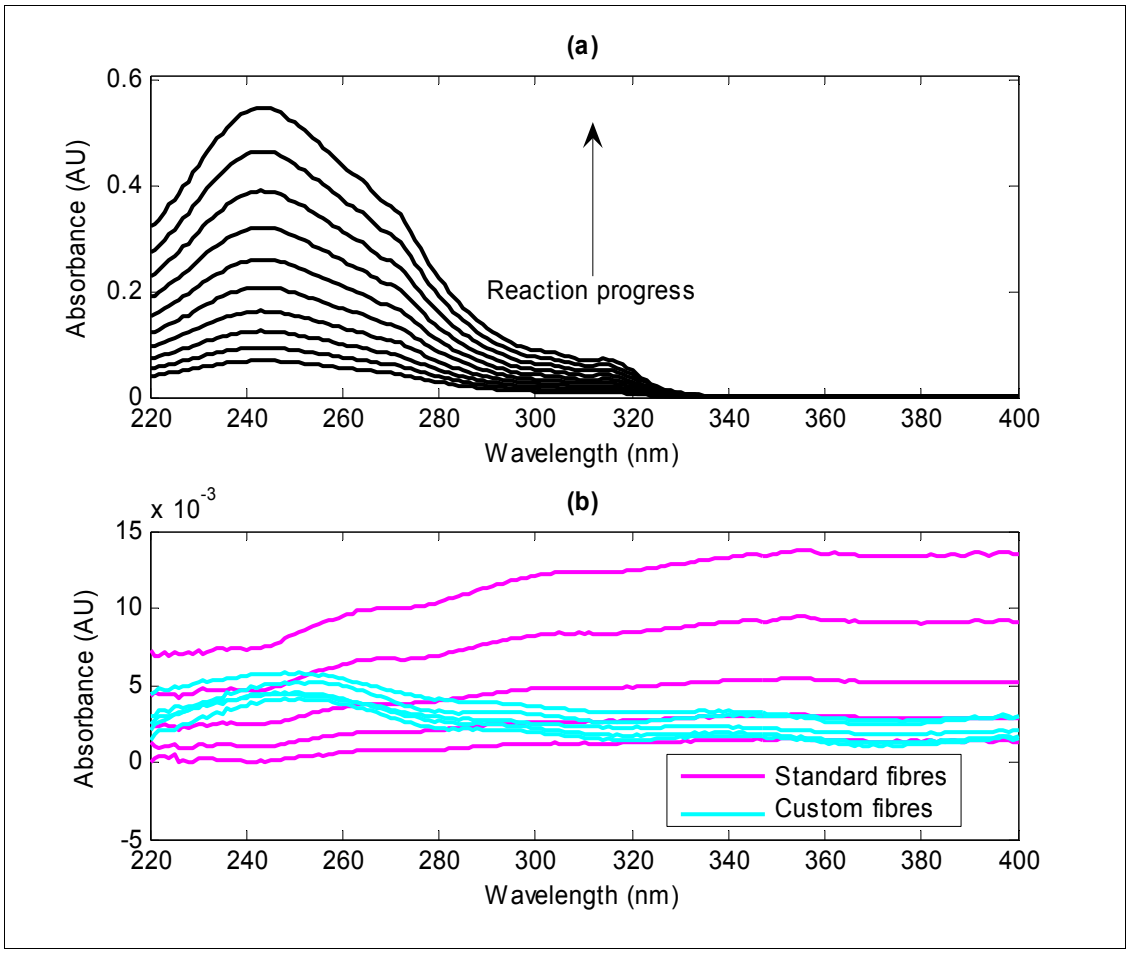


Figure 4.18: (a) Synthetic UV reaction spectra before addition of baseline noise - every tenth spectrum is shown for clarity; (b) baselines for each fibre type that were added to each synthetic data set. The baselines calculated from the double beam energy spectra acquired using standard fibres exhibit greater variation than those acquired with custom fibres.

To assess the effect of the variation contributed by the baselines on least-squares calculations, estimates of concentration profiles calculated using $\boldsymbol{X}_{std}$ and $\boldsymbol{X}_{cust.}$ were compared. The concentration profiles were calculated directly from the spectral data matrices using the pure component spectra and are shown in Figure 4.19. The concentration profiles estimated from the un-processed absorbance spectra with baseline contributions calculated from the standard and custom fibre assemblies are shown in Figure 4.19(a) and Figure 4.19(b) respectively. The difference in the level of noise present in the two sets of concentration estimates can be clearly observed. This shows that although the baseline variation appeared to be quite a small contribution to the overall signal intensity, it had a large effect on the subsequent least-squares calculations. The concentration profiles estimated from the second-derivative spectra, shown in Figure 4.19(c) and (d) had lower noise contributions relative to the estimates derived from the un-

processed absorbance spectra. This was not surprising as transformation of the spectra to their second-derivative form removed a constant offset and a linear sloping baseline.

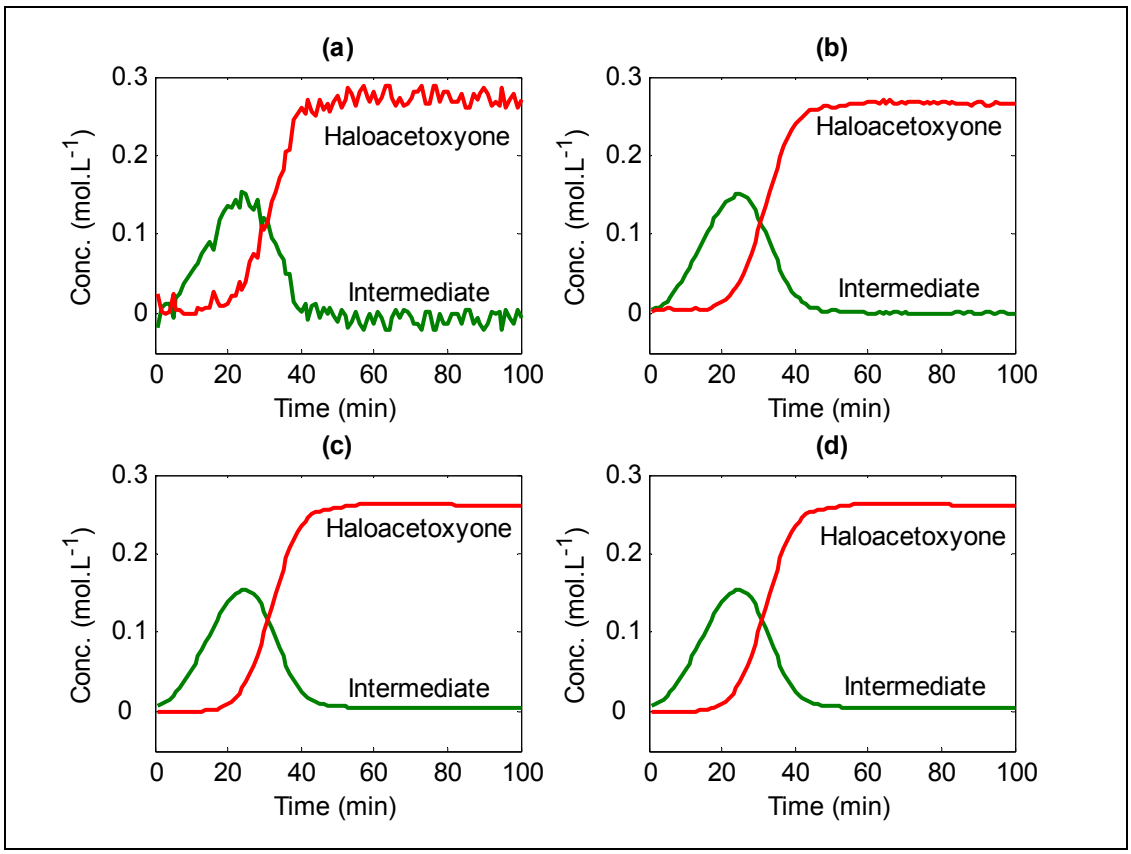


Figure 4.19: Least-squares estimates of concentration profiles calculated from synthetic spectral data with baselines added. (a) un-processed spectra with baseline contribution from standard fibre; (b) un-processed spectra with baseline contribution from custom fibre; (c) second-derivative spectra with baseline contribution from standard fibre; (d) second-derivative spectra with baseline contribution from custom fibre.

To quantify the differences between the estimated and actual concentration profiles, the root mean squared deviation (RMSD) was calculated for each set of results. These are shown in Figure 4.20. The RMSD values support the observations based upon the visual examination of the noise present in the estimated concentration profiles. For the un-processed spectra, the RMSD value decreased by approximately 69% when the profiles were estimated using the spectral data containing the reduced baseline contribution from the custom fibres. This demonstrated that although the contribution of the baseline in both data sets appeared small, the accuracy of least-squares calculations was improved considerably when data from the custom fibres were used. The RMSD values for the second-derivative spectra were two to three orders of magnitude lower than those for the unprocessed spectra. This was a consequence of two major factors: the transformation to second-derivative spectra removed a constant off-set and linear sloping baseline so the

effect of fibre movement was greatly attenuated; and transformation to second-derivative spectra also enhanced the differences between the two pure component spectra and therefore aided the calculation of their concentration profiles using least-squares.

Although the effect of the baseline variation had been greatly reduced by transformation to second-derivative spectra, the RMSD values for the custom fibre data still showed a 31% improvement relative to the standard fibre data.

This simulation experiment suggested that using a by-pass channel to actively compensate for movement of fibres, baseline artefacts in the unprocessed data were reduced. Consequently the error of least-squares calculations were reduced, even if spectral pre-processing was used to eliminate baselines.

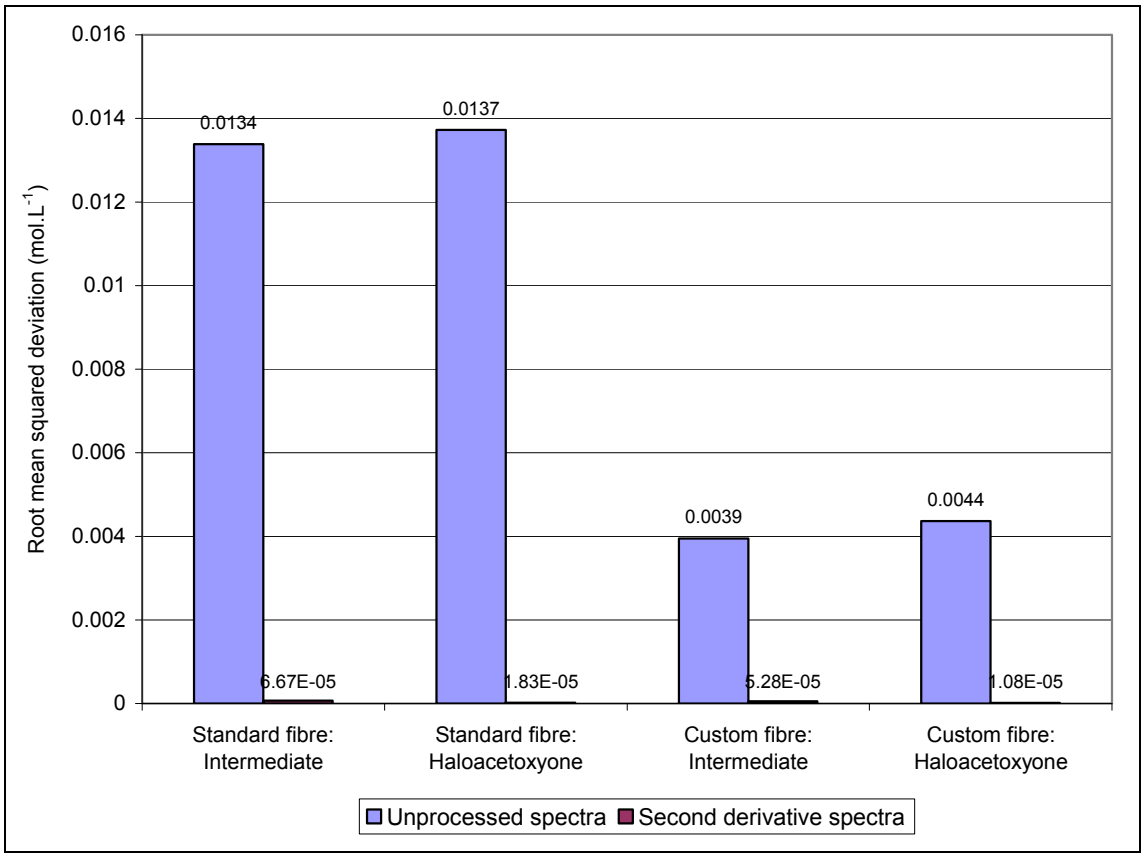


Figure 4.20: Root-mean-standard-deviation values for concentration profiles estimated using least-squares.

4.1.11 Conclusions: Testing of a custom fibre optic cable assembly

A custom fibre assembly was designed to investigate whether the effects of fibre movement could be removed from the subsequent absorbance spectra. An experiment was devised to directly compare the effect of fibre movement on a standard fibre cable and the custom fibre assembly. Comparing the optical throughput of the various legs of the custom assembly suggested that the transmission of the cable containing two fibres (fibres A and B) was approximately 34% the transmission of the cable containing a single fibre (fibre D). This was perhaps a consequence of the two fibres being off-centre and not illuminated as well as a single fibre positioned in the centre of the SMA connector. The use of a bifurcated fibre with a single illuminated leg may help to overcome this in future designs. The use of a mirror to reflect the bypass channel light did work in principle but also resulted in poor optical throughput relative to a closed loop. It is possible that using collimating lenses to collimate the light illuminating and reflected from the mirror could improve the throughput in a future design. The results obtained by measuring the transmission after vertical displacement of the cable to introduce small bend radii indicated that the custom fibre assembly reduced the effect of fibre movement by approximately 25% relative to the standard fibre assembly. Furthermore the shape of the resulting absorbance spectra showed much less variation for the custom fibre assembly compared to the standard fibres. Although, the custom fibre did not completely remove the effects of fibre movement, it demonstrated that transmitting the bypass fibres along the same outer cables as the sample fibres does help to reduce some of the effects of movement and led to more reproducible spectra. There are clearly areas for improvement that could help to refine the design. The effect of the baseline variation obtained from a standard fibre and the custom fibre assembly were compared using a synthetic data set. This simulation experiment suggested that use of a by-pass channel to actively compensate for movement of fibres reduced baseline artefacts in the unprocessed data, and also reduced the error in subsequent least-squares calculations. For the data used in this experiment, the RMSD values for the estimated concentration profiles improved (decreased) by approximately 69%. Even if spectral pre-processing was used to eliminate baselines, the RMSD values suggest that an improvement of approximately 31% was still achieved using the custom fibre assembly.

4.2 Vectorised Kalman filtering for SMCR

4.2.1 *N*-benzylation of 1*H*-indole using benzyl bromide

The *N*-benzylation of 1*H*-indole using benzyl bromide was used as a typical reaction for model development. This reaction was selected because it used an inorganic base producing a heterogeneous reaction mixture typical of those encountered in many industrial processes. The reference spectra of the two reactants and product were acquired and used to create a synthetic data set that was used for subsequent algorithm development and testing. Three reactions were then performed using the nominal amount of cesium carbonate base (2.00 equivalents with respect to the nominal amount of 1*H*-indole). The molar ratio of the two major reactants was varied to give molar ratios 0.67, 1.00 and 1.50 equivalents of 1*H*-indole with respect to benzyl bromide.

4.2.2 Analysis of reaction samples using HPLC

During each of the experiments listed in section 3.3, several reaction samples were extracted and analysed using reversed-phase HPLC. The sample solutions were assayed against external reference standards to determine the solution concentration of 1*H*-indole, benzyl bromide and 1-benzyl-1*H*-indole. The retention times of the three components 1*H*-indole, benzyl bromide and 1-benzyl-1*H*-indole were 6.35 minutes, 9.41 minutes and 11.85 minutes respectively. The solvent front eluted at 0.80 minutes. Typical chromatograms obtained from the HPLC analysis of the assay standards and reaction samples are shown in Figure 4.21.

The heterogeneous nature of the reaction mixture made the extraction of representative samples for off-line analysis difficult; this in turn introduced additional error into the estimated concentration values.

4.2.2.1 Solution assay of reaction samples

The consumption of 1*H*-indole, benzyl bromide and 1-benzyl-1*H*-indole was determined by solution assay using external standards of known concentration. Figure 4.22 shows the calculated concentration values plotted against time for three reactions using different molar ratios and equivalents of base.

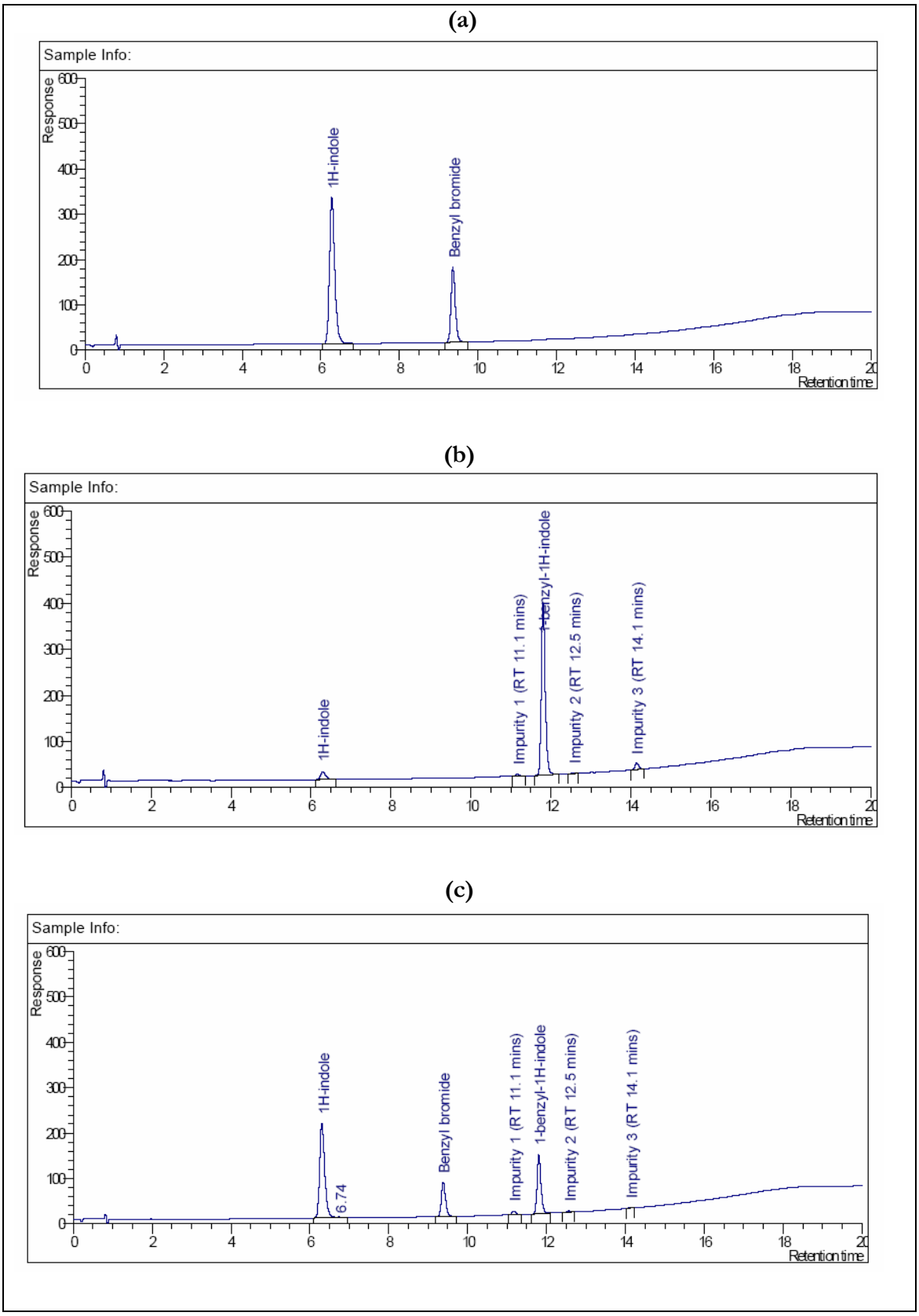


Figure 4.21: Typical chromatograms obtained from the HPLC analysis of assay standards and reaction samples for the *N*-benzylation of 1*H*-indole reaction: (a) mixed assay standard containing 1*H*-indole and benzyl bromide; (b) 1-benzyl-1*H*-indole assay standard; (c) a reaction sample.

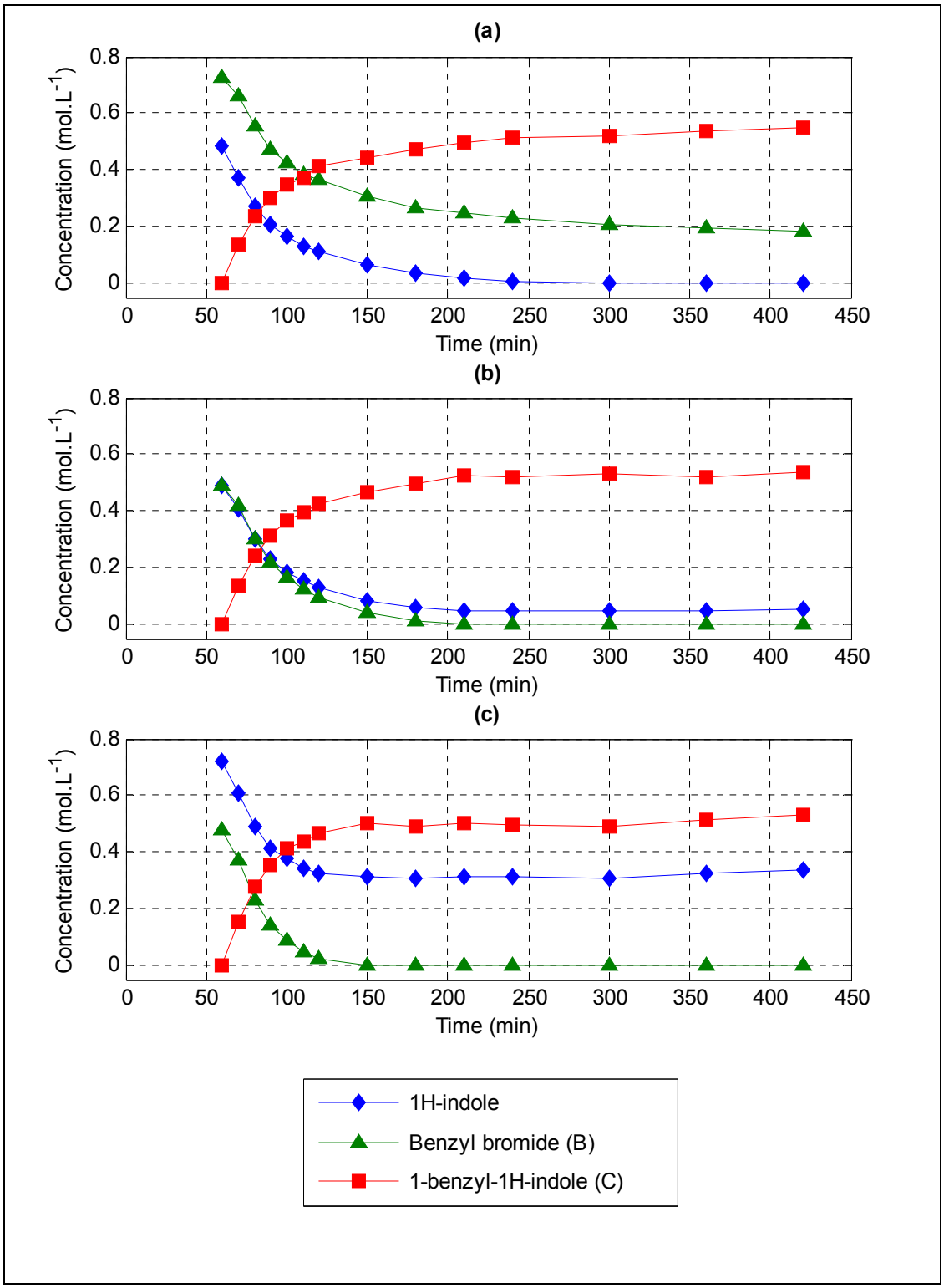


Figure 4.22: Concentration profiles derived from HPLC solution assay of reaction samples: (a) experiment BnIndole_B2.0_MR0.67 using 2.00 equivalent of base and a molar ratio of 1*H*-indole to benzyl bromide of 0.67; (b) experiment BnIndole_B2.00_MR1.00 using 2.00 equivalents of base and a molar ratio of 1*H*-indole to benzyl bromide of 1.00; (c) experiment BnIndole_B2.00_MR1.50 using 2.00 equivalents of base and a molar ratio of 1*H*-indole to benzyl bromide of 1.50.

4.2.3 Examination of reference standard spectra

To provide reference measurement functions (reference spectra) for Kalman filtering and to create a simulated data set, spectroscopic reference standard solutions of acetonitrile, tetrabutylammonium bromide, cesium carbonate, 1*H*-indole, benzyl bromide and 1-benzyl-1*H*-indole were prepared at typical reaction concentrations. The UV spectra acquired are shown in Figure 4.23; the Raman spectra acquired are shown in Figure 4.24.

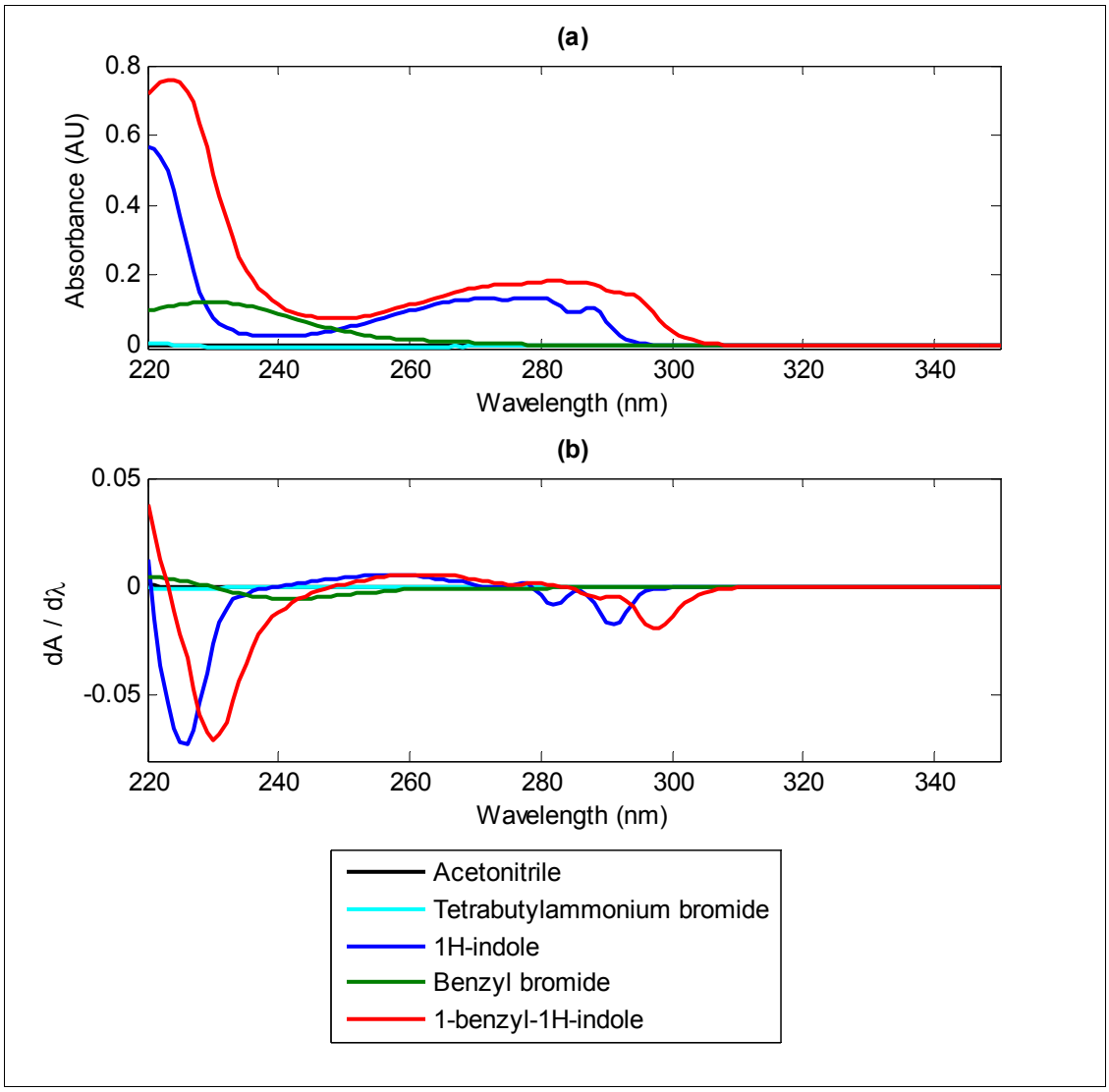


Figure 4.23: UV spectra of the main components in the N-benzylation of 1H-indole reaction: (a) un-processed absorbance spectra; (b) Savitsky-Golay first derivative absorbance spectra.

The UV spectra in Figure 4.23(a) show that over the spectral range 220 to 350 nm, acetonitrile and tetrabutylammonium bromide wereare UV inactive and did not have any significant absorbance features. Benzyl bromide did have a significant absorbance feature in the region 220 to 270 nm but it was completely overlapped by the spectra of 1*H*-indole and 1-benzyl-1*H*-indole. The difference between the spectra of 1*H*-indole and 1-benzyl-

1*H*-indole can be observed clearly in first derivative spectra shown in Figure 4.23(b), particularly in the regions 220 to 240 nm and 280 to 310 nm. However the spectra still exhibit considerable overlap. The broad absorbance feature of benzyl bromide was attenuated significantly by transformation to the first derivative spectrum. Since benzyl bromide was one of the main components of interest, data analysis was restricted to the use of the original spectra only.

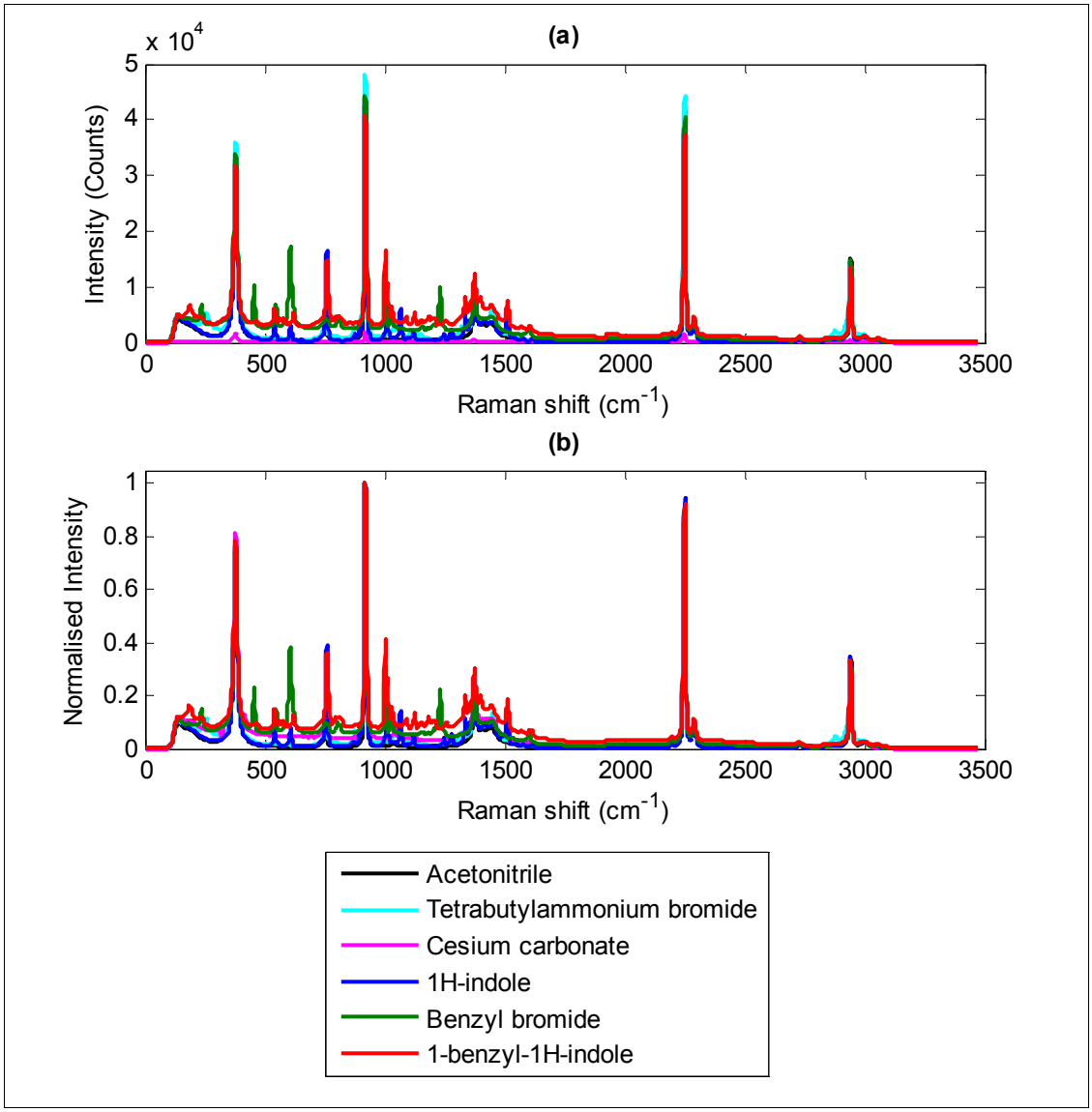


Figure 4.24: Raman spectra of the main components in the N-benzylation of 1H-indole reaction: (a) original un-processed Raman spectra; (b) Raman spectra normalised to maximum = 1.

The un-processsed Raman spectra shown in Figure 4.24(a) indicate that the peaks with the largest intensity corresponded to the C-C stretch of acetonitrile at 919 cm^{-1} and the C≡N stretch of acetonitrile at 2250 cm^{-1}. As anticipated, the intensity of the cesium carbonate spectrum was much weaker than the homogeneous solutions because the scattered Raman signal was attenuated by the suspended solid. The spectra shown in Figure 4.24(b) are

normalised to maximum = 1. The cesium carbonate spectrum had a weak feature at 1045 cm^{-1}, corresponding to carbonate (CO_3^{2-}) or bicarbonate (HCO_3^-) ions but had no significant features that could distinguish it from the acetonitrile spectrum. The spectrum of benzyl bromide has several small peaks that were not present in the other spectra. These include 237 cm^{-1}, 455 cm^{-1} (C-Br stretch), 608 cm^{-1} (ring deformation of monosubsituted benzene), 1230 cm^{-1} and 1602 cm^{-1}.

4.2.4 Preparation and pre-processing of the UV spectral data

To prepare the UV data for subsequent analysis using Kalman filtering, it was necessary to apply two forms of spectral pre-processing to remove unstructured variation. Many of the spectra contained narrow, high intensity peaks that were believed to be an artefact resulting from a skipped wavelength as the monochromator scanned across the across the wavelength range. A custom Matlab script, `MedianFilter.m`, was used to remove these spikes by applying moving window median filtering to each spectrum using a window size of three. The effect of pre-processing the UV spectra using median filtering is shown in Figure 4.25. Figure 4.25(a) shows the un-processed UV spectra with the characteristic spikes where the absorbance dropped to zero. Figure 4.25(b) shows the median filtered spectra obtained using a window width of three. All the useful spectral features were allowed to pass through the median filter, preserving the structured variation. Figure 4.25(c) shows the residual spectra calculated by subtraction of the median filtered spectra from the original spectra. The residual spectra correspond to the unwanted spikes.

The median filtered absorbance spectra still had a significant baseline contribution that increased as the reaction proceeded. The baseline was believed to be a consequence of probe fouling and was assumed to have a linear sloped baseline contribution that could be corrected by subtraction of a linear baseline with the function $f(x) = b_0 + b_1 x$ from each spectrum. The Matlab functions `polyfit` and `polyval` were used to calculate the polynomial coefficients b_0 and b_1 over the wavelength range 320 to 370 nm and extrapolate each baseline over the full spectral range 220 to 400 nm. The extrapolated baselines were subtracted from their corresponding spectra to give the final pre-processed data. This is illustrated in Figure 4.26.

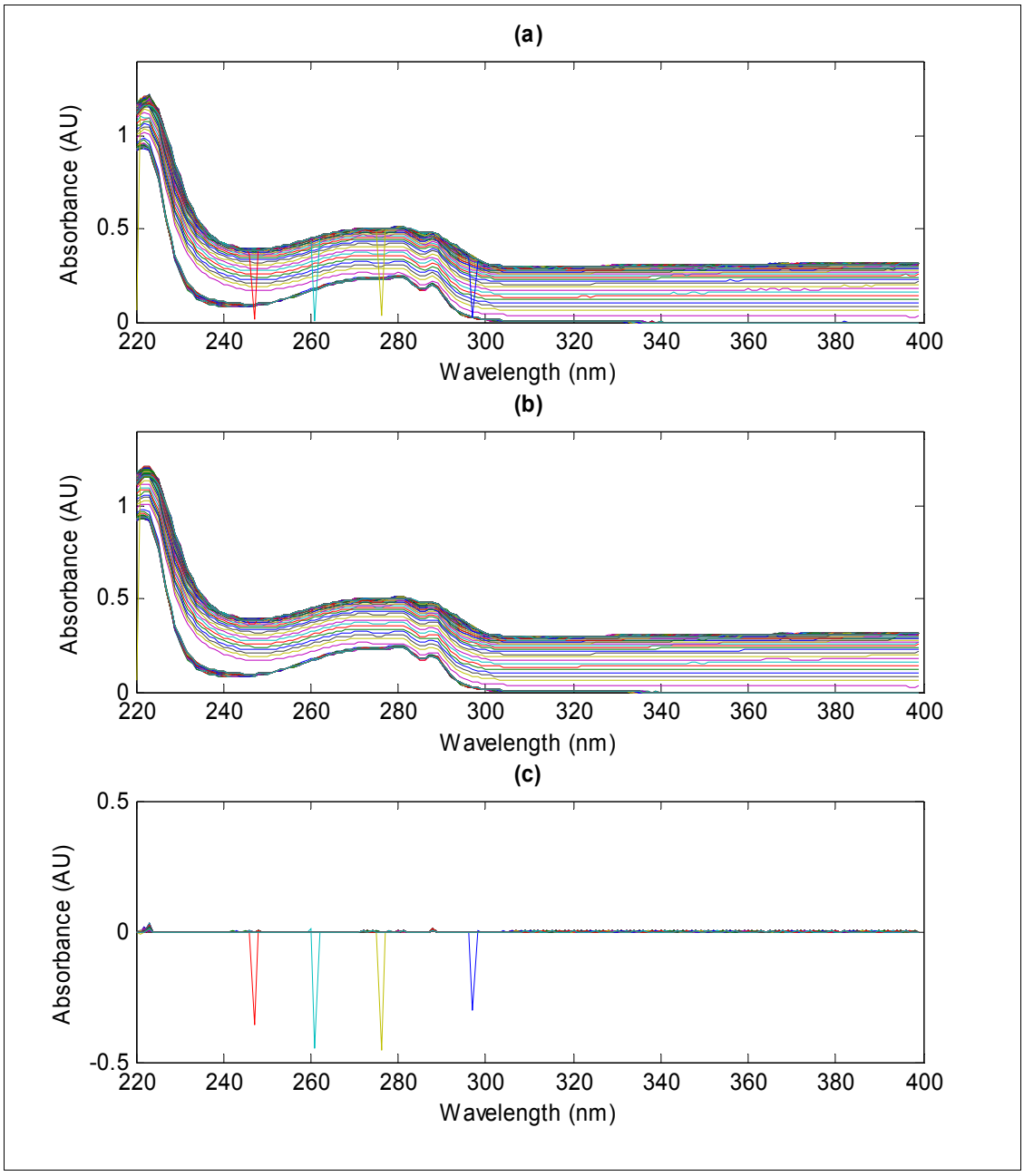


Figure 4.25: Example of applying a moving window median filter to UV/ATR data acquired using a Varian Cary 50 scanning UV spectrometer; (a) original un-filtered data; (b) data filtered using a 3-point moving window; (c) residual spectra after median filtering to remove the spikes.

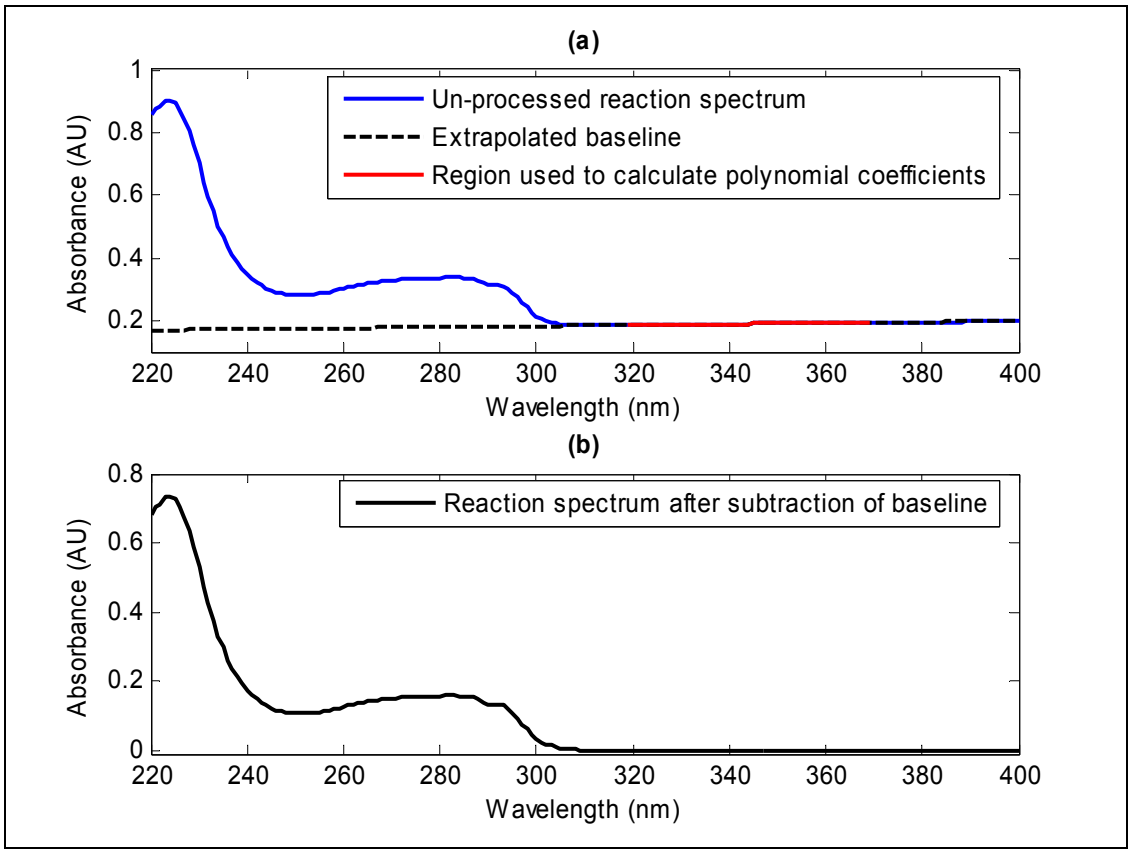


Figure 4.26: (a) illustration of how a linear sloped baseline was calculated for each spectrum; (b) spectrum after baseline correction

4.2.4.1 Conclusions: Pre-processing of the UV data using median filtering

Median filtering was found to be a useful pre-processing technique for UV spectroscopy. A median filter was originally implemented as a Matlab function (`MedianFilter.m`) to remove unwanted 'spikes' from UV spectra acquired using a Cary 50 UV spectrometer. This instrument used a high intensity xenon flash lamp and a fast scanning monochromator to rapidly scan across a wavelength range of several hundred nanometers in just a few seconds. The cause of these spectral artefacts is suspected to be caused by the monochromator and detector becoming briefly un-synchronised with the xenon flash lamp. Using a median filter with a window width of three points successfully removed these artefacts from each data set without changing any of the spectral features. Another approach that could have been utilised to remove random contributions such as this would be to re-produce the data set using only the primary principal components. The advantage of the median filter is that it can be applied to an individual spectrum.

4.2.5 Optimisation of the IPBS method parameters

Selection of the appropriate window width for the median filtering step

The custom Matlab function `IPBS.m` uses several parameters that can influence the results obtained when applying automated iterative polynomial baseline subtraction. To find the optimal values of the key parameters n (degree of the polynomial) and w_p (window width) an exhaustive search was performed over a specific range of values. Before the search of the parameters n and w_p could be started, a suitable window width to use during the median filtering step was chosen. `MedianFilter.m` was applied to the test spectrum using a window width of 101, 201, 301, 401 and 501 points.

The effect of using different window widths is shown in Figure 4.27. Increasing the window width to values above approximately 500 points did not offer any significant improvement. The plot in Figure 4.27(a) shows the un-processed Raman test spectrum from the *N*-benzylation of 1*H*-indole reactions. This spectrum was acquired at $t = 30$ minutes and corresponds to a spectrum of *1H*-indole, a small amount of tetra butyl ammonium bromide and the un-dissolved inorganic base (cesium carbonate) in acetonitrile. Also shown is the reference spectrum of *1H*-indole in acetonitrile described in section 3.3. The large baseline contribution in the reaction test spectrum was a consequence of the un-dissolved cesium carbonate. Figure 4.27(b) shows the corrected spectrum after subtraction of the median filtered spectrum using different window widths. Figure 4.27(c) shows the median filtered spectrum obtained using different window widths. The median filtered spectrum was a good approximation of the baseline contribution. Examination of the spectra in Figure 4.27(b) and Figure 4.27(c) revealed that using smaller window widths allowed part of the true Raman peaks in the spectrum to pass through the median filter. The peaks at 400 to 500 cm^{-1}, 1300 to 1500 cm^{-1} and 2300 to 2400 cm^{-1} illustrate this most clearly. As the window width was increased, less of the true Raman spectrum passed through the median filter although it did introduce more negative regions to the corrected spectrum. The median filter also completely eliminated the peak at 100 to 300 cm^{-1}. As the median filter is only used by IPBS to provide an initial estimate of the underlying baseline, a window width of 501 seemed to be the most appropriate.

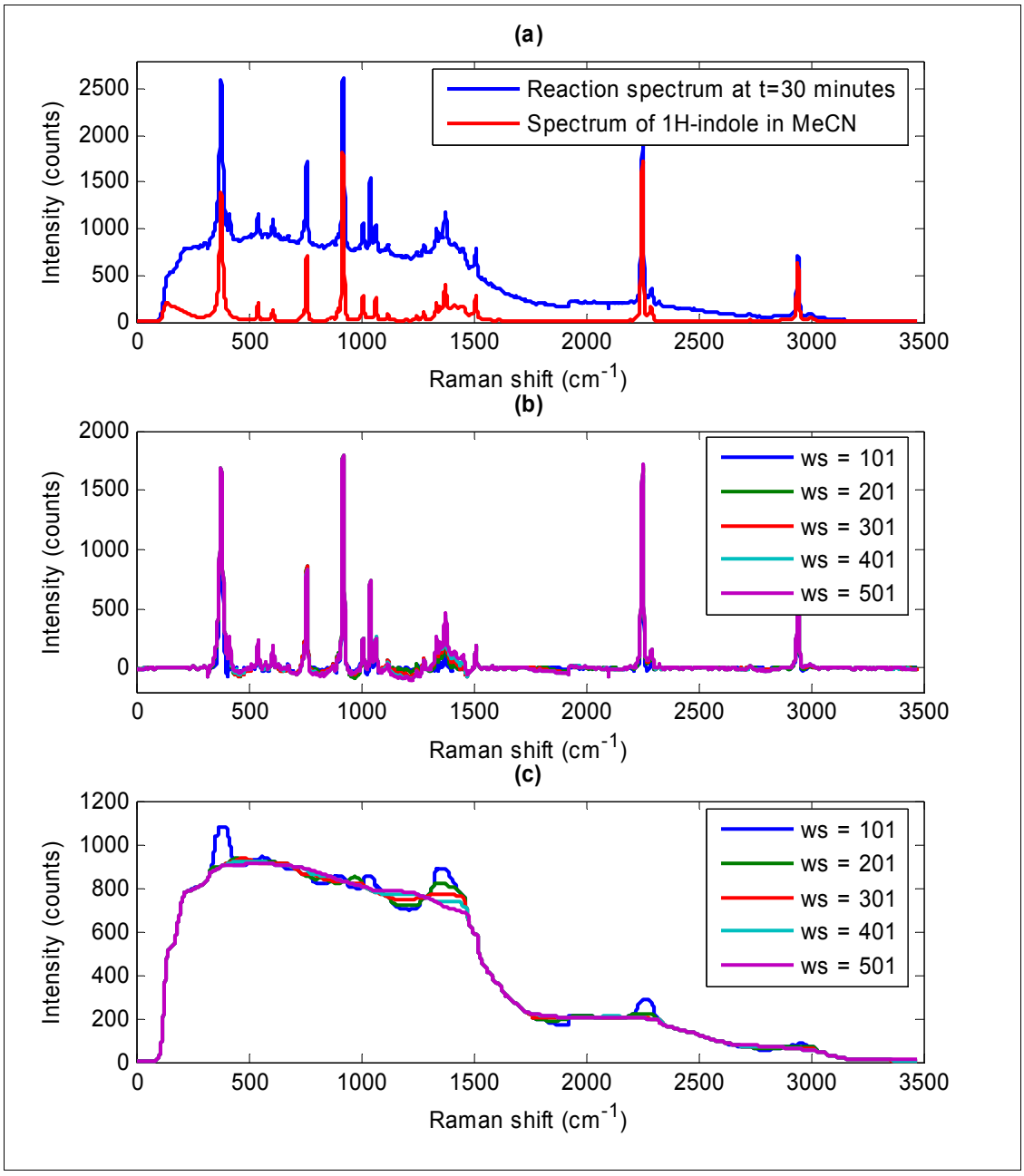


Figure 4.27: Median filtering a reaction Raman spectrum using various window widths: (a) the original unprocessed Raman spectrum from the *N*-benzylation of 1*H*-indole at t=30 minutes, and a Raman spectrum of 1*H*-indole in acetonitrile acquired in the absence of cesium carbonate; (b) Raman spectra after subtraction of the median filtered spectrum; (c) the estimated background spectrum (median filtered spectrum).

Assessment of the degree of polynomial and window width parameters

The effect of the degree of polynomial (n), window width (w_p) was investigated using a fixed median filter window width of 501. For each combination of n and w_p, the sum-of-squares of the residual spectrum obtained by subtraction of the normalised, baseline subtracted test spectrum from normalised reference spectrum of *1H*-indole in acetonitrile was calculated. This produced three (71×5) matrices. The minimum sum-of-squares values

and the corresponding window width for each degree of polynomial and degree of overlap are summarised in Table 4.3. The results in Table 4.3 suggested that for a fixed overlap of 0.10 (10%), a second degree polynomial ($f(x) = b_0 + b_1x + b_2x^2$) with a window width of 141 produced the lowest residual sum-of-squares (SSQ = 2.1095).

Increasing the degree of overlap set to 0.20 (20%) only offered a marginal improvement. A third degree polynomial ($f(x) = b_0 + b_1x + b_2x^2 + b_3x^3$) with a window width of 157 gave a minimum sum-of-squares value of 2.1081. This was the lowest SSQ obtained for all combinations of parameters. Increasing the degree of overlap to 0.30 (30%) did not improve the SSQ values any further.

Table 4.3: Minimum sum-of-squares (SSQ) values and corresponding window width for each degree of polynomial and degree of overlap. The lowest values for each column are shown in italics.

Degree of polynomial	Overlap = 0 .10		Overlap = 0 .20		Overlap = 0 .30	
(n)	Min(SSQ)	Window width	Min(SSQ)	Window width	Min(SSQ)	Window width
2	*2.1095*	*141*	2.2081	165	2.2173	77
3	2.1355	157	*2.1081*	*157*	*2.1577*	*133*
4	2.1923	133	2.1697	101	2.1963	101
5	2.1752	133	2.1759	133	2.1807	101
6	2.1698	109	2.1669	133	2.1762	101

These results suggested that the optimal parameter values to use with this particular set of Raman data were a median filter window width of 501; a third degree polynomial (n = 3); an overlap factor of 0.20 and a window width of 157. The spectra obtained by application of these parameter values to the test spectrum are shown in Figure 4.28. The unprocessed test spectrum is shown in Figure 4.28(a). Figure 4.28(b) shows the corrected spectra after subtraction of the baselines estimated using median filtering and IPBS. This plot shows that the corrected spectrum obtained using IPBS exhibit good correlation with the reference spectrum of 1H-indole in acetonitrile. The spectrum obtained using median filtering only was also a reasonable estimate although it did have several regions that were negative and some Raman spectral features had reduced intensity or were missing completely. Figure 4.28(c) shows the subtracted baseline spectra estimated using median filtering and IPBS. The baseline spectrum estimated using IPBS was iteratively refined so that it did not remove any valuable Raman features of interest.

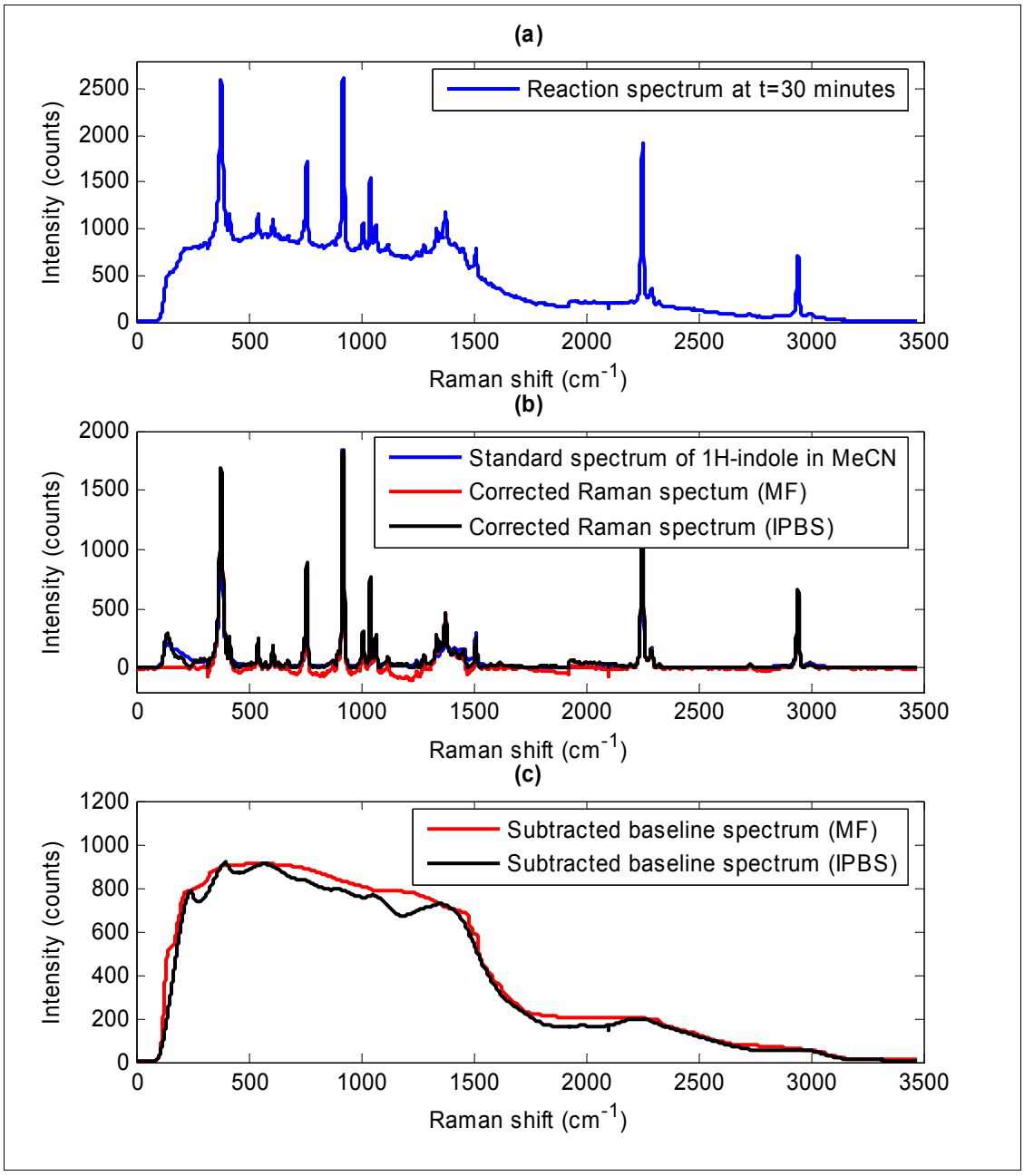


Figure 4.28: Applying IPBS to the test spectrum using the optimal parameters (median filter window width = 501 points, polynomial fitting window width = 157 points, degree of polynomial = 3, degree of overlap = 0.20): (a) the unprocessed Raman test spectrum; (b) comparison of the corrected spectra obtained using median filtering and IPBS with the standard spectrum of 1*H*-indole in acetonitrile; (c) the subtracted baseline spectra estimated using median filtering and IPBS.

To allow this algorithm to be compared with the original algorithm published by Lieber and Mahadevan-Jansen[97], the original method (`LMJ_IPBS.m`) was applied to the same test spectrum using second, third, fourth, fifth, sixth, seventh and eighth degree polynomials. In the original Lieber and Mahadevan-Jansen method, it was not necessary to specify a window size as the fitting was applied to the whole data set. Typically the data would be manually truncated to a much smaller region before applying the baseline correction. However the purpose of this test was to compare the ability of each algorithm to correctly subtract the baseline from a full range spectrum. The residual sum-of-squares values for each order of polynomial are shown below in Table 4.4. The residual sum-of-squares values in the table show that the original baseline subtraction method was less successful at removing the complex, polynomial baseline from a full range Raman spectrum. The best correction was obtained using a fourth degree polynomial but the residual sum-of-squares value was 61.7 (*cf.* 2.11 for the modified IPBS method). The baseline corrected spectrum and the subtracted baseline obtained using a fourth degree polynomial are shown in Figure 4.29. This figure shows that a significant amount of the original baseline contribution still remained in the corrected spectrum.

Table 4.4: Residual sum-of-squares (SSQ) values obtained by applying the original IPBS method to the Raman test spectrum using different orders of polynomial.

Degree of polynomial (*n*)	Residual SSQ
2	103.9
3	77.5
4	*61.7*
5	109.6
6	112.7
7	115.1
8	116.4

The original IPBS method was more successful if the two halves of the spectrum were fitted independently. However the modified IPBS method still gave a better result because it used many windows to fit the full spectrum. Furthermore, each window used a small section of the previous window to ensure the final baseline function was continuous.

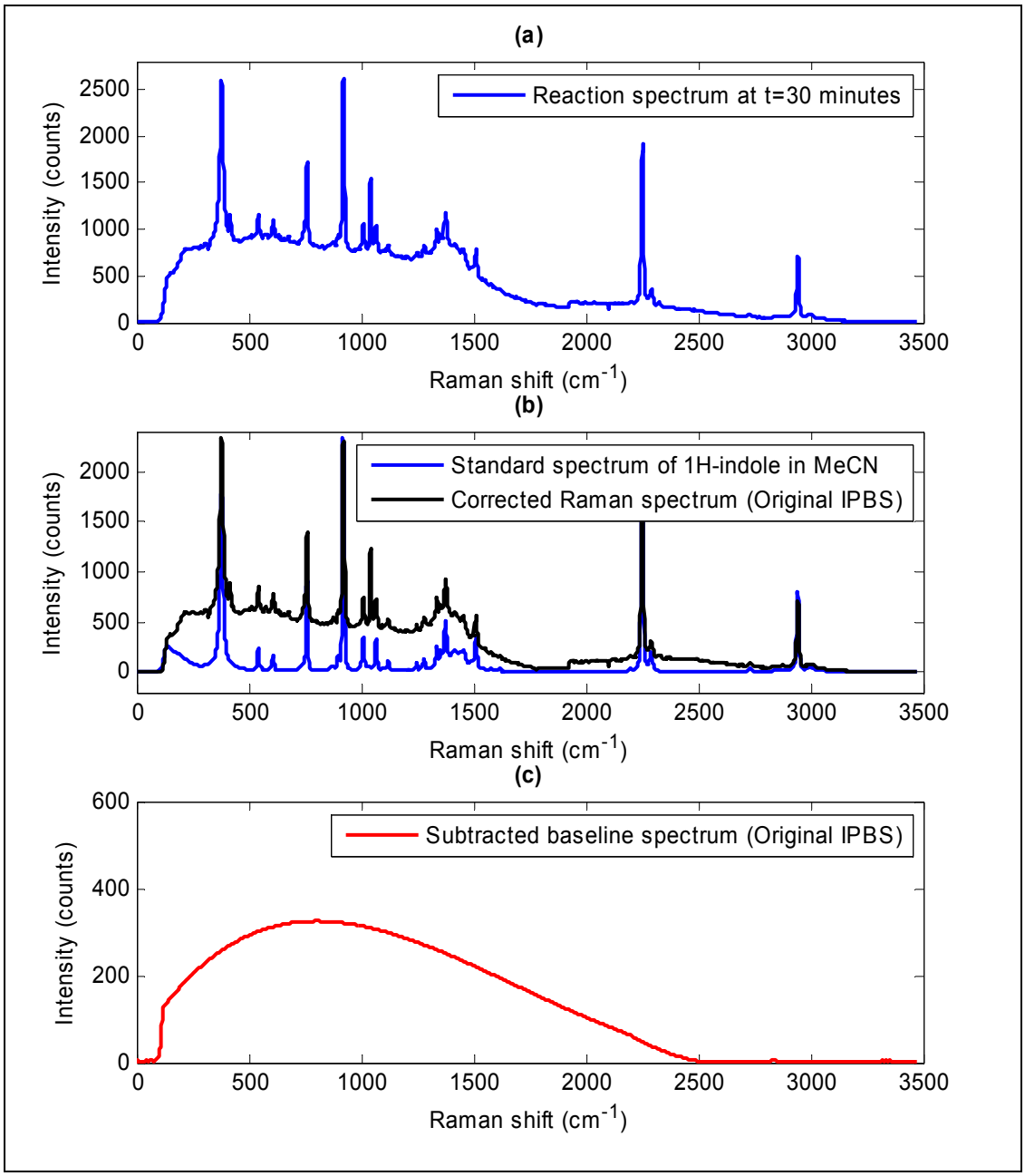


Figure 4.29: Application of the original IPBS method to the test spectrum using the optimal parameters (a fourth degree polynomial): (a) the unprocessed Raman test spectrum; (b) comparison of the baseline corrected spectrum with the standard spectrum of 1*H*-indole in acetonitrile; (c) the subtracted baseline spectra estimated using the original IPBS method.

4.2.5.1 Conclusions: Iterative Polynomial Baseline Subtraction

A median filter was applied to Raman spectra acquired during the *N*-benzylation of 1*H*-indole reactions. The Raman spectra featured a broad baseline feature that contributed additional, unwanted variation. To correctly normalise the Raman spectra, it was necessary to minimise the baseline contribution as much as possible. The median filter was applied to a Raman spectrum (3400 variables) using window widths of 101, 201, 301, 401 and 501 points. A window width of 501 points was found to offer the best approximation of the

underlying baseline but the baseline subtracted spectrum did have several negative regions where the baseline was overestimated. However to obtain a first approximation of the underlying baseline, the median filter proved to be a useful approach

To obtain a more accurate estimation of the underlying baseline contribution to the Raman spectra described in the previous section, a custom algorithm called iterative polynomial baseline subtraction (`IPBS.m`) was written. The iterative polynomial fitting was based upon the method reported by Lieber and Mahadevan-Jansen[97]. In this paper, the authors used an iterative fourth-order polynomial fitting routine to remove fluorescence from Raman spectra of biological samples. However, the Raman spectrum shown was truncated to between 600 and 1800 cm^{-1}. This simplified the fitting of the underlying fluorescent background but required the user to select the specific region of interest before performing the baseline correction. The method described in this thesis was extended to allow a full Raman spectrum to be corrected by splitting the spectrum into a number of windows defined by the window width. An additional feature of the iterative polynomial baseline subtraction method is that it allows the windows to overlap using an overlap parameter (typically 10 to 20% of the window width). This was found to minimise the discontinuities that were observed if the polynomial fitting was applied to each window independently.

Although computationally intensive, the results were very good and the complex baseline contribution of each spectrum were accurately modelled. The best parameters for correcting this particular set of Raman spectra were a median filter width of 501 points; a third degree polynomial (n =3); an overlap factor of 0.20 and a window width of 157 points. The baseline corrected spectra had a completely flat baseline, did not feature any negative regions yet retained all of the useful Raman features. This was a significant improvement over the Pearson[105] correction function available in the instruments HoloReact software. The time taken to correct each spectrum was between 30 and 60 seconds using the laptop PC described in chapter 3.1. This was a big disadvantage when the method is applied to a large data set comprising several hundred spectra, as the processing time was two to three hours or more. However, it would still be possible to apply the method in real-time as Raman acquisition times for heterogeneous mixtures are often one to two minutes in order to improve the signal to noise.

4.2.6 Preparation and pre-processing of the Raman spectral data

The Raman spectral data acquired during the *N*-benzylation of 1*H*-indole reactions were pre-processed to remove the variable baseline contributing to each spectrum. The automated iterative polynomial baseline subtraction method (`IPBS`) was applied using the optimal settings described in section 4.2.5. The data set were then reproduced using the first four principal components calculated by applying singular value decomposition to the un-centred data. This was necessary to remove the cosmic rays that occured randomly throughout the data. The data contained cosmic rays because the option to automatically remove cosmic rays was disabled in the Raman spectrometer software (HoloGrams). For real-time applications where de-noising the spectra by data reproduction using principal components analysis is not an option, the 'cosmic ray filter' must be enabled. The time required to acquire each spectrum would be doubled but random cosmic rays would be eliminated. Finally, to remove spectrum to spectrum intensity variation from each data set, the spectra were all normalised to max = 1. The maximum value in each spectrum corresponded to the 920 cm^{-1} C-C stretch of acetonitrile. The final step of spectral pre-processing was to truncate the spectra to exclude the data outside the range 100 to 1800 cm^{-1}. The variance spectrum of the normalised data revealed that the spectra contained no significant structured variance outside this range and would not be useful in resolving the spectra.

4.2.7 Creation of a simulated UV data set for algorithm testing

Two simulated data sets comprising UV spectra representing the reaction spectra acquired during the non-aqueous *N*-benzylation of 1*H*-indole using benzyl bromide were created. These data sets were used during the development and implementation of the VVSP and Kalman filter algorithms.

The simulated concentration profiles and the corresponding pure component spectra for the three major species (1*H*-indole, benzyl bromide and 1-benzyl-1*H*-indole) are shown in Figure 4.30.

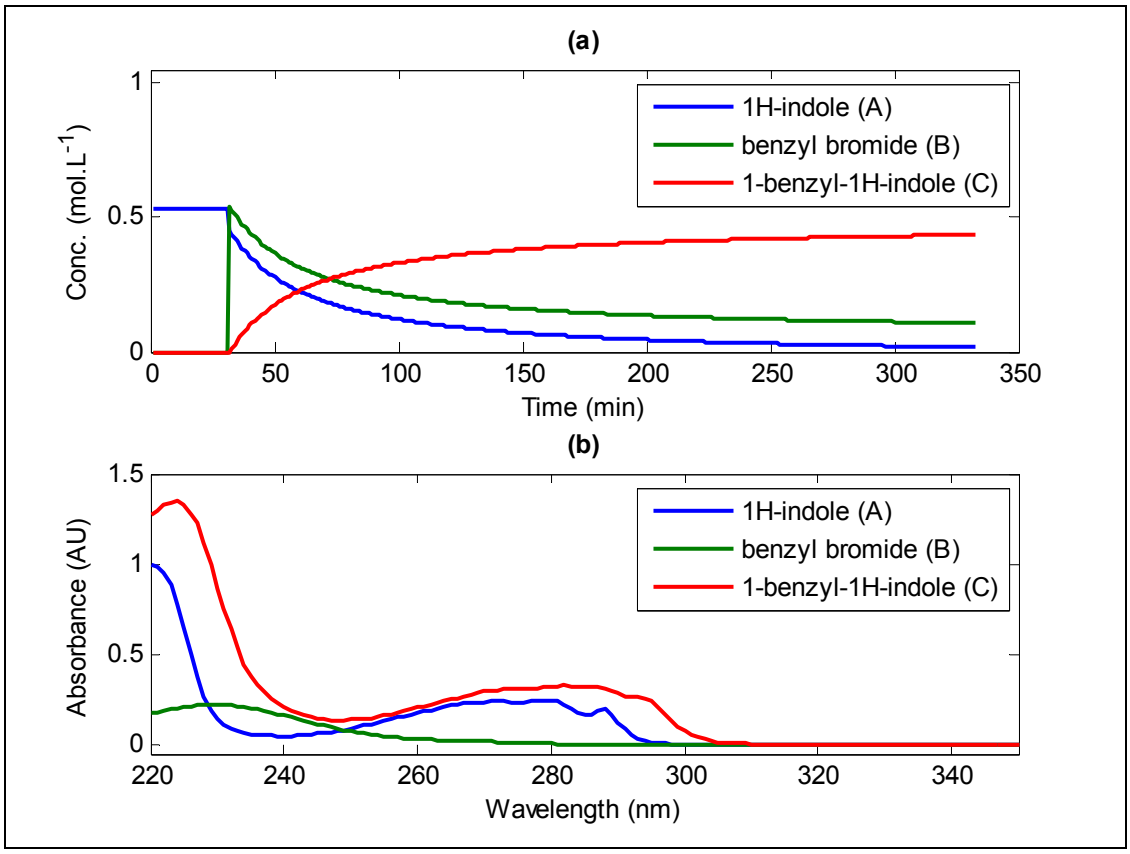


Figure 4.30: (a) simulated concentration profiles; (b) actual pure component spectra for the three major components used to create the synthetic reaction data. The pure component spectra were acquired using reference standards of each component.

The outer product of the concentration and spectral profiles was calculated to produce the noise-free reaction spectra in a (333×181) matrix, $\boldsymbol{D1}_{sim}$. Homoscedastic noise with zero mean and variance 1.0×10^{-6} was added to the data matrix to give the final simulated data matrix, $\boldsymbol{D2}_{sim}$. The Savitsky-Golay first-derivative spectra were calculated from $\boldsymbol{D2}_{sim}$ using a 13-point smoothing window and third-order polynomial to give $\boldsymbol{D3}_{sim}$. The overlaid spectra of $\boldsymbol{D2}_{sim}$ and $\boldsymbol{D3}_{sim}$ are shown in Figure 4.31. The reaction spectra $\boldsymbol{D2}_{sim}$ and $\boldsymbol{D3}_{sim}$, and the reference spectra $\boldsymbol{S1}_{sim}$ and $\boldsymbol{S2}_{sim}$ were then truncated to 220 to 350 nm to give (333×131) and (131×3) matrices respectively.

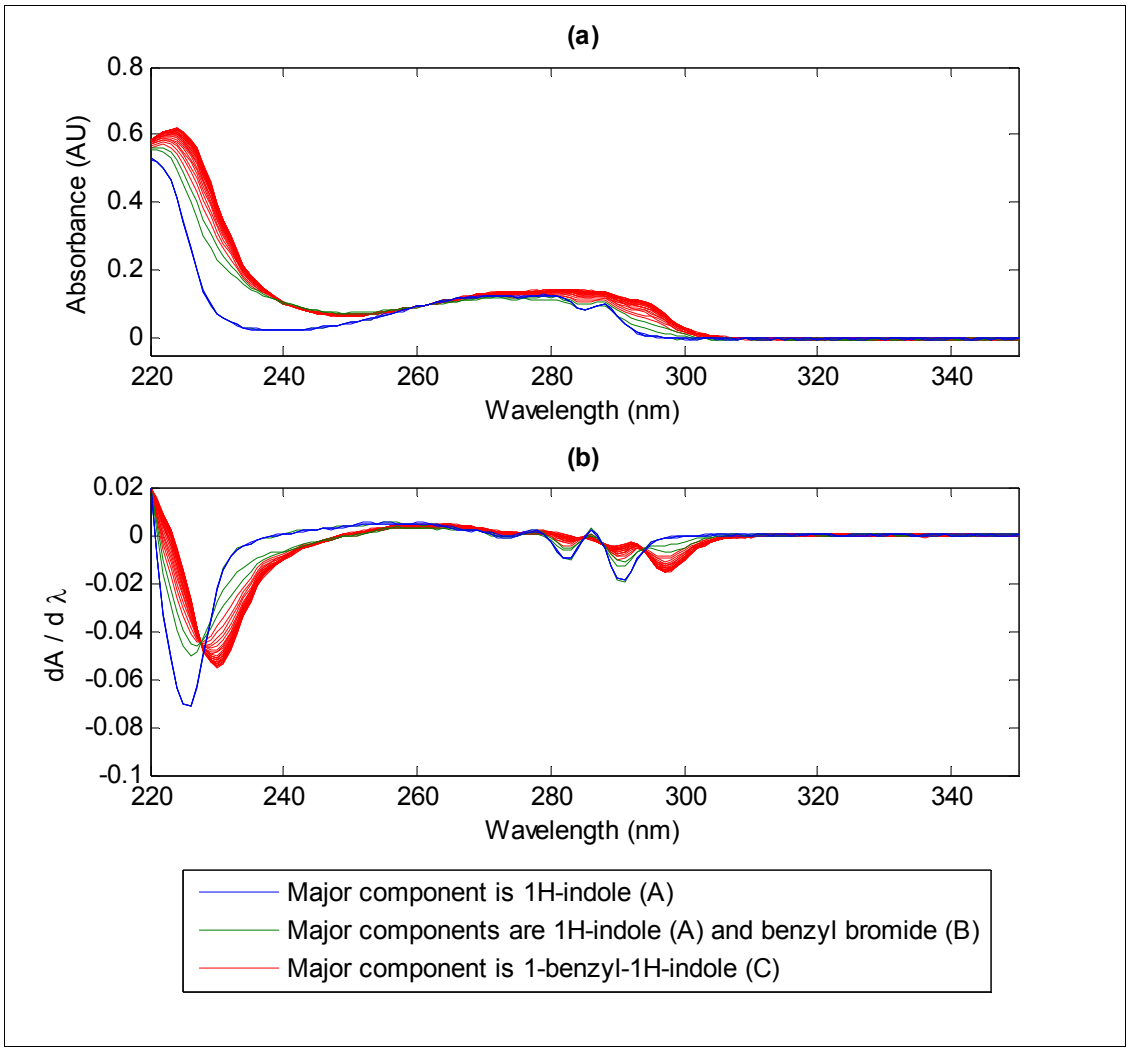


Figure 4.31: Simulated UV data sets; (a) $D2_{sim}$; (b) $D3_{sim}$. Every tenth spectrum is shown to improve clarity.

These data sets are particularly challenging for SMCR methods as the concentration of B (benzyl bromide) is in excess and never reaches zero. Also, there are no selective regions (pure variables) for each species in the UV spectra.

4.2.8 Demonstration of the equivalence of the standard and vectorised linear Kalman filter

4.2.8.1 Determination of measurement noise variance, R

The measurement noise variance, R is a critical parameter of the linear Kalman filter. If the value of R is too large, the Kalman filter will not produce accurate results because the data are assumed to have a large measurement error associated with each data point. As the elements of the error covariance matrix $\boldsymbol{P}$ are calculated using R, using a value of R smaller than the actual measurement variance will give a misleading estimate of the error

associated with each state parameter estimate. The optimal value of R is therefore a value equal to the actual measurement noise variance. To determine a suitable value of R for a data set of unknown measurement noise variance, the innovations sequence of a spectrum can be assessed using a range of values. The optimum value will be located at the point where the innovations sum-of-squares begins to approach its minimum value and shows no significant decrease when the value of R is decreased by one or two more orders of magnitude.

The linear Kalman filter (`linearKF.m`) was applied to the last spectrum of the data sets $\boldsymbol{D2}_{sim}$ and $\boldsymbol{D3}_{sim}$, using R values ranging from 1.0×10^{-10} through to 1.0×10^{-3} with ten points for each order of magnitude. The spectral root-mean-square lack-of-fit (RMS-LOF) was calculated using the expression shown in equation 3.6 and plotted against $\log_{10}(R)$. The simple first derivative of the RMS-LOF values was also calculated. The results are shown in Figure 4.32.

The LOF values for the unprocessed spectra had a defined minimum at $R=3.16\times10^{-6}$, ($\log_{10}(R)=-5.5$). The variance of the noise matrix added to the simulated data was 1.00×10^{-6}, so the value estimated from spectral residuals was a good approximation of the true noise variance. Figure 4.32(a) shows that the LOF values for the first derivative spectra reached a local minimum at a similar value of R as the unprocessed spectra. However the profile then approached the global minimum at approximately $R=1.00\times10^{-9}$, ($\log_{10}(R)=-9.0$). The values of R that were used to directly compare the original and vectorised linear Kalman filter were $R=3.16\times10^{-6}$ for the unprocessed spectra or $R=1.00\times10^{-9}$ for the first derivative spectra.

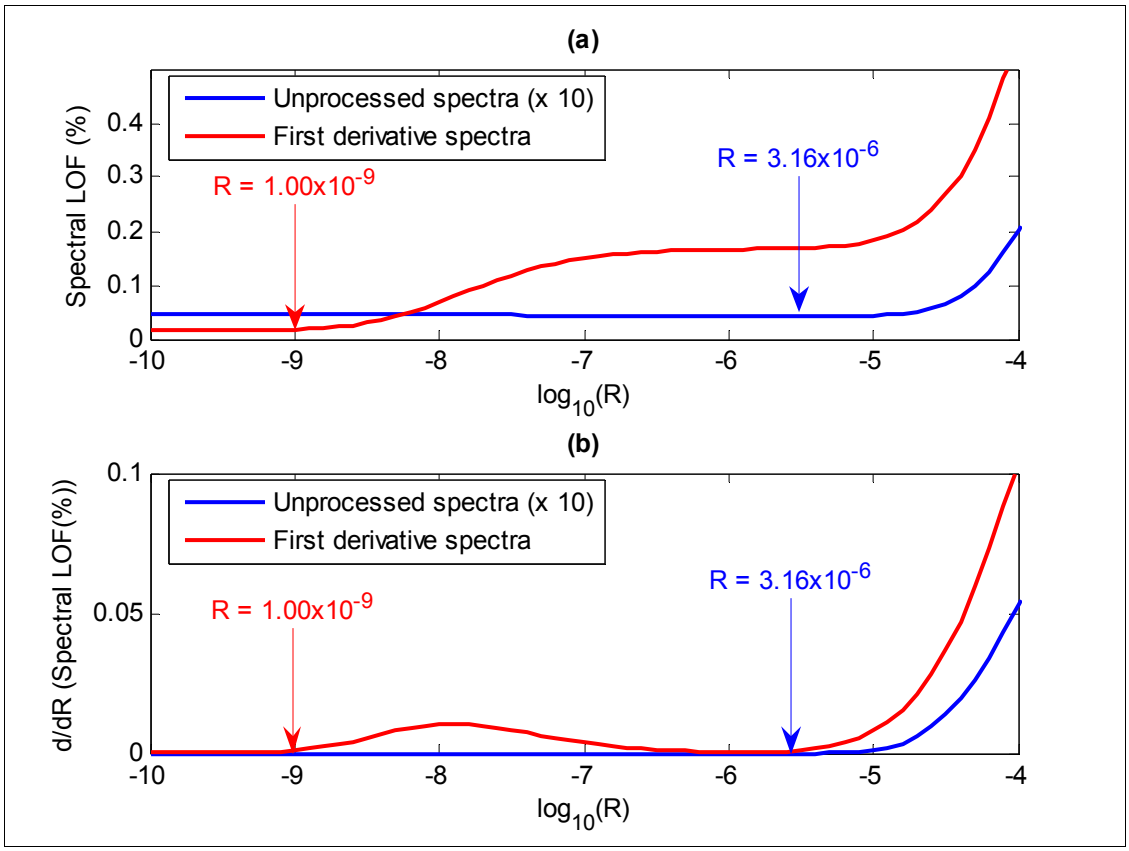


Figure 4.32: The spectral root-mean-square lack-of-fit values calculated using the innovations sequence produced by the linear Kalman filter using different values of R: (a) LOF values for the unprocessed and first derivative spectra; (b) LOF values differentiated with respect to $\log_{10}$(R). The unprocessed spectra approach a minimum at approximately R = 3×10^{-6}, the first derivative spectra approach a minimum at approximately R = 1×10^{-9}.

4.2.8.2 Application of the linear Kalman filter functions to simulated UV data

To confirm that the vectorised linear Kalman filter (`VecLinearKF.m`) produced the same results as the standard linear Kalman filter (`linearKF.m`), both functions were applied to the simulated data sets data sets $\boldsymbol{D2}_{sim}$ and $\boldsymbol{D3}_{sim}$. The resulting Kalman gains, state parameters, state parameter error covariances and the spectral lack-of-fit values were then directly compared. The outputs from the two Kalman filter functions are shown in Figure 4.33.

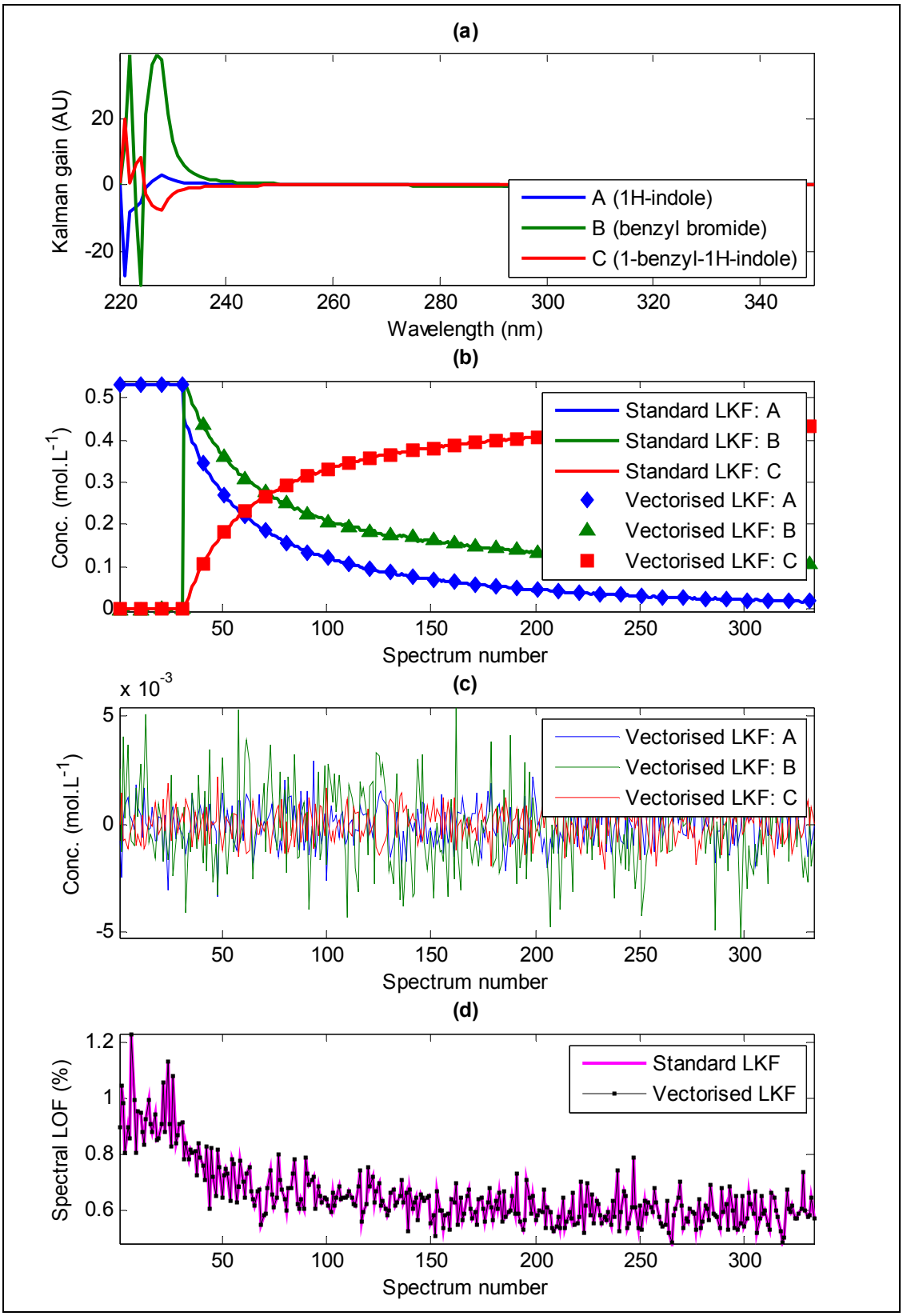


Figure 4.33: Comparison of standard and vectorised linear Kalman filters: (a) Kalman gain vectors for components A, B and C; (b) estimated state parameters; (c) state parameter residuals (actual – estimated) for vectorised Kalman filter – the values obtained from the standard linear Kalman filter were identical; (d) spectral lack-of-fit (integrated innovations).

The Kalman gain vectors shown in Figure 4.33(a) confirm that the vectorised Kalman filter produces identical Kalman gain trajectories to the standard linear Kalman filter. This result was expected and demonstrated that performing the Kalman filtering using a column vector of measurements, rather than a single element does not affect the operation of the filter. The state parameter estimates and spectral lack-of-fit values shown in Figure 4.33(b), Figure 4.33(c) and Figure 4.33(d) also provide confirmation that the filter calculations can be vectorised without any detriment to the filters performance or output. The calculated variance of the final state parameter estimates was also identical for both functions. There was a significant decrease in the execution time as the standard Kalman filter calculated the state parameters for 333 spectra, using 131 variables in approximately 1.14 seconds, whilst the vectorised Kalman filter produced the same results in approximately 0.03 seconds (38 times faster). This considerable improvement in calculation time became more significant as the number of spectral variables is increased. For example, when the standard and vectorised linear Kalman filters were applied to a data set comprising 427 Raman spectra with a spectral range 100 to 3400 cm^{-1} (3301 variables per spectrum), the observed calculation time for the vectorised linear Kalman filter (0.74 seconds) was almost 95 times faster than the standard linear Kalman filter (70 seconds). The reduction in calculation time was more significant when comparing the standard adaptive and vectorised adaptive Kalman filters. The standard adaptive Kalman filter took almost 400 seconds to process the Raman data set described earlier, whilst the vectorised adaptive Kalman filter took approximately 1.60 seconds. This represents a decrease in calculation time by a factor of 250. This reduction in calculation time was important for the implementation of the adaptive Kalman filter in the VAKFISO method as it would be necessary to repeatedly fit the entire data set several times at each iterative cycle during the optimisation step. It would not have been feasible to develop and implement the VAKFISO method using the standard adaptive Kalman filter as the time required to perform adaptive Kalman filtering several hundred times would have led to execution times of two days or more. The vectorised adaptive Kalman filter reduced this time to less than one hour.

4.2.8.3 Conclusions: Comparison of the standard and vectorised linear Kalman filters

The potential of using the Kalman filter for SMCR applications was investigated. The original, linear Kalman filter was implemented as a Matlab function (`LinearKF.m`) and applied to the simulated UV reaction data. This confirmed that if accurate reference measurement functions (pure spectral profiles) were provided; and the measurement noise

was homoscedastic, the linear Kalman filter could be used to accurately predict the state parameters (concentration profiles). In the case where the data was fully modelled by the reference measurement functions, the linear Kalman filter produced identical results to least-squares ($\boldsymbol{C} = \boldsymbol{X}\boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S})^{-1}$). The difference between the two methods is that the Kalman filter uses recursive estimation-correction calculations to specifically minimise the state estimate error covariance matrix; the least-squares approach calculates a solution that directly minimises the elements of the residual matrix.

During the implementation of the linear Kalman filter it was noted that there was considerable redundancy in the calculations when applied to a matrix of data. Initially, the linear Kalman filter function (`LinearKF.m`) was written such that the Kalman gains and error covariance update calculations were repeated for each measurement vector (spectrum) in the data set. A simple loop was used to apply the full set Kalman filter calculations to each spectrum in the $(J \times K)$ matrix of spectral data. When the all of the spectra in a data set are fully modelled by the reference measurement functions and all have the same measurement noise, the Kalman gain and error covariance matrices are identical for each spectrum. To avoid unnecessary recalculation of the Kalman gain ($\boldsymbol{G}$) and error covariance matrices ($\boldsymbol{P}$), they are only calculated for the first spectrum. The recursive Kalman filter calculations are then applied to subsequent spectra in the data set but the Kalman gain matrix is then applied directly and is not re-estimated each time. As this will produce the same state estimate error-covariance matrix, it is not necessary to recalculate $\boldsymbol{P}$.

When the reference measurement functions are applicable to each spectrum in the data set, the resulting Kalman gains computed during the recursive calculations are identical. To further improve the computational efficiency, the Kalman filter equations were vectorised. In the original linear Kalman filter, an outer loop applied the Kalman filter calculations to each of the J measurement vectors (spectra) individually. The $(1 \times N)$ vector of state parameter estimates is optimised for each spectrum before moving on to the next spectrum. Since the Kalman gain and error covariance matrices are identical and applicable to all J spectra, the Kalman filter equations were vectorised so that the state estimate update step simultaneously updates the entire $(J \times N)$ matrix of state parameters (denoted $\boldsymbol{X}$). This approach was implemented as a Matlab function called `VecLinearKF.m`. The calculations still operate recursively along the variable direction but the outer loop to apply the filter calculations to each spectrum individually was now removed. To confirm

that this approach did indeed yield identical results (in terms of the estimated state parameters and innovations based lack-of-fit), the equivalence of the standard and vectorised linear Kalman filters was demonstrated using the simulated UV data. Using the optimal value of the measurement noise variance ($R = 3.16 \times 10^{-6}$), `LinearKF.m` and `VecLinearKF.m` were applied to the (333×131) matrix of simulated data. As expected, the Kalman gain vectors, estimated state parameters and innovations based spectral lack-of-fit values were all identical. However, there was a significant decrease in the time taken to execute the vectorised Kalman filter relative to the standard Kalman filter. The standard Kalman filter took 74 seconds but the vectorised version only took 0.03 seconds (approximately 2400 times faster). This is a considerable improvement in the calculation time and became even more significant as the number of spectral variables was increased. For example NIR, MIR or Raman spectra may have up to two thousand variables or more. Furthermore, the aim of this work was to utilise the unique features of the Kalman filter to develop a SMCR method. The method that was developed (VAKFISO) requires the adaptive Kalman filter to be applied several hundred times and the vectorised implementation therefore offered a huge improvement in computation time.

4.2.9 Application of the adaptive Kalman filter to simulated UV spectra

The purpose of these experiments was to assess the performance of the adaptive Kalman filter (`AdaptiveKF.m`) when applied to a simulated UV data set comprising highly overlapped spectra.

4.2.9.1 Application of the adaptive Kalman filter using incomplete reference functions

The adaptive Kalman filter was applied using incomplete reference measurement functions to assess how accurately it could estimate the state parameters of known components in the presence of model errors. The model errors arise from the appearance of new chemical species not included in the reference measurement function matrix. The adaptive Kalman filter also allows the reference measurement function to be augmented with an approximation of the unknown components spectrum, calculated from the vector of innovations. The augmented reference functions estimated using the adaptive Kalman filtered were directly compared with true spectral profiles.

Re-scaling of the pure component concentration and spectral profiles

The estimated reference measurement functions of each component were normalised to unit length (2-norm) prior to running the Kalman filter. The scaling of the spectral profiles was different to the unit concentration spectra used to create the simulated data so the concentration profiles estimated using the adaptive Kalman filter also had different scaling. To allow the spectral and concentration profiles estimated using the adaptive Kalman filter to be directly compared with the true profiles used to create the data set, the true profiles were re-scaled as described below.

The 2-norms of each component (column) in $\boldsymbol{S1}_{sim}$ were calculated and stored in the diagonal elements of a (3×3) diagonal-matrix, $\boldsymbol{N}$. The true spectral and concentration profiles were rescaled using the expressions $\boldsymbol{S1}_N = \boldsymbol{S1}_{sim} \cdot \boldsymbol{N}^{-1}$ and $\boldsymbol{C1}_N = \boldsymbol{C1}_{sim} \cdot \boldsymbol{N}$

Estimation of the state parameters for component A in the presence of component B

The adaptive Kalman filter was applied to the first 32 spectra of $\boldsymbol{D2}_{sim}$ using a reference measurement function comprising a normalised estimate of component A only. Some of the outputs from `AdaptiveKF.m` are shown in Figure 4.34. In the simulated data $\boldsymbol{D2}_{sim}$, component B was absent for the first 31 spectra, and was then present in high concentration at spectrum 32, representing the fast addition of a final reagent to initiate the reaction. The aim of this test was to assess the ability of the adaptive Kalman filter to predict the concentration of component A in the presence of component B.

Figure 4.34(a) shows the estimated state parameters for component A using an incomplete reference measurement function. As anticipated, the estimated state parameters for component A show excellent agreement with the expected values for the first 30 spectra. This was because over this time range, component A was the only species contributing to the reaction spectra and the reference measurement function was completely modelling the measurement data. At spectrum 32, component B was present so the measurement data was no longer fully modelled by the reference measurement function. This is illustrated by the deviation of the estimated state parameter for component A at spectrum 32. This also confirmed by the spectral innovations shown in Figure 4.34(b) and Figure 4.34(c). For the first 31 spectra, the spectral innovations represent zero-mean white noise, indicating the Kalman filter was operating optimally. This resulted in low spectral lack-of-fit values as

shown in Figure 4.34(c). At spectrum 32, the reference measurement function did not completely model the measurement data and the spectral innovations and spectral lack-of-fit values increased significantly.

Re-estimation of the state parameters for components A and B using an augmented reference measurement function

In the previous experiment, the adaptive Kalman filter was applied to the first 32 spectra and a model error corresponding to the rapid appearance of the un-modelled component B was observed. The `AdaptiveKF.m` function was then used to augment the reference measurement function with an estimate of the spectral profile for component B, calculated from the adaptive measurement error values (R_k). This spectrum was normalised to unit length and the adaptive Kalman filter was then applied to spectra 1 to 32 a second time to provide new state parameter estimates for components A and B. The normalised spectral profile for component B, estimated from the adaptive measurement error values (R_k) is shown in Figure 4.34(d). This spectrum exhibited a reasonable correlation with the true spectral profile of component B, but also shows significant deviation at those wavelengths where component B was most overlapped with component A (approximately 220 to 230 nm and 255 to 280 nm).

The estimated state parameter values for components A and B, calculated using the augmented reference measurement function are shown in Figure 4.34(a). The estimated state parameter for component A showed very little improvement at spectrum 32. The previous prediction error at spectrum 32 was 0.273 mol.L^{-1} (14.5% relative error) whilst the updated prediction error was 0.264 mol.L^{-1} (14.0% relative error). Component B was not previously estimated when the Kalman filtering was performed using an incomplete reference measurement function. The estimated state parameter at spectrum 32 using augmented reference measurement function resulted in a prediction error of 0.111 mol.L^{-1} (20.3% relative error).

As before, the spectral innovations for the first 31 spectra resembled zero-mean white noise, indicating the Kalman filter is operating optimally. At spectrum 32, the values of innovations sequence were lower than those obtained previously using an incomplete reference measurement function, but still did not resemble zero-mean white noise. Correlated structure in the innovations sequence at spectrum 32 indicates that although the model error had been reduced, it had not been eliminated completely. The remaining

model error was contributed by the estimated spectral profile of component B used to augment the reference measurement function as it did not completely match the true spectral profile. The measurement model was therefore improved but was not fully resolved (rotational ambiguity).

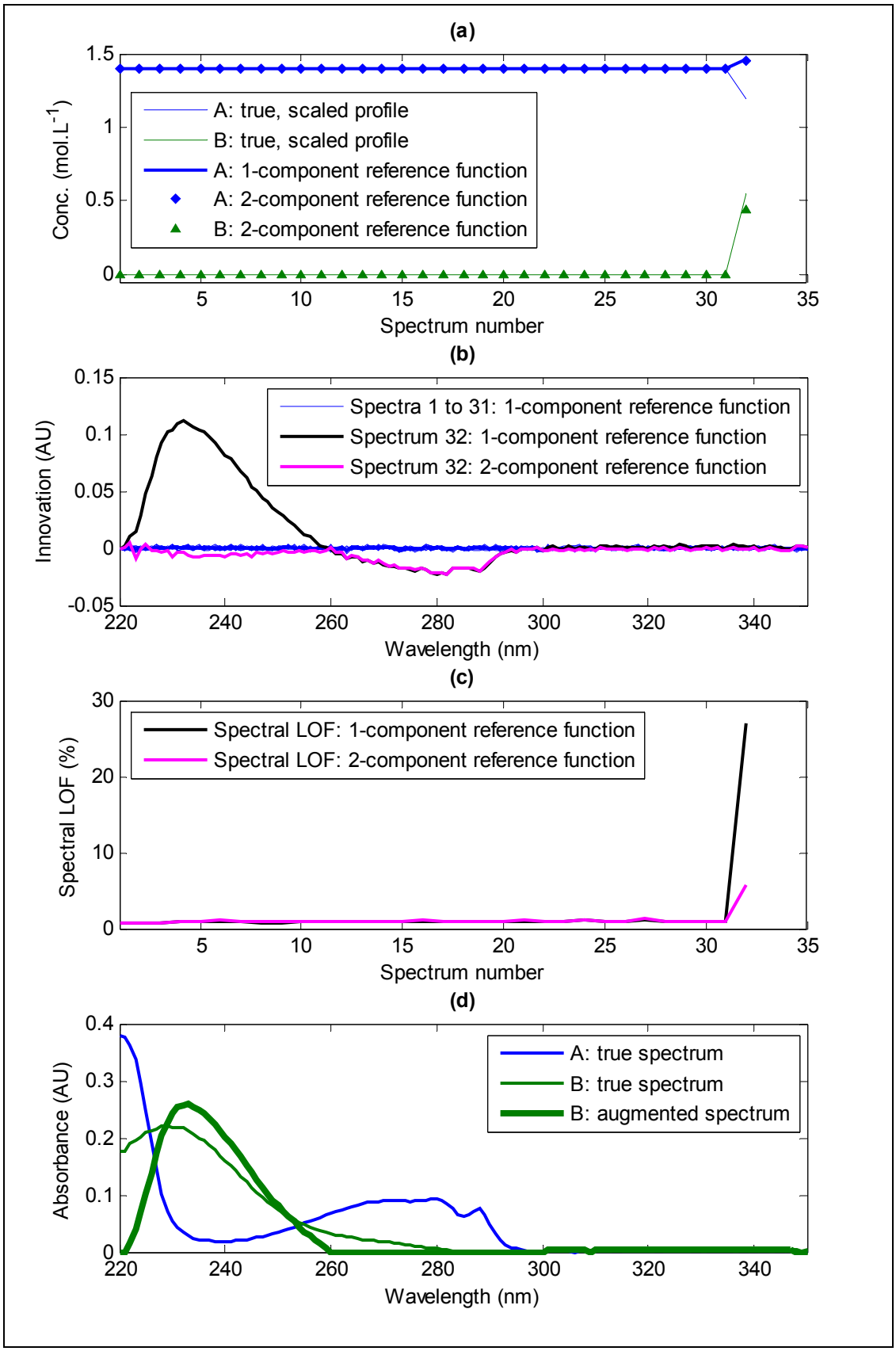


Figure 4.34: Outputs from the adaptive Kalman filter applied to the first 32 spectra of the simulated data set $D2_{sim}$ using incomplete and augmented reference measurement functions: (a) estimated state parameters; (b) spectral innovations; (c) spectral lack-of-fit; (d) true spectral profiles and augmented reference measurement functions.

Estimation of the state parameters for components A and B in the presence of component C

After augmenting the reference measurement function with an estimated spectral profile of component B, the adaptive Kalman filter was then applied to the full data set (333 spectra). The reaction was initiated at spectrum 32 by addition of component B. From spectrum 32 through to spectrum 333, the product (component C) began to increase in concentration as it was slowly formed. The results obtained from this test are shown in Figure 4.35.

Figure 4.35(a) shows the estimated state parameters for components A and B using an incomplete reference measurement function. The reference measurement function comprising components A and B was inaccurate because the estimated spectral profile of component B did not completely match the true profile. This measurement model error led to errors in the estimated state parameters for both components A and B. This was characterised by the deviation of the estimated state parameters for components A and B at spectrum 32 and 33, even though component C was only present at very low levels and should not have greatly influenced the measurement. The estimated state parameters deviated further from the true values as the concentration of component C increased. This was because component C was not included in the reference measurement function and therefore contributed to the total measurement model error.

The spectral innovations shown in Figure 4.35(b) and Figure 4.35(c) indicate that the spectral innovations were a combination of the model errors arising from the model error for component B and un-modelled component C.

Re-estimation of the state parameters for components A, B and C using an augmented reference measurement function

In the previous experiment, the adaptive Kalman filter was applied to the full set of spectra and a measurement model error corresponding to the slow formation of un-modelled component C was observed. The `AdaptiveKF.m` function was used to augment the reference measurement function with an estimate of the spectral profile for component C, calculated from the adaptive measurement error values (R_k). This spectrum was normalised to unit length and the adaptive Kalman filter was then applied a second time to provide new state parameter estimates for components A, B and C. The normalised spectral profile for component C, estimated from the adaptive measurement error values (R_k) is shown in Figure 4.35(d). The estimated spectrum exhibited poor correlation with

the true spectral profile of component C and was set to zero over most of the spectrum. Comparison of the estimated spectrum with the true spectral profiles for components A, B and C revealed that the estimated spectrum was set to zero wherever it is overlapped with the other reference spectra. This was expected as the adaptive measurement error calculations set R_k to zero whenever the innovations were negative. The estimated spectrum of component C was only correlated with the true spectrum in the spectral region 290 to 305 nm, which corresponded to the wavelengths that were selective for component C.

The estimated state parameters calculated using the augmented reference measurement function are shown in Figure 4.35(a). The results indicate that the accuracy of the state parameter estimates for component B were decreased slightly by incorporating the estimated spectrum of component C.

The innovations for spectrum 32 and 333 in Figure 4.35(b) show that there was still a significant measurement model error using a two-component reference function. After augmenting the matrix reference measurement functions with an estimated spectrum of component C, the innovations over the spectral range 290 to 305 nm were significantly reduced, whilst over the region 220 to 290 nm, the innovations remained almost unchanged. Figure 4.35(c) shows that the spectral lack-of-fit increased from 17.5% to 18.5% when the matrix of reference measurement functions was augmented with a poor estimate of the spectral profile for component C. This was because the model was not accounting for all of the spectral contribution from component C, so the Kalman filter was attempting to minimise the innovations by increasing the state parameter values for component B.

Another purpose for performing these experiments was to determine whether the adaptive Kalman filter could be implemented as a method for SMCR. The initial idea was to sequentially locate new spectral species from the adaptive variance values and using these augmented spectra as initial estimates for iterative target testing factor analysis. However, it was apparent from the results above, and from early tests using ITTFA that the augmented adaptive variance spectra for overlapped species such as component C did not have enough structure for the target testing to converge to feasible pure component spectra. Furthermore, existing pure variable methods such as OPA or VVSP could equally be applied to identify the purest spectra.

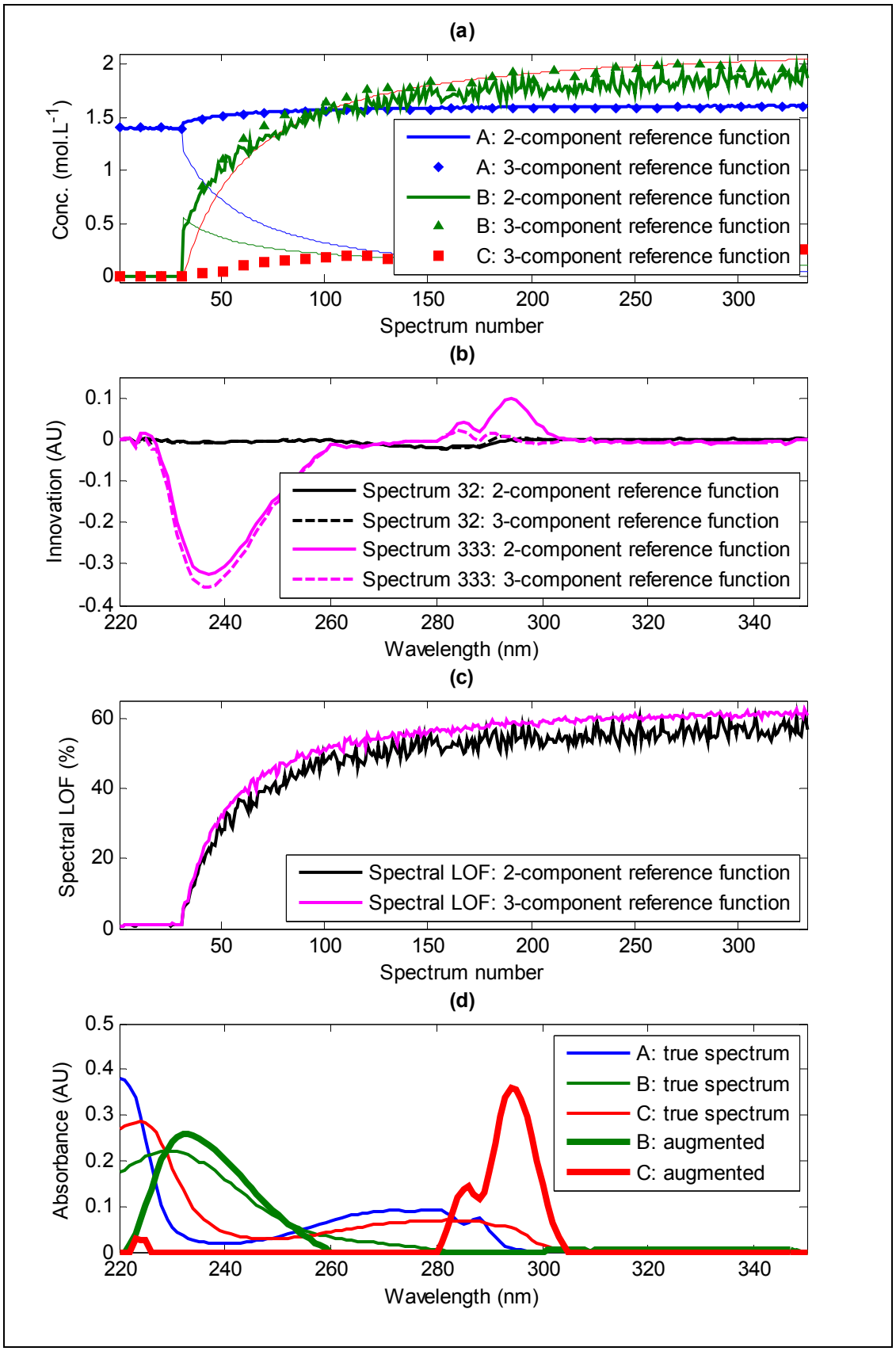


Figure 4.35: Outputs from the adaptive Kalman filter applied to the first 40 spectra of the simulated data set $D2_{sim}$ using incomplete and augmented reference measurement functions: (a) estimated state parameters; (b) spectral innovations; (c) spectral lack-of-fit; (d) true spectral profiles and augmented reference measurement functions.

4.2.9.2 Conclusions: Application of the adaptive Kalman filter to simulated UV data

Throughout the analytical chemistry and chemometrics literature, one of the most flexible and widely used versions of Kalman filter is the adaptive Kalman filter. The linear Kalman filter described in the previous paragraph assumes that the measurement noise variance (R) is constant for each of the variables. It is possible to modify the Kalman filter so that a vector of measurement noise variances can be supplied for data with heteroscedastic noise, but the noise distribution must be determined and provided before applying the Kalman filter. The adaptive Kalman filter does not use a constant value for R but allows it to adapt during the recursive estimation-correction calculations. A moving window is used to calculate the RMS value of the previous m values in the innovations sequence and R is updated at each step to adapt to the innovations values. The useful feature of this approach is that it allows the Kalman filter to compensate for certain model errors by reducing the sensitivity of the Kalman gain and state update calculations in regions where the measurement model is in error. Furthermore, the vector of adaptive measurement noise variances can be used to augment the matrix of reference measurement functions or correct an existing reference measurement function suspected to be inaccurate.

The initial motivation for investigating the adaptive Kalman filter was assess whether it could be used as the basis of a SMCR method to provide good initial estimates of the pure component concentration profiles when a full measurement model is not available. These estimates may be optimal already (in a constrained least-squares sense) or could be further refined using constrained ALS. The basic requirement for using the adaptive Kalman filter in this manner is that the number of components to recover must be pre-determined. Furthermore, the Kalman filter calculations require at least one reference measurement function so it is necessary to identify a suitable spectrum that may represent a pure spectral profile. The adaptive Kalman filter was implemented as Matlab function (`AdaptiveKF.m`) and included the additional calculations described by Rutan *et al.*[57, 58, 66] that allow the reference measurement function to be augmented or updated using the adaptive variance spectrum. To test the approach, the adaptive Kalman filter was applied to a simulated UV reaction data set ($\boldsymbol{D2}_{sim}$) using an incomplete reference measurement function. One of the restrictions of using the adaptive Kalman filter to sequentially estimate the spectral profiles of each component is that the data must first be examined to identify where the largest model errors will occur. This can be done by assessing the innovations based lack-of-fit values at each step; or it could be done using PCA or a SMCR

method such as EFA, OPA, VVSP, SIMPLISMA *etc.* to indicate when new components are introduced or reach their maximum concentration.

For the simulated data, it was known that the maximum concentration of component B occured at spectrum 32 and component C reached its maximum concentration at spectrum 333. Starting with a 1-component reference measurement function comprising the first spectrum of the data set, the adaptive Kalman filter was applied to the first 32 spectra. As expected the measurement model error increased significantly at spectrum 32 and the resulting adaptive measurement variance spectrum was used to augment the reference measurement function. The spectrum derived from the vector of adaptive measurement noise variances was a reasonable approximation of the true spectral profile of component B but also exhibited significant error where the spectrum of component B was highly overlapped with the spectrum of component A. Applying the Kalman filter a second time using the augmented (2-component) matrix of reference measurement functions offered only a small improvement in prediction error for component A at spectrum 32 (the error for a 1-component reference measurement function was 14.5%; the error for 2-component reference measurement function was 14.0%). The procedure was then repeated using the complete data set to obtain an estimate of component C. However, the estimated spectrum for component C was only non-zero over the region that was not completely overlapped with the reference measurement functions for components A and B. Using an augmented (3-component) reference measurement function did not improve the accuracy of the final state parameter estimates and they were unsuitable as initial estimates for further refinement.

This experiment demonstrated that for highly overlapped data such as UV spectra, there was not sufficient selectivity in variable mode for the adaptive Kalman filter to accurately estimate the state parameters. This limitation of the adaptive Kalman filter is well reported in the literature. However the objective was to assess whether the Kalman filter could be used to provide suitable starting estimates for further refinement using other methods such as CALS. Although the results were not shown, iterative target transformation factor analysis was applied to the pure component spectral profiles estimated using the adaptive Kalman filter. This did offer some improvement as the zero values corresponding to regions of high overlap were replaced with non-zero values. However, the approach did not seem to provide any advantage over existing methods and was not pursued further.

4.2.10 Application of VVSP to simulated UV data

The Vertex Vector Sequential Projection (VVSP) method is a SMCR method that was published in 2006 by Wang *et al.*[95, 96]. The VVSP method is a pure variable approach that identifies the purest spectra (vertex vectors) of a normalised bilinear data set. After the first VVSP component has been identified, the null matrix of that spectrum is calculated and used to identify the next component. The null matrix will span the spectral space orthogonal to the previously located components and should therefore yield pure component spectra that are most dissimilar to previous components, even if they only account for a small percentage of the variance in the data. The VVSP method is therefore suitable for providing initial estimates of the pure component spectra. The initial estimates can then be further refined using a constrained ALS approach as described by the authors. In this work, VVSP was used to provide a set of reference measurement functions (spectral profiles) to initiate the VAKFISO method.

The Vertex Vector Sequential Projection method was implemented as Matlab script (`VVSP.m`) and then applied to the simulated UV data set $\boldsymbol{D2}_{sim}$. The aim of this experiment was to investigate the ability VVSP to detect the true number of components contributing to an overlapped data set and also to locate the spectra that best resemble the true, pure spectral profiles. This would also demonstrate that pure variable based SMCR methods such as VVSP, OPA, SIMPLISMA *etc.* are often unable to successfully recover the true spectral (or concentration) profiles from data that comprise heavily overlapped spectra.

When VVSP was applied to $\boldsymbol{D2}_{sim}$, the spectral normalisation was set to p = 2 (normalise each spectrum to unit length). The number of components to locate (`NL`) was set to 8. The results obtained are shown in Figure 4.36. Figure 4.36(a) shows the $f(\boldsymbol{w}_j)$ values used to locate each VVSP spectrum. The profiles show that the magnitude of the $f(\boldsymbol{w}_j)$ values decreased with each successive VVSP component located. This is to be expected as each normalised spectrum $\boldsymbol{y}_j^T$ is projected into the null space of the previously located VVSP spectra stored in the matrix $\boldsymbol{Z}_M$. This plot indicates that the $f(\boldsymbol{w}_j)$ values for the fourth VVSP component are very low and resemble unstructured random noise. This indicates that three components were sufficient to model the data. This result was expected as the true rank of the data was three. The Durbin-Watson and

$\log_{10}(SSQ(\delta \boldsymbol{Y}))$ values shown in Figure 4.36(b, c) also indicate that the rank of the data was three.

The pure component spectral profiles located using VVSP are shown in Figure 4.36(d). VVSP component 1 (corresponding to spectrum number 327) was found to be an excellent match with the true spectral profile of component C. VVSP component 2 (corresponding to spectrum number 24) was also found to be an excellent match with the true spectral profile of component A. VVSP did successfully locate the spectrum number corresponding to the maximum concentration of component B at spectrum number 32, but because this spectrum was actually a mixture of components A and B, the VVSP spectrum did not match the true spectral profile of component B.

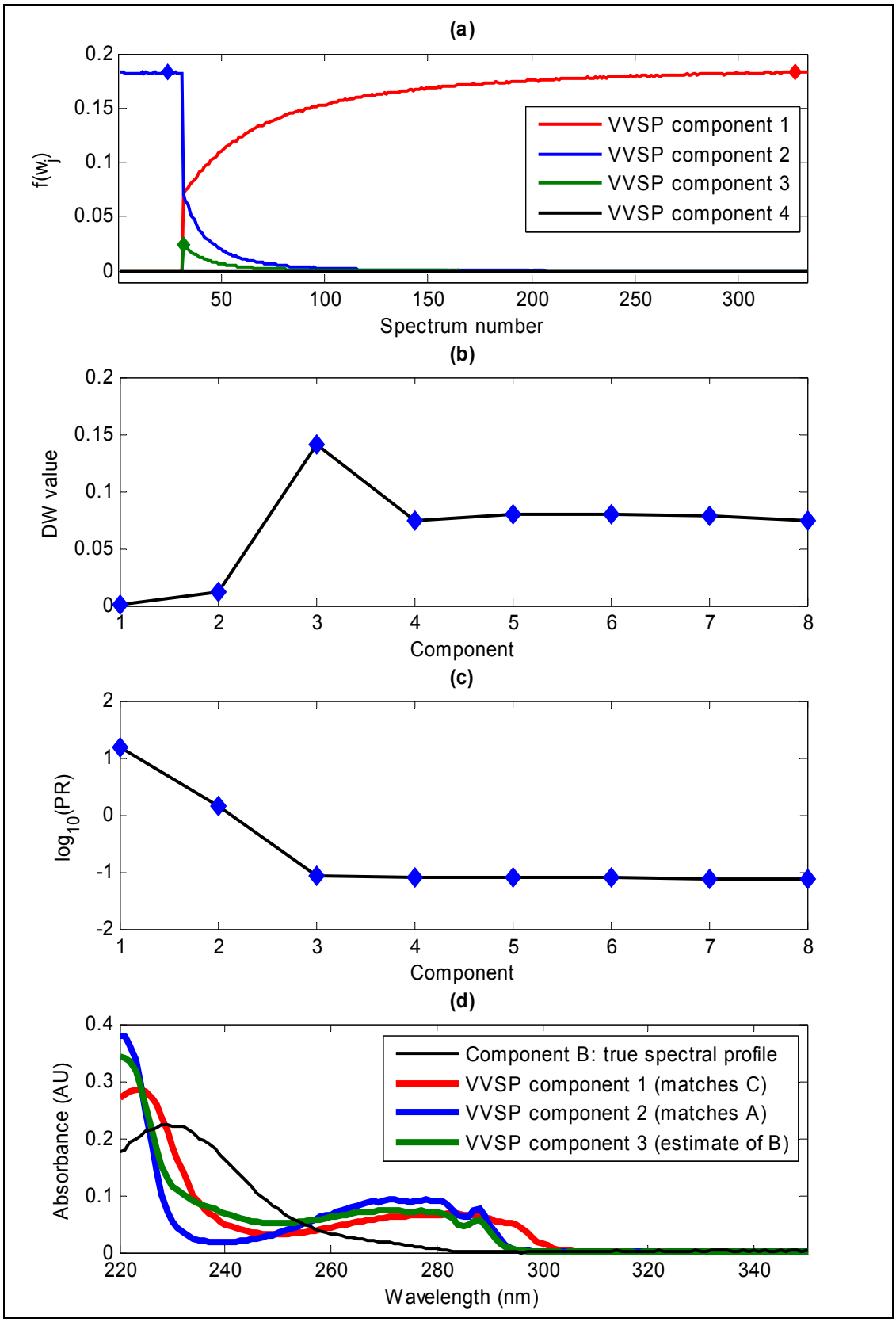


Figure 4.36: Results obtained from VVSP analysis of the simulated data set D2sim. (a) $f(w_j)$ values for the first four VVSP components; (b) the Durbin-Watson values for each VVSP spectrum; (c) $\log_{10}(SSQ(\delta Y))$ values plotted against number of VVSP components; (d) spectral profiles for the first three VVSP components – VVSP component 3 was an estimate of the true spectral profile of component B.

The concentration profiles estimated using least-squares using the VVSP spectra ($\boldsymbol{S}_{VVSP}$) are shown in Figure 4.37. As neither the spectral or concentration profiles have any negative regions, applying a constrained alternating least squares optimisation did not improve $\boldsymbol{S}_{VVSP}$ or $\boldsymbol{C}_{VVSP}$. The least-squares estimated concentration profile of VVSP component 1 exhibited excellent correlation with the true, scaled concentration profile of component C. Only a small scaling factor discrepancy between the two profiles was evident. The least-squares estimated concentration profile of VVSP component 2 was well correlated with the true, scaled concentration profile of component A for the first 31 spectra, but exhibited a step change to zero when component B appeared. The least-squares estimated concentration profile of VVSP component 3 also exhibited excellent correlation with the true, scaled concentration profile of component B but a there was a significant scaling discrepancy between the two profiles. This scaling ambiguity is common to all SMCR methods but can be minimised by invoking additional closure and equality constraints using additional knowledge about the system.

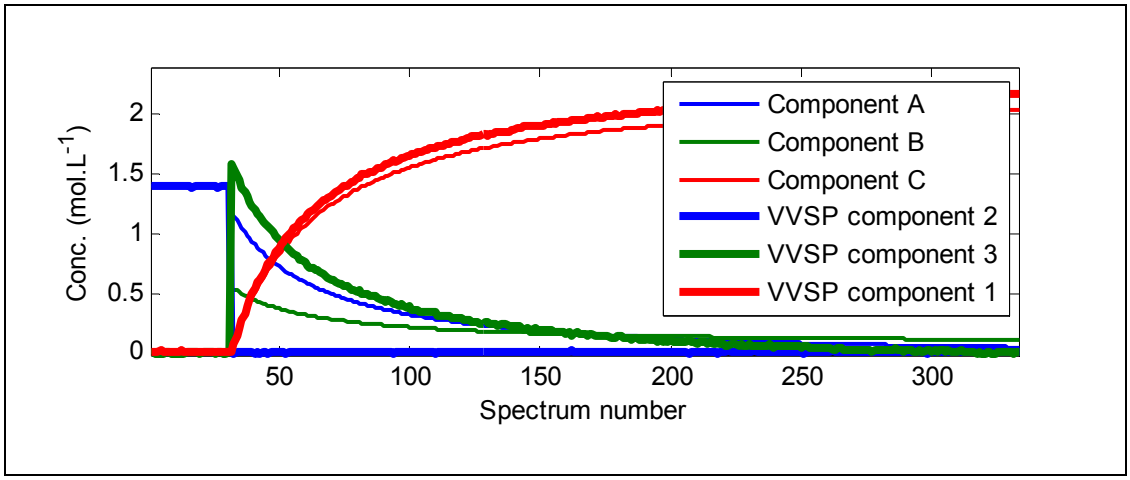


Figure 4.37: Comparison of the true, scaled concentration profiles used to generate the simulated data set $D2_{sim}$ and the least-squares estimate of the concentration profiles calculated using the VVSP pure component spectra. Note the step change in the concentration profile of VVSP component 2 (corresponding to component A) at spectrum 32.

VVSP was applied to the simulated data to find the correct number of components. Although the pure component concentration and spectral profiles were optimal in a least-squares sense, they did not completely match the true pure component profiles used to create the data set. This experiment demonstrated that VVSP can provide suitable starting estimates of the pure component profiles, but further optimisation was required.

Another useful feature of a pure variable SMCR method such as VVSP is that the purity of spectra can be assessed by comparing the correlation coefficients of each VVSP spectrum

and the corresponding least-squares estimated spectrum calculated using the normalised matrix of $f(\boldsymbol{w}_j)$ values.

To calculate the correlation coefficients, the dimensionality of the original data set $\boldsymbol{D2}_{sim}$ was first reduced by reconstructing the data using the first three principal components to yield $\bar{\boldsymbol{D}}$. The least-squares spectra were estimated using the matrix of normalised $f(\boldsymbol{w})$ values, denoted $\boldsymbol{Fw}$, using the equation $\hat{\boldsymbol{S}} = \left(\boldsymbol{Fw}^{+}\bar{\boldsymbol{D}}\right)^{T}$. The columns of $\hat{\boldsymbol{S}}_{LS}$ were normalised to unit length and the correlation coefficients of each column of $\boldsymbol{S}_{VVSP}$ and the corresponding column of $\hat{\boldsymbol{S}}_{LS}$ were calculated. The correlation coefficient for VVSP spectrum 1 and the corresponding least-squares spectrum was 0.9998; the correlation coefficient for VVSP spectrum 2 and the corresponding least-squares spectrum was 0.9999; the correlation coefficient for VVSP spectrum 3 and the corresponding least-squares spectrum was 0.8872. This indicated that for VVSP components 1 and 2, the "pure" spectra identified using VVSP as pure spectral profiles were very likely to correspond to actual pure spectra, whilst VVSP component 3 had clearly located the purest, mixture spectrum. This approach can be extremely useful for identifying which VVSP spectra are closest to isolated, pure spectral profiles in the absence of any prior knowledge of the system. The VVSP and least-squares estimated spectra are shown in Figure 4.38.

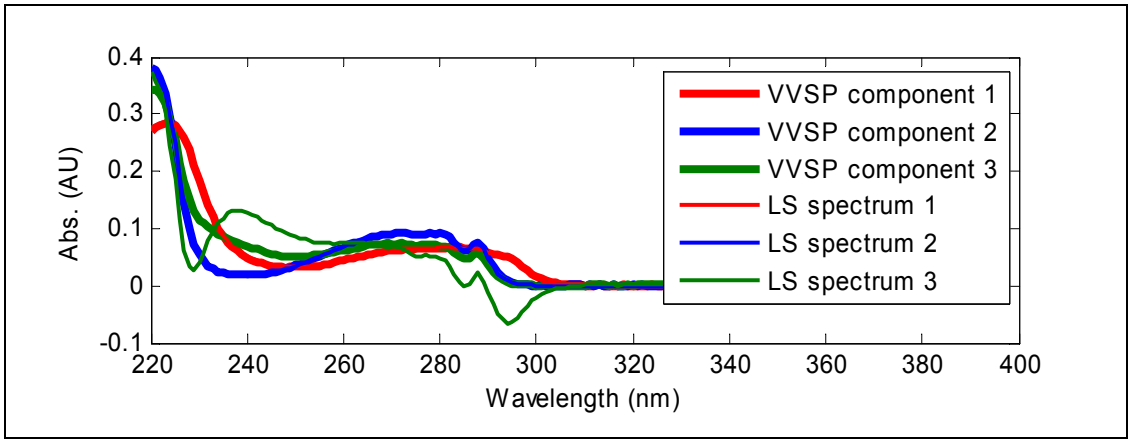


Figure 4.38: Comparison of the VVSP and least-squares estimated spectra. The least-squares estimated spectra were calculated using the normalised vectors of f(w) values, that formed the columns of the matrix Fw. The correlation between VVSP spectrum 1 and the least-squares spectrum was 0.9998, the correlation between VVSP spectrum 2 and the least-squares spectrum was 0.9999 and the correlation between VVSP spectrum 3 and the least-squares spectrum was 0.8872. This seems to be a convenient method for identifying which spectra are purest without requiring any prior knowledge.

4.2.10.1 Conclusions: Application of VVSP method to simulated UV data

A relatively new SMCR method called Vertex Vector Sequential Projection[95, 96] was implemented as a Matlab function (`VVSP.m`). This method locates the mixture spectra that most closely resemble pure spectral profiles (vertex vectors). The fundamental principal of VVSP is that by applying p-normalisation to each spectrum in the data set, the spectra can be represented as a points distributed across a polyhedral hyper-"spherical" surface. The purest spectrum will be located at the vertices of the hyper-surface. Although the calculations used to locate each pure spectrum are quite different to those used by OPA, both methods were found to produce almost identical results for the UV data.

4.2.11 Application of VAKFISO to simulated UV data

4.2.11.1 Examination of the effect of the weighting coefficients

The aim of this set of experiments was to learn how the VAKFISO method performed when different weighting coefficients (α_1, α_2, α_3 and α_4) were used to calculate the weighted residual matrix $\boldsymbol{E}$ (used during the NGL/M optimisation step).

Identification of suitable method parameters

To identify suitable starting parameters, the magnitude of the elements in an innovations vector and state-parameter error covariance matrix obtained from a fully modelled system were assessed. As VAKFISO was developed as a SMCR method, it was necessary to approximate a fully modelled system using the first N primary eigenvectors that span the spectral space of the data set. The primary eigenvectors (denoted $\bar{\boldsymbol{V}}$) obtained by application of singular value decomposition to the data set $\boldsymbol{D2}_{sim}$ were set as the matrix of reference measurement functions, $\boldsymbol{S}$. The vectorised adaptive Kalman filter (`VecAdaptiveKF.m`) was applied to the data set $\boldsymbol{D2}_{sim}$ using a window size of 4.

The approximate magnitude of the elements in the innovations vector were summarised by calculating the root-mean-squared value. The RMS value was 0.0011

The sum of the diagonal elements in state-estimate covariance matrix ($\boldsymbol{P}$) was 9.44×10^{-6}. The second term of the weighted variance covariance shown in equation 2.44 is the product of the innovations ($\boldsymbol{V}$) and the sum of the diagonal elements of $\boldsymbol{P}$ (denoted Π). For a three component model of the data set $\boldsymbol{D2}_{sim}$, the approximate value of Π for a matrix of optimised reference measurement functions was 9.44×10^{-6}.

The contribution from first term ($\alpha_1 \boldsymbol{V}$) can be given an approximately equivalent weighting to the second term ($\alpha_2 \Pi \boldsymbol{V}$) by using a weighting coefficient value $\alpha_1 = 1\times10^{-7}$ to 1×10^{-5}. However, when testing the initial parameters using $\alpha_1 = 1\times10^{-6}$, it was found that the trace of the error variance matrix was larger (approximately 1×10^{-5}). Experimentation revealed that if the RMS value of the second term ($\alpha_2 \Pi \boldsymbol{V}$) was approximately one to ten percent of the RMS value of the first term ($\alpha_1 \boldsymbol{V}$), the method would converge faster. For data set $\boldsymbol{D2_{sim}}$, a value of $\alpha_1 = 1\times10^{-3}$ was used. The `VAKFISO` script displays the RMS value for each of the four terms contributing the weighted residual matrix so that the user can adjust the values of α_1, α_2, α_3 and α_4.

The third and fourth terms of equation 2.44 are $\alpha_3 \Sigma \boldsymbol{V}$ and $\alpha_4 \Xi \boldsymbol{V}$ respectively. The values Σ and Ξ can range from 0 (complete non-negativity) to 1 (complete negativity) in the test spectra and estimated state-parameters. The product of $\alpha_3 \Sigma \boldsymbol{V}$ and $\alpha_4 \Xi \boldsymbol{V}$ terms will approach zero as Σ and Ξ approach zero. A value of 1 for the weighting coefficients α_3 and α_4 should be appropriate, although they can be increased to 10 or larger to make the product of the terms even larger when Σ and Ξ are non-zero.

Application of VAKFISO using the initial method parameters

The initial examination of the data in the previous section provided suitable weighting coefficients that could be used to apply VAKFISO to the simulated data set. Using the parameters listed in Table 3.9, VAKFISO was applied to the data set $\boldsymbol{D2_{sim}}$. The algorithm did not achieve convergence and therefore terminated when the maximum number of iterations was reached. However, observing the plots of the estimated spectra and state-estimates during the optimisation process revealed that the algorithm reached the final solution after approximately fifty iterations and did not visibly improve during subsequent iterations.

The RMS residuals between the normalised spectra obtained using VAKFISO and the true normalised spectra are shown in Table 4.5. For reference, the spectral and concentration profile RMS residuals were also calculated for the vectorised adaptive Kalman filter using the true spectra; the pure component spectra located using VVSP and the corresponding LS concentration profiles; and the VVSP pure component spectra used as reference

measurement functions for the vectorised Kalman filter. The rescaled, true concentration profiles calculated as described in section 4.2.9.1 were used to calculate the RMS error.

The second column of Table 4.5 shows the results obtained from applying the vectorised adaptive Kalman filter to the simulated data using normalised, true spectral profiles as the reference measurement functions. This provided a bench mark against which the other results could be compared. As the true, normalised spectral profiles were used as the reference measurement functions, calculation of the spectral model residuals was not applicable. As expected, the value of the concentration residuals was also low, ranging from 3.10×10^{-3} to 1.09×10^{-2} mol.L^{-1}. This resulted in a concentration RMSE of 7.27×10^{-3} mol.L^{-1} and was the lowest concentration RMSE obtained. The root-mean-square sum of residuals calculated from the innovation matrix was 2.00×10^{-3} AU and the innovations vectors resemble zero-mean white noise as expected. The state parameter variances for components A, B and C were 1.02×10^{-4}, 3.95×10^{-5} and 2.11×10^{-4} mol^2.L^{-2} respectively. These values represent the state-estimate variances for a model that was optimal in the sense that the true spectral profiles were provided.

To allow a comparison to be made, the same set of statistics were calculated for the estimates of the pure component spectra located using VVSP and the corresponding concentration profiles calculated using least squares. The results are shown in the third column of Table 4.5. The spectral RMSE values indicate how closely the estimated pure component profiles match the true spectral profiles. As described previously in section 4.2.9.2, it could be seen visually that the spectrum for VVSP component 1 exhibited excellent correlation with the true spectral profile of component C, whilst VVSP component 2 exhibited excellent correlation with component A. This was reflected in the spectral RMSE values of 7.59×10^{-4} AU for component A and 2.30×10^{-3} AU for component C. The RMSE value for component B was larger as the correlation of the corresponding VVSP spectrum was not so good. The overall spectral RMSE value was 1.88×10^{-2} AU and was a measure of how closely $\hat{\boldsymbol{S}}$ matched $\boldsymbol{S}$. The concentration RMSE values were calculated using the LS estimated concentration profiles. The concentration RMSE was 2.21×10^{-1} mol.L^{-1}. As the concentration matrix was calculated using least-squares, the RMS of the data residual matrix was minimised. RMS of the data residual matrix was 1.40×10^{-3} AU. This value is actually slightly lower than the innovations matrix RMS value calculated for the true model.

To allow the performance of VAKFISO to be compared with the solution provided using VVSP, the vectorised Kalman filter was used to estimate the concentration profiles of the simulated data using the VVSP spectra as reference measurement functions. The results are shown in column four of Table 4.5. The spectral matrix RMSE values are unchanged and the concentration RMSE values were very similar to those obtained using least-squares. The concentration RMSE was reduced slightly (1.99×10^{-1} mol.L^{-1} for the VAKF compared to 2.21×10^{-1} mol.L^{-1} for the least-squares estimates). The state-estimate variances for each component are also listed. Since only two of the three spectral profiles were a good match to the true spectral profiles, one would expect the state-estimate variance values to be increased relative to those obtained using the true spectral profiles. The state estimate variance for component A (corresponding to VVSP component 2) was 1.30×10^{-3} mol^2.L^{-2} (*cf.* 1.02×10^{-4} mol^2.L^{-2} for the true model). This demonstrates that although the VVSP spectrum corresponding to component A showed excellent correlation with the true spectral profile, it is the simultaneous performance of all components contributing to the model that will influence the error variances. The state estimate variance for component B (corresponding to VVSP component 3) was 2.20×10^{-3} mol^2.L^{-2} (*cf.* 3.95×10^{-5} mol^2.L^{-2} for the true model). This was the largest state estimate standard deviation value and indicated that the VVSP spectrum corresponding to component B contributed the largest error to the measurement model. This supports the observation that the VVSP spectral profile corresponding to component B was not accurate. Finally, the state estimate standard deviation for component C (corresponding to VVSP component 1) was 9.57×10^{-4} mol^2.L^{-2} (*cf.* 2.11×10^{-4} mol^2.L^{-2} for the true model). A small state estimate variance indicated that the VVSP spectrum of the component corresponding to component C was a good estimate and did not contribute significant error to the model. This was also supported by the concentration RMSE value for component C which was lower than for the other components.

Table 4.5: Comparison of VAKFISO and VVSP applied to simulated data.

	VAKF using known spectra	**VVSP spectra (LS)**	**VAKF using VVSP spectra**	**VAKFISO with $\alpha_2 = 1$**	**VAKFISO with $\alpha_2 = 0$**
Spectral RMSE (A)	N/A	7.59×10^{-4}	7.59×10^{-4}	9.00×10^{-3}	8.40×10^{-3}
Spectral RMSE (B)	N/A	5.32×10^{-2}	5.32×10^{-2}	1.10×10^{-3}	2.80×10^{-2}
Spectral RMSE (C)	N/A	2.30×10^{-3}	2.30×10^{-3}	9.00×10^{-3}	3.55×10^{-2}
Total /3	**N/A**	**1.88×10^{-2}**	**1.88×10^{-2}**	**6.37×10^{-3}**	**2.40×10^{-2}**
Concentration RMSE (A)	7.80×10^{-3}	3.15×10^{-1}	3.21×10^{-1}	2.41×10^{-1}	6.87×10^{-1}
Concentration RMSE (B)	3.10×10^{-3}	2.40×10^{-1}	2.22×10^{-1}	2.81×10^{-1}	2.50×10^{-1}
Concentration RMSE (C)	1.09×10^{-2}	1.08×10^{-1}	5.39×10^{-2}	4.17×10^{-1}	7.35×10^{-1}
Total / 3	**7.27×10^{-3}**	**2.21×10^{-1}**	**1.99×10^{-1}**	**3.13×10^{-1}**	**0.557**
Innovation / Residual matrix RMSE	2.00×10^{-3}	1.40×10^{-3}	4.50×10^{-3}	1.00×10^{-3}	1.00×10^{-3}
State parameter variance (A)	1.02×10^{-4}	N/A	1.30×10^{-3}	8.68×10^{-6}	7.72×10^{-6}
State parameter variance (B)	3.95×10^{-5}	N/A	2.20×10^{-3}	1.02×10^{-5}	2.18×10^{-5}
State parameter variance (C)	2.11×10^{-4}	N/A	9.57×10^{-4}	2.19×10^{-5}	1.19×10^{-5}
trace(***P***)	**3.52×10^{-4}**	**N/A**	**4.46×10^{-3}**	**4.08×10^{-5}**	**4.14×10^{-5}**

The results obtained from the application of VAKFISO to the simulated data using the weighting coefficients $\alpha_1 = 1.0\times10^{-3}$ and $\alpha_2 = 1$ are shown in the fifth column of Table 4.5 and are illustrated in Figure 4.39. The quality of the pure spectral profiles estimated using VAKFISO was quantified by the spectral RMSE values shown in the table. The estimated spectra and true spectral profiles are shown in Figure 4.39(a) and the residuals used to calculate the spectral RMSE values are shown in Figure 4.39(b). Visually, the VAKFISO spectra show very good correlation with the true spectral profiles and this was confirmed by the RMSE values. The RMSE values for components A and C were both 9.00×10^{-3} AU; this was not quite as good as the RMSE values produced using VVSP (7.59×10^{-4} and 2.30×10^{-3} AU respectively). The spectra in Figure 4.39(a) do indicate that there was some rotational ambiguity for components A and C. However, the pure spectral profile of component B estimated using VAKFISO exhibited excellent correlation with the true

spectrum. For component B, the corresponding VVSP pure component spectrum was least accurate and the improvement is reflected in the spectral RMSE values for component B. The RMSE value for the VVSP spectrum corresponding to component B was 5.32×10^{-2} AU whilst the RMSE value for the VAKFISO spectrum was 1.10×10^{-3} AU. The improvement in the pure component spectral profile of component B meant that the total error of the matrix of estimated spectral profiles ($\hat{\boldsymbol{S}}$) produced using VAKFISO was slightly lower than that produced using VVSP. However, examination of the individual spectral RMSE values revealed that VVSP produced better estimates of the spectral profiles for components A and C, whilst VAKFISO, produced a much better estimate of component B's spectral profile.

As the spectral profiles for the three components were all highly overlapped, there was some rotational ambiguity in the spectral profiles estimated using VAKFISO. Consequently, this led to some rotational ambiguity in the estimated concentration profiles. The estimated concentration and true concentration profiles are shown in Figure 4.39(c) and the residual used to calculate the concentration RMSE values are shown in Figure 4.39(d). It is apparent from Figure 4.39(c) that the concentrations of components A and B were over-estimated at the expense of component C, which was under estimated when compared with the true concentration profiles. The concentration RMSE values suggest that overall, the VAKFISO model was slightly worse than the VVSP model. The concentration RMSE value for the VAKFISO model was 3.13×10^{-1} mol.L^{-1}; the concentration RMSE values for VVSP model was 1.99×10^{-1} mol.L^{-1}. However, it is worth noting that the concentration profile of component A estimated using VVSP, shown in Figure 4.37(b), did not have the correct features and displayed a step change to zero at spectrum 32. The concentration profiles estimated using VAKFISO (Figure 4.39(c)) all had the correct features and were more feasible reaction profiles but did exhibit intensity ambiguities.

Two of the contributions to the weighted residual matrix minimised by the NGL/M optimisation are the Kalman filter innovation matrix and the sum of the diagonal elements of the state estimate variance covariance matrix. The innovations RMSE for the VAKFISO model was 1.00×10^{-3} AU and was lower than the value obtained using the true spectral profiles (2.00×10^{-3} AU). This was a consequence of the optimisation method as NGL/M is a least-squares optimisation method that will minimise the model residuals. Each innovations vector for the VAKFISO model resembled zero-mean, white noise as

expected. The innovations RMSE value indicated that the VAKFISO optimisation process converged to a feasible solution that minimised the weighted residual matrix whilst adhering to the non-negativity requirements.

The second parameter that contributes to the weighted residual matrix is the trace of the state estimate variance covariance matrix. The individual state estimate variance values are shown in Table 4.5. The sum of the individual state estimate variance values for the VAKFISO model was 4.08×10^{-5} $mol^2.L^{-2}$. This was lower than the value obtained using the true spectral profiles (3.52×10^{-4} $mol^2.L^{-2}$) and shows that VAKFISO successfully converged to a solution that minimised the elements of innovations matrix and the diagonal elements of the state estimate error covariance matrix. It was noted that when the vectorised adaptive Kalman filter was applied to the same data set using the first three right singular vectors, the innovations RMSE value was 1.00×10^{-3} AU and the sum of the state estimate standard deviations was 9.44×10^{-6} $mol^2.L^{-2}$. This suggested that whilst neither the right singular vectors, nor the calculated state parameters were strictly non-negative, they were a solution that minimised the residual matrix. As the singular vectors were orthogonal, the degree of spectral overlap was minimised and this helped to reduce the values of the diagonal elements of the state-estimate covariance matrix.

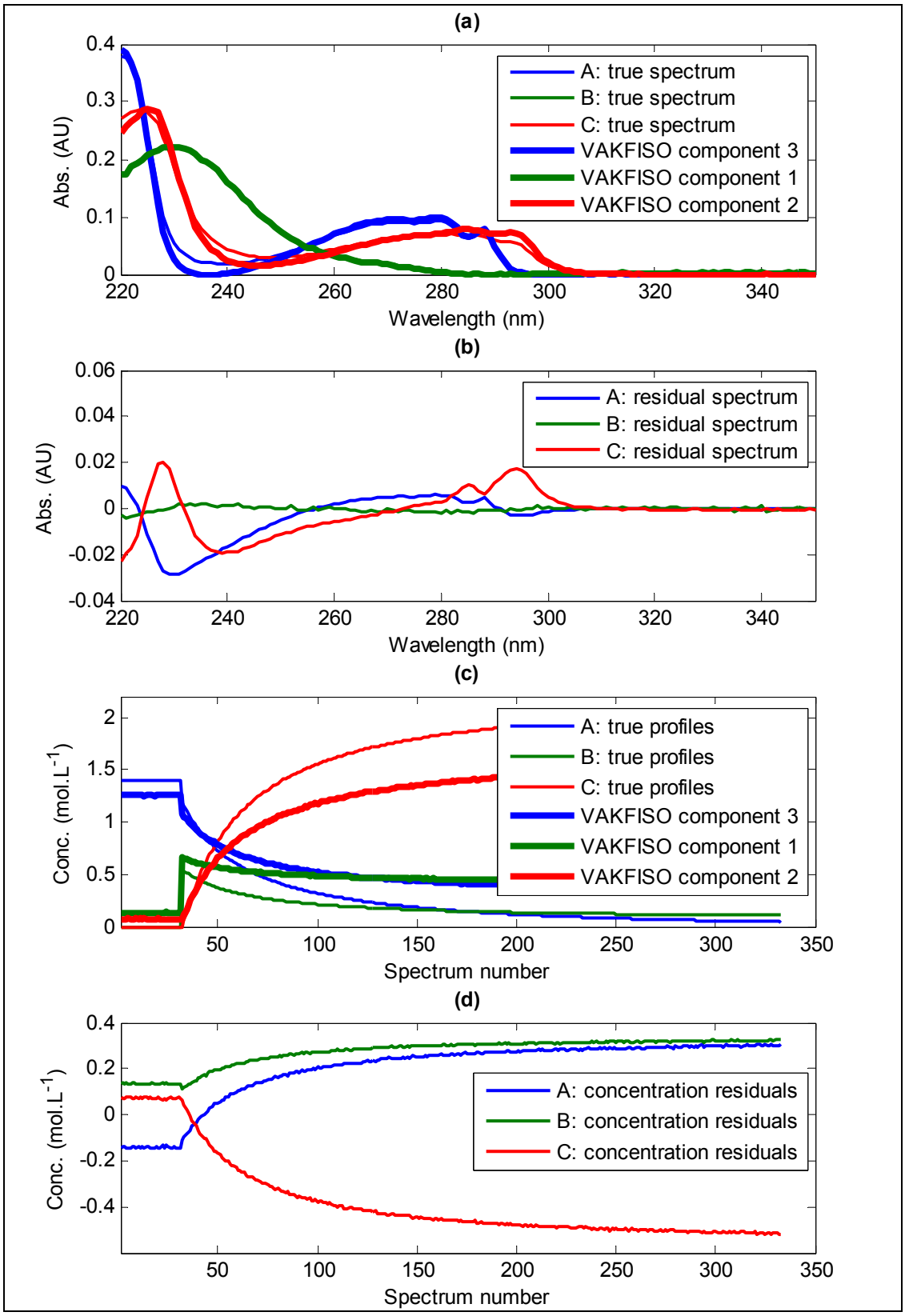


Figure 4.39: Results obtained by the application of VAKFISO to simulated data $D2_{sim}$ using the weighting coefficients $\alpha_1 = 1.0\times10^{-3}$ and $\alpha_2 = 1$: (a) comparison of pure component spectra estimated using VAKFISO and the true, normalised spectral profiles; (b) Residual spectra; (c) comparison of the pure component concentration profiles estimated using VAKFISO and the true, scaled concentration profiles; (d) concentration residuals calculated by subtraction of the VAKFISO estimated concentration profiles from the true, scaled concentration profiles.

Application of VAKFISO with exclusion of the state parameter error covariance term

To demonstrate that using the Kalman filter during the optimisation process provides a unique advantage over simple constrained least squares methods, the contribution of state parameter error covariance term was eliminated from the calculation of the weighted residual matrix by setting the value of α_2 to zero. All other parameters were set to the values listed in Table 3.9. Using these parameters, the VAKFISO method will optimise the model in the sense that it minimises the residual sum-of-squares. The same constraints of spectral and state-parameter non-negativity were applied during the optimisation. As the sum of the state estimate variances were not used in the calculation of the weighted residual matrix, they would not be minimised directly.

The results obtained by applying VAKFISO using these parameters are shown Table 4.5 and Figure 4.40. Visual inspection of the estimated spectral profiles in Figure 4.40(a) revealed that the optimised spectra were not as highly correlated to the true spectral profiles as those obtained in the previous model. This was confirmed by the spectral RMSE values. The spectral RMSE value for this model was 2.40×10^{-2} AU, which was larger than both the previous VAKFISO model (6.37×10^{-3} AU) and the VVSP pure component spectra (1.88×10^{-2} AU). It was apparent that in this model, the estimated spectral profiles for components B and C were not as accurate as those obtained in the previous VAKFISO model.

The errors in the estimated spectral profiles also led to a larger error in the calculation of the concentration profiles. This is illustrated in Figure 4.40(c) and Figure 4.40(d). Component C had the correct profile but exhibited considerable intensity ambiguity and was under estimated; whilst components A and B were both over estimated. Furthermore, component A did not have the correct features and still exhibited a large step at spectrum 32. The concentration RMSE value for this VAKFISO model was 0.557 $mol.L^{-1}$.

Despite the large errors in the estimated pure component spectral and concentration profiles, the RMS of the innovations matrix was still minimised to the same value as the previous model (1.00×10^{-3} AU). As with the previous VAKFISO model, the innovations vectors resemble zero-mean, white noise. Consequently, the sum of the state estimate error standard deviations was also very small (1.09×10^{-2} $mol.L^{-1}$). Although this value was not as low as the value obtained with the previous VAKFISO model (2.02×10^{-3} $mol.L^{-1}$) it did confirm that when the elements of the innovations matrix residual are minimised, the

diagonal elements of the error covariance matrix are also minimised. This suggests that although the innovations and error covariance statistics of the two VAKFISO models are very similar, including the error covariance term in the calculation of the weighted residual matrix had a significant effect on the accuracy of the estimated pure component spectra.

Comments on the performance of VAKFISO applied to simulated data

The testing of the VAKFISO method using simulated data generated from known spectral and concentration profiles allowed the performance of this algorithm to be assessed. The algorithm is computationally intensive and requires the Kalman filter to be run many times. In the tests described above, the maximum number of iterations was 250 and in each test, this limit was reached. Although the algorithm did appear to converge to the final solution well before 250 iterations were performed, it is clear that the convergence criterion need to be chosen carefully. During the each iteration of the optimisation, the vectorised adaptive Kalman filter is called N^2 times, where N is the number of components to refine. This is because NGL/M optimisation will individually adjust the value of each element in the $(N \times N)$ transformation matrix $\boldsymbol{T}$. This meant that during each test, the vectorised adaptive Kalman filter was called approximately 2250 times. The calculation time for 250 iterations was approximately twenty minutes; if the original adaptive Kalman filter was called instead, this time would be increased to several hours.

The results of the tests using simulated data revealed that the concept of utilising the diagonal elements of the error covariance matrix to construct a weighted residual matrix did seem to offer an advantage over least-square minimisation alone. However, the final solution was not completely accurate and it is clear that in this case, VAKFISO can not be used in isolation. A benefit of VAKFISO is that it arrives at a set of feasible pure component spectral profiles using only basic non-negativity constraints. If necessary, this solution can be refined further by utilising knowledge of the system commonly utilised in MCR-ALS for example. Additional constraints that would be applicable to this example would be to utilise known regions of zero concentration. These can be identified from the data directly using by examining the VVSP $f(\boldsymbol{w})$ vectors or applying EFA.

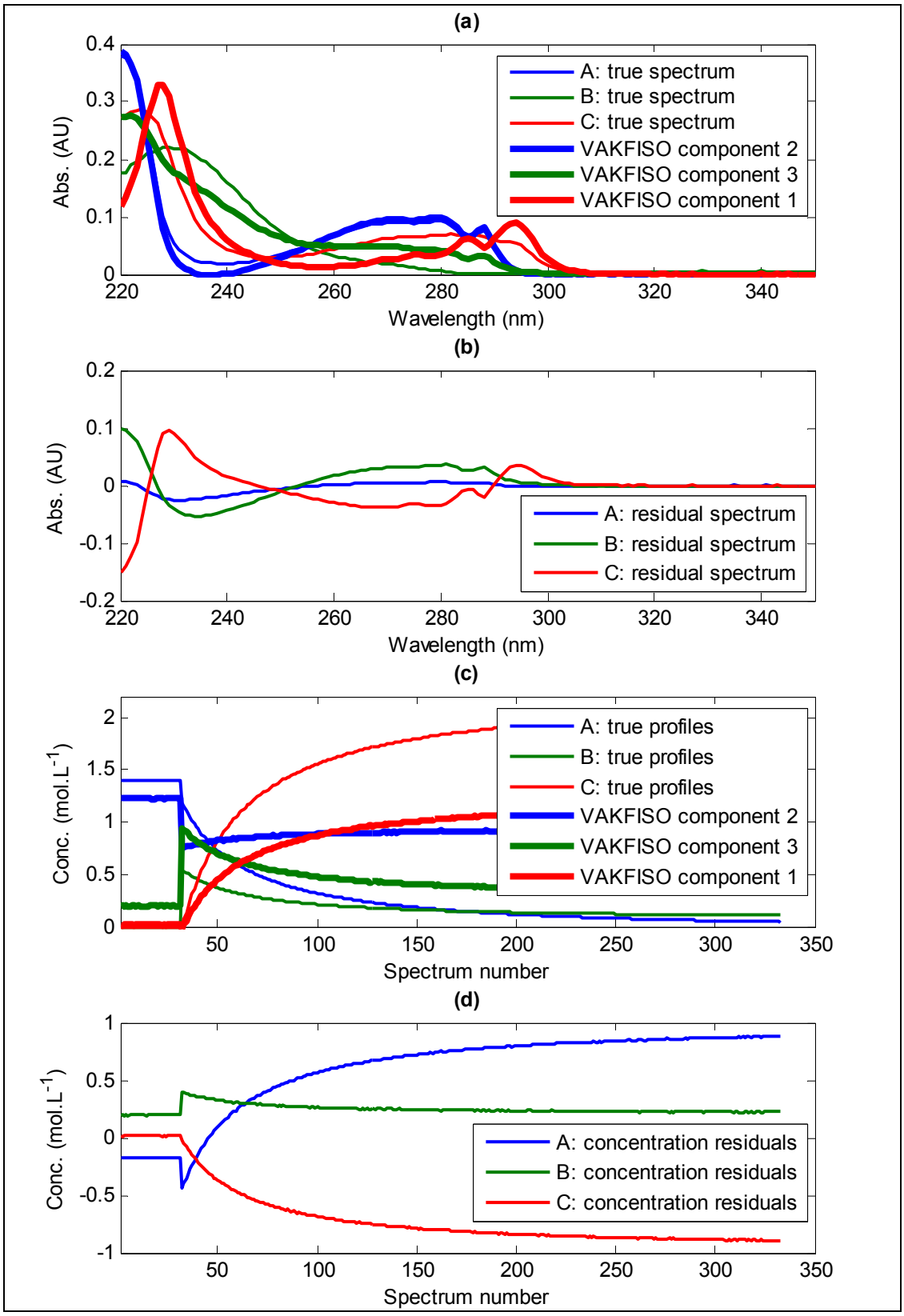


Figure 4.40: Results obtained by the application of VAKFISO to simulated data $D2_{sim}$ using the weighting coefficients $\alpha_1 = 1.0\times10^{-3}$ and $\alpha_2 = 0$: (a) comparison of pure component spectra estimated using VAKFISO and the true, normalised spectral profiles; (b) Residual spectra; (c) comparison of the pure component concentration profiles estimated using VAKFISO and the true, scaled concentration profiles; (d) concentration residuals calculated by subtraction of the VAKFISO estimated concentration profiles from the true, scaled concentration profiles.

4.2.12 Application of VAKFISO to real UV data

4.2.12.1 Examination of the UV spectral data sets using PCA

PCA was applied each data set to allow the correct number of independent components contributing to the data to be determined. The eigenvalues, variance explained for each PC and cumulative variance explained values obtained by applying PCA to both un-centred and mean-centred data are shown in Table 4.6. The eigenvalues and variance explained values for the un-centred data show that the first PC explained 99.80% of the total variance. For the PCA models calculated using the un-centred data, the first principal component corresponded to the mean spectrum. The loading of principal component 1 matched the normalised mean spectrum of the data set. The scores profile for the second principal component correlated to the expected formation of product and the third component corresponded to the addition of benzyl bromide and its subsequent consumption. The scores vector for principal component 4 resembled random noise and its loading vector did not have any spectral features. This indicates that there were not any additional chemical species contributing to the data. It also indicates that there were no significant peak shifts that required additional principal components to model their contribution to the data.

The PCA models for the mean-centred data confirmed that two principal components were sufficient to model most of the structured variance about the mean. For the mean-centred data, the scores for the first principal component corresponded to a combination of the addition of benzyl bromide and the formation of 1-benzyl-1*H*-indole. The second component also corresponded to the addition of benzyl bromide and its subsequent consumption. In both cases, the absence of a significant third (mean-centred data) or fourth component (un-centred data) indicated that two of the chemical species co-vary.

Table 4.6: Table of eigenvalues and variance explained obtained by applying principal components analysis to the UV data set BnIndole_B2.00_MR0.67_UV.

	Un-centred data			Mean-centred data		
PC number	**Eigen-value**	**Var. (%)**	**Cum. Var. (%)**	**Eigen-value**	**Var. (%)**	**Cum. Var. (%)**
1	6.316	99.8001	99.8001	2.309×10^{-2}	81.9919	81.9919
2	1.045×10^{-2}	0.1652	99.9653	4.247×10^{-3}	15.0792	97.0711
3	2.021×10^{-3}	0.0319	99.9972	6.584×10^{-4}	2.3373	99.4084
4	3.140×10^{-5}	0.0005	99.9977	3.117×10^{-5}	0.1107	99.5190
5	1.730×10^{-5}	0.0003	99.9980	1.730×10^{-5}	0.0614	99.5805
6	1.653×10^{-5}	0.0003	99.9983	1.620×10^{-5}	0.0575	99.6379

The variance of each column of the residual matrices calculated using three-component PCA models were used to estimate the approximate measurement noise variance. The average variance value for the 131 spectral variables was approximately 1×10^{-6} AU^2.

4.2.12.2 Derivation of initial spectral profile estimates using VVSP

VVSP was individually applied to each of the three data sets to allow the resulting sets of initial estimates to be compared. The reason for performing the reaction with different reagent stoichiometries was to allow the VAKFISO algorithm to be initiated with different initial estimates of the spectral profiles.

Estimation of the number of components required

When VVSP was applied to the real UV data sets, it was more difficult to determine the correct number of components from the Durbin-Watson and $\log_{10}(SSQ(\delta \boldsymbol{Y}))$ values (shown in Figure 4.41). The Durbin-Watson calculation is used to determine the correlation or randomness of a vector. The DW values will tend towards zero if the values in the vector are highly correlated (and therefore represent a true signal or profile); conversely the DW values will increase as the degree of randomness increases[106]. The $\log_{10}(SSQ(\delta \boldsymbol{Y}))$ values indicate how the projection residuals decrease as additional VVSP components are located. If all of the structured variation in the data is captured by the VVSP components, the projection residuals for addition components will resemble random noise. When interpreting plots of the DW and $\log_{10}(SSQ(\delta \boldsymbol{Y}))$ values, one must consider the value and the difference between consecutive pairs of values. For the data sets BnIndole_B2.00_MR0.67_UV and BnIndole_B2.00_MR100_UV, the difference between consecutive DW values was largest between components three and four. This indicated that three VVSP components were appropriate. For the data set, BnIndole_B2.00_MR1.50_UV, the largest difference between consecutive DW values occured between components four and five. This suggested that four VVSP components were the most appropriate, but on examination of the spectral profiles for the first four VVSP components, it was obvious that VVSP component four was very highly correlated to VVSP component one (corresponding to 1-benzyl-1*H*-indole). Based upon this observation, it was concluded that only three components were required to model each data set. The $\log_{10}(SSQ(\delta \boldsymbol{Y}))$ projection values shown in Figure 4.41(b) also suggested that three VVSP components were adequate to model most of the structured variance in the data.

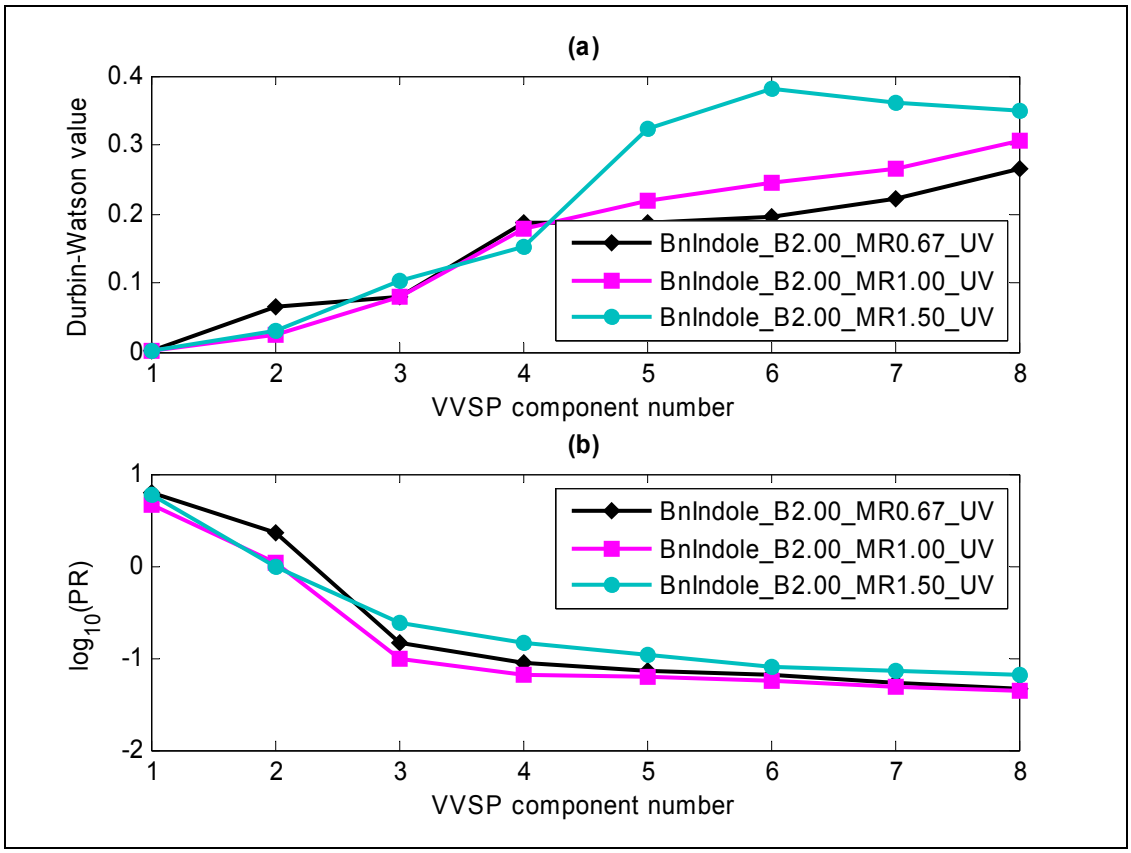


Figure 4.41: (a) Durbin-Watson values plotted against VVSP component number, calculated during the application of VVSP to real UV data sets acquired during the *N*-benzylation of 1*H*-indole reactions; (b) $\log_{10}$(SSQ(δY)) values plotted against VVSP component number.

Comparison of the VVSP spectral profiles obtained from each data set

The initial estimates of the pure component spectral profiles obtained by application of VVSP to each of the three data sets were compared visually (shown in Figure 4.42). As there was a region at the beginning of each data set where 1*H*-indole was the only component contributing to the data, one would expect the VVSP spectral profiles corresponding to this component to be the most similar. The spectral profiles for VVSP component 2 were overlaid and found to be almost identical. For the VVSP components that corresponded to1-benzyl-1*H*-indole (VVSP component 1) and benzyl bromide (VVSP component 3), there was a larger difference between the spectra obtained from each data set. The reason for this was that in each of the experiments, one or both of the reagents were not consumed completely. In experiment BnIndole_B2.00_MR1.00, the reagents 1*H*-indole and benzyl bromide were charged in equal amounts. The HPLC profiles for this experiment indicated that 1*H*-indole was not completely consumed and approximately 10% of the initial amount charged remained at the end of the reaction. The VVSP spectrum corresponding to 1-benzyl-1*H*-indole therefore included a small contribution from un-consumed 1*H*-indole.

In experiment BnIndole_B2.00_MR0.67, 1.5 molar equivalents of benzyl bromide were charged. The HPLC profiles for this experiment indicated that at the end of the reaction, 1*H*-indole was completely consumed but approximately 25% of the initial charge of benzyl bromide remained. Therefore approximately 25% of the contribution to VVSP spectrum corresponding to 1-benzyl-1*H*-indole was from benzyl bromide. The VVSP pure component spectral profiles corresponding to benzyl bromide were actually a weighted mixture of 1*H*-indole and benzyl bromide. As the reagent stoichiometries were different in each experiment, the VVSP spectra derived from them were also different.

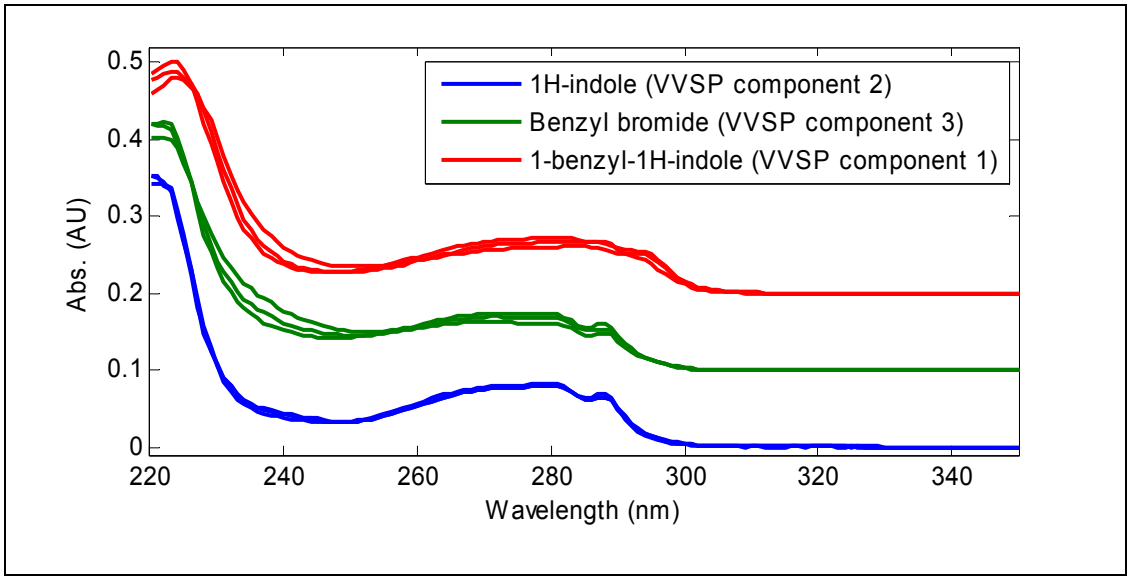


Figure 4.42: Comparison of the spectral profile estimates obtained by application of VVSP to each UV data set individually. The profiles for the VVSP components corresponding to benzyl bromide and 1-benzyl-1H-indole were translated by +0.1 AU and +0.2 AU respectively.

4.2.12.3 Application of VAKFISO using VVSP initial spectral profile estimates

Examination of the data sets using PCA and VVSP indicated that three-components were sufficient to model the data. Once the chemical rank of the data had been established, VAKFISO was applied to each UV data set individually using the parameters described in Table 3.9. The measurement noise variance was estimated to be 1.0×10^{-6} AU^2.

The first observation from these experiments was that VAKFISO did not converge to the same final estimates of the pure component spectral profiles when using the different initial estimates provided using VVSP. VAKFISO performed better for the experiments BnIndole_B2.00_MR0.67 and BnIndole_B2.00_MR1.00 than it did for experiment BnIndole_B2.00_MR1.50. The final estimates of the pure component spectral profiles obtained for experiment BnIndole_B2.00_MR1.50 were all very similar; consequently the resulting concentration profiles calculated using the Kalman filter were very noisy. It was

suspected that the reason VAKFISO produced very similar final spectral profiles for this experiment was because the reaction was performed using an excess of 1*H*-indole. The spectrum of 1*H*-indole was very similar to the spectrum of 1-benzyl-1*H*-indole and since an excess of 1*H*-indole was charged, it was not fully consumed at the end of the reaction. The 'pure' spectrum identified using VVSP at the end of the data set was therefore a mixture of 1*H*-indole and 1-benzyl-1*H*-indole. For this reason, the three VVSP spectra obtained from this experiment were all very similar and VAKFISO did not successfully separate them during the optimisation process.

As with the application of VAKFISO to the simulated data set, VAKFISO modified the initial spectral profiles obtained using VVSP to produce a new set of spectral profiles. The new spectral profiles were optimal in the sense that when they are used by the Kalman filter as reference measurement functions, they minimised the diagonal elements of the resulting state parameter error covariance matrix whilst also satisfying the spectral and concentration non-negativity constraints. Despite invoking non-negativity constraints through the use of weighted penalty functions during the NGL/M optimisation step, there remained sufficient rotational freedom in the spectral subspace to produce a range of feasible solutions. This rotational freedom prevented the recovery of a single final solution that matched the true underlying system and is a problem encountered in most curve resolution methods. When applied to real data, the VAKFISO method was not able to converge to a solution that was as close to the true model as that achieved for the simulated data. The reduced accuracy of the recovered spectral profiles from real data is believed to be a combination of the inherent heteroscedastic measurement noise and less selectivity in the concentration mode.

The initial results obtained by application of VAKFISO to the each of the data sets are shown in Figure 4.43 Visual comparison of the VAKFISO and true spectral profiles in Figure 4.43 indicates that 1*H*-indole (VAKFISO component 2) was the most accurately modelled component. This was not surprising as this component had a small selective region in the concentration mode during the first several spectra of each data set.

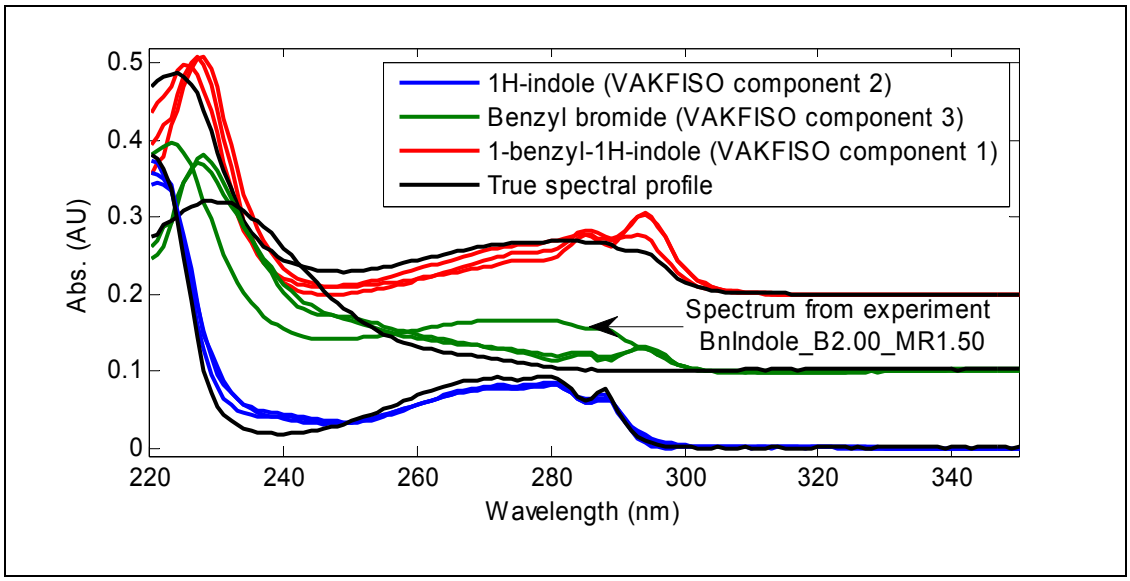


Figure 4.43: Comparison of the pure component spectral profiles estimated using VAKFISO and the true spectral profiles of 1*H*-indole, benzyl bromide and 1-benzyl-1*H*-indole. The profiles for the spectra corresponding to benzyl bromide and 1-benzyl-1H-indole were translated by +0.1 AU and +0.2 AU respectively. The estimated spectral profile of benzyl bromide obtained from experiment BnIndole_B2.00_MR0.67 is highlighted because it was very similar to the estimated spectral profiles of 1*H*-indole and 1-benzyl-1*H*-indole. This resulted in very poor of the concentration profiles.

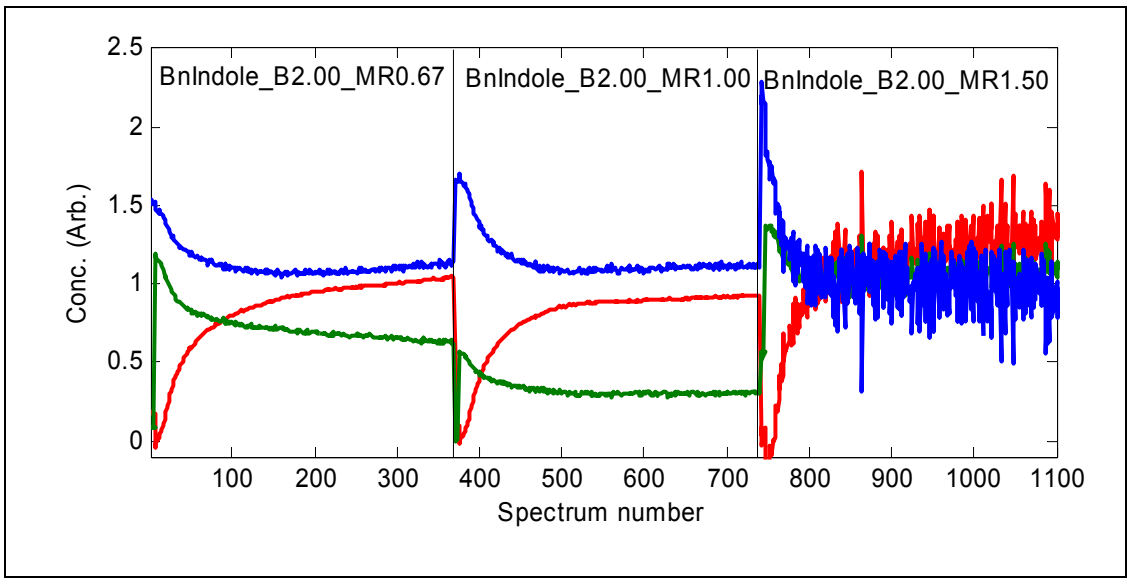


Figure 4.44: Concatenated concentration profiles calculated using the final spectral estimates produced using VAKFISO. VAKFISO was performed using initial spectral estimates obtained by applying VVSP to each data set individually. The noisy profiles obtained for the third reaction were a consequence of VAKFISO producing very similar spectral profiles for all three components.

4.2.12.4 Application of VAKFISO using a random transformation matrix

A set of initial spectral estimates were created using a (3×3) transformation matrix of random numbers. The spectra were normalised to unit length and VAKFISO was then applied to each UV data set using the parameters described in Table 3.9. The initial spectra comprised of positive and negative values so the VAKFISO algorithm would be iteratively modify the spectra to create non-negative profiles. Starting with random spectral profiles ensured that the initial state parameter error variance and innovation values were large, allowing the NGL/M optimisation to converge to smaller values.

Starting from a set of random spectral profiles did produce better results than starting from the VVSP derived spectra. The use of random spectral profiles overcame the problem of VAKFISO allowing the profiles of two components to converge to the same solution in an attempt to minimise the innovations and state parameter error variance values.

The results obtained from this experiment are shown in Figure 4.45. It was encouraging to note that although the VAKFISO algorithm was initiated using a set of random spectra, the spectral profile of each component obtained from the three different data sets were very similar. The estimated spectral profiles of 1*H*-indole were all very similar and show the highest correlation to the true spectrum for this component. This is a promising feature of the VAKFISO method as it is was able to produce a very good estimate of the spectral profile for 1*H*-indole starting from a set of random, unrelated spectra as initial estimates. The spectral profiles for the VAKFISO components corresponding to benzyl bromide and 1-benzyl-1*H*-indole possessed a higher degree of rotational ambiguity. Some of the characteristic features of 1-benzyl-1*H*-indole in the region 280 to 305 nm were also present in the estimated spectral profiles of benzyl bromide. The maxima of the estimated profiles for benzyl bromide and 1-benzyl-1*H*-indole both occured at approximately the same wavelength (228 nm), whereas the maxima in the true spectra occured at approximately 225 nm and 230 nm respectively.

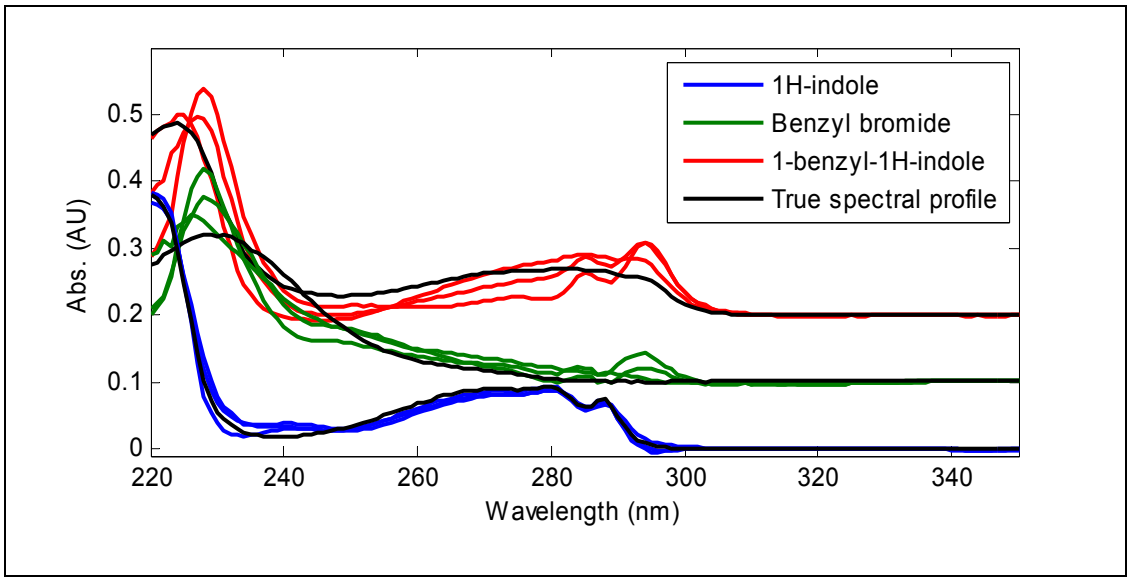


Figure 4.45: Comparison of the pure component spectral profiles estimated using VAKFISO using initial random estimates, and the true spectral profiles of 1*H*-indole, benzyl bromide and 1-benzyl-1*H*-indole. The profiles for the spectra corresponding to benzyl bromide and 1-benzyl-1H-indole were translated by +0.1 AU and +0.2 AU respectively. The estimated spectral profiles of each component were more similar between data sets than those obtained in the previous experiment, although rotational ambiguity was still evident.

The concentration profiles calculated using the final estimates of the spectral profiles are shown in Figure 4.46. An approximation of the underlying features of the data can be interpreted from these results although there was still a significant degree of ambiguity. It may be possible to refine these estimates further by applying equality constraints such as those used by the MCR-ALS method. For example, if it was known that only one species is present at the start of the reaction, the concentration profiles can be manually corrected to meet this requirement. Similar equality constraints can be applied to the spectral profiles but this requires further user knowledge of the system and the expected result. VAKFISO may therefore be used as an alternative method to deconvolute the data and provide a means to obtain these initial estimates. However, the purpose of these experiments was to investigate the ability of VAKFISO to recover useful spectral and concentration estimates using only non-negativity constraints. These particular data sets represent a significant challenge because of the lack of selectivity in both the spectral and concentration modes but are representative of the type of data that is often produced during chemical process development.

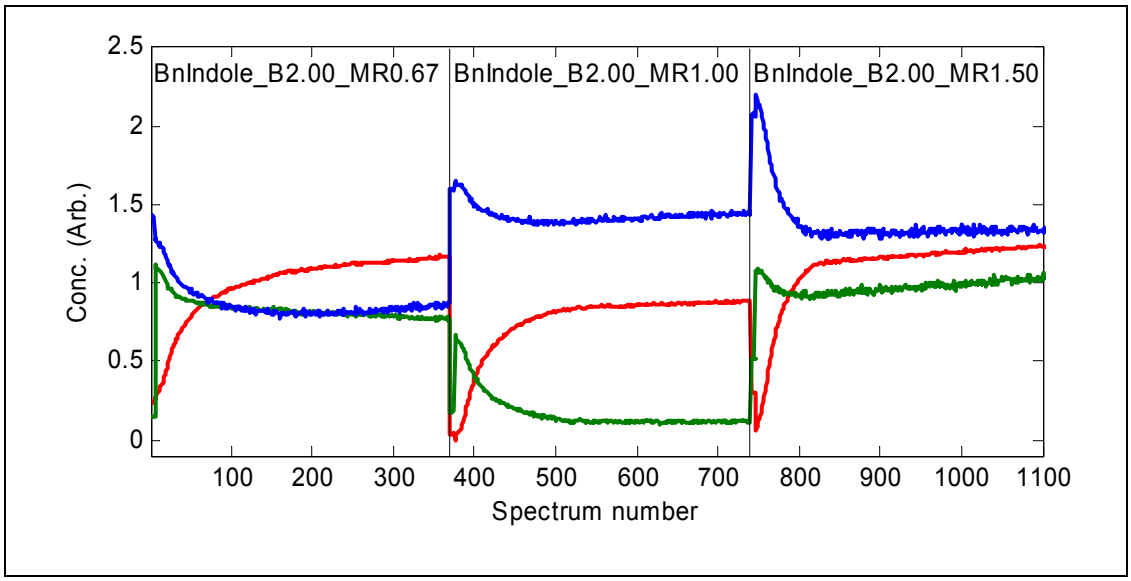


Figure 4.46: Concatenated concentration profiles calculated using the final spectral estimates produced using VAKFISO. VAKFISO was performed using randomly generated spectra. Although the profiles for each component possessed the expected characteristic features, both rotational and intensity ambiguity were evident.

4.2.13 Application of VAKFISO to real Raman spectra

VAKFISO was applied to Raman data sets using initial spectral profiles calculated from a (3×3) transformation matrix of random numbers and the (1701×3) matrix of eigenvectors ($\bar{\boldsymbol{V}}$). An example of the initial spectral estimates produced using this method is shown in Figure 4.47.

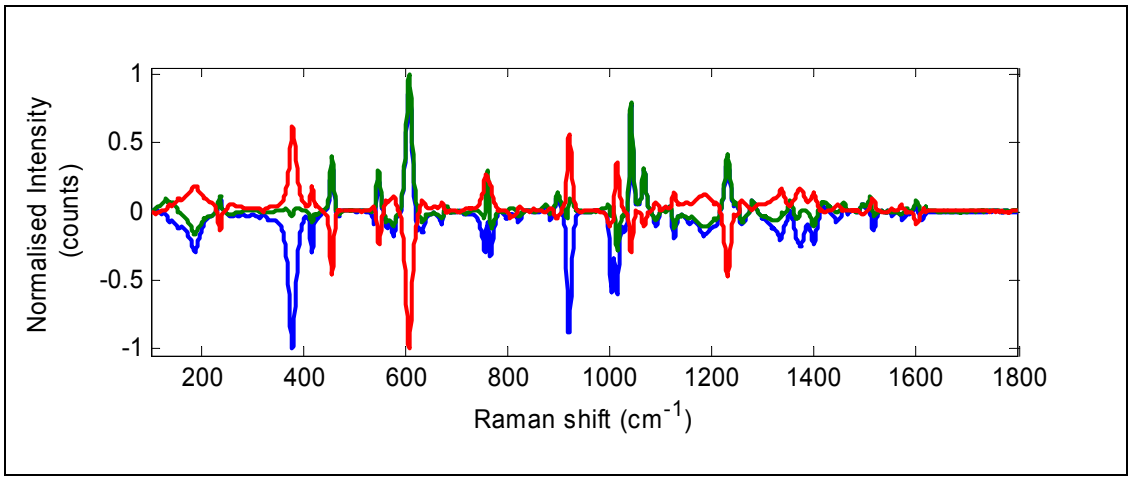


Figure 4.47: Initial spectral profiles created using a transformation matrix comprising of random numbers. The negative peaks were retained and the spectra were refined using VAKFISO.

VAKFISO was applied to each data set (BnIndole_B2.00_MR0.67, BnIndole_B2.00_MR1.00 and BnIndole_B2.00_MR1.50) using a different set of randomly generated initial spectral profiles. Although the starting spectra were different for the three experiments, VAKFISO converged to similar final estimates of the pure component spectral profiles. The estimated pure component spectral profiles are shown in Figure 4.48. It was confirmed that the peak with the largest intensity in each was the solvent band at approximately 830 cm^{-1}. This was a positive result as the initial spectra were not initially normalised to this peak.

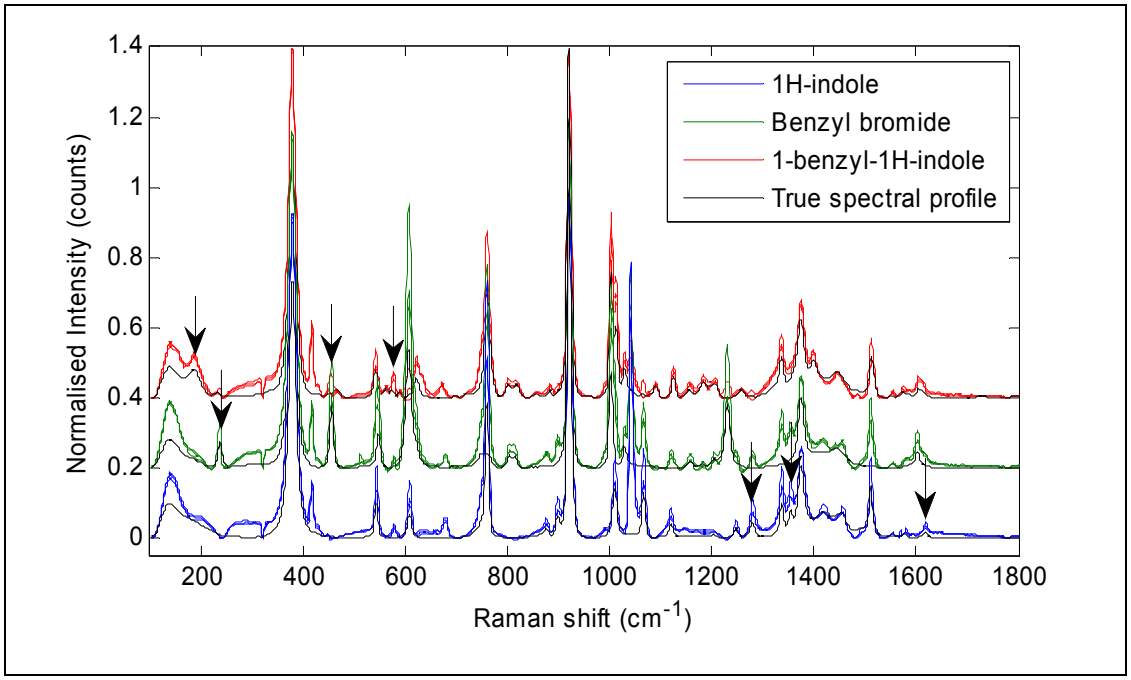


Figure 4.48: Comparison of the pure component spectral profiles estimated using VAKFISO using initial random estimates, and the true spectral profiles of 1*H*-indole, benzyl bromide and 1-benzyl-1*H*-indole. The profiles for the spectra corresponding to benzyl bromide and 1-benzyl-1H-indole were translated by +0.2 and +0.4 units respectively. The estimated spectral profiles of each component were all very similar although rotational ambiguity was evident. The selective peaks for each component are indicated by arrows. Presence of these peaks in the estimated profiles of the other components was a result of rotational ambiguity during the spectral optimisation step.

The estimated spectral profiles were compared with the expected spectra measured experimentally. The selective peaks for each component are indicated by arrows and VAKFISO had successfully recovered some of these features. For example, the distinctive peak at 450 cm^{-1} in the true spectral profile of benzyl-bromide, corresponding to a C-Br stretch was clearly visible in the estimated profile for that component, but was absent from the estimated spectrum of 1*H*-indole. There was also a peak at approximately 210 cm^{-1} in the true spectral profile of 1-benzyl-1*H*-indole, corresponding to deformation of C-C aliphatic chain of the benzyl group. This peak was present in the estimated spectral profiles of 1-benzyl-1*H*-indole but was absent from the other components. Other selective

peaks were less well resolved. For example, the band at 1280 cm^{-1} (the asymmetric C-N-C stretch) was selective for 1*H*-indole but was present in the estimated spectra of both 1*H*-indole and benzyl bromide. This rotational ambiguity can also be observed for several other minor peaks and consequently, the spectral profiles of each component were not completely recovered.

The rotational ambiguity in the spectral profiles affected the recovery of their corresponding concentration profiles. The estimated concentration profiles for the three experiments are shown in Figure 4.49. VAKFISO was applied to each data set individually so the expected differences in the profiles intensities for the three experiments observed using HPLC was not be observed in the spectral profiles. However, it was possible to see that characteristic profiles for the appearance and subsequent consumption of benzyl bromide and the formation of 1-benzyl-1H-indole were recovered successfully. Unfortunately, the profile of 1H-indole was not accurately predicted. Although not shown, other curve resolution methods such as OPA, VVSP, SIMPLISMA and EFA produce similar results, although there was often larger ambiguity between 1*H*-indole and benzyl bromide.

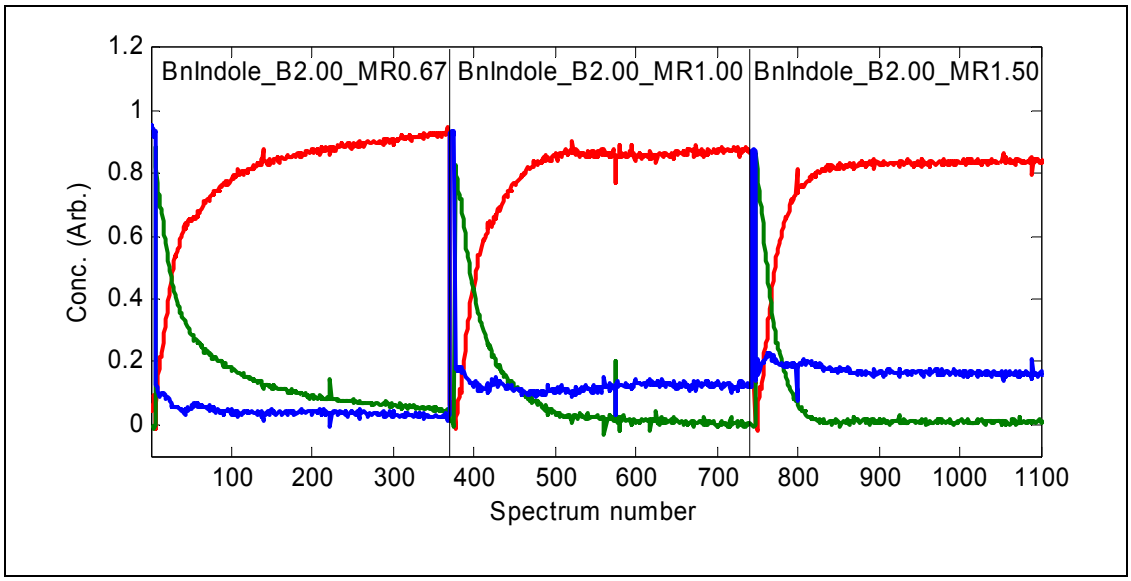


Figure 4.49: Concatenated concentration profiles calculated using the final spectral estimates produced using VAKFISO. VAKFISO was performed using randomly generated spectra. Although the profiles for benzyl bromide and 1-benzyl-1*H*-indole possessed the expected characteristic features, rotational ambiguity prevented the correct recovery of the profile for 1*H*-indole.

These experiments suggest that although VAKFISO operates in a very different manner to the other curve resolution methods described above, the results produced are often very similar. It was demonstrated that the VAKFISO method can be applied using a random

set of initial spectra to produce very similar final results. The disadvantage of this method is the time taken to perform the calculations, which can often take an hour or more depending upon the number of variables, the number of components and the number of iterations.

4.2.13.1 Conclusions: VAKFISO

The method of VAKFISO was developed to investigate whether the unique features of the Kalman filter could be employed for self-modelling curve resolution. The Kalman filter estimates the state parameters (concentration of the pure component contributing to the mixture signal) using a recursive estimation-correction approach. The objective of the Kalman filter is to minimise the state estimate covariance matrix, denoted $\boldsymbol{P}$. To perform Kalman filtering, a matrix of reference measurement functions, denoted $\boldsymbol{S}$ must be provided. When the Kalman filter is applied to spectroscopic data, the reference measurements functions represent the pure component spectral profiles of each chemical species contributing the measured signal. If the reference measurement functions are accurate, the Kalman filter will correctly estimate the state parameters. The quality of the state-parameter estimates are described by the state-estimate variance (the diagonal elements of $\boldsymbol{P}$). For an ideal system with homoscedastic noise, the matrix of innovations values (spectral prediction residuals) will resemble zero-mean white noise. If the reference measurement functions are not accurate, the estimated state parameters will also be in error. This will be characterised by larger state-estimate error variances and innovations sequences that have structured features.

The objective of VAKFISO is to find the best estimates of the pure component spectral profiles for an un-modelled system. The unique aspect of VAKFISO is that it employs the Kalman filter to test each set of feasible spectra and the resulting state-estimate variances and innovations sequences are used define the quality of the current estimates. To create each set of spectral estimates, a transformation matrix is used to calculate linear combinations of the primary eigenvectors spanning the spectral space of the original data. Newton-Gauss-Levenberg / Marquardt non-linear optimisation is used to optimise the elements of the transformation matrix. The algorithm was originally developed using simplex optimisation but this was found to be too slow and did not seem to converge to a final solution (either correct or incorrect).

The use of a transformation matrix to create estimates of the pure spectral profiles is employed in a method called Band Target Entropy Minimisation (BTEM)[46-49, 107]. This

method uses simulated annealing to find a combination of several (primary and secondary) eigenvectors that produce feasible spectra based upon the minimisation of an objective function. The objective function is constructed from penalty functions such as spectral negativity and spectral entropy to ensure that the pure spectral estimates have true features. The simulated annealing search routine is re-started several times and each minimum of the objective function corresponds to a possible spectrum. BTEM will produce a set of feasible spectra but the user must select the correct subset of spectra to calculate the corresponding concentration profiles. However, any non-negative spectrum that lies within the subspace spanned by the primary eigenvectors and has characteristic spectral features will correspond to a minimum of the objective function. The objective of VAKFISO was to combine the steps of calculating feasible spectra and testing them using the Kalman filter.

4.2.13.2 Conclusions: Application of VAKFISO to a simulated UV data set

The VAKFISO method was implemented as a Matlab function (`VAKFISO.m`) and tested using a simulated data set to determine appropriate parameters. In this case, initial estimates of the pure spectral profiles were provided using VVSP; although any method for obtaining initial estimates can be employed. The maximum number of iterations was limited to 250 cycles as preliminary tests indicated that convergence was not achieved. This was because a small change to the elements of the transformation matrix produced a large change in the resulting matrix of spectral profiles. This in turn can produced a significant change to the residual sum-of-squares used to calculate the convergence. Despite the failure to achieve the convergence criterion, it was observed visually that VAKFISO reached an approximation of the final solution after approximately 75 to 100 iterations. The algorithm produced very good estimates of the true spectral profiles although they did exhibit some rotational ambiguity. This was characterised by rotational and scaling ambiguity of the corresponding concentration profiles. Although VAKFISO successfully obeyed the non-negativity constraints invoked through the weighted penalty function, there was still sufficient freedom of rotation in the spectral subspace to prevent a unique resolution. Other established methods such as MCR-ALS minimise this problem by invoking additional constraints such as equality and closure constraints during each iterative cycle. These additional constraints require specific knowledge of the system but are a powerful way to guide the resolution of the data to a final solution that meets a specific set of criterion. It would not be possible to directly apply these types of constraints during each iterative cycle of VAKFISO as it would conflict with the NGL/M optimisation

process. However, one could perhaps consider running the VAKFISO for many iterative cycles until convergence is achieved, applying the equality constraints to the current estimate of either the concentration and / or spectral matrix and restarting the VAKFISO calculations using the corrected spectral estimates. This process could be repeated several times if necessary and could be readily automated.

4.2.13.3 Conclusions: Application of VAKFISO to real data

The VAKFISO method was applied to UV and Raman data acquired during the N-benzylation reactions used a model for the simulated data. Although the use of the adaptive Kalman filter made the method quite robust to the heteroscedastic measurement noise, the VAKFISO approach did not successfully recover the true spectral and concentration profiles. For the UV data, there was significant rotational ambiguity between the estimated profiles of benzyl bromide and 1-benzyl-1*H*-indole. However, the approach did successfully identify the regions that best characterise the differences between the true spectral profiles of each component. The concentration profiles calculated from the estimated spectra did have the recognisable features of the true concentration profiles but also exhibited significant scaling and rotational ambiguity. Similar results were obtained from the Raman data and it was concluded that VAKFISO produces very similar results to existing curve resolution methods such as VVSP and SIMPLISMA. However, unlike VVSP, OPA and other "pure" spectrum based methods which simply locate the most dissimilar spectra, the spectral profiles produced using VAKFISO are calculated as a linear combination of the spectral basis vectors. The combination of spectra that best models the data set whilst obeying the non-negativity constraints are considered the best solutions.

Initially, VAKFISO was developed to use initial spectral estimates provided by another curve resolution method such as VVSP. However, it was found that if the data was highly overlapped and lack selectivity in the concentration mode, the initial spectral profiles provided by application ofVVSP (and OPA) were very similar. VAKFISO may then converge to a solution where two or more of the estimated spectral profiles were identical. Starting from a set of random spectra often produced better results as the algorithm had an opportunity to converge to a reasonable solution and then refine it slowly.

These experiments demonstrated that for particularly challenging data sets, VAKFISO will not produce a unique solution but does produce a feasible solution that can be optimised further using additional knowledge. The spectral solutions are feasible in the sense they are already a linear combination of the basis eigenvectors, they are non-negative (if required)

and produce a set of non-negative concentration profiles for which the error covariance matrix has been minimised. VAKFISO can therefore be used as a means of gaining a valuable insight to the underlying structure of an unknown data set, and identification of the features that most distinguish the spectral profiles of each component. Other soft-modelling methods that allow a large number of constraints to be applied simultaneously (such as MCR-ALS) or even a hard-modelling approach could be used to further refine the model.

5 Conclusions

5.1 Background and project aims

In this work, approaches for developing qualitative or quantitative models for the real-time monitoring of chemical reactions using process spectroscopy were investigated. Of specific interest was the development of such models when reference values were either unavailable or only a minimal number of reference measurements could be acquired.

Throughout the literature, there are many examples describing the use of quantitative multivariate calibration models to correlate the structured variation in a set of spectral measurements to some physical or chemical property of interest, such as chemical concentration. The calibration models are then used for the prediction of the same chemical property in future batches using only the spectroscopic measurement. There are numerous algorithms available for constructing the calibration models such a multivariate linear regression (MLR), principal components regression (PCR), partial least squares or projection to latent structure (PLS), ridge regression (RR) and neural networks (NN) to name a few. Most of the reported applications describe situations where a large number of samples with good quality reference values were available. The focus of these publications is therefore the application and comparison of various spectral pre-processing and calibration methods to increase the robustness and accuracy of model, whilst reducing its complexity (the number of factors required). The objective of pre-processing methods is to minimise the amount of spectral variation contributed by sources other than the property of interest; whilst maximising the correlation between the corrected spectra and property of interest.

In a manufacturing environment where there is a high throughput of materials, a number of calibration strategies are possible. The first method is to collect samples from a large number of batches. For each sample, the values of all the properties of interest must be measured using a reliable reference method and a spectrum of each sample (or the original batch) must be acquired. Often the most difficult aspect of this approach is to ensure that the sample is representative of the entire batch or process stream. An alternative approach is to use an experimental design to create a set of synthetic mixtures that span the range of property values expected during normal production. This approach requires that each constituent of the mixture is available as an isolated material so it can be added in various amounts to create the synthetic mixture samples. If representative synthetic mixtures can

be prepared, the difficulty of sampling from a large batch is removed. Also, it is often unnecessary to perform any reference analysis, as the reference values for the properties of interest can be calculated directly from the known composition of the mixtures. With this approach, the most difficult aspect is to acquire spectra of the mixtures under normal process conditions (temperature, pressure, pH, flow-rate *etc.*).

When the aim of the application is to monitor a chemical reaction, the approaches described above are often rendered impractical by a number of complications. Many industrial reactions are heterogeneous owing to the use of insoluble materials such as inorganic salts or catalysts on a carbon or silica based support. The reaction mixtures can also have multiple phases such as dissolved gas, liquid-liquid or solid-liquid interfaces. The concentration of the reactants and products is often high (close to super-saturation) to reduce the volume of solvent required and can be performed at elevated or sub-ambient temperatures and pressures. All of these factors can make sampling very difficult. Cooling (or heating), precipitation and crystallisation, solvent loss and degassing when taking a sample can change its composition so that it is no longer the same composition as the mixture observed using an *in-situ* spectroscopic probe. The alternate approach of creating synthetic mixtures may not be possible because the reaction mechanism may proceed via a reactive intermediate species that cannot be isolated. Reactivity of the reactants may also prevent the preparation of synthetic mixtures with known composition. The time and expense required to develop a full calibration may not be justified during the early stage of development for a process because the chemistry or process conditions may change regularly.

The difficulties described in the previous paragraph are commonly encountered in a chemical R&D department so alternative approaches for constructing qualitative, semi-quantitative or fully quantitative spectroscopic methods are required. One of the most powerful tools that can be exploited for this purpose is the mathematical approach known as self-modelling curve resolution (SMCR). The objective of this research was to investigate methods for pre-processing typical spectroscopic process data and extracting the underlying structure using only the minimal amount of reference data.

5.2 Combining SMCR and PLS regression

Curve resolution is the name given to any method that can be applied to deconvolute (factorise) a matrix of bilinear data set into two smaller matrices representing the true, underlying profiles that characterise the structure in the data. For spectroscopic data acquired during a reaction monitoring experiment, the matrix of data represents the series of spectra acquired at different times throughout the course of the reaction. Curve resolution can then be applied to deconvolute the data set into its major dyads. The first matrix (often denoted $\boldsymbol{C}$) will contain the concentration profiles for each individual component that can be extracted from the data; the second matrix (often denoted $\boldsymbol{S}$) will contain the corresponding pure spectral profiles for each component in $\boldsymbol{C}$. Since the concentration profiles are derived directly from the spectral data, reference values are not required to establish a correlation between the structured variation in the reference data and the spectral data.

To assess how curve resolution could be used to develop a quantitative spectroscopic reaction monitoring method using only minimal reference data, an approach that combined SMCR and PLS regression was explored.

The reaction used to test this approach was the chlorination of 7-methoxy-4-oxo-3,4-dihydroquinazolin-6-yl acetate ('acetoxyone') using phosphorus oxychloride. This reaction was particularly problematic because it was very difficult to take samples for off-line analysis during the course of the reaction. Taking samples from the reaction mixture was precluded by a number of factors: the reaction mixture started as a heterogenous slurry owing to poor solubility of the starting material (acetoxyone); the reaction was performed at 70°C and a sample would cool rapidly, leading to precipitation of acetoxyone and haloacetoxyone (product). Furthermore, the reaction proceeds via an intermediate species for which no reference material was available so it could not be quantified using offline analysis. The only reference data that could be measured was the initial concentration of acetoxyone based upon its solubility limit at normal process conditions; and the final concentration of haloacetoxyone at the end of reaction. The reference data was provided using reversed-phase HPLC.

The objective was to construct a quantitative UV/ATR method that could be transferred to a large scale laboratory facility to monitor several 50 L scale batches. The UV data sets obtained from small scale laboratory experiments were initially examined using principal

component analysis (PCA), the orthogonal projection approach (OPA) and evolving factor analysis (EFA). The purpose of this initial examination was to determine the spectroscopic rank of the data (*i.e.* how many independent, spectroscopically active components were contributing to each mixture spectrum). The various methods all suggested that there were three components contributing to the data set, corresponding to acetoxyone (starting material), an intermediate species and haloacetoxyone (product). The initial concentration profiles obtained using EFA and the initial spectral profiles obtained using OPA were scaled using the known concentration of each species at the start and end of reaction. MCR-ALS was then applied using spectral and concentration equality constraints to refine the concentration profiles and the spectral profile of the intermediate species. MCR-ALS was found to be a powerful technique for refining the deconvolution of the spectral data set into its pure component profiles through the incorporation of additional information and the use of valid constraints to guide the process.

Once feasible concentration profiles had been derived from the spectral data, the MCR-ALS model was translated to a PLS regression model. There were several reasons why it was necessary and advantageous to translate the MCR-ALS derived model into a PLS model. The standard process monitoring software that was currently available for the Zeiss MCS501 UV/vis diode array spectrophotometer was called 'ProcessXplorer' and only supported the use of Thermo Scientific GRAMS PLSplus/IQ PLS models. Since neither ProcessXplorer nor GRAMS PLSplus/IQ currently support the use of pure spectral profiles to perform least-squares calculations, translation of the MCR-ALS model to a PLS model was required. Another advantage of translating a MCR-ALS model to a PLS model is that it facilitates the use of the standard calibration transfer corrections available within the GRAMS PLSplus/IQ software. Using a calibration transfer algorithm is necessary when transferring a model created on one instrument to another instrument such as a process analyser, as this corrects for shifts of peak position and different instrument intensity profiles. A final consideration for the use of a PLS model is that it should be more robust to additional non-structured noise present in the process spectra. This is because PLS is a subspace projection method. This means that each measured process spectrum is projected onto a set of basis vectors that span the spectral subspace containing the part of the spectrum that is correlated to the property of interest. Any additional noise components that are not spanned by the basis vectors do not contribute to the prediction. A least-squares calculation applied to the process spectrum using the pure spectral profiles will attempt to minimise the residual vector by maximising the contribution of the

additional noise in the estimation of the concentrations. It is possible to de-noise the spectrum prior to the least-squares calculation using a PCA model calculated using the calibration set. However, PLS combines these steps in one simple model.

Using the matrix of concentration profiles derived using MCR-ALS as the Y-block, and the corresponding matrix of second derivative spectra as the X-block, a series of PLS1 and PLS2 models were constructed. The performance of each model was assessed and compared using the RMSECV statistics. The number of factors required for the pure component spectral profiles calculated from PLS regression vectors to match the pure spectral profiles derived using MCR-ALS was also examined.

The RMSECV values returned by the three-factor PLS1 and PLS2 models for the prediction of haloacetoxyone (product) were 2.68×10^{-3} mol.L^{-1} and 2.78×10^{-3} mol.L^{-1} respectively. At the reaction end-point, this corresponds to a prediction error of 0.99% (PLS1) and 1.02% (PLS2). Although including additional factors would marginally reduce the RMSECV, a three factor model provided a prediction error equivalent, if not better than that expected from a HPLC assay method. A PLS2 model was chosen because it returned low prediction errors for all components within a single model. To quantify the differences between the PLS and MCR-ALS pure spectral profiles, the root-mean-square-error (RMSE) of the spectral residuals were calculated. The RMSE values for the intermediate species began to tend towards a minimum starting at three factors, and only showed marginal improvement for four or five factor models. The RMSE values for haloacetoxyone reached a minimum at three factors. The RMSE values were almost identical for PLS1 and PLS2 three-factor models. This confirmed that both methods were reproducing the MCR-ALS pure spectral profiles with the same accuracy.

The three-factor PLS2 model was then used to monitor five batches in a large scale laboratory facility (50 L) over a period of four weeks. For the first batch, the concentration profiles predicted using the PLS model were similar to the profiles observed in the laboratory. The predicted end-point concentration showed excellent agreement with both the expected theoretical value and the actual value measured using off-line HPLC analysis (0.27 mol.L^{-1}). The predictions of the product concentrations for the subsequent four batches were less accurate. Although each batch exhibited a similar profile to the first batch, the largest prediction error was attained during batch three, for which a final concentration of 0.22 mol.L^{-1} was predicted, whilst the true measured value was 0.271 mol.L^{-1} (corresponding to a prediction error of 18.8%). Despite the prediction errors, the

real-time display of the reaction profiles proved to be extremely useful during the manufacturing campaign.

When the spectral data for the five batches were examined *post hoc*, it was observed that the although the spectral profiles were comparable for all five batches, the intensity of the original absorbance spectra acquired during batches two to five were lower than those acquired during the first batch. This measurement error was the reason for the prediction error described previously.

Two experimental factors thought to have contributed to the spectroscopic measurement error were probe fouling and movement of the fibre-optic cables. A consequence of probe fouling is that an adsorbed solid or film can reduce the effective pathlength of the light through the continuous phase, and can also reduce the effective measurement area on the ATR crystal. A reduced pathlength would attenuate the measured absorbance spectra and can also cause an increase in the baseline height across the spectrum. Probe fouling can occur during the reaction and can affect the measurement of sample spectra during the experiment; or it can result from ineffective cleaning between batches and contaminate the new reference (background) spectrum.

The increased baseline height observed in the spectra could also have been a consequence of movement of the fibre-optic cables. The movement of fibre-optic cables whilst they are in use is often unavoidable but is known to cause baseline shifts. The baseline shifts arise from a change in the transmissitivity of the fibre core as the cable is moved and is caused by a change in the number of internal reflections and other optical losses. To investigate whether this effect could be minimised when using a double-beam spectrophotometer, a custom fibre-optic cable was designed. The basis of this design was to utilise the second detector channel to actively correct for the changes in transmissitivity of the fibre caused by movement. To achieve this, the fibre transmitting the signal for the bypass channel was designed to follow a path very similar to the sample signal, without interacting with the sample.

The custom fibre assembly was compared with a standard pair of fibre-optic cables in a double beam configuration by measuring the effect of displacing the cable by different amounts. This was achieved by vertically displacing the cables at their centre point to introduce increasing amounts of curvature; starting with the cables lying in a straight line along the ground to represent zero displacement. The energy spectra acquired at each

stage of displacement were used to compare the two configurations. The results obtained indicated that the custom fibre assembly reduced the effect of fibre movement by approximately 25% relative to a standard pair of fibre optic cables. Two simulated data sets, each containing the spectral variability expected from the standard and custom fibre-optic cable configurations were created. The results suggested that using a by-pass channel to actively compensate for movement of fibres will reduce baseline artefacts in the unprocessed data. Reducing the baseline artefacts will consequently reduce the prediction error in concentration profiles estimated using least-squares calculations. The RMSD values for the estimated concentration profiles improved (decreased) by approximately 69% using the custom fibre assembly. Even if spectral pre-processing is used to eliminate baseline variation, the RMSD values suggest that an improvement of approximately 31% can still be achieved using the custom fibre assembly. Although the spectral measurements can still be affected by probe fouling, the amount of additional variation introduced through fibre movement can be minimised using the custom fibre assembly.

5.3 Kalman filtering for SMCR

In the previous section, an application demonstrating the use of various SMCR methods to extract concentration profiles directly from the spectral data was presented. These concentration profiles were then used to construct a PLS regression model for the real-time prediction of future batches. This approach was necessary because of the problems experienced when trying to sample the reaction mixture to obtain reference measurements using an offline method. In this particular case, it was possible to extract the pure component concentration and spectral profiles because each component was relatively well resolved in time mode owing to the semi-batch nature of the reaction. In many cases, the pure spectral profiles of each component are highly overlapped and the corresponding concentration profiles are correlated to each other as reactant is transformed to product. A number of hard-modelling methods devised to overcome this issue have been reported in the literature.

Although hard-modelling methods can be extremely powerful and have enjoyed a lot of success, they do rely on the user to decide upon a feasible reaction mechanism to use as the basis of the optimisation. The use of these methods can also be hindered in those situations where the data obtained from process monitoring experiments that are non-ideal. The main limitation of these methods is that the optimisation process seeks to find a rate constant, k. The rate constant is applied to the data using the rate equations for the

proposed mechanism to create a set of estimated concentration profiles; in turn these profiles are used to calculate the pure spectral profiles using least-squares. If the correct reaction mechanism has been chosen, the optimal solution of k will minimise the sum-of-squares of the residual matrix. However rate constants are temperature dependent and typical industrial reactions are rarely operated using isothermal conditions. This is because temperature ramps are often necessary to aid the dissolution of reactants or to achieve the desired reaction temperature following charge of solvents, catalysts, reactants *etc.* The consequence of using a temperature ramp is that the initial concentration of reactant (denoted $[A]_0$ or $[B]_0$) will be unknown and will change with temperature. To include this behaviour in the rate equations will require additional information (such as temperature-solubility curves) and will increase the complexity of the model significantly. For this reason, soft-modelling curve resolution methods are still required to obtain an initial estimate of a model that describes the data set under investigation.

In the hard-modelling approach described above, the matrix of concentration profiles is continuously refined by solving the rate equations until the sum-of-squares of the residual matrix is minimised. During each iterative cycle, the pure component spectral profiles are calculated using least-squares to complete the model and calculate the residual sum-of-squares. An alternate approach employed by many soft modelling SMCR methods is to identify the purest spectra in a data set and then calculate the concentration profiles using least-squares. Alternating least squares is often applied to refine these estimates whilst incorporating constraints such as non-negativity, closure and equality constraints. The success of this approach requires each species to have a selective window of existence where it is the major contribution to the mixture spectrum. In many reaction systems, pure spectral profiles for one or more of the reactants can be retrieved, along with the spectrum of the product(s), providing the reaction has reached completion and the reactants have been fully consumed. It is more difficult to extract the pure spectral profiles of additional reagents or intermediates that are added or formed during the course of the reaction. If the reaction does not proceed to full conversion, the pure spectral profile of the product(s) cannot be extracted because the mixture spectra will contain significant contributions from other components.

In an attempt to address this problem, alternative approaches for modelling this type of data using the Kalman filter were considered. The *N*-benzylation of 1*H*-indole was selected as a model reaction to represent a typical process encountered in industry. This

reaction was characterised by the relatively high concentrations of reactants and products, the use of an insoluble base to yield a heterogeneous reaction mixture, and a relatively slow reaction time of six to eight hours. The rapid introduction of a final reactant to initiate reaction is a typical approach for investigating the effect of different reagent ratios upon the reaction rate as it produces a defined start point (t_0). The reference UV spectra of 1*H*-indole, benzyl bromide and 1-benzyl-1*H*-indole were acquired and used to create a synthetic data set that featured the true reaction profiles. This data set was used for algorithm development and testing. The reference spectra of the two reactants and product were heavily overlapped and had very few selective regions (pure variables).

A relatively new SMCR method called Vertex Vector Sequential Projection[95, 96] was implemented as a Matlab function (`VVSP.m`) and then applied to simulated data. This method locates the mixture spectra that most closely resemble pure spectral profiles (vertex vectors). Although the calculations used to locate each pure spectrum are quite different to those used by OPA, both methods were found to produce almost identical results for the UV data. VVSP was also used as a method for providing initial estimates of the pure spectral profiles that could be refined using VAKFISO.

A novel SMCR method called the Vectorised Adaptive Kalman Filter with Iterative Spectral Optimisation (VAKFISO) was developed as an alternative approach for estimating the pure spectral profiles from a bilinear data set. The objective of VAKFISO is to find the best estimates of the pure component spectral profiles for an un-modelled system. The unique aspect of VAKFISO is that it employs the Kalman filter to test each set of feasible spectra and the resulting state-estimate variances and innovations sequences are used to define the quality of the current estimates. Each set of spectral estimates is created using a transformation matrix to calculate linear combinations of the primary eigenvectors spanning the spectral space of the original data. Newton-Gauss-Levenberg / Marquardt non-linear optimisation is used to optimise the elements of the transformation matrix. VAKFISO shares some similarities with another spectral search method called Band Target Entropy Minimisation (BTEM)[47, 48, 107]. BTEM will produce a set of feasible spectra, but the user must then select a subset of spectra to calculate the corresponding concentration profiles. The objective of VAKFISO was to combine the steps of calculating feasible spectra and testing them using a vectorised adaptive Kalman filter.

The VAKFISO method was tested using a simulated UV data set to determine appropriate weighting parameters. The algorithm produced very good estimates of the true spectral

profiles although they did exhibit some rotational ambiguity. This then produced rotational and scaling ambiguity in the corresponding concentration profiles. The most challenging component to recover was 'component B' because it its spectrum was completely overlapped with the other components. However, the estimated spectral profile for component B produced using VAKFISO was much closer to the true spectrum than that produced by other methods such as VVSP/MCR-ALS, OPA/MCR-ALS, ITTFA or SIMPLISMA. This result demonstrated that the VAKFISO was a viable alternative to existing SMCR methods.

The VAKFISO method was also applied to UV and Raman data acquired during the *N*-benzylation reactions. This data resembled true process data and was particularly challenging. Unfortunately the VAKFISO approach did not successfully recover the true spectral and concentration profiles. For the UV data, there was significant rotational ambiguity between the estimated profiles of benzyl-bromide and 1-benzyl-1*H*-indole. Similar results were obtained from the Raman data and it was concluded that VAKFISO produces very similar results to existing curve resolution methods. However, unlike VVSP, OPA, SIMPLISMA and other "pure" spectrum based methods that simply locate the most dissimilar spectra or variables, the spectral profiles produced using VAKFISO are calculated as a linear combination of the spectral basis vectors. The combination of spectra that best models the data set whilst obeying the non-negativity constraints are considered the best solutions. It is also possible to start VAKFISO from a set of random spectra and this often produced better results as the algorithm had an opportunity to converge to a reasonable solution and then refine it slowly.

These experiments demonstrated that for particularly challenging data sets, VAKFISO will not produce a unique solution but does produce a feasible solution that could be optimised further using additional knowledge. The spectral solutions are feasible in the sense they are already a linear combination of the basis eigenvectors, they are non-negative (if required) and produce a set of non-negative concentration profiles for which the error covariance matrix has been minimised. VAKFISO can therefore be used as a means of gaining a valuable insight to the underlying structure of an unknown data set, and identification of the features that most distinguish the spectral profiles of each component. Other methods that allow a large number of constraints to be applied simultaneously (such as MCR-ALS) or even a hard-modelling approach could be used to further refine the model.

To pre-process the UV spectra acquired during these experiments, median filtering was found to be very useful. A median filter was implemented as a Matlab function (`MedianFilter.m`) to remove unwanted 'spikes' from UV spectra acquired using a Cary 50 UV spectrophotometer. Using a median filter with a window width of three points successfully removed these artefacts without changing any of the useful spectral features. The median filter was also applied to Raman spectra acquired during the same experiments. The Raman spectra featured a broad baseline feature that would contribute additional, unwanted variation. The median filter was applied to a Raman spectrum (3400 variables) using window widths of 101, 201, 301, 401 and 501 points. A window width of 501 points was found to offer the best approximation of the underlying baseline, but after subtraction of the baseline the corrected spectrum had several negative regions where the baseline was overestimated. However to obtain a first approximation of the underlying baseline, the median filter proved to be a useful approach.

A custom algorithm called iterative polynomial baseline subtraction (`IPBS.m`) was written to obtain a more accurate estimation of the underlying baseline contribution to the Raman spectra. This method uses a median filter to obtain an initial estimate of the baseline. A fixed size moving window then moves through the data and a polynomial of degree n is fitted to the data. The polynomial coefficients for each window are then iteratively optimised to prevent the subtraction of any high frequency Raman features. The iterative polynomial fitting is based upon the method reported by Lieber and Mahadevan-Jansen[97]. The method described in this thesis was extended to allow a full Raman spectrum to be corrected by splitting the spectrum into a number of windows defined by the window width. An additional feature of the modified iterative polynomial baseline subtraction method is that it allows the windows to overlap using an overlap parameter (typically 10 to 20% of the window width). The discontinuities observed if the polynomial fitting was applied to each window independently were removed by incorporating an overlap between windows.

Although computationally intensive (30 to 60 seconds per spectrum), the results were very good and the complex baseline contribution of each spectrum was accurately modelled. The best parameters for correcting this particular set of Raman spectra were a median filter width of 501 points; a third degree polynomial (n=3); an overlap factor of 0.20 and a window width of 157 points. The baseline corrected spectra have a completely flat baseline, do not feature any negative regions yet retain all of the useful Raman features.

This was a significant improvement over the Pearson[105] correction function available in the instruments HoloReact software.

5.4 Overall conclusions

The goal of this research was to investigate the use and development of chemometric methods that could be applied to facilitate the recovery and prediction of component concentrations using real process spectra. An important component of this research was the investigation and development of self modelling curve resolution methods that would allow quantitative or semi-quantitative models to be developed in the absence of external reference data.

A reaction that exhibited many of the difficulties encountered when trying to model an industrial process was the chlorination of acetoxyone using phoshorus oxychloride. This reaction was used to investigate an approach that combined SMCR with PLS regression. In the absence of reliable reference data, SMCR methods such as EFA and OPA were used to extract the underlying concentration profiles of the major components of interest. The concentration profiles were then used to construct a PLS model that could be implemented in real-time to predict future batches. The pure component spectra calculated from the PLS regression coefficients were compared with the SMCR spectra to ensure that the correct number of principle components were used. This work was published in *Applied Spectroscopy* (2007, volume 61, number 9, pp 940-949).

Improving the robustness of the spectral measurements is also an important element for the reliable application of process spectroscopy. During the application of the UV/ATR PLS method used to determine the end-point of the chlorination reaction in a large scale laboratory facililty (chlorination of acetoxyone using phosphorus oxychloride), the fibre-optic cables were subjected to significant movement during the process. To investigate how fibre-movement affected spectral measurements, a series of experiments were performed. A custom fibre-optic cable assembly that used the spectrophotometers by-pass channel to compensate for fibre movement was designed and compared with standard fibre-optic cables. The experiments demonstrated that although the custom fibre-optic assembly did not completely eliminate the effects of fibre movement, it did offer a significant improvement relative to standard fibres.

Process spectra often contain contributions from additional physical or chemical factors caused by the process or the measurement technique itself. The additional contributions can hinder the recovery of the desired chemical information and need to be removed prior as part of the data analysis step. Both the UV and Raman spectra acquired during the N-benzylation of *1H*-indole reactions contained additional spectral contributions that required specific methods to be implemented. A moving window median filter was implemented as a Matlab script remove the high intensity spikes present in the UV data. The moving window median filter could also be used to remove subtle baseline contributions that are often present in FTIR and Raman spectra. However the median filter was not sufficient to remove significant, complex baselines such as those observed in the Raman spectra acquired during the N-benzylation of *1H*-indole reactions described in chapters 3.3 and 4.2. The original iterative polynomial baseline subtraction method (IPBS) was implemented as Matlab script and trialled, but this was also unsuccessful when applied to a full Raman spectrum. The original IPBS method was modified to utilise a moving window median filter to provide an initial estimate of the underlying baseline. The modified IPBS method also applies the iterative polynomial fitting to a number of windows across the spectrum, using a certain degree of overlap between the windows to ensure that the baseline function is continuous. This modified IPBS algorithm was successfully applied to the Raman data described above and removed the complex baseline function without perturbing the spectral information of interest. The modified IPBS algorithm also offered significant improvement compared to the Pearson correction function available in the Raman spectrometers control and acquisition software.

The use of adaptive Kalman filter for SMCR was investigated. The linear and adaptive Kalman filters were implemented as vectorised algorithms in Matlab. The vectorisation offered significant computational efficiency that allowed the Kalman filter to be applied to large data sets. A novel SMCR method called vectorised adaptive Kalman filtering with iterative spectral optimisation (VAKFISO) was developed. This method was applied to both simulated and real data obtained from UV and Raman spectroscopy acquired during the *N*-benzylation of indole reactions. This reaction data was specifically used because it was particularly challenging. The spectral profiles of each major component were highly overlapped and the reaction profiles were also linearly dependent. When applied to the simulated UV reaction data, VAKFISO successfully recovered the spectral profiles with very little rotational ambiguity. Indeed, VAKFISO seemed to offer an advantage over other established SMCR methods such as OPA-ALS, VVSP-ALS and EFA-MCR-ALS.

When applied to real reaction spectra, VAKFISO was less successful, although other established SMCR methods also failed to resolve the same data sets. For these particular data sets, VAKFISO performed better when initiated using random spectra than it did when initiated using spectral profiles derived by application of VVSP. VAKFISO therefore shows promise as an alternative SMCR method and there are several potential areas for further development.

5.5 Further work

The reactions investigated during this research were performed in jacketed glass reaction vessels that provided excellent temperature control, although some small variation in the reaction temperature was still observed. Larger reaction vessels such as those used on pilot- or full-scale manufacturing plants would exhibit larger temperature variations as the temperature controllers and heating components respond much slower than laboratory scale reaction vessels.

Temperature variation can introduce small, non-linear perturbations into the measured spectra such as small peak shifts, and can also cause subtle changes to the peak intensity. When employing a typical calibration strategy, it is possible to incorporate temperature variation by recording the spectra of synthetic mixtures at a number of different temperatures, or by running the reaction at several different temperatures. The temperature can then be implicitly or explicitly modelled by the calibration model. A more robust approach would be to correct the spectra to remove the temperature dependent contributions prior to construction or application of a calibration model. Several methods that model and remove the effects of temperature from a spectrum have been reported. Amongst the most promising are Loading Space Standardisation (LSS)[108], Optical Pathlength Error Correction (OPLEC)[109] and Extended Loading Space Standardisation (ELSS)[110]. In the work reported in this thesis, the concentration profiles were not obtained by off-line analysis of reaction samples, but were derived directly from the spectral data by application of self-modelling curve resolution methods. To apply this approach to a calibration that is also required to span a range of temperatures, it would be necessary to acquire several data sets by performing the reaction at different temperatures. As each data set would be acquired at a different temperature, it would not be appropriate to concatenate the data sets prior to the application of SMCR methods because the pure spectral profiles in each data set would be slightly different owing to the temperature variation. However, by correcting the spectra of each data set to remove the effects of

temperature, they can be concatenated whilst retaining the true chemical rank. The advantages of performing SMCR to multiple data sets (to overcome scaling ambiguity) can then be exploited. The feasibility of utilising LSS for this purpose was investigated by the author and shows great promise.

During the *N*-benzylation of *1H*-indole reactions, both UV/ATR and Raman spectroscopy were used to acquire spectra simultaneously. VVSP and VAKFISO were applied to each set of data individually, or to multiple data sets concatenated vertically. It was demonstrated that if the true pure spectral profiles of each chemical species contributing to the system were known, and an accurate estimation of the measurement noise variance was also provided, the Kalman filter could accurately estimate the underlying concentration profiles, even if the data is rank deficient (*e.g.* a linearly dependent reaction). An area of research that could further be exploited to obtain such models is the analysis of two different spectral data sets acquired simultaneously, for example FTIR and UV. One method that could be employed to analyse the structure of two data sets is O2-PLS[111, 112]. This method could be employed to filter each data set to remove the specific structured variation inherent to each type of spectroscopy (for example, fluorescence in Raman spectroscopy), whilst retaining the underlying chemical variation common to both data sets (the desired profiles). The regression spectra obtained from O2-PLS could also be used to calculate initial pure component spectral profiles for use by VAKFISO or other SMCR methods to refine the model of each data set individually. Other two-block methods such as Canonical Correlation Analysis[113] and Common Components Specific Weights Analysis[114-118] were also investigated by the author and show considerable promise. However, directly integrating estimates obtained from the multi-block methods described above with SMCR methods such as VAKFISO or MCR-ALS was not investigated and could offer great potential for reaction modelling.

There are also opportunities for further development of the VAKFISO algorithm. With particularly challenging data sets, the VAKFISO method would occasionally converge to a solution that produced almost identical pure spectral profiles for two of the components. This did not occur when the algorithm was initiated using random spectra, but there may be situations when it would be desirable to utilise initial estimates of the pure spectral profiles obtained by measurement of standards or by application of other SMCR methods. To prevent convergence to identical spectra, a spectral dissimilarity criterion could be included as a penalty function when calculating the weighted residual matrix.

6 References

[1] *Guidance for industry: PAT - A framework for innovative pharmaceutical manufacturing and quality assurance.* US Food and Drug Administration, Rockville, MD, USA, **2004**. http://www.fda.gov/cder/guidance/6419fnl.pdf (Last viewed 8th October 2008)

[2] *Pharmaceutical Development Q8*. International Conference on Harmonisation of Technical Requirements for Registration of Pharmaceuticals for Human Use, Geneva, Switzerland, **2005**. http://www.ich.org/LOB/media/MEDIA1707.pdf (Last viewed 8th October 2008)

[3] *Pharmaceutical Development Annex to Q8*. International Conference on Harmonisation of Technical Requirements for Registration of Pharmaceuticals for Human Use, Geneva, Switzerland, **2007**. http://www.ich.org/LOB/media/MEDIA4349.pdf (Last viewed 8th October 2008)

[4] K. A. Bakeev, *Process Analytical Technology*, Blackwell Publishing Ltd., Oxford, UK, **2005**.

[5] D. E. Booth, T. L. Isenhour, J. K. M. Jr, M. Suh and C. Wright in *Quality Control and Data Analysis*, *Vol. 1* (J. Gasteiger), Wiley -VCH, Weinheim, Germany, **2003**,

[6] T. Kourti, Multivariate dynamic data modeling for analysis and statistical process control of batch processes, start-ups and grade transitions, *Journal of Chemometrics,* 17 **(2003)**, 93-109.

[7] T. Kourti and J. F. MacGregor, Process analysis, monitoring and diagnosis, using multivariate projection methods, *Chemometrics and Intelligent Laboratory Systems,* 28 **(1995)**, 3-21.

[8] C. Duchesne and J. F. MacGregor, Multivariate analysis and optimization of process variable trajectories for batch processes, *Chemometrics and Intelligent Laboratory Systems,* 51 **(2000)**, 125-137.

[9] J. A. Westerhuis, T. Kourti and J. F. MacGregor, Comparing alternative approaches for multivariate statistical analysis of batch process data, *Journal of Chemometrics,* 13 **(1999)**, 397-413.

[10] S. J. Qin, Statistical process monitoring: basics and beyond, *Journal of Chemometrics,* 17 **(2003)**, 480-502.

[11] S. P. Gurden, J. A. Westerhuis and A. K. Smilde, Monitoring of Batch Processes Using Spectroscopy, *American Institute of Chemical Engineers Journal,* 48 (10) **(2002)**, 2283-2297.

[12] J. M. Chalmers, *Spectroscopy in Process Analysis*, Sheffield Academic Press, Ltd, Sheffield, UK, **2000**.

[13] R. J. Ampiah-Bonney and A. D. Walmsley, Monitoring of the acid catalysed esterification of ethanol by acetic acid using Raman spectroscopy, *Analyst,* 124 **(1999)**, 1817-1821.

[14] K. D. Braekeleer, A. D. Juan, F. C. Sánchez, P. A. Hailey, D. C. Sharp, P. Dunn and D. L. Massart, Determination of the End Point of a Chemical Synthesis Process Using On-Line Measured Mid-Infrared Spectra, *Applied Spectroscopy,* 54 (4) **(2000)**, 601-607.

[15] E. N. M. v. Sprang, H.-J. Ramaker, H. F. M. Boelens, J. A. Westerhuis, D. Whiteman, D. Baines and I. Weaver, Batch process monitoring using on-line MIR spectroscopy, *Analyst,* 128 **(2003)**, 98-102.

[16] B. Ma, P. J. Gemperline, E. Cash, M. Bosserman and E. Comas, Characterizing batch reactions with *in-situ* spectroscopic measurements, calorimetry and dynamic modeling, *Journal of Chemometrics,* 17 **(2003)**, 470-479.

[17] P. Geladi and J. Forsström, Monitoring of a batch organic synthesis by near-infrared spectroscopy: modeling and interpretation of three-way data, *Journal of Chemometrics,* 16 **(2002)**, 329-338.

[18] M. Garrido, M. S. Larrechi and F. X. Rius, Near infrared spectroscopy and multivariate curve resolution-alternating least squares incorporating ^{13}C-NMR information for monitoring epoxy resins reactions, *Journal of Chemometrics,* 21 **(2007)**, 263-269.

[19] G. D. Christian, *Analytical Chemistry*, John Wiley & Sons, Inc., Toronto, Canada, **1994**.

[20] R. Kramer, *Chemometric techniques for quantitative analysis*, Marcel Dekker, Inc., New York, NY, USA, **1998**.

[21] R. G. Brereton, *Chemometrics. Data Analysis for the Laboratory and Chemical Plant*, John Wiley & Sons, Ltd, Chichester, UK, **2003**.

[22] P. Gemperline, *Practical guide to chemometrics; Second edition*, CRC Press, Boca Raton, FL, USA, **2006**.

[23] T. Næs, T. Isaksson, T. Fearn and T. Davies, *A user-friendly guide to Multivariate Calibration and Classification*, NIR Publications, Chichester, UK, **2002**.

[24] A. Smilde, R. Bro and P. Geladi, *Multi-way Analysis with applications in the chemical sciences*, John Wiley & Sons, Ltd, Chichester, UK, **2004**.

[25] B. G. M. Vandeginste, D. L. Massart, L. M. C. Buydens, S. D. Jong, P. J. Lewi and J. Smeyers-Verbeke, *Handbook of Chemometrics and Qualimetrics: Part B*, Elsevier, Amsterdam, NL, **1998**.

[26] M. Kubista, J. Nygren, A. Elbergali and R. Sjöback, Making Reference Samples Redundant, *Critical Reviews in Analytical Chemistry,* 29 (1) **(1999)**, 1-28.

[27] R. M. Dyson, M. Hazenkamp, K. Kaufmann, M. Maeder, M. Studer and A. Zilian, Modern tools for reaction monitoring; hard and soft modelling of 'non-ideal', on-line acquired spectra, *Journal of Chemometrics,* 14 **(2000)**, 737-750.

[28] P. Gemperline, G. Puxty, M. Maeder, D. Walker, F. Tarczynski and M. Bosserman, Calibration-Free Estimates of Batch Process Yields and Detection of Process Upsets Using in Situ Spectroscopic Measurements and Nonisothermal Kinetic Models: 4-(dimethylamino)pyridine-Catalysed Esterification of Butanol, *Analytical Chemistry,* 76 (9) **(2004)**, 2575-2582.

[29] J.-H. Jiang, Y. Liang and Y. Ozaki, Principles and methodologies in self-modeling curve resolution, *Chemometrics and Intelligent Laboratory Systems,* 71 **(2004)**, 1-12.

[30] W. J. Krzanowski, *Principles of Multivariate Analysis; A Users Perspective*, Oxford University Press, Oxford, UK, **2000**.

[31] E. R. Malinowski, *Factor Analysis in Chemistry*, John Wiley & Sons, Inc., New York, NY, USA, **2002**.

[32] R. G. Brereton, Introduction to multivariate calibration in analytical chemistry, *Analyst,* 125 **(2000)**, 2125-2154.

[33] P. Geladi, Chemometrics in spectroscopy. Part 1. Classical chemometrics, *Spectrochimica Acta Part B,* 58 **(2003)**, 767-782.

[34] J. R. Schott, *Matrix Analysis for Statistics*, John Wiley & Sons, Inc., New York, NY, USA, **1997**.

[35] A. d. Juan and R. Tauler, Chemometrics applied to unravel multicomponent processes and mixtures. Revisiting latest trends in multivariate resolution., *Analytica Chimica Acta,* 500 **(2003)**, 195-210.

[36] M. Garrido, F. X. Rius and M. S. Larrechi, Multivariate curve resolution-alternating least squares (MCR-ALS) applied to spectroscopic data from monitoring chemical reaction processes, *Analytical and Bioanalytical chemistry,* 390 **(2008)**, 2059-2066.

[37] S. E. Richards and A. D. Walmsley, Quantitative iterative target transformation factor analysis, *Journal of Chemometrics,* 22 **(2007)**, 63-80.

[38] S. Richards, M. Ropic, D. Blackmond and A. Walmsley, Quantitative determination of the catalysed asymmetric transfer hydrogenation of 1-methyl-6,7-dimethoxy-3,4-

dihydroisoquinoline using in-situ FTIR and multivariate curve resolution., *Analytica Chimica Acta,* 519 **(2004)**, 1-9.

[39] M. Amrhein, B. Srinivasan, D. Bonvin and M. M. Schumacher, On the rank deficiency and rank augmentation of the spectral measurement matrix, *Chemometrics and Intelligent Laboratory Systems,* 33 **(1996)**, 17-33.

[40] C. Ruckebusch, A. D. Juan, L. Duponchel and J. P. Huvenne, Matrix augmentation for breaking rank-deficiency: A case study, *Chemometrics and Intelligent Laboratory Systems,* 80 **(2006)**, 209-214.

[41] M. Garrido, M. S. Larrechi, F. X. Rius and R. Tauler, Calculation of band boundaries of feasible solutions obtained by Multivariate Curve Resolution - Alternating Least Squares of multiple runs of a reaction monitored by NIR spectroscopy, *Chemometrics and Intelligent Laboratory Systems,* 76 **(2005)**, 111-120.

[42] G. Puxty, M. Maeder and K. Hungerbühler, Tutorial on the fitting of kinetics models to multivariate spectroscopic measurements with non-linear-squares regression, *Chemometrics and Intelligent Laboratory Systems,* 81 **(2006)**, 149-164.

[43] J. Jaumot, P. J. Gemperline and A. Stang, Non-negativity constraints for elimination of multiple solutions in fitting multivariate kinetic models to spectroscopic data, *Journal of Chemometrics,* 19 **(2005)**, 97-106.

[44] E. Furusjö and L.-G. Danielsson, A method for the determination of reaction mechanisms and rate constants from two-way spectroscopic data, *Analytica Chimica Acta,* 373 **(1998)**, 83-94.

[45] A. d. Juan, M. Maeder, M. Martínez and R. Tauler, Combining hard- and soft-modelling to solve kinetic problems, *Chemometrics and Intelligent Laboratory Systems,* 54 **(2000)**, 123-141.

[46] E. Widjaja and M. Garland, Pure Component Spectral Reconstruction from Mixture Data Using SVD, Global Entropy Minimization and Simulated Annealing. Numerical Investigations of Admissible Objective Functions Using a Synthetic 7-Species Data Set, *Journal of Computational Chemistry,* 23 (9) **(2002)**, 911-919.

[47] W. Chew, E. Widjaja and M. Garland, Band Target Entropy Minimization (BTEM): An Advanced Method for Recovering Unknown Pure Component Spectra. Application to the FTIR Spectra of Unstable Organometallic Mixtures, *Organometallics,* 21 **(2002)**, 1982-1990.

[48] E. Widjaja, C. Li and M. Garland, Semi-Batch Homogeneous Catalytic In-Situ Spectroscopic Data. FTIR Spectral Reconstructions Using Band-Target Entropy

Minimization (BTEM) without Spectral Preconditioning, *Organometallics,* 21 **(2002)**, 1991-1997.

[49] H. Zhang, W. Chew and M. Garland, The Multi-Reconstruction Entropy Minimization Method: Unsupervised Spectral Reconstruction of Pure Components from Mixture Spectra, Without the use of *a Priori* Information, *Applied Spectroscopy,* 61 (12) **(2007)**.

[50] E. Widjaja, Y. Y. Tan and M. Garland, Application of Band-Target Entropy Minimization to On-Line Raman Monitoring of an Organic Synthesis. An Example of New Technology for Process Analytical Technology, *Organic Process Research & Development,* 11 (1) **(2007)**, 98-103.

[51] M. S. Grewal and A. P. Andrews, *Kalman filtering: theory and practice using MATLAB*, John Wiley & Sons, Inc., New York, NY, USA, **2001**.

[52] R. E. Kalman, A New Approach to Linear Filtering and Prediction Problems, *Transactions of the ASME–Journal of Basic Engineering,* 82 (Series D) **(1960)**, 35-45.

[53] P. F. Seelig and H. N. Blount, Kalman Filter Applied to Anodic Stripping Voltammetry: Theory, *Analytical Chemistry,* 48 (2) **(1976)**, 252-258.

[54] P. F. Seelig and H. N. Blount, Application of Recursive Estimation to the Real Time Analysis of Trace Metal Analytses by Linear Sweep, Pulse, and Differential Pulse Anodic Stripping Voltammetry, *Analytical Chemistry,* 51 (8) **(1979)**, 1129-1134.

[55] P. F. Seelig and H. N. Blount, Experimental evaluation of Recursive Estimation Applied to Linear Sweep Anodic Stripping Voltammetry for Real-time Analysis, *Analytical Chemistry,* 51 (3) **(1979)**, 327-337.

[56] S. D. Brown, The Kalman filter in analytical chemistry., *Analytica Chimica Acta,* 181 **(1986)**, 1-26.

[57] S. C. Rutan, Kalman filtering approaches for solving problems in analytical chemistry., *Journal of Chemometrics,* 1 **(1987)**, 7-18.

[58] S. C. Rutan, Recursive parameter estimation, *Journal of Chemometrics,* 4 **(1990)**, 103-121.

[59] R. L. Tranter, The Kalman Filter: An Alternative Approach to Quantitative Analysis, *Analytical Proceedings,* 27 **(1990)**, 134-138.

[60] L. Shi, Z. Li, Z. Xu, Z. Pan and L. Wang, Simultaneous analysis of Co(II), Ni(II), Cu(II), Zn(II) and Cd(II) by spectrophotometry and the Kalman filter, *Journal of Chemometrics,* 5 **(1991)**, 193-199.

[61] K. Volka, M. Suchánek and P. Urban, The Kalman filter in quantitative infrared spectral analysis, *Vibrational Spectroscopy,* 3 **(1992)**, 155-160.

[62] N. Yongnian, M. Selby, S. Kokot and M. Hodgkinson, Curve Resolution and Quantification of Pyrazines by Differentil-pulse Polarography Using a Kalman Filter Approach, *Analyst,* 118 **(1993)**, 1049-1053.

[63] Y. Ni, Y. Wu and S. Kokot, Improved ICP-OES analysis of trace calcium in rare-earth matrices with the use of iterative transformation factor analysis and Kalman filter, *Journal of Analytical Atomic Spectrometry,* 17 **(2002)**, 596-602.

[64] L. V. Pérez-Arribas, F. Navarro-Villoslada, M. E. León-Gonzalez and L. M. Polo-Díez, Use of the Kalman filter for multivariate calibration in a real system and its comparison with CLS and pure component calibration methods, *Journal of Chemometrics,* 7 **(1993)**, 267-275.

[65] D. L. Alspach, A Parallel Processing Solution To The Adaptive Kalman Filtering Problem With Vector Measurements, *Computing and Electrical Engineering,* 1 **(1973)**, 83-94.

[66] S. C. Rutan and S. D. Brown, Adaptive Kalman filtering used to compensate for model errors in multicomponent methods, *Analytica Chimica Acta,* 160 **(1984)**, 99-119.

[67] D. D. Gerow and S. C. Rutan, Background subtraction for fluorescence detection in thin-layer chromatography with derivative spectrometry and the adaptive Kalman filter., *Analytica Chimica Acta,* 184 **(1986)**, 53-64.

[68] D. D. Gerow and S. C. Rutan, Background Correction for Fluorescence Detection in Thin-Layer Chromatography Using Factor Analysis and the Adaptive Kalman Filter., *Analytical Chemistry,* 60 **(1988)**, 847-852.

[69] S. C. Rutan and S. D. Brown, Simplex optimization of the adaptive Kalman filter., *Analytica Chimica Acta,* 167 **(1985)**, 39-50.

[70] D. F. Sittig and K.-H. Cheung, A parallel implementation of a multi-state Kalman filtering algorithm to detect ECG arrhythmias, *International Journal of Clinical Monitoring and Computing,* 9 **(1992)**, 13-22.

[71] P. D. Wentzell and S. J. Vanslyke, Parallel Kalman filter networks for kinetic methods of analysis, *Analytica Chimica Acta,* 257 **(1992)**, 173-181.

[72] S. J. Vanslyke and P. D. Wentzell, Real-time Principal Component Analysis Using Parallel Kalman Filter Networks for Peak Purity Analysis, *Analytical Chemistry,* 63 **(1991)**, 2512-2519.

[73] P. D. Wentzell, S. G. Hughs and S. J. Vanslyke, Parallel Kalman filters for peak purity analysis: extensions to non-ideal detector response, *Analytica Chimica Acta,* 307 **(1995)**, 459-470.

[74] S. C. Rutan and S. D. Brown, Two-dimensional and three-dimensional fitting of enzyme kinetic data with the Kalman filter, *Analytica Chimica Acta,* 175 **(1985)**, 219-229.

[75] B. M. Quencer and S. R. Crouch, Extended Kalman Filter for Multiwavelength, Multicomponent Kinetic Determinations, *Analyst,* 118 **(1993)**, 695-701.

[76] M. B. Smith and J. March, *March's Advanced Organic Chemistry: Reactions, Mechanisms and Structure*, Jon Wiley & Sons, Inc., Hoboken, NJ, USA, **2007**.

[77] H. Heaney and S. V. Ley, *N*-Alkylation of Indole and Pyrroles in Dimethyl Sulphoxide, *Journal of Chemical Society: Perkin Transactions 1,* (5) **(1973)**, 499-500.

[78] Y. Kikugawa and Y. Miyake, A Simple Synthesis of *N*-Alkylindoles, *Synthesis,* **(1981)**, 461-462.

[79] A. Barco, S. Benetti and G. P. Pollini, The Use of Phase-Transfer Catalysis for the *N*-Alkylation of Indole, *Synthesis,* **(1976)**, 124-125.

[80] V. Bocchi, G. Casnati, A. Dossena and F. Villani, Synthesis of *N*-alkylindoles using Tetraalkylammonium Salt Catalysis, *Synthesis,* (6) **(1976)**, 414-416.

[81] J. M. Hollas, *Modern Spectroscopy*, John Wiley &Sons, Ltd, Chichester, UK, **2004**.

[82] D. A. Skoog and J. J. Leary, *Principles of Instrumental Analysis*, Harcourt Brace & Company, Orlando, FL, USA, **1992**.

[83] E. Smith and G. Dent, *Modern Raman Spectroscopy: A Practical Approach*, John Wiley & Sons Ltd, Chichester, UK, **2005**.

[84] M. J. Pelletier, *Analytical Applications of Raman Spectroscopy*, Blackwell Science Ltd, Oxford, UK, **1999**.

[85] M. Maeder, Evolving Factor Analysis Analysis for the Resolution of Overlapping Chromatographic Peaks, *Analytical Chemistry,* 59 (3) **(1987)**, 527-530.

[86] H. Gampp, M. Maeder, C. J. Meyer and A. D. Zuberbühler, Calculation of equilibrium constants from multiwavelength spectroscopic data—III : Model-free analysis of spectrophotometric and ESR titrations, *Talanta,* 32 (12) **(1985)**, 1133-1139.

[87] M. Maeder and A. D. Zuberbuehler, The resolution of overlapping chromatographic peaks by evolving factor analysis, *Analytica Chimica Acta,* 181 **(1986)**, 287-291.

[88] M. Maeder and A. Zilian, Evolving Factor Analysis, a New Multivariate Technique in Chromatography, *Chemometrics and Intelligent Laboratory Systems,* 3 **(1988)**, 205-213.

[89] F. C. Sánchez, J. Toft, B. v. d. Bogaert and D. L. Massart, Orthogonal Projection Approach Applied to Peak Purity Assessment, *Analytical Chemistry,* 68 **(1996)**, 79-85.

[90] R. Tauler, B. Kowalski and S. Fleming, Multivariate Curve Resolution Applied to Spectral Data from Multiple Runs of an Industrial Process, *Analytical Chemistry,* 65 **(1993)**, 2040-2047.

[91] R. Tauler, A. Smilde and B. Kowalski, Selectivity, local rank, three-way data analysis and ambiguity in multivariate curve resolution, *Journal of Chemometrics,* 9 **(1995)**, 31-58.

[92] R. Tauler, Calculation of maximum and minimum band boundries of feasible solutions for species profiles obtained by multivariate curve resolution, *Journal of Chemometrics,* 15 **(2001)**, 627-646.

[93] A. d. Juan, Y. V. Heyden, R. Tauler and D. L. Massart, Assessment of new constraints applied to the alternating least squares method, *Analytica Chimica Acta,* 346 **(1997)**, 307-318.

[94] J. Jaumot, R. Gargallo, A. d. Juan and R. Tauler, A graphical user-friendly interface for MCR-ALS: a new tool for multivariate curve resolution in MATLAB, *Chemometrics and Intelligent Laboratory Systems,* 76 **(2005)**, 101-110.

[95] Z.-G. Wang, J.-H. Jiang, Y.-Z. Liang, H.-L. Wu and R.-Q. Yu, Vertex vectors sequential projection for self-modelling curve resolution of two-way data, *Chemometrics and Intelligent Laboratory Systems,* 82 **(2006)**, 154-164.

[96] Z.-G. Wang, J.-H. Jiang, Y.-Z. Liang, H.-L. Wu and R.-Q. Yu, Vertex vector sequential projection for the resolution of three-way data, *Talanta,* 68 **(2006)**, 1371-1377.

[97] C. A. Lieber and A. Mahadevan-Jansen, Automated Method for Subtraction of Flourescence from Biological Raman Spectra, *Applied Spectroscopy,* 57 (11) **(2003)**, 1363-1367.

[98] K. D. Braekeleer and D. L. Massart, Evaluation of the orthogonal projection approach (OPA) and the SIMPLISMA approach on the Windig standard spectral data sets, *Chemometrics and Intelligent Laboratory Systems,* 39 **(1997)**, 127-141.

[99] F. C. Sánchez, S. C. Rutan, M. D. G. García and D. L. Massart, Resolution of multicomponent overlapped peaks by the orthogonal projection approach, evolving factor analysis and window factor analysis, *Chemometrics and Intelligent Laboratory Systems,* 36 **(1997)**, 153-164.

[100] S. Gourvénec, D. L. Massart and D. N. Rutledge, Determination of the number of components during mixture analysis using the Durbin-Watson criterion in the Orthogonal Projection Approach and in the SIMPLe-to-use Interactive Self-modelling Mixture Analysis approach, *Chemometrics and Intelligent Laboratory Systems,* 61 **(2002)**, 51-61.

[101] A. W. Moore and J. W. Jorgenson, Median filtering for removal of low-frequency background drift, *Analytical Chemistry,* 65 **(1993)**, 188-191.

[102] *Website for MCR-ALS toolbox* **2005**. www.ub.edu/mcr/als2004.htm (Last viewed 1st October 2008)

[103] J. Tyrgg, Prediction and spectral profile estimation in multivariate calibration, *Journal of Chemometrics,* 18 **(2004)**, 166-172.

[104] P. W. Atkins, *Physical Chemistry*, Oxford University Press, Oxford, UK, **1994**.

[105] G. A. Pearson, A General Baseline-Recognition and Baseline-Flattening Algorithm, *Journal of Magnetic Resonance,* 27 **(1977)**, 265-272.

[106] D. N. Rutledge and A. S. Barros, Durbin-Watson statistic as a morphological estimator of information content, *Analytica Chimica Acta,* 454 **(2002)**, 277-295.

[107] E. Widjaja, C. Li, W. Chew and M. Garland, Band Target Entropy Minimization. A Robust Algorithm for Pure Component Spectral Recovery. Application to Complex Randomized Mixtures of Six Components, *Analytical Chemistry,* 75 (17) **(2003)**, 4499-4507.

[108] Z.-P. Chen, J. Morris and E. Martin, Correction of Temperature-Induced Spectral Variations by Loading Space Standardization, *Analytical Chemistry,* 77 (5) **(2005)**, 1376-1384.

[109] Z.-P. Chen, J. Morris and E. Martin, Extracting Chemical Information from Spectral Data with Multiplicative Light Scattering Effects by Optical Path-Length Estimation and Correction, *Analytical Chemistry,* 78 **(2006)**, 7674-7681.

[110] Z.-P. Chen and J. Morris, Improving the linearity of spectroscopic data subjected to fluctuations in external variables by the extended loading space standardization, *Analyst,* 133 **(2008)**, 914-922.

[111] J. Trygg and S. Wold, O2-PLS, a two-block (X-Y) latent variable regression (LVR) method with an integral OSC filter, *Journal of Chemometrics,* 17 **(2003)**, 53-64.

[112] J. Trygg, O2-PLS for qualitative and quantitative analysis in multivariate calibration, *Journal of Chemometrics,* 16 **(2002)**, 283-293.

[113] M. F. Devaux, P. Robert, A. Qannari, M. Safar and E. Vigneau, Canonical Correlation Analysis of Mid- and Near-Infrared oil Spectra, *Applied Spectroscopy,* 47 (7) **(1993)**, 1024-1029.

[114] R. Karoui, É. Dufour and J. D. Baerdemaeker, Common components and specific weights analysis: a tool for monitoring the molecular structure of semi-hard cheese throughout ripening, *Analytica Chimica Acta,* 572 **(2006)**, 125-133.

[115] R. Karoui, É. Dufour, L. Pillonel, E. Schaller, D. Picque, T. Cattenoz and J.-O. Bosset, The potential of combined infrared and fluorescence spectroscopies as a method of determination of the geographic origin of Emmental cheeses, *International Dairy Journal,* 15 **(2005)**, 287-298.

[116] G. Mazerolles, M. F. Devaux, E. Dufour, E. M. Qannari and P. Courcoux, Chemometric methods for the coupling of spectroscopic techniques and for the extraction of the relevant information contained in the spectral data tables, *Chemometrics and Intelligent Laboratory Systems,* 63 **(2002)**, 57-68.

[117] G. Mazerolles, M. Hanafi, E. Dufour, D. Bertrand and E. M. Qannari, Common components and specific weights analysis: A chemometric method for dealing with complexity of food products, *Chemometrics and Intelligent Laboratory Systems,* 81 **(2006)**, 41-49.

[118] E. M. Qannari, P. Courcoux and E. Vigneau, Common components and specific weights analysis performed on preference data, *Food Quality and Preference,* 12 **(2001)**, 365-368.

Appendix I

Combining Self-Modeling Curve Resolution Methods and Partial Least Squares to Develop a Quantitative Reaction Monitoring Method with Minimal Reference Data

NICHOLAS I. PEDGE* and ANTHONY D. WALMSLEY

Process Research and Development Department, AstraZeneca R&D Charnwood, Loughborough, Leicestershire, LE11 5RH, United Kingdom (N.I.P.); and Department of Chemistry, The University of Hull, Cottingham Road, Hull, HU6 7RX, United Kingdom (A.D.W.)

An example of combining self-modeling curve resolution (SMCR) methods and partial least squares (PLS) to construct a quantitative model using minimal reference data is presented. The objective was to construct a quantitative calibration model to allow real-time *in situ* ultraviolet–attenuated total reflection (UV/ATR) measurements to determine the end-point during a chlorination reaction. Time restrictions for development combined with difficult reaction sampling conditions required the method to be developed using only a few key reference measurements. Utilizing evolving factor analysis (EFA) and the orthogonal projection approach (OPA), initial estimates of the concentration and spectral profiles for the intermediate and product were obtained. Further optimization by multivariate curve resolution–alternating least squares (MCR-ALS) led to refined estimates of the concentration profiles. A PLS2 model was then constructed using the calculated concentration profiles and the preprocessed UV spectra. Using a standard PLS model compatible with the spectrometer's standard process software facilitated real-time predictions for new batches. This method was applied to five 45 liter batches in a large-scale laboratory facility. The method successfully predicted the product concentration of batch 1 but exhibited larger prediction error for subsequent batches. The largest prediction error was attained during batch 3, for which a final concentration of 0.22 mole·L^{-1} was predicted, while the true measured value was 0.271 mole·L^{-1} (an error of 18.8%). However, the qualitative real-time profiles proved to be extremely useful as they allowed the end-point to be determined without sampling or performing off-line analysis. Furthermore, the concentration profile of the intermediate species, which could not be observed by the off-line method, could also be observed in real-time and gave further confidence that the process was approaching the end-point. Another benefit of real-time reaction profiles was encountered during the manufacture when the formation of product in batch 3 appeared to be progressing slower than was observed in previous batches. This prompted a check of the batch temperature and it was found to be 10 °C lower than the required set-point. The temperature was corrected and the batch successfully reached completion in the expected time.

Index Headings: **Ultraviolet spectroscopy; UV spectroscopy; Attenuated total reflection; ATR; Self-modeling curve resolution; SMCR; Evolving factor analysis; EFA; Orthogonal projection approach; OPA; Partial least squares; PLS.**

INTRODUCTION

Process monitoring using spectroscopic techniques such as near-infrared (NIR), ultraviolet (UV), Raman, mid-infrared (MIR), and fluorescence spectroscopy offer plant operators the opportunity to measure their processes remotely, in real-time, and noninvasively.[1–4] The spectral data obtained from these measurements can contain a wealth of information about the chemical composition and physical properties of the process. A common application of process spectroscopy is to monitor the progress of a reaction and indicate when an end-point has been reached. This can be done using a qualitative or quantitative model. For a qualitative method, one might use a peak height, peak area, peak ratio, or principal components score values for a selective peak or spectral region. By plotting these derived values as a function of time (or some other suitable process parameter), a profile that is correlated to the product or reactant concentration is obtained. The method is qualitative because the actual concentration of product or reactant is not predicted from this measurement. A quantitative model can be obtained from the same spectral attributes (peak area, peak height, etc.), but the values are transformed into actual concentrations using a previously developed calibration model.

In order to construct a calibration model, a data set comprising a series of spectral measurements and the corresponding response values (e.g., concentrations) obtained by an external reference method (e.g., high-performance liquid chromatography (HPLC), gas chromatography (GC), nuclear magnetic resonance (NMR), titration, assay, etc.) is required.[5] This can either be a set of nonreacting mixtures with known composition or samples taken during a process while acquiring spectral data. For reactions or processes for which it is not possible to prepare mixtures of known composition, perhaps due to the absence of isolated reference materials or stability issues, sampling from the process is usually the only practical alternative.

It is the difficulties described above that naturally led to the investigation and development of "calibration-free" methods of data analysis and prediction. Kubista et al.[6] describe a method suitable for equilibrium systems such as pH, concentration, temperature, and ionic strength titrations. The method is based on principal components analysis (PCA) of the spectral data followed by derivation of a rotation matrix that transforms the abstract spectra (principal component loadings) and concentration profiles (principal component scores) into the true spectral and concentration profiles. This approach uses measurement of physical properties such as pH, volume, temperature, pressure ionic strength, etc., that effect the concentrations of the components in a predictable way. By fitting the data using a known thermodynamic model, the concentration parameters are derived indirectly. Although this method has been demonstrated for systems with well-understood equilibria, it is not so well suited to dynamic systems such as reactions.

A comparison of hard and soft modeling for reaction modeling was published by Dyson et al.[7] In this work, the catalytic hydrogenation of 1-chloro-2-nitrobenzene was monitored by Raman and IR spectroscopy. Using self-modeling curve resolution methods such as evolving factor analysis (EFA), window factor analysis (WFA), and iterative target

Received 6 February 2007; accepted 8 June 2007.
* Author to whom correspondence should be sent. E-mail: nicholas.pedge@astrazeneca.com.

0003-7028/07/6109-0940$2.00/0

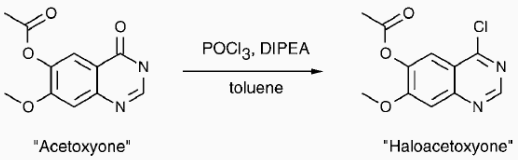


FIG. 1. Reaction scheme for the chlorination of acetoxyone using phosphoryl chloride. The reaction solvent is toluene.

transformation factor analysis (ITTFA), the relative concentration profiles of the major species were derived and compared well with the off-line NMR results. The relative concentration profiles calculated by hard modeling (explicit kinetic modeling) also showed good agreement with the off-line NMR results. The qualitative (relative) concentration profiles were obtained *post-hoc*.

A calibration-free method using NIR reaction spectra acquired during the 4-(dimethylamino)pyridine catalyzed esterification of butanol was reported by Gemperline et al.[8] The authors used a nonlinear fitting routine to fit a third-order reaction mechanism to five batches simultaneously. In this work, the data from four batches was used to predict the profiles of the fifth batch. Prediction of new data requires augmentation of the new spectra with spectra acquired in previous runs and fitting a multi-way kinetic model.

More recently, Richards et al.[9] applied multivariate curve resolution–alternating least squares (MCR-ALS) to Fourier transform infrared (FT-IR) data acquired during a rhodium catalyzed asymmetric transfer hydrogenation reaction. This technique was applied to the spectral data retrospectively and the results predicted by MCR-ALS showed good agreement with the validation measurements using HPLC.

Although there are many papers in which curve resolution methods have been applied to spectral data sets retrospectively,[8–19] to our knowledge there have been no papers that describe the use of data obtained from curve resolution to construct a quantitative calibration model that can be used in the prediction of new measurements during scale-up.

In this work, the reaction of interest is the chlorination of "acetoxyone" using phosphoryl chloride to produce "haloacetoxyone", as shown in Fig. 1. This reaction is part of a longer sequence to produce a key intermediate in the synthesis of an active pharmaceutical ingredient (API). An *in situ* real-time method for monitoring this reaction was required for the manufacturing campaign in a large-scale laboratory (LSL) to indicate the end-point and to provide valuable scale-up information. Furthermore, as the reaction mixture was initially heterogeneous due to undissolved starting material, a quantitative method was necessary to confirm that all starting material had dissolved and reacted to form product.

METHODS

Evolving Factor Analysis. Evolving factor analysis was originally developed by Gampp and Maeder et al.[20–23] for the model-free resolution of spectrophotometric titration data and was later applied to multi-wavelength chromatographic data.

An $I \times J$ matrix of reaction spectra $\mathbf{X}$, where I is the number of observations (spectra) and J is the number of variables (wavelengths) can be expressed as the matrix product

$$\mathbf{X} = \mathbf{C}\mathbf{S}^{\mathrm{T}} + \mathbf{E} \tag{1}$$

where $\mathbf{C}_{(I,K)}$ is the matrix of concentration profiles for K components, $\mathbf{S}_{(J,K)}$ is the matrix of spectral profiles for the same k components and $\mathbf{E}_{(I,J)}$ is the matrix of residuals (remaining part of $\mathbf{X}$ not modeled by $\mathbf{C}$ and $\mathbf{S}$ and usually accounts for measurement noise). To simplify the description without loss of generality, in the absence of measurement noise the expression can be simplified to

$$\mathbf{X} = \mathbf{C}\mathbf{S}^{\mathrm{T}} \tag{2}$$

Singular value decomposition of $\mathbf{X}$ would yield K non-zero singular values and indicates the rank of data. The corresponding scores and loadings calculated from the eigenvalues and eigenvectors would loosely resemble concentration and spectral profiles. However, these are abstract solutions and would require additional transformations to obtain the true profiles. Performing PCA on the complete data matrix $\mathbf{X}$ would reveal the maximum rank of the data but does not provide any information about local rank, i.e., how many species are contributing to the signal at each time point.

One advantage of EFA is that it utilizes the inherent evolutionary structure in the data to determine the local rank at each time point. The basis of EFA is evolving principal components analysis. Starting from the first spectrum, a sub-matrix is created and principal components analysis is performed to calculate the principal components and their corresponding eigenvalues. The analysis is then continued by incrementing the size of the sub-matrix by one spectrum and calculating a new set of eigenvalues. This process is then repeated until the last spectrum in the data matrix is reached. The reverse analysis is performed by starting with the last spectrum and incrementing the window in the opposite direction. The log of the eigenvalues obtained from both forward and backward analysis are then plotted as a function of spectrum number (time). The forward analysis shows the appearance of new components, while the backward analysis shows the disappearance of components. The concentration window of each component k can be calculated by combining the kth eigenvalues obtained from the forward EFA with the $K - k + 1$ set of eigenvalues obtained from the backward EFA ($K = \min[I, J]$). The resulting concentration windows are then re-scaled and collected into the matrix $\mathbf{C}$. The corresponding spectral profiles can be calculated by least squares using the expression

$$\mathbf{S} = \mathbf{X}^{\mathrm{T}}\mathbf{C}(\mathbf{C}^{\mathrm{T}}\mathbf{C})^{-1} \tag{3}$$

The matrix $\mathbf{S}$ calculated above can then be used to calculate a new estimate of $\mathbf{C}$ using

$$\mathbf{C} = \mathbf{X}\mathbf{S}(\mathbf{S}^{\mathrm{T}}\mathbf{S})^{-1} \tag{4}$$

Alternating least squares is performed by repeating the calculations shown in Eqs. 3 and 4 to further improve the concentration and spectral profiles utilizing basic constraints such as non-negativity. This iterative approach produces physically meaningful solutions that are optimized in a least squares sense.

Orthogonal Projection Approach. The orthogonal projection approach (OPA) was developed by Sanchez et al.[24] for peak purity assessment of HPLC-DAD data. The aim of OPA is to find the purest set of spectra (rows) or variables (columns)

in a data matrix and works on the basic assumption that the purest spectra are mutually more dissimilar than the corresponding mixture spectra.[25] The measure of dissimilarity d_i is defined as the determinant of the dispersion (covariance) matrix of $\mathbf{Y}_i$, given by

$$d_i = \det(\mathbf{Y}_i^{\mathrm{T}}\mathbf{Y}_i) \tag{5}$$

where $\mathbf{Y}_i$ comprises one or more of the reference spectra and the spectrum $\mathbf{x}_i$, the ith spectrum from $\mathbf{X}$. The determinant of the dispersion matrix $\mathbf{Y}_i$ measures the area of the parallelogram defined by the reference spectrum (or spectra) and $\mathbf{x}_i$. If the spectrum $\mathbf{x}_i$ is very similar to the reference spectra, the angle between the two vectors in the J-dimensional space will be small, so the resulting parallelogram will have small area and thus a low dissimilarity value. A spectrum that is quite different from the previously found reference spectra will give a high dissimilarity value.

The routine is usually initiated by selecting the mean spectrum, $\bar{\mathbf{x}}$ as the first reference spectrum. The mean spectrum is normalized to unit length and the dissimilarity of the unnormalized spectrum ($\mathbf{x}_i$, $i = 1$ to I) is calculated iteratively using Eq. 5. The spectrum with the highest dissimilarity value will replace the mean spectrum as the first reference spectrum and the process will be repeated. The spectrum with the highest dissimilarity value with respect to the first reference spectrum is normalized and added to $\mathbf{Y}$. The procedure is continued until the desired number of pure spectra has been identified, the magnitude of the dissimilarity values drops below a threshold value, or the dissimilarity profiles resemble noise. Plotting the dissimilarity values for each pure spectrum as a function of spectrum number (time) will often provide a good approximation of its corresponding concentration profile. OPA can be applied in either the row (spectral) direction or the column (variable) direction.

Multivariate Curve Resolution–Alternating Least Squares. The purpose of multivariate curve resolution–alternating least squares (MCR-ALS)[12,13,16,26,27,29,34] is to resolve a multi-component system assumed to have an additive bilinear model as shown in Eqs. 1 and 2. MCR-ALS is an iterative method that gives equal priority to both the concentration and spectral profile matrices and optimizes both $\mathbf{C}$ and $\mathbf{S}^{\mathrm{T}}$ during each iteration cycle. All components (columns of $\mathbf{C}$ or $\mathbf{S}^{\mathrm{T}}$) are modeled simultaneously during the procedure.

Methods such as EFA and OPA can be applied to a two-way data matrix to produce estimates of the underlying concentration or spectral profiles; alternating least squares can then be used to refine the estimates using simple constraints such as non-negativity. The true power and flexibility of MCR-ALS lies in its ability to use and combine initial estimates obtained from various methods and apply a number of physically meaningful constraints such as non-negativity, unimodality, and closure. Additional external information such as known concentration or spectral profiles for one or more species can be incorporated and zero-concentration regions can be specified when it is known that a particular species will not be present over a specific time period. These additional options require user input, and thus a good understanding of the data and its underlying structure is necessary to prevent the use of invalid assumptions. The purpose of including additional constraints and external information is that it further reduces the range of feasible solutions ($\mathbf{C}$ and $\mathbf{S}^{\mathrm{T}}$) and is therefore more likely to produce a solution that describes the true underlying structure.

Partial Least Squares. Partial least squares (also referred to as projection to latent structures)[2,5,25,28–33] is a commonly used multivariate regression method throughout many disciplines such as chemometrics, physical and biological sciences, social sciences, economics, etc. The history and details of this technique can be found in most chemometrics textbooks so only a brief description of the PLS2 method used in this work will be given here.

Partial least squares is often used in spectroscopic calibration applications as it can help to overcome the problem of high collinearity between spectral variables that can lead to unstable regression coefficients if traditional multi-linear regression (MLR) were applied. The problem of collinearity is reduced by representing both the $\mathbf{X}$ and $\mathbf{Y}$ blocks (spectra and concentrations, respectively) as linear combinations of the original variables.

$$\mathbf{X} = \mathbf{T}\mathbf{P}^{\mathrm{T}} + \mathbf{E} \tag{6}$$

$$\mathbf{Y} = \mathbf{U}\mathbf{Q}^{\mathrm{T}} + \mathbf{F} \tag{7}$$

The objective of PLS is to find a set of latent variables that maximizes the covariance between $\mathbf{X}$ and $\mathbf{Y}$. The factors are calculated to maximize the correlation between the $\mathbf{X}$ and $\mathbf{Y}$ blocks while also accounting for the maximal amount of structured variance in each block. The number of factors to use in the PLS model is usually chosen by selecting the value that minimizes the root mean square error of cross-validation (RMSECV) or root mean square error of prediction (RMSEP).

During the calculation of each set of latent variables, the loading vectors $\mathbf{p}^{\mathrm{T}}$ and $\mathbf{q}^{\mathrm{T}}$ are iteratively improved. Projection of $\mathbf{X}$ and $\mathbf{Y}$ onto the loading vectors results in a pair of scores vectors $\mathbf{t}$ and $\mathbf{u}$. The final scores vectors will be found when the difference between two consecutive iterations is below a predefined threshold. The inner relationship that relates the scores matrices $\mathbf{T}$ and $\mathbf{U}$ ($\mathbf{X}$ and $\mathbf{Y}$ block respectively) is $\mathbf{U} = \mathbf{T}\mathbf{W}$. Although Eqs. 6 and 7 would suggest that the loading vectors are calculated for each block independently, this can actually lead to poor correlation between the two sets of scores. In practice, the inner relationship is improved by exchanging the scores vectors $\mathbf{t}$ and $\mathbf{u}$ during the iterative cycle. Once all the latent vectors have been found, the regression matrix is calculated using

$$\hat{\mathbf{B}} = \mathbf{P}(\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}\mathbf{W}\mathbf{Q}^{\mathrm{T}} \tag{8}$$

Each column of $\mathbf{B}$ contains the regression spectrum for the corresponding column of $\mathbf{Y}$. Future values of $\mathbf{Y}$ are predicted from measured spectra using the expression

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{B}} \tag{9}$$

EXPERIMENTAL AND MODEL DEVELOPMENT

Instrument and Software. This work was undertaken using a Zeiss MCS501 single-beam UV/Vis diode array spectrophotometer fitted with a deuterium source lamp coupled to an attenuated total reflection (ATR) immersion probe. The spectral data was acquired using Zeiss ProcessXplorer (version 1.2). Data analysis was performed using MATLAB 7.0.1 [R14] (The

Mathworks, Natick, MA) running on a Toshiba Satellite Pro 6100 laptop computer (Intel® Pentium® 4 processor, 2 GHz, 1 Gb RAM, Windows XP Professional). The Eigenvector PLS Toolbox (version 3.5, Eigenvector Research, Inc., WA) was used for preliminary data analysis. The EFA and OPA programs were written in-house (NIP) based on the algorithms published in the literature. MCR-ALS was performed using the MCR-ALS toolbox written by Jaumot et al.[34] The final PLS model was calculated using the GRAMS PLSplus/IQ toolbox (Thermo Scientific, Waltham, MA).

Initial Laboratory Reactions. The initial laboratory reactions were performed in a 100 mL jacketed glass vessel with a Huber CC3 thermoregulator unit providing temperature control. The various ports on the vessel lid were used to accommodate an overhead agitator, a Pt100 temperature probe, condenser/nitrogen purge, a polytetrafluoroethylene (PTFE) addition line introduced through a septum, and a UV-ATR probe. Prior to charging any materials to the reaction vessel, an air background spectrum was acquired at room temperature. The reaction was prepared by charging 5.00 g (1.0 equivalent) of the starting reactant (acetoxyone) to the reaction vessel, followed by 70 mL toluene (14 relative volumes) and diisopropylethylamine base solution (3.0 g, 1.10 equivalents). The mixture was then heated to 70 °C. When the set-point temperature had been reached, the reaction mixture was a thick but mobile slurry due to the very low solubility of the acetoxyone starting material. The reaction was then started by charging phosphoryl chloride (8.30 g, 2.50 equivalents) using a syringe driver over 20 minutes. At the end of the addition, the reaction mixture was still a slurry due to undissolved starting material. After approximately two hours, the starting material had completely dissolved.

Throughout the course of the experiment, several attempts were made to sample the mixture for analysis by HPLC. Unfortunately, the low solubility of the starting material and product resulted in almost immediate precipitation as the sample cooled in the sampling pipette.

Practical Difficulties Observed During the Initial Experiments. Although initial examination of the UV/ATR reaction data suggested that the reaction progress could be monitored spectroscopically, there were a number of practical issues that would preclude attempts to obtain reliable reference measurements for subsequently developing a calibration model. A calibration model is required to quantitatively monitor a reaction using *in situ* spectroscopy, as the end-point criterion is often defined as a molar ratio of product to starting material or simply a minimum target concentration of product. To build a PLS model in the traditional manner would typically require off-line analysis (e.g., by HPLC) of many reaction samples to determine the concentration of each reactant and product of interest. Observations made during the initial trial experiment indicated that it would not be possible to simply sample the reaction mixture during the course of the reaction to provide the necessary concentration reference data due to the following difficulties:

(1) The reaction mixture starts as a slurry and the starting material (acetoxyone) is only partially soluble but dissolves slowly during the course of the reaction.

(2) The reaction solution rapidly cools on sampling, causing a mixture of acetoxyone/haloacetoxyone to precipitate out of solution; the measured concentrations in a filtered solution would no longer be representative of what the ATR probe measured in the solution phase. This would lead to discrepancies between the *in situ* spectra and the off-line reference measurements.

(3) Following complete addition of phosphoryl chloride, the UV reaction spectra indicate the presence of an intermediate species for which there was no reference material available. Without this reference material, it would not be possible to measure a pure UV reference spectrum or prepare an HPLC assay standard of the isolated intermediate species.

Performing a "Calibration" Reaction to Obtain Key Reference Measurements. The solubility of the acetoxyone starting material was established by adding small amounts of the material to 100 mL of a toluene/diisopropyethylamine solution heated to 70 °C. When undissolved solid was observed in the bottom of the reaction vessel, the supernatant liquid was sampled and assayed against a standard of known concentration using reverse-phase HPLC. The solubility of the starting material in the reaction solvent was found to be 2.52×10^{-4} mole·L^{-1}. Examination of the UV/ATR spectra also revealed that due to the low solubility of the starting material, its UV spectrum could not be distinguished from the solvent. As a consequence, an *in situ* reaction monitoring method based on the consumption of acetoxyone was not possible. A "calibration" reaction was then performed by repeating the experiment described in the Initial Laboratory Reactions subsection. At the end of the reaction, when all starting material had been consumed and the mixture was homogenous, a sample was taken for off-line analysis. The concentration of product (haloacetoxyone) was determined by assaying against a standard of known concentration using reverse-phase HPLC. A value of 0.272 mole·L^{-1} was obtained. The extent of reaction, based upon a peak area ratio of product to starting material was greater than 98%.

Application of Self-Modeling Curve Resolution Methods to the Laboratory Spectral Data. ***Overview of the Spectral Data.*** During the calibration reaction a spectral data matrix comprising 172 spectra at 1 minute intervals over the wavelength range 220 to 400 nm at 1 nm intervals (resulting in 180 variables) was acquired. Figures 2a and 2b show the reaction spectra obtained. The appearance and disappearance of the intermediate species is more clearly observed in the three-dimensional (3D) plot of the second-derivative data (shown in Fig. 3). Visual examination and a preliminary examination of the data by principal components analysis also indicated that the region 220 to 265 nm is highly overlapped and is significantly noisier than the region 265 to 400 nm. The main reason for this is that the absorbance values in the original spectra between 220 and 265 nm are almost 3 AU (Fig. 2a). The region from 350 to 400 nm also contained no spectral features. Therefore, prior to further analysis, the data was truncated to only include the region 265 to 350 nm.

Principal components analysis of the mean-centered, second-derivative data indicates that two major components could be resolved from the spectra. The first two principal components account for 98.96% of the total variance, while the third and fourth components would contribute a further 0.51 and 0.37% respectively. The eigenvalues, scores, and loadings are shown in Fig. 4.

The scores for the first two principal components clearly show features that correlate with the expected profiles of the intermediate and product.

Initial Estimates of the Concentration Profiles Using Evolving Factor Analysis. To provide an estimate of the

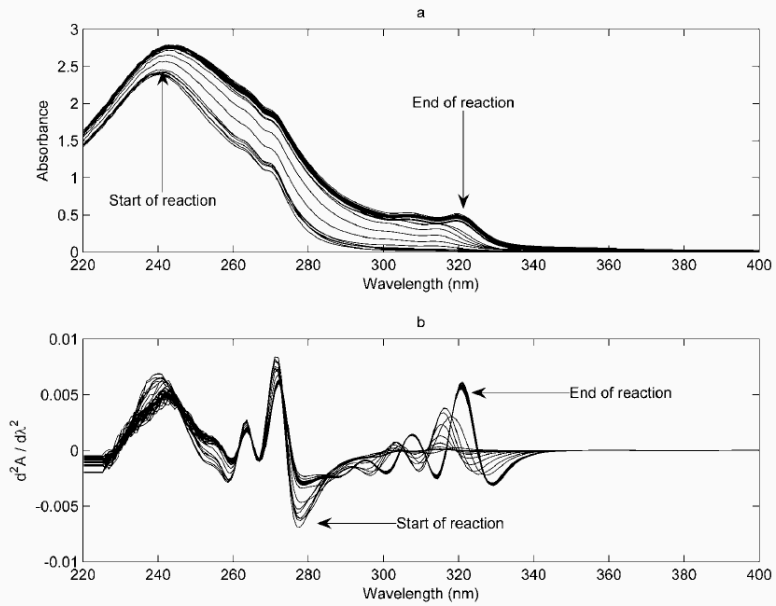


FIG. 2. (a) UV spectra from laboratory scale reaction. Every fifth spectrum is shown to improve clarity; (b) second-derivative spectra from laboratory scale reaction. Every fifth spectrum is shown to improve clarity.

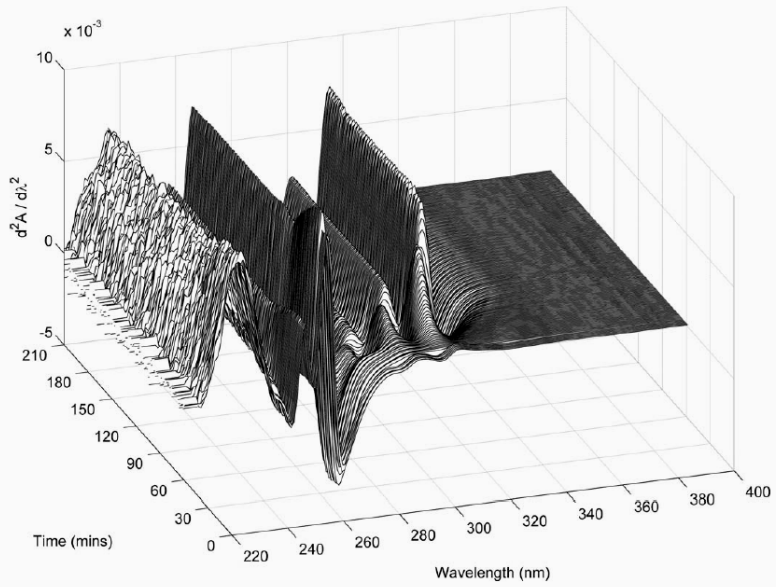


FIG. 3. 3D plot of second-derivative UV spectra acquired during the laboratory scale experiments.

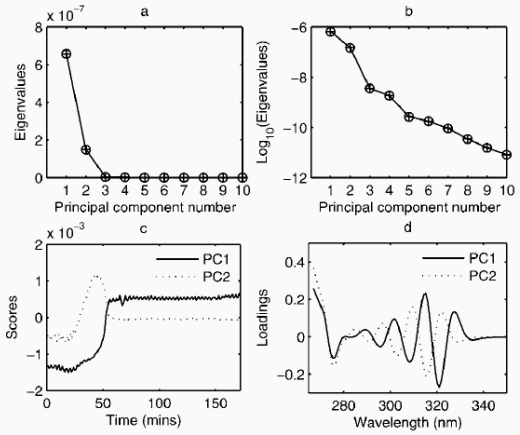


FIG. 4. Summary plots from PCA of mean-centered, second-derivative UV spectra acquired during the laboratory scale calibration reaction; (**a**) plot of eigenvalue versus principal component number; (**b**) plot of $\log_{10}$ (eigenvalue) versus principal component number; (**c**) plot of scores for first two principal components versus time (min); principal component 1 (——), and principal component 2 (- - -); and (**d**) plot of loading vectors for the first two principal components versus wavelength; principal component 1 (——), principal component 2 (- - -).

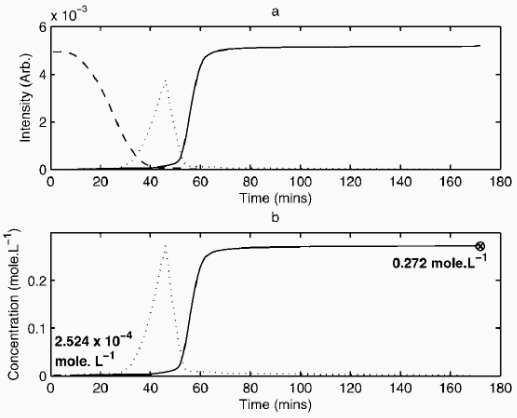


FIG. 5. (**a**) Estimates of concentration profiles obtained by applying evolving factor analysis to the second-derivative UV spectra; (**b**) scaled initial estimates of the concentration profiles.

concentration profiles of the spectroscopically active major species, evolving factor analysis was applied to the second-derivative data. The profiles obtained from EFA are shown in Fig. 5a. Component 1 appears to correspond to the starting mixture spectrum (toluene/acetoxyone) and its subsequent dilution on addition of phosphoryl chloride. It is the ability of EFA to perform a local rank analysis of the first several spectra that allowed this feature to be detected. The remaining two components again seemed to approximate the expected concentration profiles of the intermediate and product. Using the experimental values obtained from the HPLC assays described previously, the EFA profiles could be scaled to provide initial estimates of the underlying concentration profiles. As these profiles will later be refined, the concentration profile of the intermediate species was also scaled to the same maximum value as the product.

Initial Estimates of the Pure Component Spectra Using the Orthogonal Projection Approach. In addition to the concentration profile estimates obtained by EFA, orthogonal projection analysis was also applied to second-derivative UV data to obtain initial estimates of the pure component spectra for the intermediate and product species. Examination of the EFA profiles indicates that the most selective spectra would be observed at approximately 0, 45, and 172 minutes, corresponding to the starting spectrum (predominately toluene), the maximum concentration of the intermediate species, and the product at end of the reaction, respectively. The most dissimilar spectra were identified at 0, 43, and 168 minutes. The resulting un-normalized "pure" spectra were collected into a matrix $\mathbf{S}_{est}$ and scaled to unit concentration (1.0 mole·L^{-1}). The pure spectrum corresponding to the product was scaled using the known concentration of this species obtained by HPLC solution assay at the end of the reaction. Since reference standard material for the intermediate species was not available, an assumption regarding its unit concentration intensity was necessary. The spectrum for the intermediate species displays very similar shape but exhibits a hypsochromic shift with respect to the product spectrum. It was therefore assumed that the 2-norm of the intermediate and product spectra would be equivalent. After scaling the product spectrum to unit concentration, the spectrum of the intermediate species was scaled to give the same 2-norm as the unit concentration product spectrum ($\|S_i\|_2 = \|S_p\|_2$).

Although EFA identified three profiles, the first component is believed to correspond to the dilution of toluene during the addition of phosphoryl chloride. PCA of the same data indicated that no significant factors corresponding to this dilution were observed. Since none of the first three principal components seem to model this behavior, the dilution accounts for less than 0.35% of the total variance. The emphasis is therefore on resolving the profiles for the intermediate and product only. The scaled spectra are shown in Fig. 6b.

Refining the Concentration Profiles Using Multivariate Curve Resolution–Alternating Least Squares. Using the initial estimates obtained as described in the previous sections, the pure spectral and concentration profiles for the intermediate species and product were refined using multivariate curve resolution–alternating least squares. The concentration profiles were constrained to non-negative solutions and a concentration selectivity matrix was used to include the starting reactant and product concentration information provided by the reference analysis (HPLC solution assay). The concentration selectivity matrix was also used to constrain the concentration of the product to zero during the early part of the reaction. A spectral selectivity matrix was used to constrain the product spectrum as obtained from OPA. The spectra for the starting reactant/toluene and intermediate species were not initially constrained. In this particular application the second-derivative spectra were used, but in those cases where the original un-transformed spectra are used, the spectral non-negativity constraint could also be applied.

The results obtained from the optimization are shown in Fig. 6.

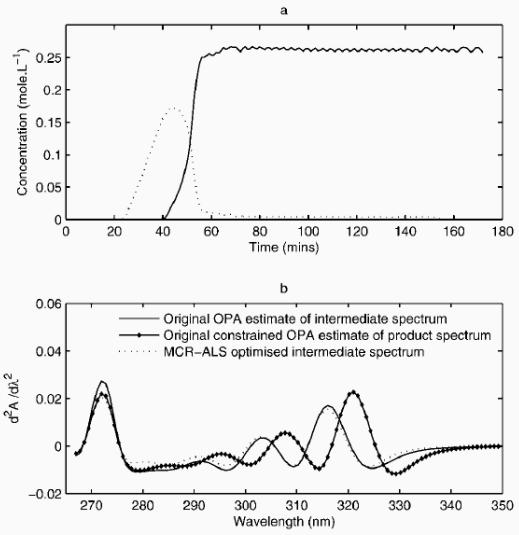


FIG. 6. (**a**) MCR-ALS optimized concentration profiles; (**b**) MCR-ALS optimized spectral profiles. Initial estimates of the pure component spectra were obtained by orthogonal projection approach and were scaled to unit concentration. The OPA spectrum for the intermediate species was scaled to have the same 2-norm as the scaled product spectrum.

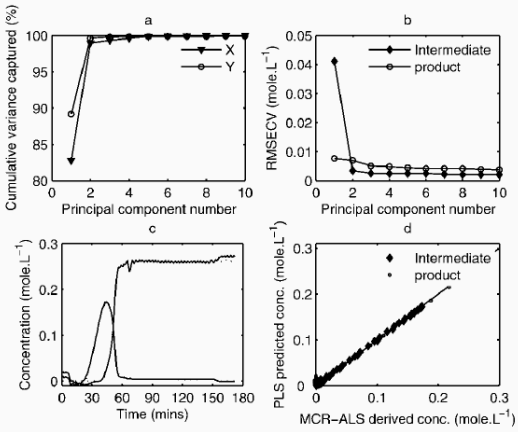


FIG. 7. (**a**) Cumulative variance explained (%) versus principal component number for spectral matrix (**X**) and concentration matrix (**Y**); (**b**) RMSECV versus principal component number; (**c**) MCR-ALS derived concentrations (···) and PLS predicted concentrations (——) versus time (min); and (**d**) MCR-ALS derived concentrations versus PLS predicted concentrations (analogous to actual versus predicted).

Calculating a Partial Least Squares-2 Model to Predict the Concentrations in Real Time. Although the unit concentration pure analyte spectra had been derived using curve resolution, it was not possible to incorporate them directly into a ProcessXplorer method as a standard model for predicting the concentrations (for example, using CLS). The reaction spectra and the corresponding MCR-ALS optimized concentration vectors for the intermediate and product were therefore used to calculate a PLS2 model using Grams PLSplus/IQ chemometric software. Using a standard PLS model would allow the predictions to be made in real-time as Grams PLSplus/IQ chemometric models can be imported and used by the instrument software.

A PLS2 model was selected as it allows a matrix of dependent variables (**Y**-block) to be fitted simultaneously within a single model. PLS1 models fit each component of **Y** independently, and although this is reported to give better RMSECV and RMSEP values, it results in a set of scores, weights, and loadings matrices for each component of **Y**. A PLS2 model was chosen because the initial trials using PLS2 exhibited good prediction errors within a single model and, as with MCR-ALS, it models all components simultaneously.

The spectral data collected during the laboratory experiment were imported and preprocessed as described previously (265 to 350 nm, transformation to second-derivative spectra). Both the **X** (spectral) and **Y** (concentrations) matrices were mean centered.

The main figures of interest for the resulting three-factor PLS model are shown in Fig. 7.

Applying the Method to 45 L Batches in a Large-Scale Laboratory. The instrument used during method development was transferred to the large-scale laboratory facility and coupled to a process ATR probe via two 4-meter lengths of fiber-optic cable terminating with standard SMA905 connectors. The probe was installed using a customized flange fitted to the charging port of the vessel lid. This configuration was problematic as it was not possible to acquire a background spectrum with the probe in position. As the spectrometer was a single-beam instrument using a deuterium source lamp, a new background spectrum was required prior to starting each batch. For the first batch, it was possible to collect a spectrum with the probe installed on the vessel lid and the ATR crystal positioned in the reactor headspace. In subsequent batches, the background spectra had to be acquired prior to re-installation of the probe.

RESULTS AND DISCUSSION

Model Development. Principal components analysis of the second-derivative spectra suggested that the first two principal components would account for 98.96% of the total variance in the data. Subsequent evolving factor analysis of the same data revealed three potential components that were subsequently scaled using the concentration values provided by the HPLC solution assays. The scores of the first two principal components could have been used as initial estimates of the concentration profiles for optimization by MCR-ALS. However, the scaled EFA profiles were considered better estimates because they were non-negative and also captured the additional component that was either not detected by PCA or was embedded in the first few principal components.

Applying OPA also identified three major components and provided initial estimates of their pure spectral profiles. The unnormalized pure spectra were scaled to unit concentration using the HPLC reference data. Furthermore, OPA confirmed that assigning the data as rank 3 would be appropriate as additional components were only describing small peak shifts. The dissimilarity plots also provided further evidence that the EFA

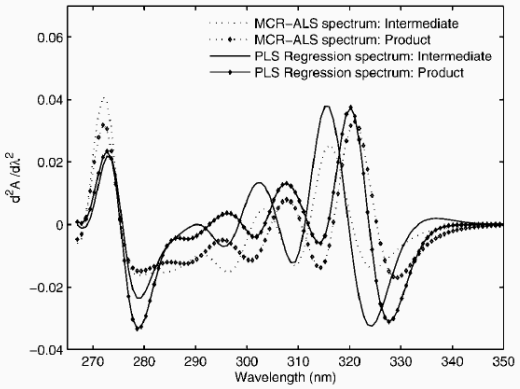


FIG. 8. Comparison MCR-ALS derived pure spectral profiles (···) and pure spectral profiles (——) calculated from PLS regression vectors. The profiles were normalized to unit length to allow them to be compared on the same plot.

profiles were appropriate estimates of the concentration profiles.

The initial estimates of the pure concentration and spectral profiles were combined with suitable constraints and refined using MCR-ALS. Figure 6 shows the resulting optimized pure concentration and spectral profiles. The lack-of-fit of the MCR-ALS optimized factors with respect to reduced data reconstructed from the first three principal components was 0.0085%. Percentage of variance explained by the three MCR-ALS components was 99.87% (c.f. PCA 99.47%).

The MCR-ALS derived concentrations were then used as the **Y** block to construct a PLS2 model. Figures 7a and 7b show that the first three principal components account for 99.32 and 99.85% of the total variance in the **X** and **Y** blocks respectively. The root mean squared error of cross-validation (RMSECV) using "leave-one-out" cross-validation was 0.0025 mole·L^{-1} for the intermediate species and 0.0053 mole·L^{-1} for the product. These values should obviously be interpreted with some care as the **Y** block data used to calculate the PLS model was not true reference data provided by an orthogonal off-line method but rather was derived from the spectral data itself.

A least squares method for calculating the pure spectral profiles from the PLS regression vectors[35] was also employed to allow comparison with the MCR-ALS pure spectral profiles. These are calculated using the simple expression $\mathbf{K} = \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}$, where **K** is the matrix of pure spectral profile estimates and **B** is the matrix of PLS regression coefficients. Figure 8 shows that the PLS regression coefficients transformed to pure spectral profiles that are highly correlated with the pure spectral profiles derived from MCR-ALS. There were some small discrepancies in the magnitude of each pair of spectral profiles, but this can still be a useful way to check that the PLS regression coefficients are indeed an accurate transformation of the known pure spectral profiles and do not contain any additional features arising from poor selectivity.

The difference between the MCR-ALS spectra and the pure spectral profiles calculated from the PLS regression vectors may be a consequence of the way in which they are derived by each method. Both self-modeling curve resolution and PLS are based on a bilinear model of the spectral data. The aim of curve resolution is to factorize the spectral data matrix into two appropriately sized matrices representing the pure analyte concentration and spectral profiles. The factors recovered by curve resolution are optimized in a least squares sense, i.e., they minimize the total sum of squares of the residual unmodeled part of the data. The factors recovered by curve resolution are not constrained to be orthogonal and represent a linear combination of all possible basis vectors spanning the data. If applied to the original data matrix, curve resolution therefore represents a full-rank model. However, PLS will search for a set of orthogonal basis vectors that both maximize the covariance of X/Y and the amount of variance explained in X and Y. These basis vectors form a subspace that spans only part of the spectral space. One would expect the two sets of spectra to converge by increasing the number of PLS factors. However, three factors were chosen initially as there were no significant differences between the profiles obtained by applying curve resolution methods to the original data or to data reproduced from three principal components, suggesting that a three-factor model would be appropriate.

Implementing the Model. The final PLS model developed using the reaction spectra acquired in the laboratory was applied to five 45 L batches in a large-scale laboratory facility. The same spectrometer used in the development work was used with a custom process ATR probe. Prior to installing the probe, the optical throughput of the process ATR probe and the longer fiber-optic cables was assessed. Although the instrument required a slightly higher integration time to give a background spectrum of comparable intensity to those observed in the laboratory, the correlation between the two background spectra was very good. This suggested that the longer fiber length and probe should not have a detrimental affect on the measured absorbance spectra.

The PLS model was used to monitor five batches over a period of four weeks. The results from the first batch were successful as the concentration profiles were similar to those observed in the laboratory. More importantly, the predicted end-point concentration showed good agreement with both the expected theoretical value and the actual measured value obtained by HPLC analysis (0.27 mole·L^{-1}). The results for batch 1 are shown in Fig. 9a. The profiles at the start of the reaction are not zero, as would be expected, and clearly show the point at which the addition of phosphoryl chloride is added at 39 minutes.

The product concentration prediction results obtained from the subsequent four batches were not so accurate. Figure 9b shows the overlaid predicted product concentration profiles for all five batches. The batches were aligned by setting the time at which phosphoryl chloride addition was started to t_0. The five overlaid batches all show very similar profiles but are not correctly predicting the true concentration. The largest prediction error attained was during batch 3 for which a final concentration of 0.22 mole·L^{-1} was predicted, while the true measured value was 0.271 mole·L^{-1} (an error of 18.8%).

There are two potential experimental factors contributing to the observed prediction errors. The first factor is a consequence of the way in which the background spectrum was acquired for batches 2 to 5. In batch 1, the background spectrum was acquired with the probe installed on the vessel lid but not fixed to the full immersion depth, allowing the tip to be positioned in

the headspace above the charged reactants. Therefore, both the probe and the fiber-optic cables were very close to their final orientation when the background spectrum was acquired. Unfortunately, this approach could not be adopted in subsequent batches, so the reference spectra were acquired prior to installing the probe. The change in the optical transmissivity of the fibers when installing the probe would result in subsequent single-beam spectra with higher or lower intensity than the background spectrum. Upon conversion to an absorbance spectrum, the difference between the intensities of the background and sample single-beam spectra would be observed as an apparent increase or decrease in calculated absorbance values. This is usually observed as a baseline shift that should be removed by zero-minimum offset correction or transforming the spectra to the first or second derivative. As the predictions were based upon second-derivative spectra, the effect of fiber movement should be minimized.

The second major factor is the observed magnitude of the absorbance values in the measured spectra and their subsequent derivative spectra. Examination of the second-derivative spectra indicated that the magnitude of the absorbance values for the final four batches were lower than those for the development reaction and batch 1. The peak positions and peak shapes for the spectra acquired in all batches show excellent correlation but clearly display a scaling discrepancy. As described above, a variable baseline offset is present in all six data sets, but is more severe in the LSL batches. Although this could be due to movement of fibers as discussed above, it could also suggest that probe fouling was occurring. One possible consequence of probe fouling is that the adsorbed solid can reduce the absorbance by the continuous phase due to a reduced measurement area on the ATR crystal. The reduced absorbance values would account for the values predicted by the PLS model showing the correct reaction profiles but having low predicted concentrations.

Despite the problems with the quantitative predictions described above, the real-time data still proved to be very useful during manufacture. With the exception of batch 3 (the bottom profile in Fig. 9b, marked with a dotted line), the qualitative shape of both the intermediate and product profiles for batches 2 through 5 showed good agreement with those observed in the laboratory and batch 1 in the large-scale reactor. It was noted at the time of manufacture that the profile for batch 3 suggested that the reaction was proceeding more slowly than was observed for previous batches. This prompted the operators to check the process conditions and identify that the temperature set-point was 10 °C lower than required. The inflection point in the profile at approximately 110 minutes corresponds to the time at which the operators corrected the temperature set-point. The appropriate time at which to take a sample for off-line analysis to confirm the end-point was also correctly identified from the reaction profiles. Further confidence was taken from the ability to see the appearance and disappearance of the intermediate species, as well as the appearance of the product.

CONCLUSION

In this paper, an example of deriving and implementing a PLS model for *in situ* reaction monitoring by UV spectroscopy was presented. It was shown that a good approximation of the true concentration profiles can be achieved using a small number of reference measurements to provide scaling information, combined with self-modeling curve resolution to reveal the underlying structure in the spectral data. Multivariate curve resolution–alternating least squares allowed this data to be combined and utilized to provide realistic constraints during the optimization steps.

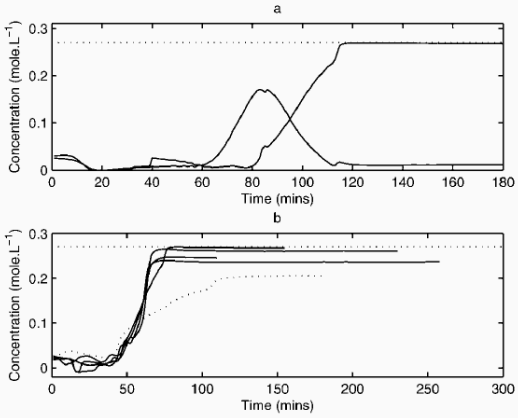


FIG. 9. (**a**) Predicted concentrations for batch 1, manufactured in a large-scale laboratory; (**b**) comparison of predicted product concentrations for five batches manufactured in a large-scale laboratory. The dashed line shows the theoretical end of reaction limit. The dotted line (···) is batch 2. The batches were aligned by setting the start of the phosphoryl chloride addition to t_0.

In addition to providing a concentration matrix, the spectral profiles obtained from curve resolution were also useful for confirming that the subsequent PLS model regression vectors were indeed using spectral features inherent in each species' pure spectrum.

The outcome of using the derived PLS model for predicting new batches was a mixed success. The predicted concentrations from batch 1 showed good agreement with the expected values and the measured reference value provided by HPLC. The predicted concentrations for the remaining batches were less accurate. However, the resulting qualitative reaction profiles all displayed the same characteristic shape and successfully indicated the reaction end-point. The ability to observe the appearance and disappearance of the intermediate species gave confidence when interpreting the qualitative profiles to judge the end-point. The apparent rate of intermediate and product formation also allowed the low set-point temperature of batch 3 to be identified. The temperature was corrected and the batch successfully reached completion in the expected time.

The major cause of inaccuracy in the predictions is believed to be a consequence of probe fouling or moving the fiber-optic cables after acquiring the initial background spectrum. In this particular application it was unavoidable, but in future applications, a dedicated port in the vessel lid for mounting the probe, combined with the use of a double-beam instrument, should help to minimize this. Improved agitation of the reaction mixture would help to minimize probe fouling.

ACKNOWLEDGMENTS

Dr. Mike Baker, Dr. Steve Eyley, and Kevin Sutcliffe (PR&D PAT group) are acknowledged for their help to implement and promote the use of process spectroscopy within AstraZeneca PR&D Development Manufacture. The

author would also like to thank Dr. Eric Merifield (PR&D Process Chemistry) for his helpful advice while performing the development reactions. Annabelle Germon (PR&D Analytical Chemistry) is thanked for providing the analytical reference data, and Simon Watkins (PR&D Development Manufacture) is gratefully acknowledged for helping to implement the use of the UV method during the LSL manufacture.

1. K. A. Bakeev, *Process Analytical Technology* (Blackwell Publishing Ltd., Oxford, UK, 2005).
2. J. M. Chalmers, *Spectroscopy in Process Analysis* (Sheffield Academic Press Ltd, Sheffield, UK, 2000).
3. J. Jerome Workman, M. Koch, and D. Veltkamp, Anal. Chem. **77,** 3789 (2005).
4. M. T. Riebe and D. J. Eustace, Anal. Chem. **62,** 65A (1990).
5. R. G. Brereton, Analyst (Cambridge, U.K.) **125,** 2125 (2000).
6. M. Kubista, J. Nygren, A. Elbergali, and R. Sjöback, Crit. Rev. Anal. Chem. **29,** 1 (1999).
7. R. M. Dyson, M. Hazenkamp, K. Kaufmann, M. Maeder, M. Studer, and A. Zilian, J. Chemom. **14,** 737 (2000).
8. P. Gemperline, G. Puxty, M. Maeder, D. Walker, F. Tarczynski, and M. Bosserman, Anal. Chem. **76,** 2575 (2004).
9. S. Richards, M. Ropic, D. Blackmond, and A. Walmsley, Anal. Chim. Acta **519,** 1 (2004).
10. B. Ma, P. J. Gemperline, E. Cash, and M. B. a. E. Comas, J. Chemom. **17,** 470 (2003).
11. Z.-L. Zhu, W.-Z. Cheng, and Y. Zhao, Chemom. Intell. Lab. Syst. **64,** 157 (2002).
12. A. d. Juan, M. Maeder, M. Martinez, and R. Tauler, Chemom. Intell. Lab. Syst. **54,** 123 (2000).
13. R. Tauler, B. Kowalski, and S. Fleming, Anal. Chem. **65,** 2040 (1993).
14. J.-H. Jiang, S. Sasic, R.-Q. Yu, and Y. Ozaki, J. Chemom. **17,** 186 (2003).
15. J.-H. Jiang, Y. Liang, and Y. Ozaki, Chemom. Intell. Lab. Syst. **71,** 1 (2004).
16. A. d. Juan and R. Tauler, Anal. Chim. Acta **500,** 195 (2003).
17. W. Chew, E. Widjaja, and M. Garland, Organometallics **21,** 1982 (2002).
18. E. Widjaja, C. Li, and M. Garland, Organometallics **21,** 1991 (2002).
19. A. R. Carvalho, J. Wattoom, L. Zhu, and R. G. Brereton, Analyst (Cambridge, U.K.) **131,** 90 (2006).
20. H. Gampp, M. Maeder, C. J. Meyer, and A. D. Zuberbühler, Talanta **32,** 1133 (1985).
21. M. Maeder and A. D. Zuberbuehler, Anal. Chim. Acta **181,** 287 (1986).
22. M. Maeder, Anal. Chem. **59,** 527 (1987).
23. M. Maeder and A. Zilian, Chemom. Intell. Lab. Syst. **3,** 205 (1988).
24. F. C. Sanchez, J. Toft, B. v. d. Bogaert, and D. L. Massart, Anal. Chem. **68,** 79 (1996).
25. B. G. M. Vandeginste, D. L. Massart, L. M. C. Buydens, S. D. Jong, P. J. Lewi, and J. Smeyers-Verbeke, *Handbook of Chemometrics and Qualimetrics: Part B* (Elsevier, Amsterdam, 1998).
26. R. Tauler, A. Smilde, and B. Kowalski, J. Chemom. **9,** 31 (1995).
27. R. Tauler, J. Chemom. **15,** 627 (2001).
28. R. G. Brereton, "Chemometrics", in *Data Analysis for the Laboratory and Chemical Plant* (John Wiley and Sons Ltd, Chichester, UK, 2003).
29. P. Gemperline, *Practical Guide to Chemometrics* (CRC Press, Boca Raton, FL, 2006), 2nd ed.
30. R. Kramer, *Chemometric Techniques for Quantitative Analysis* (Marcel Dekker, Inc., New York, 1998).
31. E. R. Malinowski, *Factor Analysis in Chemistry* (John Wiley and Sons, New York, 2002), 3rd ed.
32. T. Næs, T. Isaksson, T. Fearn, and T. Davies, *A User-Friendly Guide to Multivariate Calibration and Classification* (NIR Publications, Chichester, UK, 2002).
33. A. Smilde, R. Bro, and P. Geladi, *Multi-Way Analysis with Applications in the Chemical Sciences* (John Wiley and Sons Ltd, Chichester, UK, 2004).
34. J. Jaumot, R. Gargallo, A. d. Juan, and R. Tauler, Chemom. Intell. Lab. Syst. **76,** 101 (2005).
35. J. Tyrgg, J. Chemom. **18,** 166 (2004).

Appendix II

ResidualComps.m

```
function [RegenData]=ResidualComps(Data, maxcomps, fig_num, Xaxis, Xlabel, Ylabel)
% [RegenData] = ResidualComps(Data, maxcomps, fig_num, Xaxis, Xlabel, Ylabel)
%
% The function of this simple script is to perform PCA (specifically SVD)
% on a data set. Using the singular values and eigenvectors calculated,
% the data is re-constructed using an increasing number of components, up
% to a maximum number of principal components specified by the user.
% The main benefit of this function is that it will display the
% re-constructed data alongside the corresponding residual data calculated
% using a different number of principal components. This allows the
% contribution of each new principal component to be assessed visually.
% For each principal component, the residual sum-of-squares are
% calculated and plotted versus principal component number to provide
% a numerical indication of the information contributed by each
% additional component. The user can then specify the number of
% principal components to use in re-constructing the data and this reduced
% data is written to the MATLAB workspace. This is useful way to filter
% or clean up data prior to further analysis.
% The resulting figures are automatically saved as '.fig' and '.emf'
% files using the filename prefix provided by the user.
%
% INPUT ARGUMENTS
% [Data] is a 2-dimensional array of data. The dimensions of this data
% matrix are (I × J) where I is the number of observations (samples or
% time points) and J is the number of experimental variables (e.g. wavelengths).
%
% [maxcomps] is the maximum number of Principal Components to be used.
%
% [fig_num] is the figure filename prefix to store the resulting figures.
% For example, if the user enters 'figure6', the resulting figure will
% be saved as 'figure6.fig' and 'figure6.emf'.
%
% [Xaxis] is the vector containing the x-axis scale.
%
% [Xlabel] is a string variable to used as the x-axis label on the plots
% generated by the script, e.g. 'wavelength (nm)'.
%
% [Ylabel] is a string variable to used as the y-axis label on the plots
% generated by the script, e.g. 'Absorbance'.
%
% OUTPUT ARUGUMENTS
% [RegenData] is the reduced data set re-constructed using the number of
% principal components specified by the user during the execution of the
% function.
%
% Note, if the user selects maxcomps greater than 2, the plots will be
% split over a number of figures with the suffix a,b,c etc added to the
% filename.
%
```

```
% Nicholas I. Pedge
% Department of Chemistry, University of Hull, Cottingham Road, Hull,
% HU6 7RX
% Process R&D, AstraZeneca R&D Charnwood, Bakewell Road, Loughborough,
% Leicestershire, LE11 5RH
%
% [RegenData] = ResidualComps(Data, maxcomps, fig_num, X_axis, X_label, Y_label);
%
% CHECK INPUT ARGUMENTS

if nargin<6 || isempty(Ylabel)
        fprintf( '\n' )
        fprintf( 'Ylabel was not specified! \n' )
        fprintf( 'The default label "Value" will be used. \n' )
        Ylabel= 'Value' ;
end
%
if ischar(Ylabel)==0
        error( 'Please enter [Ylabel] as a string variable!' )
end
%
if nargin<5 || isempty(Xlabel)
        fprintf( '\n' )
        fprintf( 'Xlabel was not specified! \n' )
        fprintf( 'The default label "Variable" will be used. \n' )
        Xlabel= 'Variable' ;
end
%
if ischar(Xlabel)==0
        fprintf( '\n' )
        error( 'Please enter [Xlabel] as a string variable!' )
end
%
if ischar(Xaxis)==1
        fprintf( '\n' )
        error( 'Please enter [Xaxis] as a column vector!' )
end
%
if nargin<4 || isempty(Xaxis)
        Xaxis=(1:size(Data,2))';
end
%
if size(Xaxis,1) ~= size(Data,2)
        error( 'The dimensions of Xaxis do not match the data set' )
end
%
if nargin <3 || isempty(fig_num)
        fprintf( '\n' )
        fprintf( 'Figure output name not specified! \n' )
        fprintf( 'Using the default name "ResidualComps_Figure". \n' );
        fig_num= 'ResidualComps_Figure' ;
end
%
if ischar(fig_num)==0
        fprintf( '\n' )
        fprintf( 'Please enter [fig_num] as a string variable! \n' )
```

```
        fprintf( 'For example "Figure01". \n' )
        fprintf( 'Using the default name "ResidualComps_Figure". \n' );
        fig_num= 'ResidualComps_Figure' ;
end
%
if nargin < 2 || isempty(maxcomps)
        error( 'maxcomps variable was not provided!' )
end
%
nsubplots=2; % Variable specifying the number of PCs per figure
%
% Calculate the number of figures required
if maxcomps < nsubplots
        nfigs=1; % Only need 1 figure
else
        % Number of full figures required
        nfigs=floor(maxcomps/nsubplots);
        % Total number of figures required
        nfigstotal=ceil(maxcomps/nsubplots);
        % Number of subplots in final figure
        nplotsfinal=maxcomps-(nfigs*nsubplots);
end
%
SSQ=zeros(maxcomps,1);
%
disp([ 'This routine will generate ' ,num2str(nfigstotal+1), ' figures' ])
% char(97)= a
% char(98)= b
% char(99)= c etc
%
% Perform Singular Value Decomposition on the data
%
[U,S,V]=svd(Data);
%
% Perform Data regeneration and calculate the residuals
figurecounter=1;
charactercounter=0;
%
for n=1:nfigs
        % create figures with nsubplots Principal Components per plot,
        % and create final figure with nplotsfinal subplots
        %
        figure() % Open new figure
        j=0;
        %
        for i=(n*nsubplots)-nsubplots+1:1:(n*nsubplots)
                RegenData=U(:,1:i)*S(1:i,1:i)*V(:,1:i)';
                ResidData=Data-RegenData;
                SSQ(i)=trace(ResidData'*ResidData);
                j=j+1;
                subplot(nsubplots,2,j)
                plot(Xaxis,RegenData);
                text1=( ' PC: Regenerated Data' );
                figuretitle=num2str(i);
                figuretitle=[figuretitle, text1];
                title(figuretitle);
```

```
                xlabel(Xlabel)
                ylabel(Ylabel)
                axis tight
                %
                j=j+1;
                %
                subplot(nsubplots,2,j)
                plot(Xaxis, ResidData);
                text1=( ' PC: Residual Data' );
                figuretitle=num2str(i);
                figuretitle=[figuretitle, text1];
                title(figuretitle);
                xlabel(Xlabel)
                ylabel(Ylabel)
                axis tight
        end
        % Save the figures
        fprintf( '*** Saving figure %i *** \n' , figurecounter)
        saveas(gcf,[fig_num, char(97+charactercounter)], 'fig' )
        saveas(gcf,[fig_num, char(97+charactercounter)], 'emf' )
        %
        charactercounter = charactercounter +1;
        figurecounter=figurecounter+1;
end
% Now create last plot (if needed).
%
if nplotsfinal>0
        figure()
        n=n+1;
        j=0;
        for i=(n*nsubplots)-nsubplots+1:1:((n-1)*nsubplots)+nplotsfinal
                j=j+1;
                RegenData=U(:,1:i)*S(1:i,1:i)*V(:,1:i)';
                ResidData=Data-RegenData;
                %
                subplot(nplotsfinal,2,j)
                plot(Xaxis,RegenData);
                text1=( ' PC: Regenerated Data' );
                figuretitle=num2str(i);
                figuretitle=[figuretitle, text1];
                title(figuretitle);
                xlabel(Xlabel)
                ylabel(Ylabel)
                axis tight
                %
                j=j+1;
                subplot(nplotsfinal,2,j)
                plot(Xaxis, ResidData);
                text1=( ' PC: Residual Data' );
                figuretitle=num2str(i);
                figuretitle=[figuretitle, text1];
                title(figuretitle);
                xlabel(Xlabel)
                ylabel(Ylabel)
                axis tight
        end
```

```
        %
        % Save the figures
        fprintf( '*** Saving figure %i *** \n' , figurecounter)
        saveas(gcf,[fig_num, char(97+charactercounter)], 'fig' )
        saveas(gcf,[fig_num, char(97+charactercounter)], 'emf' )
        figurecounter=figurecounter+1;
        charactercounter = charactercounter +1;
end
%
figure()
subplot(2,1,1)
plot((1:1:maxcomps), SSQ, 'b+-' )
hold on
plot((1:1:maxcomps), SSQ, 'bO' )
xlabel( 'Principal Component' )
ylabel( 'Sum of squares' )
title( 'Residual sum of squares vs. Number of Principal Component ' )
subplot(2,1,2)
plot((1:1:maxcomps), log10(SSQ), 'b+-' )
hold on
plot((1:1:maxcomps), log10(SSQ), 'bO' )
xlabel( 'Principal Component' )
ylabel( 'log_1_0(Sum of squares)' )
title( 'Residual sum of squares vs. Number of Principal Component ' )
%
fprintf( '*** Saving figure %i *** \n' , figurecounter)
saveas(gcf,[fig_num, char(97+charactercounter)], 'fig' )
saveas(gcf,[fig_num, char(97+charactercounter)], 'emf' )
%
% Ask user to select the number of components to retain when regenerating
% the data.
ncomp=input( 'Enter the number of components you wish to retain: ' );
RegenData=U(:,1:ncomp)*S(1:ncomp,1:ncomp)*V(:,1:ncomp)';
%
if nargout < 1 | | isempty(RegenData)
        varname= 'RegenData' ;
        assignin( 'base' ,varname,RegenData)
end
end  % END OF FUNCTION
```

OPA.m

```
function [PureSpec, Dissim, DW, SI]=OPA(Data, N, plotting)
% function [PureSpec,Dissim, DW, SI]=OPA(Data,N,plotting)
%
% This function will perform Orthogonal Projection Analysis using the OPA
% algorthm described in Analytica Chimica Acta 519 (2004) 11-21,
% S.Gourvenec, X. Capron and D.L. Massart, and also Chemometrics and
% Intelligent Laboratory Systems, 61 (2002) 51-61, S. Gourvenec, D.L
% Massart and D.N Rutledge. This function also calculates the
% Durbin-Watson values to aid identification of the correct number of
% components to model the data. The function will not perform ALS to
% calculate the corresponding concentration profiles or optimise the
% spectral profiles, but will simply output the selected number of OPA
% spectra, the normalised dissimilarity profiles and the Durbin-Watson
% values.
%
% INPUT ARGUMENTS
% [Data] is a 2-dimensional array of evolutionary experimental data such
% as spectra acquired during a reaction monitoring experiment, HPLC-DAD
% or any other bilinear experimental data matrix. The dimensions of this
% data matrix are (I x J) where I is the number of observations (samples
% or time points) and J is the number of experimental variables
% (wavelengths, response variables). The data matrix should therefore be
% row orientated so that the OPA can be performed along the 'time'
% dimension.
%
% [N]is the number of OPA components to calculate. This number should
% be equal to or greater than the number of experimentally visible
% components that are contributing to the instrument response.
%
% [Plotting] allows the user to specifiy whether plotting is 'off'
% {plotting = 0} or 'on' {plotting =1}
%
% OUTPUT ARGUMENTS
%
% [PureSpec] is the matrix of pure component spectra located by
% OPA. The number of columns is equal to N, the number of individual
% chemical components whilst the number of rows is equal to J, the number
% of spectral variables.
%
% [Dissim] is a matrix containing the calculated Dissimilarity values for
% each OPA spectrum as it compared with the previously located reference
% spectra. As explained in the literature, the dissimilarity value is
% defined as the determinant of the dispersion matrix (YiYi'). This will
% be a matrix with N columns (one column for each OPA component) and I
% rows (one row for each observation or spectrum in the data matrix).
%
% [DW] is a column vector containing the Durbin-Watson values
% calculated for each new OPA component.
%
% [SI] is an (N x 2) matrix. The first column stores the index (spectrum)
% numbers of the OPA pure component spectra and the second column stores
% the corresponding dissimilarity values.
%
% Nicholas I. Pedge
```

```
% Department of Chemistry, University of Hull, Cottingham Road, Hull,
% HU6 7RX
% Process R&D, AstraZeneca R&D Charnwood, Bakewell Road, Loughborough,
% Leicestershire, LE11 5RH
%
% [PureSpec,Dissim,DW, SI]=OPA(Data,N,plotting);
% BASIC INPUT VARIABLES CHECK
%
% Set plotting to 'on' if user does not specify
if nargin < 3, plotting = 1; end
%
% Prompt user for N if the number of OPA components is not specified
if nargin <2 ||isempty(N)
        N = input( 'Please enter the number of components' );
end
%
% Determine the dimensions of the original data matrix [Data]
[I, J]=size(Data);
%
if I ==1 || J == 1
        fprintf( 'Error! OPA cannot be performed on a vector input. \n' )
        fprintf( 'Please provide a matrix as input variable [Data] \n' )
        return
end
%
% PRE-ALLOCATE STORAGE VECTORS AND MATRICES
%
% Prepare an output matrix for dissimilarity values [Dissim]
Dissim=zeros(I, N);
%
% Prepare a column vector to store the Durbin-Watson values.
DW=zeros(N,1);
DW(1)=0;
%
% Prepare a matrix to store the index of each OPA component and their
% corresponding dissimilarity values.
SI=zeros(N,2);
%
% Calculation of initial mean spectrum xs
xs = mean(Data);
%
% Normalise initial mean spectrum xs to unit length
PureSpec=xs./norm(xs,2);
%
% ORTHOGONAL PROJECTION ANALYSIS: MAIN LOOP
%
% Outer loop counts the number of OPA components calculated, max = N
%
tic % Start timer
h = waitbar(0, 'Please wait...Calculating OPA Components' );
%
for n=1:N
        waitbar(n/N, h);
        %
        % Inner loop counts the observation (spectrum) number, max = nmeas
        for i=1:I
```

```
                Yi=[PureSpec; Data(i,:)]';
                Dissim(i,n)=det(Yi'*Yi);
        end
        %
        % Identify the most dissimilar spectrum and store its index and
        % dissimilarity value
        [xs_max, xs_index]=max(Dissim(:,n));
        SI(n,1)=xs_index;
        SI(n,2)=xs_max;
        %
        % Select spectrum with largest dissim and normalise to length = 1
        xs=Data(xs_index,:);
        xs_norm=xs./norm(xs,2);
        %
        % Assign / concatenate spectrum to PureSpec matrix
        if n==1
                PureSpec(n,:)=xs_norm; % Replace mean spectrum if n = 1
        else
                PureSpec=[PureSpec;xs_norm]; % concatenate spectrum if n > 1
        end
        %
        % Calculation of Durbin-Watson value for the dissimilarity
        % vector of component n, Dissim(:,n).
        dw_diff(1:I,1)=0;
        %
        for i=2:I
                dw_diff(i)=Dissim(i,n)-Dissim(i-1,n);
        end
        dw_sum_sq=Dissim(:,n)'*Dissim(:,n);
        dw_diff_sum_sq=dw_diff'*dw_diff;
        DW(n)=dw_diff_sum_sq/dw_sum_sq;
end % End outer loop, n
close(h)
toc % End timer
%
% Transpose PureSpec
PureSpec = PureSpec';
%
% Plot outputs
if plotting ==1
        subplot(3,1,1)
        plot(PureSpec)
        axis tight
        title( 'Pure component spectral profiles calculated by OPA' );
        xlabel( 'Variable number' )
        Str1(1)={ 'Normalised' };
        Str1(2)={ 'Intensity' };
        ylabel(Str1) ;
        %
        subplot(3,1,2)
        plot(Dissim)
        title( 'Dissimilarity values vs. measurement number' );
        xlabel( 'Measurement / Spectrum number' );
        ylabel( 'Dissimilarity' );
        %
        subplot(3,1,3)
```

```
        plot(DW, 'marker' , 'O' ); hold on ;plot(DW, '+' )
        title( 'Durbin-Watson values vs. component number' );
        xlabel( 'Component number' ); ylabel( 'DW value' );
        drawnow
end
%
% User can now specify how many components to retain
user_N=input( 'Please enter the number of OPA components to retain: ' );
PureSpec(:,user_N+1:end)=[ ];    % Remove N+1 OPA components
Dissim(:,user_N+1:end)=[ ];      % Remove N+1 dissimilarity profiles
SI(user_N+1:end,:)=[ ];  % Remove N+1 spectrum indices and dissimilarity values
% Normalise dissimilarity values to make profiles easier to see on plot
Dissim_Norm = Dissim ./ repmat((max(Dissim)),I,1);
%
figure()
subplot(3,1,1)
plot(PureSpec)
axis tight
title( 'Pure component spectral profiles calculated by OPA' );
xlabel( 'Variable number' );
Str1(1)={ 'Normalised' };
Str1(2)={ 'Intensity' };
ylabel(Str1);
%
subplot(3,1,2)
plot(Dissim_Norm)
axis([1 I 0 1.1]);
title( 'Normalised dissimilarity values vs. measurement number' );
xlabel( 'Measurement / Spectrum number' );
Str2(1)={ 'Normalised' }; Str2(2)={ 'Dissimilarity' };
ylabel(Str2);
%
subplot(3,1,3)
plot(DW, 'marker' , 'O' ); hold on ;plot(DW, '+' )
title( 'Durbin-Watson values vs. component number' );
xlabel( 'Component number' ); ylabel( 'DW value' );
drawnow
end % END OF FUNCTION
```

MedianFilter.m

```
function [Filtered, Residual]=MedianFilter(Data, WS, WE, step, NN, plotting);
% [Filtered, Residual]=MedianFilter(Data, WS, WE, step, NN, plotting);
%
% This function will perform median filtering using a moving window and is
% based upon the algorithm published by A. W. Moore and J. W. Jorgenson,
% Median filtering for removal of low-frequency background drift,
% Analytical Chemistry, 1993, 65, p188-191.
%
% Median filtering can be applied to broad spectral data to such UV spectra
% to remove high frequency 'spikes' using a small window of 3 or 5. Median
% filtering can also be used to remove low frequency baseline contributions
% from FTIR or Raman spectra or chromatograms using a larger window size.
%
% This function also permits the use of an increasingly larger (or
% decreasingly smaller) window. This option is for testing purposes
% only as the result will be the same as using a fixed window at the final
% window size. A single spectrum must be used for this test as the filtered
% spectra obtained using the different window sizes are output to the
% matrices [Filtered] and [Residual].
%
% INPUT ARGUMENTS
% [Data] is a 1- or 2-dimensional array of data. The dimension of this
% data matrix are (J x K), where J is the number of observations (1 for a
% single spectrum or chromatogram) and K is the number of experimental
% variables (e.g. wavelengths or retention times).
%
% [WS] is the initial window size to be used for median filtering. [WS]
% must be odd-numbered. The user can specify a fixed window size by only
% entering a value for [WS].
%
% [WE] is an optional input argument that allows the user to specify a
% final window size when assessing a range of windows sizes for median
% filtering. [WE] must be odd-numbered. This input argument can be omitted
% if filtering using fixed window size is required.
%
% [step] is an optional input argument used with [WE] and allows the user
% to specify the increment of the window size. This value must be
% even-numbered. If [WE] and [step] are not provided, a fixed-size window
% defined by [WS] will be used during filtering.
%
% [NN] is an optional input argument that allows the user to specify
% whether a non-negativity correction should be applied to the filtered
% data. Non-negativity is 'off' when {NN=0} and 'on' when {NN=1}.
%
% [plotting] specifies whether the function will display plots during
% run-time. Plotting is 'off' when {plotting=0} and 'on' when {plotting =1}
%
% OUTPUT ARGUMENTS
% During normal use:
% [Filtered] is a (J x K) matrix of signal component subtracted by the
% process of median filtering. If the original signal is broad and a small
% window size has been applied to remove high frequency 'spikes',
% [Filtered] will be the high-frequency part of the signal. If the signal
% contains high frequency peaks, such as FTIR or Raman spectra, and a large
```

```
% window size has been applied to remove a broad baseline contribution,
% [Filtered] will comprise of the desired Raman signal.
%
% [Residual] is a (J x K) matrix of the residual signal component after
% subtraction of the median filtered signal from the original data. If the
% original signal is broad and a small window size has been applied to
 % remove high frequency 'spikes', [Residual] will be the desired low frequency
% part of the signal. If the signal contains high frequency peaks such as
% FTIR or Raman spectra, and a large window size has been applied to remove
% a broad baseline contribution, [Residual] will be the underlying
% baseline.
%
% During testing:
% [Filtered] and [Residual] are (n x K) matrices where n is the number of
% increments defined by the values of [WS], [WE] and [step].
%
% Nicholas I. Pedge
% Department of Chemistry, Univeristy of Hull, Cottingham Road, Hull,
% HU6 7RX
% Process R&D, AstraZeneca R&D Charnwood, Bakewell Road, Loughborough,
% Leicetershire, LE11 5RH.
%
% [Filtered, Residual]=MedianFilter(Data, WS, WE, step, NN, plotting);

% BASIC INPUT VARIABLES CHECK
if nargin<6||isempty(plotting)==1
        fprintf('Plotting will be set to "Off" \n')
        plotting=0;
end
%
if nargin<5||isempty(NN)==1
        fprintf('Non-negativity will be set to "Off" \n')
        NN=0;
end
%
if nargin <4 || isempty(step)==1
        fprintf('Step value not entered...default value [2] will be used \n')
        step = 2;
end
%
if nargin <3 || isempty(WE)==1
        WE=WS;
        step=2;
        fprintf('WE = WS, no incremental filtering \n')
end
%
if nargin<2
        error('Minimum inputs are [Data] and [WS]')
end
%
if WS==WE
        IncFilter=0;
        fprintf('WE = WS, no incremental filtering \n')
        step=2;
else
        IncFilter=1;
```

```
end
%
if WS<3 || WE<3
        error('The minimum for [WS] and [WE] is 3')
end
% Check if input arguments [WS] and [WE] are odd-numbered.
if rem(WS,2)==0 || rem(WE,2)==0
        error('Input arguments [WS] and [WE] must be odd-numbered')
end
% Check if input argument [step] is even-numbered.
if rem(step,2)==1
        error('Input argument [step] must be even-numbered')
end
%
% Check that if the incremental window size is on, the input argument
% [Data] is a vector.
[nrows, ncols]=size(Data);
if IncFilter == 1
        if nrows ~=1
                error('The input argument [Data] must be a vector when', ...
                        ' used with [WE] for incremental filtering')
        end
end
%
% CHECK VALUES OF [WS], [WE] AND [step]
% Ensure that the difference between WE and WS is an integer
% number of steps sizes.
%
% Forward direction; WE is larger than WS
if WS < WE
        win_diff=WE - WS;
        n = floor(win_diff / abs(step));
        %
        % Update WE to be an integer number of step sizes from WS
        WE=WS + (n*step);
        step=(abs(step));
        fprintf('WE has been updated to match step size \n')
        fprintf('WE value has been set to %g \n',WE)
else
        win_diff=WS-WE;
        n = floor(win_diff / abs(step));
        % Update WS to be an integer number of step sizes from WE
        WS=WE + (n*abs(step));
        step=abs(step)*-1;
        fprintf('WS has been updated to match step size \n')
        fprintf('WS value has been set to %g \n',WS)
end
%
% PRE-ALLOCATE STORAGE VECTORS AND MATRICES
%
% Determine size of original data matrix
[nrows, ncols]=size(Data);
%
if IncFilter==0
        Filtered=zeros(nrows, ncols);
        Residual=zeros(nrows, ncols);
```

```
else
        Filtered=zeros(n+1, ncols);
        Residual=zeros(n+1, ncols);
end
%
%
% PERFORM MOVING MEDIAN FILTERING
%
tic
index=1;
h=waitbar(0, 'Please wait...' );
%
% THIS SECTION IS USED IF INCREMENTAL WINDOW SIZE IS SET TO ON
%
if IncFilter==1
        %
        for w=WS:step:WE
                %
                waitbar(index/n, h);
                %
                x_win=(w-1)/2; % number of points either side of window centre-point
                %
                X=Data;
                %
                % To ensure smooth filtering, the ends of the spectral data must be
                % padded with additional columns. The additional columns are a mirror
                % imag e of the first x_win or final x_win columns of the data.
                %
                Xpad=[(fliplr(X(:,1:x_win))),X,(fliplr(X(:,(end-x_win+1):end)))];
                ResidualTemp=zeros(1, ncols+(2*x_win));
                %
                for k=(1+x_win):(ncols+x_win);
                        Subset=Xpad(:,(k-x_win):(k+x_win));
                        Subset=sort(Subset);
                        ResidualTemp(k)=Subset(:,x_win+1);
                end
                %
                % Remove padded columns before updating residuals
                Residual(index,:)=ResidualTemp(:,1+x_win:end-x_win);
                Xpad=Xpad(:,1+x_win:end-x_win);
                Filtered(index,:)=Xpad-Residual(index,:);
                % Find and remove negative values in filtered data
                if NN==1
                        Residual(index,find(Filtered(index,:)<0))= ...
                                Residual(index,find(Filtered(index,:)<0)) ...
                                + Filtered(index,find(Filtered(index,:)<0));
                        Filtered(index,:)=Xpad-Residual(index,:);
                end
                %
                if plotting == 1
                        clf;
                        subplot(3,1,1);
                        plot(Data', 'linewidth' ,2);
                        title( 'O riginal data' )
                        xlabel( 'Variable number' )
                        ylabel( 'Intensity (Arb.)' )
```

```
                    subplot(3,1,2);
                    plot(Filtered(1:index,:)'); hold on ;
                    plot(Filtered(index,:), 'r' , 'linewidth' ,2)
                    title( 'Median filtered data' )
                    xlabel( 'Variable number' )
                    ylabel( 'Intensity (Arb.)' )
                    subplot(3,1,3);
                     plot(Residual(1:index,:)');hold on ;
                    plot(Residual(index,:), 'r' , 'linewidth' ,2)
                    title( 'Residual: Original data - filtered data' )
                    xlabel( 'Variable number' )
                    ylabel( 'Intensity (Arb.)' )
                    drawnow
                end
                %
                index = index +1;
        end
        close(h)
end
%
if IncFilter==0
        % THIS SECTION IS USED IF INCREMENTAL WINDOW SIZE IS OFF
        x_win=(WS-1)/2; % number of points either side of window centre-point
        %
        X=Data;
        %
        % To ensure smooth filtering, the ends of the spectral data must be
        % padded with additional columns. The additional columns are a mirror
        % image of the first x_win or final x_win columns of the data.
        %
        Xpad=[(fliplr(X(:,1:x_win))),X,(fliplr(X(:,(end-x_win+1):end)))];
        Residual=zeros(nrows, ncols+(2*x_win));
        %
        for k=(1+x_win):(ncols+x_win);
                waitbar(k/ncols, h)
                Subset=Xpad(:,(k-x_win):(k+x_win));
                Subset=sort(Subset');
                Residual(:,k)=Subset(x_win+1,:)';
        end
        close(h)
        %
        % Remove padded columns before updating residuals
        Residual=Residual(:,1+x_win:end-x_win);
        Xpad=Xpad(:,1+x_win:end-x_win);
        Filtered=Xpad-Residual;
        %
        % Find and remove negative values in filtered data
        if NN==1
                Residual(find(Filtered<0))= Residual(find(Filtered<0)) ...
                        + Filtered(find(Filtered<0));
                Filtered=Xpad-Residual;
        end
end
%
subplot(3,1,1);
plot(Data'); axis tight
```

```
title( 'Original data' )
xlabel( 'Variable number' )
ylabel( 'Intensity (Arb.)' )
subplot(3,1,2);
plot(Filtered'); axis tight
title( 'Median filtered data' )
xlabel( 'Variable number' )
ylabel( 'Intensity (Arb.)' )
subplot(3,1,3);
plot(Residual'); axis tight
title( 'Residual: Original data - filtered data' )
xlabel( 'Variable number' )
ylabel( 'Intensity (Arb.)' )
%
toc
end % END OF FUNCTION
```

LMJ_IPBS.m

```
function IPBS_OUTPUT = LMJ_IPBS(D, n, Tol, MaxIterations, PlotOn)
% IPBS_OUTPUT = LMJ_IPBS(D, n, Tol, MaxIterations, PlotOn)
%
% This function is the iterative background removal algorithm published in
% Applied Spectroscopy, Vol 57, Number 11, (2003) p1363 by Chad A. Lieber
% and Anita Mahadevan-Jansen. The function fits an n degree polynomial to
% the whole spectrum and iteratively modifies the estimated background
% spectrum until the convergence tolerance or maximum number of iterations
% 'MaxIterations' is reached.
%
% INPUT ARGUMENTS
%
% [D] is a (J x K) matrix or (1 x K) vector containing the spectra or
% spectrum to be processed.
%
% [n] is the degree of the polynomial function to be used when fitting the data.
% Minimum value is n=1, maximum value is n = 10, default value is n=4.
%
% [Tol] is the tolerance to be used when iteratively refining the estimates
% of the polynomial baseline. Warning: if the value of [Tol] is too small,
% the baseline fitting function may never reach convergence.
% The default value is 1E-03. Convergence is calculated as the
% sum-of-squares residual between two consecutive estimates of the baseline
% function.
%
% [MaxIterations] is the maximum number of iterations the function is
% permitted to perform if it does not reach the convergence tolerance
% first.
%
% [PlotOn] allows the user to specify whether plotting during runtime is 'off'
% {PlotOn = 0} or 'on' {PlotOn =1}.
%
% OUTPUT ARGUMENTS
%
% [IPBS_OUTPUT] is a structured array generated by this script to store the
% various outputs listed below.
%
% [.CorrSpec] is a (J x K) matrix or (1 x K) vector containing the
% processed spectra after subtraction of the baseline.
%
% [.Baseline] is a (J x K) matrix or (1 x K) vector containing the fitted
% baselines subtracted from the original data
%
% Nicholas I. Pedge
% Department of Chemistry, University of Hull, Cottingham Road, Hull,
% HU6 7RX
% Process R&D, AstraZeneca R&D Charnwood, Bakewell Road, Loughborough,
% Leicestershire, LE11 5RH

% CHECK INPUT ARGUMENTS
if nargin<5
        PlotOn=0;
end
%
```

```
if nargin <4 | | isempty(MaxIterations)
        MaxIterations=100; % Default maximum number of iterations
end
%
if nargin <3 | | isempty(Tol)
        Tol=1E-2; % Default tolerance
end
%
if nargin < 2 | | isempty(n)
 n = 4; % Default order of polynomial to be fitted
end
%
if n>10 | | n<1
        error( 'The value of [n] should be between 1 and 10' )
end
%
[J,K]=size(D);
%
% Display parameters to user
fprintf( 'Order of polynomial function: %g \n' ,n)
fprintf( 'Convergence tolerance: %e \n' ,Tol)
%
% Dimension temporary storage matrices
Baseline=zeros(J,K);
%
tic
warning off all
%
h1=gcf;
if PlotOn==1
        figure(); h2=gcf;
end
%
hw=waitbar(0, 'Performing IPBS calculations...please wait' );
%
for j=1:J % Outer loop to process each row (spectrum) in D
        %
        Convergence=1000; % Reset initial convergence
        Counter=1; % Reset counter (iterations) to 1
        Dj=zeros(1,K); % Temporary vector to store current spectrum
        EBG_new=zeros(1,K); % Initialise vector to store estimate of baseline
        EBG_old=D(j,:); % Initial estimate of the baseline
        %
        while Convergence > Tol % Loop to check convergence of baseline
                if J>1
                        waitbar(j/J, hw);
                else
                        waitbar(Counter/MaxIterations, hw);
                end
                %
                M=zeros(K,n+1);
                x=1:1:K;
                M(:,1)=1;
                %
                for p=1:n
                        M(:,p+1)=x.^p;
```

```
        end
        %
        % To obtain an initial estimate of the baseline, a
% polynomial function will be fitted to original data.
% The estimated baseline function will then be refined
        % during each successive iterative cycle.
        %
        if Counter==1
                Dj=D(j,:); % Use original data for first estimate
        else
                Dj=Baseline(j,:); % Refine baseline estimate
        end
        %
        % Calculate polynomial coefficients
        A=M\Dj';
        % Estimate background using calculated polynomial coefficients
        EBG=(M*A)';
        %
        % Find values in EBG that are larger than original signal
        Diff=Dj-EBG;
        z=find(Diff<0);
        EBG(z)= Dj(z); % Check EBG is not larger than data
        z=find(EBG<0);
        EBG(z)= Dj(z); % Check for negativity in EBG
        %
        EBG_new(1,x)=EBG;
        %
        %
        % Check for convergence
        EBG_Diff = EBG_new-EBG_old;
        EBG_DiffSSQ = trace(EBG_Diff'*EBG_Diff);
        EBG_newSSQ = trace(EBG_new'*EBG_new);
        EBG_oldSSQ = trace(EBG_old'*EBG_old);
        Convergence = EBG_DiffSSQ ./ EBG_oldSSQ;
        %
        if PlotOn==1
                clf(h2)
                plot(D(j,:)', 'k' ); hold on
                plot(EBG_new', 'r' );
                plot((D(j,:)-EBG_new), 'b' );
                axis tight
                xlabel( 'Variable number' )
                ylabel( 'Intensity (Arb.)' )
                drawnow
        end
        % Rename baseline estimate before next iteration
        Baseline(j,:) = EBG_new;
        EBG_old=EBG_new;
        %
        Counter=Counter+1;
        %
        if Counter == MaxIterations
                break
        end
        %
end     % end loop for convergence check
```

```
end       % end loop for j
%
close(hw)
if PlotOn==1
          close(h2)
end
%
fprintf( '\nIterative Polynomial Baseline Subtraction complete! \n' )
toc % Stop timer
%
CorrSpec=D-Baseline;
%
 if PlotOn==1
          figure()
          subplot(2,1,1)
          plot(CorrSpec', 'b' ); axis tight
          title( 'Baseline subtracted spectra' , 'fontweight' , 'bold' )
          xlabel( 'Variable number' )
          ylabel( 'Intensity (Arb.)' )
          subplot(2,1,2)
          plot(Baseline', 'r' ); hold on
          plot(D', 'k' ); axis tight
          title( 'Subtracted baselines' , 'fontweight' , 'bold' )
          xlabel( 'Variable number' )
          ylabel( 'Intensity (Arb.)' )
end
%
IPBS_OUTPUT.CorrSpec=CorrSpec;
IPBS_OUTPUT.Baseline=Baseline;
IPBS_OUTPUT.Parameters.n=n;
IPBS_OUTPUT.Parameters.Tol=Tol;
%
end
% END OF FUNCTION
```

IPBS.m

```
function IPBS_OUTPUT = IPBS(D, n, Tol, win, MFwin, OLP, PlotOn)
% IPBS_OUTPUT = IPBS(D, n, Tol, win, MFwin, OLP, PlotOn)
%
% This function is based upon the iterative background removal algorithm
% published in Applied Spectroscopy, Vol 57, Number 11, (2003) p1363 by
% Chad A. Lieber and Anita Mahadevan-Jansen. It has been adapted to use
% an initial estimate of the underlying baseline function calculated using
% a moving window median filter (MedianFilter.m). The data is then divided
% into a series of windows of width [win] and a polynomial function of
% degree [n] is iteratively fitted to each window. The overlap parameter
% [OLP] allows the user to specify what fraction of the fitted polynomial
% from the previous window is added to the current window. This feature
% eliminates any discontinuities that could appear if the baseline is fitted
% as a series of discrete windows. The whole spectrum is fitted once, and
% then the convergence calculations are performed, rather than refine
% each window to convergence before fitting the next window.
%
% INPUT ARGUMENTS
%
% [D] is a (J x K) matrix or (1 x K) vector containing the spectra or
% spectrum to be processed.
%
% [n] is the degree of the polynomial function to be used when fitting the data.
% Minimum value is n=1, maximum value is n = 10, default value is n=4.
%
% [Tol] is the tolerance to be used when iteratively refining the estimates
% of the polynomial baseline. Warning: if the value of [Tol] is too small,
% the baseline fitting function may never reach convergence.
% The default value is 1E-03. Convergence is calculated as the
% sum-of-squares residual between two consecutive estimates of the baseline
% function.
%
% [win] is the window size to be used when fitting the polynomial
% baseline. This value may be an odd- or even-number.
%
% [MFwin] is the window size to be used when applying median filtering to
% obtain an initial estimate of the underlying baseline function. This
% number must be an odd-number.
%
% [OLP] is the overlap parameter and defines the fraction of the data in
% the previous window to be added to the current window. The total size of
% the window used to fit the polynomial function then becomes
% [win] + [win]*[OLP]. The value of [OLP] must be in the range 0.00 to
% 1.00. Note, a value of 1.00 is equivalent to doubling the size of [win].
% The default value is 0.20.
%
% [PlotOn] allows the user to specify whether plotting during runtime is 'off'
% {PlotOn = 0} or 'on' {PlotOn =1}.
%
% OUTPUT ARGUMENTS
%
% [IPBS_OUTPUT] is a structured array generated by this script to store the
% various outputs listed below.
%
% [.CorrSpec] is a (J x K) matrix or (1 x K) vector containing the
```

```
% processed spectra after subtraction of the baseline.
%
% [.Baseline] is a (J x K) matrix or (1 x K) vector containing the fitted
% baselines subtracted from the original data
%
% [.MF_Residual] is a (J x K) matrix or (1 x K) vector containing the
% initial estimates of the baselines provided by median filtering of the
% unprocessed spectra.
%
% [.MF_Filtered] is a (J x K) matrix or (1 x K) vector containing the
% processed spectra after median filtering.
%
% [.MF_CorrSpec] is a (J x K) matrix or (1 x K) vector containing the
% processed spectra after subtraction of a polynomial baseline estimated
% during the first iteration. This allows the user to compare the spectra
% obtained by subtraction of a polynomial baseline with the spectra
% obtained by median filtering.
%
% [.MF_Baseline] is a (J x K) matrix or (1 x K) vector containing the
% polynomial baselines estimated suring the first iteration. This allows
% the user to compare the baselines calculated by polynomial fitting with
% the baselines obtained by median filtering.
%
% Nicholas I. Pedge
% Department of Chemistry, University of Hull, Cottingham Road, Hull,
% HU6 7RX
% Process R&D, AstraZeneca R&D Charnwood, Bakewell Road, Loughborough,
% Leicestershire, LE11 5RH

% CHECK INPUT ARGUMENTS
if nargin<7
        PlotOn=0;
end
%
if nargin<6 || isempty(OLP)
        OLP=0.20;
end
%
if OLP<0 || OLP>1
        error( 'The value of [OLP] should be between 0 and 1' )
end
%
if nargin <5 || isempty(MFwin)
        MFwin=ceil(sqrt(size(D,2))*2);
end
%
if nargin <4 || isempty(win)
        win=ceil(sqrt(size(D,2)));
end
%
if nargin <3 || isempty(Tol)
        Tol=1E-2; % Default number of iterations
end
%
if nargin < 2 || isempty(n)
        n = 4; % Default order of polynomial to be fitted
```

```
end
%
if n>10 || n<1
        error('The value of [n] should be between 1 and 10' )
end
%
[J,K]=size(D);
%
% Calculate size of window overlap.
OL=floor(win*OLP);
%
% Display parameters to user
fprintf('Window size: %g \n' ,win)
fprintf('Median filter window size: %g \n' ,MFwin)
fprintf('Order of polynomial function: %g \n' ,n)
fprintf('Convergence tolerance: %e \n' ,Tol)
fprintf('Overlap window size factor: %g \n' ,OLP)
fprintf('Overlap window: %g \n' ,OL)
%
% This algorithm splits the data into a series of windows with window size
% defined by the user. The iterative algorithm is then applied to each
% window to find the polynomial that best describes the underlying
% baseline.
%
% Calculate total number of windows
nwin=ceil(K/win);
% Calculate size of last window
LastWin=K-((nwin-1)*win);
if LastWin > 0
        lwi=((nwin-1)*win)+1; % Index of the start point for the last window
        lws=K-((nwin-1)*win); % Size of the last window
else
        nwin=nwin+1;
end
%
% Dimension temporary storage matrices
Baseline=zeros(J,K);
MF_Baseline=zeros(J,K);
MF_CorrSpec=zeros(J,K);
%
tic
warning off all
MaxIterations=50;
%
% Apply median filtering to obtain initial estimate of the baseline
fprintf( 'Applying median filtering...please wait \n' )
[MF_Filtered, MF_Residual]=MedianFilter(D, MFwin, [], [], [], 1);
fprintf( 'Median filtering complete. \n' )
%
h1=gcf;
if PlotOn==1
        figure(); h2=gcf;
end
%
hw=waitbar(0, 'Performing IPBS calculations...please wait' );
%
```

```
for j=1:J % Outer loop to process each row (spectrum) in D
        %
        Convergence=1000;               % Reset initial convergence
        Counter=1;                      % Reset counter (iterations) to 1
        Dj=zeros(1,K);          % Temporary vector to store current spectrum
        EBG_new=zeros(1,K);  % Initialise vector to store estimate of baseline
        EBG_old=MF_Residual(j,:);       % Initial estimate of the baseline
        %
        while Convergence > Tol         % Loop to check convergence of baseline
                if J>1
                        waitbar(j/J, hw);
                else
                        waitbar(Counter/MaxIterations, hw);
                end
                %
                WinCounter=1;
                for w=1:win:((nwin-1)*win)
                        if WinCounter==1
                                x=1:1:(1+win-1); x=x';
                                M=zeros(win,n+1);
                        else
                                x=w-OL:1:(w+win-1); x=x';
                                M=zeros(win+OL,n+1);
                        end
                        %
                        M(:,1)=1;
                        %
                        for p=1:n
                                M(:,p+1)=x.^p;
                        end
                        %
                        % To a obtain good initial estimate of the baseline, a
                        % polynomial function will first be fitted to the estimate
                        % of the underlying baseline obtained by median filtering
                        % the data. Copy the current window from the median
                        % filtered data to Dj
                        %
                        if Counter==1
                                Dj=MF_Residual(j,:);    % Use MF baseline estimate
                        else
                                Dj=Baseline(j,:);       % Refine baseline estimate
                        end
                        %
                        if WinCounter==1
                                Dw=zeros(1,win);
                                Dw(1,1:win)=Dj(1,1:win);
                                Ds=D(j,1:win);
                        else
                                Dw=zeros(1,win+OL);
                                Dw(1,1+OL:win+OL)=Dj(1, w:w+win-1); % Data
                                Dw(1,1:OL)=EBG_new(1,w-OL:w-1); % Previous window
                                Ds=D(j,x);
                        end
                        %
                        % Calculate polynomial coefficients
                        A=M\Dw';
```

```
        % Estimate background using calculated polynomial coefficients
        EBG=(M*A)';
        %
        % Find values in EBG that are larger than original signal
        Diff=Ds-EBG;
        z=find(Diff<0);
        EBG(z)= Ds(z);          % Check EBG is not larger than data
        z=find(EBG<0);
        EBG(z)= Ds(z);          % Check for negativity in EBG
        %
        EBG_new(1,x)=EBG;
        WinCounter=WinCounter+1;
        %
end % end loop w=1:win:((nwin-1)*win)
%
% Fit final window if necessary.
if LastWin>0
        x=lwi-OL:1:(lwi+lws-1); x=x';
        M=zeros(lws+OL,n+1);
        % M=zeros(lws+OL+1,n+1);
        M(:,1)=1;
        for p=1:n
                M(:,p+1)=x.^p;
        end
        %
        if Counter==1
                Dj=MF_Residual(j,:);    % Use MF baseline estimate
        else
                Dj=Baseline(j,:);       % Refine baseline estimate
        end
        %
        Dw=zeros(1,lws+OL);
        Dw(1,1+OL:lws+OL)=Dj(1, lwi:lwi+lws-1); % Data
        Dw(1,1:OL)=EBG_new(1,lwi-OL:lwi-1); % Previous window
        Ds=D(j,x);
        %
        % Calculate polynomial coefficients for spectrum j, window w
        A=M\Dw';
        % Estimate background (EBG) using calculated polynomial coefficients
        EBG=(M*A)';
        %
        % Find values in EBG that are larger than original signal
        Diff=Ds-EBG;
        z=find(Diff<0);
        EBG(z)= Ds(z);          % Check EBG is not larger than data
        z=find(EBG<0);
        EBG(z)= Ds(z);          % Check for negativity in EBG
        EBG_new(1,x)=EBG;
end
%
% Check for convergence
EBG_Diff = EBG_new-EBG_old;
EBG_DiffSSQ = trace(EBG_Diff'*EBG_Diff);
EBG_newSSQ = trace(EBG_new'*EBG_new);
EBG_oldSSQ = trace(EBG_old'*EBG_old);
Convergence = EBG_DiffSSQ ./ EBG_oldSSQ;
```

```
                %
                if PlotOn==1
                        clf(h2)
                        plot(D(j,:)', 'k' ); hold on
                        plot(EBG_new', 'r' );
                        plot((D(j,:)-EBG_new), 'b' );
                        axis tight
                        xlabel( 'Variable number' )
                        ylabel( 'Intensity (Arb.)' )
                        drawnow
                end
                % Smooth estimate before next iteration
                Baseline(j,:) = EBG_new;
                EBG_old=EBG_new;
                %
                if Counter==1
                        MF_Baseline(j,:)=Baseline(j,:);
                        MF_CorrSpec(j,:)=D(J,:)-Baseline(j,:);
                end
                %
                Counter=Counter+1;
                %
                if Counter == MaxIterations
                        break
                end
                %
        end % end loop for convergence check
end % end loop for j
%
close(hw)
if PlotOn==1
        close(h2)
end
%
fprintf( '\n Iterative Polynomial Baseline Subtraction complete! \n' )
toc % Stop timer
%
CorrSpec=D-Baseline;
%
if PlotOn==1
        figure()
        subplot(2,1,1)
        plot(CorrSpec', 'b' ); axis tight
        title( 'Baseline subtracted spectra' , 'fontweight' , 'bold' )
        xlabel( 'Variable number' )
        ylabel( 'Intensity (Arb.)' )
        subplot(2,1,2)
        plot(Baseline', 'r' ); hold on
        plot(D', 'k' ); axis tight
        title( 'Subtracted baselines' , 'fontweight' , 'bold' )
        xlabel( 'Variable number' )
        ylabel( 'Intensity (Arb.)' )
end
%
IPBS_OUTPUT.CorrSpec=CorrSpec;
IPBS_OUTPUT.Baseline=Baseline;
```

```
IPBS_OUTPUT.MF_Residual=MF_Residual;
IPBS_OUTPUT.MF_Filtered=MF_Filtered;
IPBS_OUTPUT.MF_CorrSpec=MF_CorrSpec;
IPBS_OUTPUT.MF_Baseline=MF_Baseline;
IPBS_OUTPUT.Parameters.n=n;
IPBS_OUTPUT.Parameters.Tol=Tol;
IPBS_OUTPUT.Parameters.win=win;
IPBS_OUTPUT.Parameters.MFwin=MFwin;
IPBS_OUTPUT.Parameters.OLP=OLP;
IPBS_OUTPUT.Parameters.OL=OL;
%
end % END OF FUNCTION
```

VVSP.m

```
function VVSP_output = VVSP(X, NL, NR, p_norm, plotting)
% VVSP_output = VVSP(X, NL, NR, p_norm, plotting)
%
% This function will perform Vertex Vector Sequential Projection analysis
% on a two-dimensional data array using the algorithm described in "Vertex
% Vectors Sequential Projection for self-modelling curve resolution of
% two-way data", Zhi-Guo Wang et al., Chemometrics and Intelligent
% Laboratory Systems, 82 (2006) 154-164.
% The function is based upon the principle that after p-normalisation (p>1),
% all points in a two-way data matrix lie on a polyhedral hyper-"spherical"
% surface, with the pure variables (spectra) forming the vertices (vertex
% vectors). A certain quadratic expression f(w)= yj'.A.(yj')'is maximised
% at those spectra that form the vertex vectors, allowing the closest
% estimates of the pure component spectral profiles to be located.
% To aid the selection of the number of components to retain the
% Durbin-Watson values (a measure of autocorrelation) are calculated using
% the vector of quadratic values fw used to locate each successive VVSP
% component. The log10 of the sum of projection residuals is also
% calculated for each component.
% The measurement data X is assumed to have a bilinear model X=CS'+E. This
% function does not apply constrained alternating least squares to refine
% the C or S'.
%
% INPUT ARGUMENTS
%
% [X] is the row-orientated 2-way data matrix with dimensions (J x K) where
% J is the number of sample or observations and K is the number of
% variables.
%
% [NL] is the number of VVSP pure component profiles to initially locate.
%
% [NR] is the number of VVSP pure component profiles to retain.
%
% [p_norm] is the type of spectral normalisation to apply during VVSP
% analysis. Acceptable values are;
% [p_norm] = 2, Normalise to unit length
% [p_norm] = inf, Normalise to maximum value = 1 (normalise height)
% [p_norm] may also have values of 3, 4, 5 but this is uncommon.
%
% [plotting] allows the user to specify whether plotting is 'off'
% {plotting = 0} or 'on' {plotting =1}
%
% OUTPUT ARGUMENTS
%
% [VVSP_output] is a structured array containing the following output
% variables;
%
% [.X] is the workspace variable name of data matrix that VVSP was
% applied to.
%
% [.p_norm] is the value of p-normalisation applied to the original spectra.
%
% [.Sopt] is a (K x NR) matrix of VVSP pure component spectral profiles.
%
% [.Copt] is a (J x NR) matrix of concentration profiles estimated by LS.
```

```
%
% [.SI] is a (NR x 2) matrix storing the spectrum number (column 1) and
% corresponding f(w) value (column 2) for each retained VVSP spectrum.
%
% [.fw] is a (J x NR) matrix storing the vector of solutions to the
% quadratic expression yj'.A.(yj')', calculated for each VVSP component.
%
% [.fwNorm] is a (J x NR) matrix storing the normalised vector of
% solutions to the quadratic expression yj'.A.(yj')'.
%
% [.DW] is an (NL x 1) vector storing the Durbin-Watson value for each VVSP
% component and is calculated from the corresponding vector of f(w) values.
%
% [.PR] is an (NL x 1) vector storing the logarithmic SSQ projection residuals
% calculated for each new VVSP component located.
%
% Nicholas I. Pedge
% Department of Chemistry, University of Hull, Cottingham Road, Hull,
% HU6 7RX
% Process R&D, AstraZeneca R&D Charnwood, Bakewell Road, Loughborough,
% Leicestershire, LE11 5RH
%
% VVSP_output = VVSP(X, NL, NR, p_norm, plotting)

% BASIC INPUT VARIABLES CHECK
if nargin < 5 || isempty(plotting)
        fprintf( '[plotting] input variable not specified. \n' )
        fprintf( 'Setting plotting to "on" \n' )
        plotting = 1;
end
%
if nargin < 4 || isempty(p_norm)
        fprintf( 'Spectral p-norm not specified. \n' )
        fprintf( 'The default p-norm {2} will be used \n' )
        p_norm=2;
end
%
if nargin < 2 || isempty(NL)
        error( 'The number of components to locate [NL] was not specified' )
end
%
if nargin < 1 || isempty(X)
        error( 'Please provide [X]' )
end
%
if nargout < 1
        error( 'Output variable [VVSP_output] not specified! ' )
end
%
% PRE-ALLOCATE STORAGE VECTORS AND MATRICES
%
% Establish the size of the input data matrix
[J, K]= size(X);
%
% Vector to store the spectral norms of each spectrum in X
u_norm_s=zeros(J,1);
```

```
p_norm_s=zeros(J,1);
%
% Matrix to select the spectral norms and indices of each VVSP component.
SI=zeros(NL, 2);
%
% Matrix to store the solutions of the quadratic expression yi'.A.(yi')'
fw=zeros(J,NL);
fwNorm=zeros(J,NL);
%
% Vector to store the Durbin-Watson values calculated from the values
% derived from the quadratic form yi'.A.(yi')'
DW=zeros(NL,1);
%
% Vector to logartithmic projection residuals
PR=zeros(NL,1);
%
% Matrix to store normalised data calculated from original data
Y=zeros(J,K);
%
% MAIN VVSP CALCULATIONS
%
tic % Start timer
fprintf( 'Finding spectrum to initialise VVSP calculations \n' )
%
% Calculate the p-norms (p=pnorm) of each spectrum in original data
for j=1:J
        u_norm_s(j)=norm(X(j,:),1);          % Calculate the 1-norm of each spectrum
        p_norm_s(j)=norm(X(j,:),p_norm);     % Calculate the p-norm of each spectrum
        Y(j,:)=X(j,:)./ p_norm_s(j);         % Normalise each spectrum in X
end
%
% Identify the spectrum with the largest 1-norm value from u_norm_s and
% store in SI
[SI(1,1), SI(1,2)]=max(u_norm_s);
%
% Normalise the spectrum with largest 1-norm to give r. The spectrum is
% normalised using the 1-norm
%
r=X(SI(1,2),:) ./ u_norm_s(SI(1,2));
%
% Find the spectrum z1 that maximises ||r'-yj'||_2
%
R=repmat(r,J,1); % Create a matrix filled with the spectrum r
E=R-Y; % Calculate the residual between the r spectrum and
 % each normalised spectrum in Y
%
%Calculate the p-norm of each spectrum in the residual matrix E
for j=1:J
        ResidNorm(j)=norm(E(j,:),p_norm);
end
%
% Select the spectrum with largest residual p-norm. This corresponds to the
% spectrum that maximises ||r'-yj'||_2
[SI(1,1), SI(1,2)]=max(ResidNorm);
Z=Y(SI(1,2),:)'; % Set spectrum as a column vector
% Now that the first spectrum has been selected, can start the loop to find
```

```
% NL VVSP spectra.
%
for m = 1: NL
        fprintf( 'Calculating VVSP component %i \n' , m)
        A=eye(K)-(Z*pinv(Z)); % null matrix for the vectors in z
        % A = Ij-ZnZn^+
        %
        % Use matrix expression to calculate f(w) for each normalised spectrum
        % in Y
        fw(:, m)=(diag(Y*A*Y'));
        %
        % Normalise vector of f(w) values
        fwNorm(:,m)=fw(:,m) ./ norm(fw(:,m),2);
        %
        % find the maximum value and concatenate to Z (unless first spectrum)
        [SI(m,1), SI(m,2)]=max(fw(:,m));
        if m == 1
                Z(:,1)=Y(SI(m,2),:)';
        else
                Z=[Z, Y(SI(m,2),:)'];
        end
        %
        % Calculte the Durbin-Watson value
        fw_diff = diff(fw(:,m));
        fw_diff = [fw_diff; 0];
        fw_SSQ = fw(:,m)' * fw(:,m);
        fw_diff_SSQ = fw_diff' * fw_diff;
        DW(m,1) = fw_diff_SSQ ./ fw_SSQ;
        %
        % Calculate the log10 of the projection residual sum-of-squares, PR
        % Re-calculate the null matrix A using updated Z
        A=eye(K)-(Z*pinv(Z));
        % Calculate the projection residuals using original data, X
        XA=X*A;
        % Calculate the log10 value of the projection residuals sum-of-squares
        PR(m,1)=log10(trace(XA'*XA));
end
%
toc % Stop timer
%
if plotting == 1
        figure()
        subplot(2,1,1)
        plot(fw(:,1:m), 'linewidth' ,2); axis tight
        title( 'f(w) = y_j^T.A.(y_j^T)^T' , 'fontweight' , 'bold' )
        xlabel( 'Measurement number' )
        ylabel( 'Value' )
        subplot(2,1,2)
        plot(fwNorm(:,1:m), 'linewidth' ,2); axis tight
        title( 'f(w) = y_j^T.A.(y_j^T)^T / ||y_j^T.A.(y_j^T)^T||_2' ,'fontweight' , 'bold' )
        xlabel( 'Measurement number' )
        ylabel( 'Value' )
end
```

```
if plotting == 1
        figure()
        subplot(2,1,1)
        plot(DW(1:m), 'marker' , '+' , 'linewidth' ,1); hold on ;
        plot(DW(1:m), 'marker' , 'O' , 'linewidth' ,1)
        title( 'Durbin-Watson Values' , 'fontweight' , 'bold' )
        xlabel( 'Component number' ), ylabel( 'DW Value' )
        subplot(2,1,2)
        plot(PR(1:m), 'marker' , '+' , 'linewidth' ,1); hold on ;
        plot(PR(1:m), 'marker' , 'O' , 'linewidth' ,1)
        title( 'Logarithmic Projection Residual' , 'fontweight' , 'bold' )
        xlabel( 'Component number' ), ylabel( 'Log_1_0 Projection Residual' )
        drawnow
end
% User can now specify how many components to retain
if nargin < 3 | | isempty(NR)
        NR=input( 'Please enter the number of VVSP components to retain: ' );
end

Z(:,NR+1:end)=[];
fw(:,NR+1:end)=[];
fwNorm(:,NR+1:end)=[];
SI(NR+1:end,:)=[];
Sopt=Z;
Copt=X*pinv(Sopt');

if plotting ==1
        figure()
        subplot(2,1,1)
        plot(Sopt, 'linewidth' ,2)
        string1=[ 'VVSP Spectra (p-norm = ' ,num2str(p_norm), ')' ];
        title(string1, 'fontweight' , 'bold' )
        xlabel( 'Variable number' ), ylabel( 'Intensity (Arb.)' ), axis tight
        subplot(2,1,2)
        plot(Copt, 'linewidth' ,2)
        title( 'Least squares estimate of concentration profiles' ,'fontweight' , 'bold' )
        xlabel( 'Spectrum number' ), ylabel( 'Intensity (Arb.)' ),axis tight
end
% Store name of data matrix VVSP was applied to
VVSP_output.X=inputname(1);
% Store the value of p used for p-normalisation of spectra
VVSP_output.p_norm=p_norm;
% Store the VVSP pure component profiles (Sopt)
VVSP_output.Sopt=Sopt;
% Store the LS estimates of the concentratio profiles (Copt)
VVSP_output.Copt=Copt;
% Store the index and maximum values of the expression yj'.A.(yj')'
VVSP_output.SI=SI;
% Store the calculated values of the expression yj'.A.(yj')'
VVSP_output.fw=fw;
% Store the normalised values of the expression yj'.A.(yj')'
VVSP_output.fwNorm=fwNorm;
% Store the Durbin-Watson values
VVSP_output.DW=DW;
% Store the projection residuals
VVSP_output.PR=PR;
```

LinearKF.m

```
function KF_output=LinearKF(S, Z, KF_options)
% KF_output=LinearKF(S, Z, KF_options)
%
% Linear Kalman Filter.
% Implementation based upon description of the linear and adaptive Kalman
% filter published by Sarah C. Rutan et al. in Analytica Chemica Acta, 160
% (1984) 99-119. This script will run the linear Kalman filter on the first
% spectrum to calculate the Kalman gain and state parameter error
% covariance matrices. These are then used directly in the estimation of
% the state parameters of the remaining measurement vectors. The user can
% also provide Kalman gain and state parameter error covariance matrices
% calculated previously. The script will also run through the data twice.
% The state parameters are propagated from the first pass and the Kalman
% filter will run through a second time to recalculate the innovations. This
% will provide a better estimate of the final lack-of-fit.
%
% INPUT ARGUMENTS
%
% [S] is an (N x K) row matrix of reference measurement functions, such as
% pure component spectra. N is the number of individual components and K
% is the number of measurement variables.
%
% [Z] is a (J x K) matrix or (1 x K) vector of process measurement data where
% J is the number of observations (spectra) and K is the number of
% measurement variables.
%
% [KF_options] is a structured array that can be used to provide
% additional optional input arguments. If the user does not provide any
% additional input arguments, the script will call the default values
% described below.
%
% [.R] is an estimate of the measurement noise variance. This value will
% not change during Kalman filtering so the accuracy of the estimated state
% parameters will be reduced by selecting an inappropriate value.
% The default value is 0.0001.
%
% [.G] is an (N x K) row matrix of Kalman gain values to be applied during
% Kalman filtering. The user may provide [G] if the Kalman filter has
% been run previously using the same reference measurement functions. This
% will prevent the Kalman filter from re-calulating the Kalman gain for the
% first measurement vector, j=1.
% The default value is [] (empty field).
%
% [.Xin] is a (J x N) matrix of initial estimates of the state parameters.
% Enter a row vector for a single measurement vector or a row orientated
% matrix (J rows) if a number of measurement vectors (spectra) are
% used.
% The default value is [] (empty field).
%
% [.EvoluOn] allows the user to specify whether the input matrix [Z]
% contains evolutionary data (such as HPLC-DAD or reaction spectra). If
% the data is evolutionary, the values of the state parameters at
% measurement(time) j + 1 will be similiar to the values of the state
% parameters at measurement (time) j.
% If [.EvoluOn] = 1, the final estimated values of the state parameters at
```

```
% at sample j, variable k = K are propagated to be the initial estimates for
% sample j+1, variable k = 1.
% If [.EvoluOn] = 0, the final estimate of the state parameters for
% measurement (time) j are not propagated to the measurement j+1.
% The default value is 0 ([.EvoluOn] is off).
%
% [.Pin] is an (N x N) matrix of final variances to be used by Kalman filter.
%
% [.Plotting] allows the user to specify whether results are plotted after
% Kalman filtering has been performed.
% The default value is 1 ([.Plotting] is on).
%
% OUTPUT ARGUMENTS
%
% KF_output is a structured array generated by this script to store the
% various outputs listed below.
%
% [.X] is a (J x N) matrix or (1 x N) row vector of the calculated state
% parameters, where J is the number of observations (spectra) and N is the
% number of state parameters (e.g. chemical components).
%
% [.G] is an (N x K) matrix of Kalman gains calculated for the first
% measurement vector, j = 1 and used during Kalman filtering of the
% remaining J-1 measurements.
%
% [.V] is a (J x K) matrix or (1 x K) row vector of innovation values for
% each iteration k of the Kalman filter.
%
% [.Pf] is an (N x N) matrix that stores the final error variance (the
% diagonal elements of the error covariance matrix P) at the K-th iteration.
%
% Nicholas I. Pedge
% Department of Chemistry, University of Hull, Cottingham Road, Hull,
% HU6 7RX
% Process R&D, AstraZeneca R&D Charnwood, Bakewell Road, Loughborough,
% Leicestershire, LE11 5RH
%
% KF_OUTPUT=LinearKF(S, Z, KF_options);

% CHECK INPUT ARGUMENTS
% Check that an options structure array, KF_options has been provided
if nargin < 3
        fprintf( 'No options structure provided. Default values will be used \n' )
        fprintf( 'This default options structure will be stored in the output structure \n' )
        KF_options.R=1E-4;
        KF_options.G=[ ];
        KF_options.Xin=[ ];
        KF_options.EvoluOn=0;
        KF_options.Pin=[ ];
        KF_options.Plotting=1;
end
% Check that the user provided KF_options is a structured array
if isstruct(KF_options)==0
        error( 'The input argument [KF_options] should be a structured array' )
end
% Check user has provided at least two input arguments
```

```
if nargin < 2
        error( 'Please provide [S] and [Z]' )
end
%
% Check the dimensions of [S] and [Z].
[N,K]=size(S);
S_trans=S';
% N is the number of components in reference data
% K is the number of measurement variables in reference data
 %
[J, K2]=size(Z);
% J is the number of observations or spectra (number of row vectors)
% K2 is the number of variables
%
if K ~= K2
        error( 'The number of variables in [S] and [Z] are not equal' )
end
%
% EXTRACT VALUES FROM [KF_options] OR INITIALISE AS REQUIRED.
R=KF_options.R;
fprintf( 'Measurement error variance has been set to: %E \n' , R)
%
% If user has not provided Kalman gain matrix, initialise values of KG to
% 1.0
if isempty(KF_options.G)==1
        UpdateG=1;
        G=zeros(N, K);
else
        UpdateG=0;
        G=KF_options.G;
        fprintf( 'User has provided matrix of Kalman gains \n' )
end
%
% If the user has not entered a vector or matrix [Xin] of initial estimates
% for the state parameters (e.g. component concentrations), [X] is set to a
% matrix of ones with dimensions (J,K).
if isempty(KF_options.Xin)==1
        X = ones(J,N);
        fprintf( 'No initial estimates of X were provided by user \n' )
else
        X=KF_options.Xin;
        fprintf( 'A matrix of initial estimates of X were provided by user \n' )
end
%
EvoluOn=KF_options.EvoluOn;
if EvoluOn==0
        fprintf( 'Data set will be treated as non-evolutionary \n' )
else
        fprintf( 'Data set will be treated as evolutionary \n' )
end
%
% If user has not provided error covariance matrix [.Pin], use initial
% estimate calculated from reference measurement function [S].
if isempty(KF_options.Pin)==1
        UpdateP=1;
        P = cov(S');
```

```
        P = eye(N)*P*100;
else
        UpdateP=0;
        Pin=KF_options.Pin;
        fprintf( 'User has provided matrix of state estimate errors \n' )
end
%
Plotting=KF_options.Plotting;
if Plotting==0
        fprintf( 'Display plots is set to off \n' )
else
        fprintf( 'Display plots is set to on \n' )
end
 %
% PRE-ALLOCATE STORAGE VECTORS AND MATRICES
V = zeros(J, K);
%V(:,1)=Z(:,1);
%
% Matrix to store final state paramater estimate variance values.
Pf=zeros(N, N);
%
% Vector to store the innovations square-root, sum-of-squares
LOF=zeros(J,1);
%
%*************************************************************************
% Recursive Kalman Filter Calculations                                   *
%*************************************************************************
tic
%
SecondPass=-1; % Counter so filter will make second pass through the data
%
while SecondPass < 1
        hw=waitbar(0, 'Please wait...' );
        %
        for j=1:J;
                waitbar(j/J, hw)
                %
                if j==1 && UpdateP==1
                        P = cov(S');
                        P=eye(N)*100*P;
                elseif j==1 && UpdateP==0
                        P = Pin;
                elseif j~=1 && UpdateP==0
                        P = Pf;
                end
                %
                % Update initial state estimates if evolutionary data is used.
                if (j>1) && EvoluOn == 1
                        % Propagate state parameters to next sample.
                        X(j,:)=X(j-1,:);
                end
                %
                for k = 1:K;
                        % State estimate extrapolation.
                        _old(j,:) = X(j,:);
                        % x(k|k-1) = x(k-1:k-1)
```

```
                %
                % Error covariance extrapolation.
                P_old = P;
                % P(k|k-1) = P(k-1|k-1)
                %
                % Calculate innovation INV(j,k)
                V(j,k)=Z(j,k)-(S(:,k)'*X_old(j,:)');
                % V(k) = z(k) - (S(k)' * x(k|k-1))
                %
                % Update Kalman gain
                if UpdateG==1
                        G(:,k) = (P_old * S(:,k))*inv((S_trans(k,:) ...
                         *P_old*S(:,k))+R);
                        % k(k)=(P(k|k-1).S(k)).[S(k)'.P(k|k-1).S(k)+R]^-1
                end
                %
                % State estimate update
                g=G(:,k);
                X(j,:)=X_old(j,:) + (g*V(j,k))';
                % x(k|k)= x(k|k-1) + k(k)v(k)
                %
                % Error covariance update, numerically stable solution
                if UpdateP==1
                        P=(eye(N)-(g*S_trans(k,:))) * P_old * ...
                        (eye(N)-(g*S_trans(k,:)))'+ (g * R * g');
                        % P(k|k)= [I - k(k).S(k)'].P(k|k-1).
                        % [I-k(k).S(k)']'+ k(k).R.k(k)'
                else
                        P=P_old;
                end
        end % end k
        %
        UpdateP=0;      % After first spectrum, turn off updating of P
        UpdateG=0;      % After first spectrum, turn off updating of G
        Pf=P;           % After first spectrum, Pf = P
        %
        % ANALYSE INNOVATIONS SEQUENCE
        %
        % The innovations sequence contains the difference between the
        % predicted measurement value at variable k and the actual measured
        % value. If the Kalman filter is operating correctly, the
        % innovations sequence for each observation should be white noise
        % with zero mean. The square-root, sum-of-squares of each samples'
        % innovations sequence should decrease as Kalman filtering improves
        %
        VSSQ=V(j,:)*V(j,:)'/(K);
        DATASSQ=Z(j,:)*Z(j,:)'/(K);
        LOF(j) = (sqrt(VSSQ / DATASSQ))*100;
        %
        % This LOF statistic gives the average lack-of-fit per spectral
        % data point as a percentage of the average data point in [Z].
end % end j
%
close(hw)
SecondPass=SecondPass+1; % increment counter so Kalman filter makes second pass
Pin=Pf;
```

```
end % while
%
toc

%*************************************************************************
% End of Recursive Kalman Filter Calculations, update final outputs      *
%*************************************************************************
%
% OUTPUT RESULTS
KF_output.X=X;
KF_output.G=G;
KF_output.V=V;
KF_output.LOF=LOF;
KF_output.Pf=Pf;
KF_output.KF_options=KF_options;
%
if Plotting ==1
        if J>1
                figure()
                subplot(3,1,1)
                plot(1:J,X);axis tight ;
                title( 'State parameters' , 'fontweight' , 'bold' );
                xlabel( 'Sample number' );
                ylabel( 'State parameter' );
                %
                subplot(3,1,2)
                plot(1:K,V); axis tight ;
                title( 'Innovations' , 'fontweight' , 'bold' );
                xlabel( 'Variable number' );
                ylabel( 'Innovations' );
                %
                subplot(3,1,3)
                plot(1:J,LOF); axis tight ;
                title( 'Average LOF per variable point (%)' , 'fontweight' , 'bold' );
                xlabel( 'Sample number' );
                ylabel( 'LOF (%)' );
        else
                figure()
                plot(1:K,V); axis tight ;
                title( 'Innovations' , 'fontweight' , 'bold' );
                xlabel( 'Variable number' );
                ylabel( 'Innovations' );
        end
end
end % END OF FUNCTION
```

AdaptiveKF.m

```
function KF_output=AdaptiveKF(S, Z, W, KF_options)
% KF_output=AdaptiveKF(S, Z, W, KF_options);
%
% Algorithm based upon description of the linear and adaptive Kalman
% filter published by Sarah C. Rutan et al. in Analytica Chemica Acta, 160
% (1984) 99-119. The script will run through the data twice.
% The state parameters are propagated from the first pass and the Kalman
% filter will run through a second time to recalculate the innovations. This
% will provide a better estimate of the final spectral lack-of-fit
% calculated using optimised state parameters.
% The user will then have the option to augment the matrix of reference
% measurement functions with a new component, or update an existing
% reference measurement function for one of the components. The
% augmentation or update assumes the reference measurement functions are
% non-negative as described by Rutan and Brown.
%
% INPUT ARGUMENTS
%
% [S] is an (N x K) row matrix of reference measurement functions, such as
% pure component spectra. N is the number of individual components and K
% is the number of measurement variables.
%
% [Z] is a (J x K) matrix or (1 x K) vector of process measurement data where
% J is the number of observations (spectra) and K is the number of
% measurement variables.
%
% [W] is window size to be used for adapting the measurement variance
% estimates, Rk. W must be even-numbered
%
% [KF_options] is a structured array that can be used to provide
% additional optional input arguments. If the user does not provide any
% additional input arguments, the script will call the default values
% described below.
%
% [.Rmin] is the minimum measurement noise variance permitted.
% The actual measurement noise variance will be calculated adaptively
% during the Kalman filtering but the value of R is not permitted to drop
% below Rmin. This value should be two or three orders of magnitude lower
% than the expected measurement noise variance.
% The default value is 1.0E-6
%
% [.Xin] is a (J x N) matrix of initial estimates of the state parameters.
% Enter a row vector for a single measurement vector or a row orientated
% matrix (J rows) if a number of measurement vectors (spectra) are
% used.
% The default value is [] (empty field).
%
% [.EvoluOn] allows the user to specify whether the input matrix [Z]
% contains evolutionary data (such as HPLC-DAD or reaction spectra). If
% the data is evolutionary, the values of the state parameters at
% measurement (time) j + 1 will be similar to the values of the state
% parameters at measurement (time) j.
% If [.EvoluOn] = 1, the final estimated values of the state parameters at
% at sample j, variable k = K are propagated to be the initial estimates for
```

```
% sample j + 1, variable k = 1.
% If [.EvoluOn] = 0, the final estimate of the state parameters for
% measurement (time) j are not propagated to the measurement j+1.
% The default value is 0 ([.EvoluOn] is off).
%
% [.Plotting] allows the user to specify whether results are plotted after
% Kalman filtering has been performed.
% The default value is 1 ([.Plotting] is on).
%
% OUTPUT ARGUMENTS
%
% KF_output is a structured array generated by this script to store the
% various outputs listed below.
%
% [.X] is a (J x N) matrix or (1 x N) row vector of the calculated state
% parameters, where J is the number of observations (spectra) and N is the
% number of state parameters (e.g. chemical components).
%
% [.G_ALL] is an (N x K x J) array of Kalman gains calculated for each
% measurement vector. It is necessary to recalculate the Kalman gains
% for each measurement vector as the measurement variance is adapted based
% on the innovations sequence.
% If input [Z] is a single measurement vector, the output [.G_ALL] will
% be a N x K matrix of Kalman gain values
%
% [.Pf_ALL] is a (J x N) matrix or (1 x N) row vector which stores the final
% error variance (the diagonal elements of the error covariance matrix P)
% at the K-th iteration for ach of the J measurements
%
% [.V] is a (J x K) matrix or (1 x K) row vector of innovation values for
% each iteration k of the Kalman filter.
%
% [.Rk] is a (J x K) matrix or (1 x K) vector of adaptive measurement error
% values calculated at each variable index k.
%
% [.LOF] is the average measurement lack-of-fit with respect to the original
% data for each measurement j. This produces a (J x 1) column vector. A
% large value indicates that the Kalman filter has fitted the original
% measurement vector with less accuracy and is a good indication of the
% presence of an un-modelled component.
%
% [.LOF_Index] is index number (j) of the measurement vector that yields
% the largest LOF value. This is usually the measurement in which an
% contribution of an un-modelled component is at its largest.
%
% [.S_new] is the reference measurement function [S] after augmentation or
% correction using adaptive variance (Rk) values.
%
% [.KF_options] is a structured array containing the various input options
% used to run the Kalman filter.
%
% Nicholas I. Pedge
% Department of Chemistry, University of Hull, Cottingham Road, Hull,
% HU6 7RX
% Process R&D, AstraZeneca R&D Charnwood, Bakewell Road, Loughborough,
% Leicestershire, LE11 5RH
```

```
%
% KF_output=AdaptiveKF(S, Z, W, KF_options);

% CHECK INPUT ARGUMENTS
% Check that an options structure array, KF_options has been provided
if nargin < 4
        fprintf( 'No options structure provided. Default values will be used \n' )
        fprintf( 'This default options structure will be stored in the output structure \n' )
        KF_options.Rmin=1E-6;
        KF_options.Xin=[ ];
        KF_options.EvoluOn=0;
        KF_options.Plotting=1;
end
% Check that the user provided KF_options is a structured array
if isstruct(KF_options)==0
        error( 'The input argument [KF_options] should be a structured array' )
end
% Check user has provided at least two input arguments
if nargin < 3
        error( 'Please provide [S], [Z] and [W]' )
end
%
% Check the dimensions of [S] and [Z].
[N,K]=size(S);
S_trans=S';
% N is the number of components in reference data
% K is the number of measurement variables in reference data
%
[J, K2]=size(Z);
% J is the number of observations or spectra (number of row vectors)
% K2 is the number of variables
%
if K ~= K2
        error( 'The number of variables in [S] and [Z] are not equal' )
end
%
% EXTRACT VALUES FROM [KF_options] OR INITIALISE AS REQUIRED.
Rmin=KF_options.Rmin;
fprintf( 'Minimum permitted measurement error variance has been set to: %E \n' , Rmin)
%
% If the user has not entered a vector or matrix [Xin] of initial estimates
% for the state parameters (e.g. component concentrations), [X] is set to a
% matrix of ones with dimensions (J,K).
if isempty(KF_options.Xin)==1
        X = ones(J,N);
        fprintf( 'No initial estimates of X were provided by user \n' )
else
        X=KF_options.Xin;
        fprintf( 'A matrix of initial estimates of X were provided by user \n' )
end
%
EvoluOn=KF_options.EvoluOn;
if EvoluOn==0
        fprintf( 'Data set will be treated as non-evolutionary \n' )
else
        fprintf( 'Data set will be treated as evolutionary \n' )
```

```
end
%
% Calculate initial estimate of P from reference measurement function [S].
P = cov(S');
P=eye(N)*100*P;
%
Plotting=KF_options.Plotting;
if Plotting==0
        fprintf( 'Display plots is set to off \n' )
else
        fprintf( 'Display plots is set to on \n' )
end
%
if rem(W,2)~=0 % W is odd
        W=W+1;
end
% PRE-ALLOCATE STORAGE VECTORS AND MATRICES
V = zeros(J, K);
X_old = zeros(J, N);
%
% Array to store the Kalman gains matrix for each measurement vector
G_ALL=zeros(N,K,J);
% Matrix to store final state paramater error variance values.
Pf_ALL1=zeros(J,N);
Pf_ALL2=zeros(J,N,N);
%
% Vector to store the innovations square-root, sum-of-squares
LOF=zeros(J,1);
%
% Matrix to store the Rk values calculated during the calculation of the
% adaptive measurement noise variance.
Rk=zeros(J,K); Rk(:,1)= Rmin;
%
%*************************************************************************
% Recursive Kalman Filter Calculations                                   *
%*************************************************************************
tic
%
SecondPass=-1; % Counter so filter will make second pass through the data
%
while SecondPass < 1
        hw=waitbar(0, 'Please wait...' );
        %
        for j=1:J;
                waitbar(j/J, hw)
                %
                if SecondPass==-1
                        P = cov(S');
                        P = eye(N)*P*100;
                else
                        P = squeeze(Pf_ALL2(j,:,:));
                end
                %
                % Update initial state estimates if evolutionary data is used.
                if (j>1) && EvoluOn == 1
                        % Propagate state parameters to next sample.
```

```
        X(j,:)=X(j-1,:);
end
%
for k = 2:K;
        % State estimate extrapolation.
        X_old(j,:) = X(j,:);
        % x(k|k-1) = x(k-1:k-1)
        %
        % Error covariance extrapolation.
        P_old = P;
        % P(k|k-1) = P(k-1|k-1)
        %
        % Calculate innovation INV(j,k)
        V(j,k)=Z(j,k)-(S(:,k)'*X_old(j,:)');
        % V(k) = z(k) - (S(k)' * x(k|k-1))
        %
        % Calculate adaptive variance R, use available data if k < W
        m=W;
        if m >= k
                m = k-1; % if k=2, m=1 etc.
        end
        %
        for i=1:m;
                V_sum(i)=(V(j,k-i)*V(j,k-i));
        end
        %
        V_sum=sum(V_sum);
        % Update estimate of R, measurement error (equation 11)
        Rk(j,k)=((inv(m))*(V_sum)) - (S_trans(k,:)*P_old*S(:,k));
        % R(k) = 1/m [ j=1 Sigma m v(k-j).v(k-j)] -
        %                                   S(k)'.P(k|k-1).S(k)
        %
        % Limit variance to prevent R approaching zero as that can result
        % in a singular error covariance matrix P.
        if Rk(j,k)<Rmin
                Rk(j,k)=Rmin;
        end
        % Update Kalman gain
        G(:,k) = (P_old * S(:,k))*(inv((S_trans(k,:) ...
                                        *P_old*S(:,k))+Rk(j,k)));
        % k(k)=(P(k|k-1).S(k)).[S(k)'.P(k|k-1).S(k)+R]^-1
        %
        g=G(:,k);
        G_ALL(:,k,j)=g;
        %
        % State estimate update
        X(j,:)=X_old(j,:) + (g*V(j,k))';
        % x(k|k)= x(k|k-1) + k(k)INV(k)
        %
        % Error covariance update, numerically stable solution
        P = (eye(N)-(g*S_trans(k,:))) * P_old * ...
                        (eye(N)-(g*S_trans(k,:)))' + (g * Rk(j,k) * g');
        % P(k|k)= [I - k(k).S(k)'].P(k|k-1).
        %          [I-k(k).S(k)']'+ k(k).R.k(k)'
end % end k
%
```

```
                Pf_ALL1(j,:)=diag(P);
                Pf_ALL2(j,:,:)=P;
                %
                % ANALYSE INNOVATIONS SEQUENCE
                 % The innovations sequence contains the difference between the
                % predicted measurement value at variable k and the actual measured
                % value. If the Kalman filter is operating correctly, the
                % innovations sequence for each observation should be white noise
                % with zero mean. The square-root, sum-of-squares of each samples'
                % innovations sequence should decrease as Kalman filtering improves
                %
                VSSQ=V(j,:)*V(j,:)' ./(K);
                DATASSQ=Z(j,:)*Z(j,:)' ./(K);
                LOF(j) = (sqrt(VSSQ / DATASSQ))*100;
                %
                % This LOF statistic gives the average lack-of-fit per spectral
                % data point as a percentage of the average data point in [Z].
                %
        end % end j
        %
        close(hw)
        SecondPass=SecondPass+1; % increment counter
end % while
LOF_Max, LOF_Index]=max(LOF);
Pf_ALL=Pf_ALL1; clear Pf_ALL1 Pf_All2
toc
%
%*************************************************************************
% End of Recursive Kalman Filter Calculations                            *
%*************************************************************************
%
if Plotting ==1
        if J>1
                figure()
                subplot(2,1,1)
                plot(1:J,X);axis tight ;
                title( 'Estimated state parameters' , 'fontweight' , 'bold' );
                xlabel( 'Sample number' );
                ylabel( 'State parameter' );
                %
                subplot(2,1,2)
                plot(1:J,LOF); axis tight ;
                title( 'Average LOF per variable point (%)' , 'fontweight' , 'bold' );
                label( 'Sample number' );
                label( 'LOF (%)' );
                %
                figure()
                subplot(2,1,1)
                plot(1:K,V); axis tight ;
                title( 'Innovations sequence' , 'fontweight' , 'bold' );
                xlabel( 'Variable number' );
                ylabel( 'Innovations' );
                subplot(2,1,2)
                plot(1:K,Rk); axis tight ;
                title( 'Adaptive measurement variance' , 'fontweight' , 'bold' );
                xlabel( 'Variable number' );
```

```
                ylabel( 'R' );
        else
                subplot(2,1,1)
                plot(1:K,V); axis tight ;
                title( 'Innovations sequence' , 'fontweight' , 'bold' );
                xlabel( 'Variable number' );
                ylabel( 'Innovations' );
                subplot(2,1,2)
                plot(1:K,Rk); axis tight ;
                title( 'Adaptive measurement variance' , 'fontweight' , 'bold' );
                xlabel( 'Variable number' );
                ylabel( 'R' );
        end
end
%
%*************************************************************************
% Update or augment the reference measurement function S if there is an      *
% un-modelled contribution to the data. The matrices Bk and the              *
% innovation matrix V are used to create to augment or correct the           *
% measurement function                                                       *
%*************************************************************************
%
disp( 'Press (1) to Augment the Reference Measurement Function' )
disp( 'Press (2) to Modify a component of Reference Measurement Function' )
disp( 'Press (3) to make no change to the Reference Measurement Function' )
UserOpt1=input( 'Select an option ' );
while UserOpt1 <1 || UserOpt1 >3
        disp( 'Invalid option' )
        disp( 'Do you wish to Augment Reference Measurement Function? (1)' )
        disp( 'Do you wish to Modify Reference Measurement Function? (2)' )
        disp( 'Make no changes to the Reference Measurement Functions? (3)' )
        UserOpt1=input( 'Please select an option? ' );
end
%
if UserOpt1 == 1
        [S_new]=augment(V,Rk,S,W,LOF_Index);
end
%
if UserOpt1 == 2
        [S_new]=update(V,Rk,S,W,LOF_Index);
end
%
if UserOpt1 == 3
        S_new=S;
end
%
% OUTPUT RESULTS
KF_output.X=X;
KF_output.G_ALL=G_ALL;
KF_output.Pf_ALL=Pf_ALL;
KF_output.V=V;
KF_output.Rk=Rk;
KF_output.LOF=LOF;
KF_output.LOF_Index=LOF_Index;
KF_output.S_new=S_new;
KF_output.KF_options=KF_options;
```

```
% Use the command KF_options=KF_output.KF_options; to extract options
% structure at command line.
end % END FUNCTION 'AdaptiveKF'



%*************************************************************************
% Subfunction Augment                                                    *
% ************************************************************************
function [S_new]=augment(V,Rk,S,W,LOF_Index)
% subfunction augment
[N,K]=size(S);
[J,K]=size(V);
% Vector to store augmented reference function
S_Aug=zeros(1,K);
%
% Augmentation calculation using spectrum with largest spectral LOF
Bk=zeros(1,K);
for k=2:K-(W/2)-1
        m=W;
        if m>k
                m=k-1;
        end
        b=zeros(m,1);
        for i=1:m
                b(i)=V(LOF_Index,k-i+(floor(m/2)))/m;
        end
        %
        b_sum=sum(b);
        %
        if b_sum > 0
                Bk(k)=1;
        else
                Bk(k)=-1;
        end
end
%
for k=2:K-(W/2)-1
        m=W;
        if m>k
                m=k-1;
        end
        %
        if Bk(k)==1
                S_Aug(k)=Bk(k)*(sqrt(Rk(LOF_Index,k+(floor(m/2)))));
        else
                S_Aug(k)=0;
        end
end
%
S_Aug=S_Aug/norm(S_Aug,2); % Normalise spectrum
S_new=[S;S_Aug]'; % Augment reference measurement function matrix
%
figure()
plot(1:K, S_new(:,1:end-1), 'linewidth' ,1);hold on ;
plot(1:K, S_new(:,end), 'linewidth' ,2);
```

```
legend
title( 'Augmented Reference Measurement Function' , 'fontweight' , 'bold' )
xlabel( 'Variable' )
ylabel( 'Value' )
%
end % END SUBFUNCTION 'augment'

%*************************************************************************
% Subfunction Update                                                     *
% ************************************************************************
function [S_new]=update(V,Rk,S,W,LOF_Index)
% subfunction update
[N,K]=size(S);
[J,K]=size(V);
S_new=S;
%
figure()
subplot(2,1,1)
plot(1:K, S, 'linewidth' ,1);hold on ;
legend
title( 'Original Reference Measurement Function' , 'fontweight' , 'bold' )
xlabel( 'Variable' )
ylabel( 'Value' )
subplot(2,1,2)
plot(1:K, Rk(LOF_Index,:), 'linewidth' ,2);
title( 'Rk at maximum lack-of-fit' , 'fontweight' , 'bold' )
xlabel( 'Variable' )
ylabel( 'Value' )
%
disp( 'Please select the component you wish to modify' )
string1=[ 'Select an integer number between 1 and ' ,num2str(N), ': ' ];
UserOpt2=input(string1);
while UserOpt2 <1 || UserOpt2 > N
        disp( 'The number you selected is out of range' )
        UserOpt2=input(string1);
end
%
% Update reference measurement function using spectrum with largest spectral LOF
Bk=zeros(1,K);
for k=2:K-(W/2)-1
        m=W;
        if m>k
                m=k-1;
        end
        %
        b=zeros(m,1);
        for i=1:m
                b(i)=V(LOF_Index,k-i+(floor(m/2)))/m;
        end
        %
        b_sum=sum(b);
        %
        if b_sum > 0
                Bk(k)=1;
        else
                Bk(k)=-1;
```

```
        end
end
%
for k=2:K-(W/2)-1
        m=W;
        if m>k
                m=k-1;
        end
        %
        S_new(UserOpt2,k)=S(UserOpt2,k) + ...
        Bk(k)*(sqrt(Rk(LOF_Index,k+(floor(m/2)))));
        %
        if S(UserOpt2, k)< 0
                S_new(UserOpt2, k)= 0;
        end
end
%
figure()
plot(1:K, S, 'linewidth' ,1);hold on ;
plot(1:K, S_new(UserOpt2,:), 'linewidth' ,2);
legend
title( 'Updated Reference Measurement Function' , 'fontweight' , 'bold' )
xlabel( 'Variable' )
ylabel( 'Value' )
end % END SUBFUNCTION 'update'
```

VecLinearKF.m

```
function KF_output=VecLinearKF(S, Z, KF_options)
% KF_output=VecLinearKF(S, Z, KF_options);
%
% Vectorised Linear Kalman Filter.
% This is a vectorised implementation of the linear Kalman filter and is
% based upon the description of the linear and adaptive Kalman filter
% published by Sarah C. Rutan et al. in Analytica Chemica Acta, 160
% (1984) 99-119.
% The Kalman filter calculations have been vectorised so if the user
% specifies a matrix of measurement data, the entire matrix of state
% parameter estimates will be updated at each iteration (variable k).
% The vectorised Kalman filter will produce identical results to the
% standard Kalman filter for the same data set [Z],reference measurement
% functions [S] and measurement noise variance [R]. However the script
% will run much faster. The user can also provide Kalman gain
% and state parameter error covariance matrices calculated previously.
% The script will run through the data twice. The state parameters are
% propagated from the first pass and the Kalman filter will run through a
% second time to recalculate the innovations. This will provide a better
% estimate of the final innovations (spectral) lack-of-fit calculated using
% optimised state parameters
%
% INPUT ARGUMENTS
%
% [S] is an (N x K) row matrix of reference measurement functions, such as
% pure component spectra. N is the number of individual components and K
% is the number of measurement variables.
%
% [Z] is a (J x K) matrix or (1 x K) vector of process measurement data where
% J is the number of observations (spectra) and K is the number of
% measurement variables.
%
% [KF_options] is a structured array that can be used to provide
% additional optional input arguments. If the user does not provide any
% additional input arguments, the script will call the default values
% described below.
%
% [.R] is an estimate of the measurement noise variance. This value will
% not change during Kalman filtering so the accuracy of the estimated state
% parameters will be reduced by selecting an inappropriate value.
% The default value is 0.0001.
%
% [.G] is an (N x K) row matrix of Kalman gain values to be applied during
% Kalman filtering. The user may provide [G] if the Kalman filter has
% been run previously using the same reference measurement functions.
% The default value is [ ] (empty field).
%
% [.Xin] is a (J x N) matrix of initial estimates of the state parameters.
% Enter a row vector for a single measurement vector or a row orientated
% matrix (J rows) if a number of measurement vectors (spectra) are
% used.
% The default value is [ ] (empty field).
%
% [.Pin] is an (N x N) matrix of final variances to be used by Kalman
```

```
% filter.
%
% [.Plotting] allows the user to specify whether results are plotted after
% Kalman filtering has been performed.
% The default value is 1 ([.Plotting] is on).
%
% OUTPUT ARGUMENTS
%
% KF_output is a structured array generated by this script to store the
% various outputs listed below.
%
% [.X] is a (J x N) matrix or (1 x N) row vector of the calculated state
% parameters, where J is the number of observations (spectra) and N is the
% number of state parameters (e.g. chemical components).
%
% [.G] is an (N x K) matrix of Kalman gains calculated during Kalman
% filtering.
%
% [.V] is a (J x K) matrix or (1 x K) row vector of innovation values for
% each iteration k of the Kalman filter.
%
% [.Pf] is an (N x N) matrix that stores the final error variance (the
% diagonal elements of the error covariance matrix P) at the K-th iteration.
%
% [.LOF] is a (J x 1) vector of spectral Lack-of-fit values calculated from
% the corresponding innovations sequences.
%
% Nicholas I. Pedge
% Department of Chemistry, University of Hull, Cottingham Road, Hull,
% HU6 7RX
% Process R&D, AstraZeneca R&D Charnwood, Bakewell Road, Loughborough,
% Leicestershire, LE11 5RH
%
% function KF_output=VecLinearKF(S, Z, KF_options);

% CHECK INPUT ARGUMENTS
% Check that an options structure array, KF_options has been provided
if nargin < 3
        fprintf( 'No options structure provided. Default values will be used \n' )
        fprintf( 'This default options structure will be stored in the output structure \n' )
        KF_options.R=1E-4;
        KF_options.G=[ ];
        KF_options.Xin=[ ];
        KF_options.Pin=[ ];
        KF_options.Plotting=1;
end
% Check that the user provided KF_options is a structured array
if isstruct(KF_options)==0
        error( 'The input argument [KF_options] should be a structured array' )
end
% Check user has provided at least two input arguments
if nargin < 2
        error( 'Please provide [S] and [Z]' )
end
%
% Check the dimensions of [S] and [Z].
```

```
[N,K]=size(S);
S_trans=S';
% N is the number of components in reference data
% K is the number of measurement variables in reference data
%
[J, K2]=size(Z);
% J is the number of observations or spectra (number of row vectors)
% K2 is the number of variables
%
if K ~= K2
        error( 'The number of variables in [S] and [Z] are not equal' )
end
%
% EXTRACT VALUES FROM [KF_options] OR INITIALISE AS REQUIRED.
R=KF_options.R;
fprintf( 'Measurement error variance has been set to: %E \n' , R)
%
% If user has not provided Kalman gain matrix, initialise values of G to
% 1.0
if isempty(KF_options.G)==1
        UpdateG=1;
        G=zeros(N, K);
else
        UpdateG=0;
        G=KF_options.G;
        fprintf( 'User has provided matrix of Kalman gains \n' )
end
%
% If the user has not entered a vector or matrix [Xin] of initial estimates
% for the state parameters (e.g. component concentrations), [X] is set to a
% matrix of ones with dimensions (J,K).
if isempty(KF_options.Xin)==1
        X = ones(J,N);
        fprintf( 'No initial estimates of X were provided by user \n' )
else
        X=KF_options.Xin;
        fprintf( 'A matrix of initial estimates of X were provided by user \n' )
end
%
% If user has not provided error covariance matrix [.Pin], use initial
% estimate calculated from reference measurement function [S].
if isempty(KF_options.Pin)==1
        UpdateP=1;
        P=cov(S');
        P=eye(N)*P*100;
else
        UpdateP=0;
        Pin=KF_options.Pin;
        fprintf( 'User has provided matrix of state estimate errors \n' )
end
%
Plotting=KF_options.Plotting;
if Plotting==0
        fprintf( 'Display plots is set to off \n' )
else
        fprintf( 'Display plots is set to on \n' )
```

```
end
%
% PRE-ALLOCATE STORAGE VECTORS AND MATRICES
V = zeros(J, K);
%
% Matrix to store final state paramater estimate variance values.
Pf=zeros(N, N);
%
% Vector to store the innovations square-root, sum-of-squares
LOF=zeros(J,1);
%
%*************************************************************************
% Vectorised linear Kalman Filter Calculations                           *
%*************************************************************************
tic
%
SecondPass=-1; % Counter so filter will make second pass through the data
%
while SecondPass < 1
        hw=waitbar(0, 'Please wait...' );
        %
        if UpdateP==1
                P = cov(S');
                P = eye(N)*P*100;
        else
                P = Pin;
        end
        %
        for k = 1:K;
                waitbar(k/K, hw)
                %
                % State estimate extrapolation
                X_old = X;
                % X(k|k-1) = X(k-1:k-1) Matrix notation
                %
                % Error covariance extrapolation
                P_old = P;
                % P(k|k-1) = P(k-1|k-1) Matrix notation
                %
                % Calculate innovation V(:,k)
                V(:,k)=Z(:,k)-(X_old*S(:,k));
                % V(k) = z(k)- (X(k|k-1) * h(k)) Matrix notation
                %
                % Update Kalman gain
                if UpdateG==1
                        G(:,k) = (P_old * S(:,k)) * inv((S_trans(k,:) ...
                                *P_old*S(:,k))+R);
                        % k(k)=[P(k|k-1)*h(k)]*[(h(k)'*P(k|k-1)*h(k))+ R)]-1
                end
                %
                % State estimate update
                g=G(:,k);
                X=X_old + (V(:,k)*g');
                % X(k|k)= X(k|k-1) + (INV(k)* k(k)') Matrix notation
                %
                % Error covariance update, numerically stable solution
```

```
                if UpdateP==1
                        P=(eye(N)-(g * S_trans(k,:))) * P_old * ...
                                (eye(N)-(g * S_trans(k,:)))' + (g * R * g');
                        % P(k|k) = [I - k(k).S(k)'].P(k|k-1).
                        %          [I-k(k).S(k)']' + k(k).R.k(k)'
                else
                        P=P_old;
                end
        end % end k
        UpdateP=0;      % After first pass, turn off updating of P
        UpdateG=0;      % After first pass, turn off updating of G
        Pf=P;           % After first pass, Pf = P
        %
        close(hw)
        SecondPass=SecondPass+1;
        Pin=Pf;
end % end while
%
% ANALYSE INNOVATIONS SEQUENCE
%
% The innovations sequence contains the difference between the predicted
% measurement value at variable k and the actual measured value. If the
% Kalman filter is operating correctly, the innovations sequence for each
% observation should be white noise with zero mean. The square-root,
% sum-of-squares of each samples' innovations sequence should decrease as
% Kalman filtering improves.
VSSQ=sum((V.^2),2)/(K);
DATASSQ=sum((Z.^2),2)/(K);
LOF = (sqrt(VSSQ ./ DATASSQ))*100;
%
% This LOF statistic gives the average lack-of-fit per spectral data point
% as a percentage of the average data point in original data.
%
toc
%*************************************************************************
% End of Vectorised Kalman Filter Calculations, update final outputs     *
%*************************************************************************
%
% OUTPUT RESULTS
KF_output.X=X;
KF_output.G=G;
KF_output.V=V;
KF_output.Pf=Pf;
KF_output.LOF=LOF;
KF_output.KF_options=KF_options;
%
if Plotting ==1
        if J>1
                figure()
                subplot(3,1,1)
                plot(1:J,X); axis tight
                title( 'State parameters' , 'fontweight' , 'bold' );
                xlabel( 'Sample number' );
                ylabel( 'State parameter' );
                %
                subplot(3,1,2)
```

```
                plot(1:K,V); axis tight
                title( 'Innovations' , 'fontweight' , 'bold' );
                xlabel( 'Variable number' );
                ylabel( 'Innovations' );
                %
                subplot(3,1,3)
                plot(1:J,LOF); axis tight
                title( 'Average LOF per variable point (%)' , 'fontweight' , 'bold' );
                xlabel( 'Sample number' );
                ylabel( 'LOF (%)' );
        else
                figure()
                plot(1:K,V); axis tight
                title( 'Innovations' , 'fontweight' , 'bold' );
                xlabel( 'Variable number' );
                ylabel( 'Innovations' );
        end
end % END OF FUNCTION
```

VecAdaptiveKF.m

```
function KF_output=VecAdaptiveKF(S, Z, W, KF_options)
% KF_output=VecAdaptiveKF(S, Z, W, KF_options);
%
% This is a vectorised implementation of the adaptive Kalman filter and is
% based upon the description of the linear and adaptive Kalman filter
% published by Sarah C. Rutan et al. in Analytica Chemica Acta, 160
% (1984) 99-119.
% The Kalman filter calculations have been vectorised so if the user
% specifies a matrix of measurement data, the entire matrix of state
% parameter estimates will be updated during each iteration (variable k).
% The vectorised adaptive Kalman filter will produce identical results to
% the original Kalman filter for the same data set [Z], reference measurement
% functions [S] and measurement noise variance [R]. However the script
% will run much faster as a consequence of vectorisation.
% The script will run through the data twice; the state parameters are
% propagated from the first pass and the Kalman filter will run through a
% second time to recalculate the innovations. This will provide a better
% estimate of the final spectral lack-of-fit calculated using optimised
% state parameters.
% Unlike the original adaptive Kalman filter 'AdaptiveKF.m', the user is
% not given the option to augment the matrix of reference measurement
% functions with a new component, or update an existing reference
% measurement for one of the components.
%
% INPUT ARGUMENTS
%
% [S] is an (N x K) row matrix of reference measurement functions, such as
% pure component spectra. N is the number of individual components and K
% is the number of measurement variables.
%
% [Z] is a (J x K) matrix or (1 x K) vector of process measurement data where
% J is the number of observations (spectra) and K is the number of
% measurement variables.
%
% [W] is window size to be used for adapting the measurement variance
% estimates, Rk. W must be even-numbered
%
% [KF_options] is a structured array that can be used to provide
% additional optional input arguments. If the user does not provide any
% additional input arguments, the script will call the default values
% described below.
%
% [.Rmin] is the minimum measurement noise variance permitted.
% The actual measurement noise variance will be calculated adaptively
% during the Kalman filtering but the value of R is not permitted to drop
% below Rmin. This value should be two or three orders of magnitude lower
% than the expected measurement noise variance.
% The default value is 1.0E-6
%
% [.Xin] is a (J x N) matrix of initial estimates of the state parameters.
% Enter a row vector for a single measurement vector or a row orientated
% matrix (J rows) if a number of measurement vectors (spectra) are
% used.
% The default value is [] (empty field).
%
```

```
% [.Plotting] allows the user to specify whether results are plotted after
% Kalman filtering has been performed.
% The default value is 1 ([.Plotting] is on).
%
% OUTPUT ARGUMENTS
%
% KF_output is a structured array generated by this script to store the
% various outputs listed below.
%
% [.X] is a (J x N) matrix or (1 x N) row vector of the calculated state
% parameters, where J is the number of observations (spectra) and N is the
% number of state parameters (e.g. chemical components).
%
% [.G] is an (N x K) matrix of Kalman gains calculated during Kalman
% filtering.
%
% [.V] is a (J x K) matrix or (1 x K) row vector of innovation values for
% each iteration k of the Kalman filter.
%
% [.Pf] is an (N x N) matrix that stores the final error variance (the
% diagonal elements of the error covariance matrix P) at the K-th iteration.
%
% [.Rk] is a (J x K) matrix or (1 x K) vector of adaptive measurement error
% values calculated at each variable index k.
%
% [.LOF] is the average measurement lack-of-fit with respect to the original
% data for each measurement j. This produces a (J x 1) column vector. A
% large value indicates that the Kalman filter has fitted the original
% measurement vector with less accuracy and is a good indication of the
% presence of an un-modelled component.
%
% Nicholas I. Pedge
% Department of Chemistry, University of Hull, Cottingham Road, Hull,
% HU6 7RX
% Process R&D, AstraZeneca R&D Charnwood, Bakewell Road, Loughborough,
% Leicestershire, LE11 5RH
%
% function KF_output=VecAdaptiveKF(S, Z, W, KF_options);

% CHECK INPUT ARGUMENTS
% Check that an options structure array, KF_options has been provided
if nargin < 4
        fprintf( 'No options structure provided. Default values will be used \n' )
        fprintf( 'This default options structure will be stored in the output structure \n' )
        KF_options.Rmin=1E-6;
        KF_options.Xin=[ ];
        KF_options.Plotting=1;
end
% Check that the user provided KF_options is a structured array
if isstruct(KF_options)==0
        error( 'The input argument [KF_options] should be a structured array' )
end
% Check user has provided at least two input arguments
if nargin < 3
        error( 'Please provide [S], [Z] and [W]' )
end
```

```
%
% Check the dimensions of [S] and [Z].
[N,K]=size(S);
S_trans=S';
% N is the number of components in reference data
% K is the number of measurement variables in reference data
%
[J, K2]=size(Z);
% J is the number of observations or spectra (number of row vectors)
% K2 is the number of variables
%
if K ~= K2
        error( 'The number of variables in [S] and [Z] are not equal' )
end
%
% EXTRACT VALUES FROM [KF_options] OR INITIALISE AS REQUIRED.
Rmin=KF_options.Rmin;
fprintf( 'Minimum permitted measurement error variance has been set to: %E \n' , Rmin)
%
% If the user has not entered a vector or matrix [Xin] of initial estimates
% for the state parameters (e.g. component concentrations), [X] is set to a
% matrix of ones with dimensions (J,K).
if isempty(KF_options.Xin)==1
        X = ones(J,N);
        fprintf( 'No initial estimates of X were provided by user \n' )
else
        X=KF_options.Xin;
        fprintf( 'A matrix of initial estimates of X were provided by user \n' )
end
%
Plotting=KF_options.Plotting;
if Plotting==0
        fprintf( 'Display plots is set to off \n' )
else
        fprintf( 'Display plots is set to on \n' )
end
%
% PRE-ALLOCATE STORAGE VECTORS AND MATRICES
V = zeros(J, K);
%
% Matrix to store final state paramater estimate variance values.
Pf=zeros(N, N);
%
% Vector to store the innovations square-root, sum-of-squares
LOF=zeros(J,1);
%
% Set up output matrix for the adaptive measurement variance values.
% Rk=R*ones(J,K);
Rk=zeros(J,K); Rk(:,1)=Rmin;
%
% Set up output matrix for the Kalman gains
G=zeros(N, K);
%
%*************************************************************************
% Vectorised Adaptive Kalman Filter Calculations                         *
%*************************************************************************
```

```
tic
%
SecondPass=-1; % Counter so filter will make second pass through the data
%
while SecondPass < 1
        if Plotting==1
                hw=waitbar(0, 'Please wait...' );
        end
        %
        if SecondPass==-1
                P = cov(S');
                P = eye(N)*P*100;
        else
                P = Pf;
        end
        %
        for k = 2:K;
                if Plotting ==1
                        waitbar(k/K, hw)
                end
                %
                % State estimate extrapolation
                 X_old = X;
                % X(k|k-1) = X(k-1:k-1) Matrix notation
                %
                % Error covariance extrapolation
                P_old = P;
                % P(k|k-1) = P(k-1|k-1) Matrix notation
                %
                % Calculate innovation INV(:,k)
                V(:,k)=Z(:,k)-(X_old*S(:,k));
                % V(k) = z(k)- (X(k|k-1) * h(k)) Matrix notation
                %
                % Calculate adaptive measurement variance Rk
                % Only use available data if k < m
                %
                m = W;
                if m >= k          % If m is greater than or equal to the current
                        m = k-1; % iteration k, use k-1
                end
                %
                % Initialise V_sum matrix to zeros
                %
                V_sum=zeros(J, m);
                %
                for i=1:m;
                        V_sum(:,i)=(V(:,k-i).*V(:,k-i)); % square each element
                end
                %
                V_sum2=sum(V_sum,2);
                %
                % Update estimate of R, measurement error.
                % Rk is a vector containing measurement error R for each row of the
                % original data matrix.
                %
                Rk(:,k)=(inv(m).*V_sum2') - (S_trans(k,:)*P_old*S(:,k));
```

```
                % R(k) = 1/m [ j=1 Sigma m v(k-j).v(k-j)] - S(k)'.P(k|k-1).S(k)
                % Find maximum value of R for use in update of Kalman gain
                Rnew=max(Rk(:,k));
                %
                % Limit variance to prevent R approaching zero as that can result
                % in a singular error covariance matrix P.
                if Rnew<Rmin
                        Rnew=Rmin;
                end
                %
                % Update Kalman gain
                G(:,k) = (P_old * S(:,k)) * inv((S_trans(k,:) ...
                        *P_old*S(:,k))+Rnew);
                % k(k)=[P(k|k-1)*h(k)]*[(h(k)'*P(k|k-1)*h(k))+ R)]-1
                %
                % State estimate update
                g=G(:,k);
                X=X_old + (V(:,k)*g');
                % X(k|k)= X(k|k-1) + (INV(k)* k(k)') Matrix notation
                %
                % Error covariance update, numerically stable solution
                P=(eye(N)-(g*S_trans(k,:))) * P_old * ...
                        (eye(N)-(g*S_trans(k,:)))'+(g * Rnew * g');
                % P(k|k) = [I - k(k).S(k)'].P(k|k-1).
                %          [I-k(k).S(k)']' + k(k).R.k(k)'
        end % end k
        Pf=P;
        %
        if Plotting==1
                close(hw)
        end
        SecondPass=SecondPass+1;
end % end while
%
% ANALYSE INNOVATIONS SEQUENCE
%
% The innovations sequence contains the difference between the predicted
% measurement value at variable k and the actual measured value. If the
% Kalman filter is operating correctly, the innovations sequence for each
% observation should be white noise with zero mean. The square-root,
% sum-of-squares of each samples' innovations sequence should decrease as
% Kalman filtering improves.
VSSQ=sum((V.^2),2)./(K);
DATASSQ=sum((Z.^2),2)./(K);
LOF = (sqrt(VSSQ ./ DATASSQ))*100;
%
% This LOF statistic gives the average lack-of-fit per spectral data point
% as a percentage of the average data point in original data.
%
[LOF_Max, LOF_Index]=max(LOF);
%
toc
%*************************************************************************
% End of Vectorised Adaptive Kalman Filter Calculations                  *
%*************************************************************************
%
```

```
% OUTPUT RESULTS
KF_output.X=X;
KF_output.G=G;
KF_output.V=V;
KF_output.Rk=Rk;
KF_output.LOF=LOF;
KF_output.LOF_Index=LOF_Index;
KF_output.Pf=Pf;
KF_output.KF_options=KF_options;
%
if Plotting ==1
        if J>1
                figure()
                subplot(3,1,1)
                plot(1:J,X); axis tight
                title( 'State parameters' , 'fontweight' , 'bold' );
                xlabel( 'Sample number' );
                ylabel( 'State parameter' );
                %
                subplot(3,1,2)
                plot(1:K,V); axis tight
                title( 'Innovations' , 'fontweight' , 'bold' );
                xlabel( 'Variable number' );
                ylabel( 'Innovations' );
                %
                subplot(3,1,3)
                plot(1:J,LOF); axis tight
                title( 'Average LOF per variable point (%)' , 'fontweight' , 'bold' );
                xlabel( 'Sample number' );
                ylabel( 'LOF (%)' );
                %
                figure()
                subplot(2,1,1)
                plot(1:K,Rk'); hold on ; plot(Rk(LOF_Index,:)', 'linewidth' ,2)
                title( 'Adaptive variance (R)' , 'fontweight' , 'bold' );
                xlabel( 'Variable number' );
                ylabel( 'Variance' );
                subplot(2,1,2)
                plot(1:J,mean(Rk,2), 'linewidth' ,2)
                title( 'Adaptive variance (R)' , 'fontweight' , 'bold' );
                xlabel( 'Variable number' );
                ylabel( 'Variance' );
        else
                figure()
                plot(1:K,V); axis tight
                title( 'Innovations' , 'fontweight' , 'bold' );
                xlabel( 'Variable number' );
                ylabel( 'Innovations' );
        end
end
end % END OF FUNCTION
```

Create_Spectra.m

```
function  S_hat=Create_Spectra(Z,N,p_norm)
% This simple function can be used to create as set of random spectral
% estimates using a (N x N) transformation matrix of random numbers.  The
% spectral profiles are normalised using the p-normalisation specified by
% the user.
% INPUT ARGUMENTS
% [Z] is a (J x K) matrix of process measurement data where J is the number
% of observations (spectra) and K is the number of measurement variables.
%
% [N] is the number of initial pure spectral profiles to create.
%
% [p_norm] is the type of spectral normalisation to apply.
% Commonly used values are:
% [p_norm] = 2.  Normalise to unit length
% [p_norm] = inf Normalise to maximum value = 1 (normalise height)
% [p_norm] may also have values of 3, 4, 5 but this is uncommon.
%
% OUTPUT ARGUMENTS
% S_hat is a (K x N) matrix of random spectra created from the eigenvectors
% obtained by apply SVD to a data set spanning the same spectral space.
%
% Perform SVD on original data matrix
[U,S,V]=svd(Z,  'econ' );
%
% Truncate the matrix of right eigenvectors
V_bar=V(:,1:N);
%
% Create matrix of random numbers
T=randn(N,N);
%
% Create initial spectral estimates
S_hat=V_bar*T;
%
% Normalised the spectral profiles
for  n=1:N
        S_hat(:,n)=S_hat(:,n)/norm(S_hat(:,n),p_norm);
end
%
end % END OF FUNCTION
```

VAKFISO.m

```
function VAKFISO_output=VAKFISO(Z, N, S, VAKFISO_options)
% VAKFISO_output=VAKFISO(Z, N, S, VAKFISO_options);
%
% The function of VAKFISO is to find a matrix of reference measurement
% functions (pure spectral profiles) that minimise a weighted residual
% matrix when used to calculate the corresponding state parameters using
% the vectorised adaptive Kalman filter. The diagonal elements of the
% state-parameter error covariance matrix, corresponding to the variance of
% the state estimates will be minimised when the matrix of reference
% measurement functions accurately model the measurement data. To find the
% set of reference measurement functions that minimise the diagonal
% elements of the error covariance matrix, the elements of a transformation
% matrix are optimised using Newton-Gauss-Levenberg / Marquardt non-linear
% optimisation. During each iterative cycle, a new estimate of the
% optimised transformation matrix is calculated. Each spectrum in the
% matrix of test reference measurement functions is a linear combination of
% the primary eigenvectors spanning the spectral space. The transformation
% matrix is used to transform the eigenvectors into test reference
% functions and he vectorised adaptive Kalman filter then allows the
% state-parameters of all components for all available measurement vectors
% to be calculated simultaneously. As the matrix of test reference
% measurement functions approaches a feasible solution, the diagonal
% elements of the state-parameter error covariance matrix will be
% minimised. Without invoking any penalties, minimisation of the diagonal
% elements of the error covariance matrix or the residual matrix could
% correspond to negative spectra and / or negative state parameters. To
% prevent this, a weighted residual matrix is constructed from the initial
% innovations matrix but also includes additional terms to penalise large
% state-estimate variances as well negativity in the test spectra and
% estimated state-parameters.
%
% INPUT ARGUMENTS
% [Z] is a (J x K) matrix of process measurement data where J is the number
% of observations (spectra) and K is the number of measurement variables.
%
% [N] is the number of reference measurement functions (pure spectral
% profiles) to locate and optimise.
%
% [S] is a (K x N) matrix of initial estimates of the reference
% measurement functions. If [S] is not provided, initial estimates would be
% obtained by applying VVSP to the data
%
% [VAKFISO_options] is a structured array that can be used to provide
% additional optional input arguments. If the user does not provide any
% additional input arguments, the script will call the default values
% described below. The script 'VAKFISO_SetOptions' may be run to create an
% options structure with the default values.
%
% [.W] is window size to be used for adapting the measurement variance
% estimates, Rk during adaptive Kalman filtering. W must be even-numbered
%
% [.Rmin] is the minimum measurement noise variance permitted.
% The actual measurement noise variance will be calculated adaptively
% during the Kalman filtering but the value of R is not permitted to drop
```

```
% below Rmin. This value should be two or three orders of magnitude lower
% than the expected measurement noise variance.
% The default value is 1.0E-6
%
% [.p_norm] is the type of spectral normalisation to apply during VVSP
% analysis or when calculating the state parameters using the final
% estimates of the reference measurement functions. Acceptable values are
% [p_norm] = 2. Normalise to unit length
% [p_norm] = inf Normalise to maximum value = 1 (normalise height)
% [p_norm] may also have values of 3, 4, 5 but this is uncommon.
%
% [.alpha1] is a weighting coefficient applied to the original matrix of
% innovations (spectral residuals) V.
%
% [.alpha2] is a weighting coefficient applied to penalty term PI.
% PI corresponds to the sum of the diagonal elements of the state estimate
% error covariance matrix P. The default value is 1. Set this value to 0
% if PI should not contribute to the residual matrix used during NGL/M
% optimisation.
%
% [.alpha3] is a weighting coefficient applied to spectral negativity
% penalty term SIGMA. SIGMA will have a value of 0 if none of the
% reference measurement functions have any negative regions; and a value of
% 1 if all of the reference measurement functions are completely negative.
% The weighting coefficient [.alpha3] allows the weighting of the SIGMA term
% to be customised. The default value is 1. Set this value to 0 if SIGMA
% should not contribute to the residual matrix used during NGL/M
% optimisation.
%
% [.alpha4] is a weighting coefficient applied to state-estimate negativity
% penalty term XI. SI will have a value of 0 if none of the
% state parameter estimates have any negative regions; and a value of
% 1 if all of the state parameter estimates are completely negative.
% The weighting coefficient [alpha4] allows the weighting of the XI term
% to be customised. The default value is 1. Set this value to 0 if XI
% should not contribute to the residual matrix used during NGL/M
% optimisation.
%
% [.delta] is the shift to be added to the elements of the transformation
% matrix T during NGL/M optimisation. A suitable value would be in the
% range 1E-4 to 1E-6.
%
% [.mp] is the Marquardt parameter used to prevent divergence during the
% NGL/M optimisation step. This value is typically set to 1 or a similiar
% value to [.delta]
%
% [.mu] is the convergence tolerance limit calculated from the total
% sum-of-squares of the weighted residual matrix re, defined as
% (ssq_old-ssq_new)/ssq_old.
%
% [.MaxIterations] is the maximum number of iterations permitted. The
% NGL/M optimisation will terminate if the convergence tolerance value has
% not been reached and the number of iterations performed = MaxIterations.
%
% OUTPUT ARGUMENTS
% [VAKFISO_output] is a structured array generated by this script to store
```

```
% the various outputs listed below.
%
% [.S0] is a (K x N) matrix containing the initial estimates of the reference
% measurement functions. These will either be provided by the user or
% estimated by performing VVSP analysis
%
% [.Sf] is a (K x N) matrix containing the final, optimised estimates of the
% reference measurement functions.
%
% [.X] is a (J x N) matrix of state parameters calculated using the
% vectorised adaptive Kalman filter using the final, normalised estimates
% of the reference measurement functions [.Sf].
%
% [.Pf] is an (N x N) state parameter error variance-covariance matrix for
% the state parameters stores in [.X].
%
% [.G] is an (N x K) matrix of Kalman gains obtained by applying the
% vectorised adaptive Kalman filter to Z using the optimised reference
% measurement functions [.Sf].
%
% [.V] is a (J x K) innovations matrix obtained by applying the
% vectorised adaptive Kalman filter to Z using the optimised reference
% measurement functions [.Sf].
%
% [.Rk] is a (J x K) matrix of adaptive measurement noise variances obtained
% by applying the vectorised adaptive Kalman filter to Z using the
% optimised reference measurement functions [.Sf].
%
% [.LOF] is the average measurement lack-of-fit with respect to the original
% data for each measurement j. This produces a (J x 1) column vector. A
% large value indicates that the Kalman filter has fitted the original
% measurement vector with less accuracy and is a good indication of the
% presence of an un-modelled component.
%
% [.T] is the final optimised (N x N) transformation matrix used to create [.Sf].
%
% [.counter] is the number of iterations reached before the optimisation
% step was terminated. Termination may have occurred because the
% convergence tolerance was achieved or because the maximum number of
% iterations was reached.
%
% [.convergence] is the final convergence value when the optimisation step
% was terminated.
%
% [.sigma_t] is column vector containing the standard error for each of the
% elements in the final transformation matrix T
%
% [.VVSP_fw], [.VVSP_fwNorm], [.VVSP_SI], [.VVSP_DW] and [.VVSP_PR] are the
% outputs from the initial VVSP analysis of Z and are described in the
% VVSP.m help.
%
% Nicholas I. Pedge
% Department of Chemistry, University of Hull, Cottingham Road, Hull,
% HU6 7RX
% Process R&D, AstraZeneca R&D Charnwood, Bakewell Road, Loughborough,
% Leicestershire, LE11 5RH
```

```
%
% VAKFISO_output=VAKFISO(Z, N, S, VAKFISO_options)

% CHECK INPUT ARGUMENTS
% Check that an options structure array, VAKFISO_options has been provided
%
if nargin < 4 || isempty(VAKFISO_options)==1
        fprintf( 'No options structure provided. Default values will be used \n' )
        % Call function 'VAKFISO_SetOptions' to create options structure with
        % default values
        VAKFISO_options=VAKFISO_SetOptions;
end
%
if nargin < 3 || isempty(S)==1
        fprintf( 'No initial estimates of reference measurement functions provided \n' )
        fprintf( 'Vertex Vector Sequential Projection will be used to provide [S] \n' )
        S=[];
end
%
% Check the dimensions of [S] and [D].
if isempty(S)==0
        [K1]=size(S,1);
        % K1 is the number of measurement variables in reference data
        %
        [K2]=size(Z,2);
        % K2 is the number of measurement variables in Z
        %
if K1 ~= K2
        error( 'The number of variables in [S] and [D] are not equal' )
end
clear K1 K2 ;
end
%
[J, K]=size(Z);
%
if nargin < 2|| isempty(Z)==1 || isempty(N)==1
        error( 'VAKFISO requires the input arguments [Z] and [N]' )
end
%
%*************************************************************************
% Start VAKFISO                                                          *
%*************************************************************************
% Assign function handles. This improves performance when functions are
% called repeatedly.
fh_VVSP=@VVSP;
fh_VAKF=@VecAdaptiveKF;
fh_NGLM=@NGLM;
%
% Extract relevant options values from options structure 'VAKFISO_options'
p_norm=VAKFISO_options.p_norm;
%
%*************************************************************************
% Call VVSP to obtain initial estimate of S or use input argument S      *
%*************************************************************************
if isempty(S)==1
        % Call VVSP to obtain initial estimates S0
```

```
        VVSP_output=feval(fh_VVSP, Z, N+2, N, p_norm, 0);
        S0=VVSP_output.Sopt;
        Call_VVSP=1;
else
        % Use initial estimates provided by user
        S0=S;
        Call_VVSP=0;
end
%
%*************************************************************************
% Perform SVD to obtain right singular vectors (V)                       *
%*************************************************************************
[U, S, V]=svd(Z, 'econ' );
clear U; clear S;
V_bar=V(:,1:N);
clear V;
%
%*************************************************************************
% Call subfunction NGLM to optimise T                                    *
%*************************************************************************
% Calculate initial transformation matrix T0
T0 = pinv(V_bar) * S0;
%
% Vectorise the initial estimate of the transformation matrix
t0 = T0(:);
%
% Call the Newton-Gauss-Levenberq/Marquardt optimisation function
[T, Jn, counter, convergence]= feval(fh_NGLM, Z, t0, V_bar, VAKFISO_options);
%
%*************************************************************************
% Use final estimate of T to re-estimate Sf and X using VAKF             *
%*************************************************************************
% Extract relevant options values from options structure 'VAKFISO_options'
W=VAKFISO_options.W;
Rmin=VAKFISO_options.Rmin;
p_norm=VAKFISO_options.p_norm;
%
% Create options structure for Vectorised Adaptive Kalman filter
KF_options.Rmin=Rmin;
KF_options.Xin=[];
KF_options.Plotting=1;
%
% Use final estimate of transformation matrix T to calculate Sf
Sf = V_bar * T;
%
% Normalise the spectra
for n=1:N
Sf(:,n)=Sf(:,n) ./ norm(Sf(:,n), p_norm);
end
%
% Perform Kalman filtering using final estimate of S
VAKF_output=feval(fh_VAKF, Sf',Z, W, KF_options);
%
% Extract output variables from 'VAKF_output'
X = VAKF_output.X;
G = VAKF_output.G;
```

```
V = VAKF_output.V;
Rk = VAKF_output.Rk;
LOF = VAKF_output.LOF;
Pf = VAKF_output.Pf;
%
% Calculate standard errors for the transformation parameters
t=T(:);
SSQ=trace(V'*V);
% Degrees of freedom, nu
nu=(J*K)-length(t)-(N*K);
sigma_Z = SSQ ./ nu;
% Standard error for transformation parameters in T
sigma_t = sigma_Z * sqrt(diag(inv(Jn'*Jn)));
%
%*************************************************************************
% Output variables to structured array                                   *
%*************************************************************************
VAKFISO_output.S0=S0; % Initial estimate of reference functions
VAKFISO_output.Sf=Sf; % Final estimate of reference functions
VAKFISO_output.X=X; % Final estimate of state-parameters
VAKFISO_output.Pf=Pf; % Error covariance matrix for final state estimates
VAKFISO_output.G=G; % Kalman gain matrix obtained using Sf
VAKFISO_output.V=V; % Innovations matrix obtained using Sf
VAKFISO_output.Rk=Rk; % Adaptive variance values obtained using Sf
VAKFISO_output.LOF=LOF; % Spectral LOF values obtained using Sf
VAKFISO_output.T=T; % Optimised transformation matrix T used to create Sf
VAKFISO_output.counter=counter;
VAKFISO_output.convergence=convergence;
VAKFISO_output.sigma_t=sigma_t; % Standard error for elements of T
VAKFISO_output.options=VAKFISO_options;
if Call_VVSP==0;
        VAKFISO_output.VVSP_fw=[];
        VAKFISO_output.VVSP_fwNorm=[];
        VAKFISO_output.VVSP_SI=[];
        VAKFISO_output.VVSP_DW=[];
        VAKFISO_output.VVSP_PR=[];
else
        VAKFISO_output.VVSP_fw=VVSP_output.fw;
        VAKFISO_output.VVSP_fwNorm=VVSP_output.fwNorm;
        VAKFISO_output.VVSP_SI=VVSP_output.SI;
        VAKFISO_output.VVSP_DW=VVSP_output.DW;
        VAKFISO_output.VVSP_PR=VVSP_output.PR;
end
end % END OF FUNCTION 'VAKFISO'
%
%*************************************************************************
% Subfunction NGLM                                                       *
%*************************************************************************
function [T, Jn, counter, convergence]=NGLM(Z, t0, V_bar, VAKFISO_options)
% Assign function handle. This improves performance when functions are
% called repeatedly.
fh_VAKF_opt=@VAKF_opt;
%
% Extract relevant options values from options structure 'VAKFISO_options'
delta=VAKFISO_options.delta;
mp=VAKFISO_options.mp;
```

```
mu=VAKFISO_options.mu;
MaxIterations=VAKFISO_options.MaxIterations;
% Calculate initial sum-of-squares (ssq_old) using data matrix
ssq_old=trace(Z'*Z);
t=t0;
[J,K]=size(Z);
[N]=size(V_bar, 2);
% Pre-allocate storage array for Jacobian matrix Jn
Jn=zeros((J*K),(N*N));
% Initialise counter
counter = 0;
mpp=mp;
%
hw=waitbar(0, 'Performing spectral optimisation, please wait...' );
while 1
        % First call of VAKF to calculate vector of residuals re0
        re0 = feval(fh_VAKF_opt, Z, V_bar, t, VAKFISO_options);
        % Calculate sum-of-squares for starting estimate of t
        ssq_new=sum(re0.*re0);
        % Calculate convergence
        convergence=(ssq_old-ssq_new)/ssq_old;
        fprintf( 'Convergence %6.4f \n' , convergence)
        %
        % Determine whether convergence tolerance has been reached
        if abs(convergence) <= mu
                if mp==0
                        % If Marquardt parameter is also zero, exit 'while' loop
                        break
                else
                        % If Marquardt parameter is not zero, set mp=0 and confirm
                        % convergence by performing another iteration.
                        mp=0;
                        re0_old=re0;
                end
        elseif convergence > mu
                % If convergence is greater than convergence tolerance, reduce
                % value of Marquardt parameter and estimate new values of the
                % transformation vector t
                mp=mp/3;
                ssq_old=ssq_new;
                re0_old=re0;
                % Loop to update each element of the transformation vector t.
                % This is slice-wise numerical differentiation to create the
                % Jacobian matrix Jn
                for q=1:length(t)
                        % Add delta to current value of t(q)
                        t(q)=(1+delta)*t(q);
                        % Calculate residuals for shifted element t(q)
                        re = feval(fh_VAKF_opt, Z, V_bar, t, VAKFISO_options);
                        % Populate the q-th column of the Jacobian matrix Jn
                        Jn(:,q) = (re - re0)/(delta*t(q));
                        % Calculate new shift value for t(q)
                        t(q) = t(q)/(1 + delta);
                end
        elseif convergence < -mu
                if mp==0
```

```
                        % If divergence is observed and the Marquardt parameter is set to
                        % 0, set the parameter back to 1.
                        mp=mpp;
                else
                        % If divergence is observed but the Marquardt parameter is not
                        % set to 0, increase the value of the parameter.1.
                        mp=mp*5;
                end
                % If divergence is observed, take back the shifts added previously.
                t=t-delta_t;
        end
        % Augment Jacobian matrix with diagonal matrix of Marquardt parameters
        Jn_mp=[Jn; mp*eye(length(t))];
        % Augment residual vector with a vector of zeros
        re0_mp=[re0_old; zeros(size(t))];
        % Calculate transformation vector parameter shifts
        delta_t=-Jn_mp \ re0_mp;
        % Add the transformation vector parameter shifts to current t
        t=t+delta_t;
        % Increment counter
        counter = counter + 1;
        if counter >= MaxIterations
                fprintf( 'Maximum number of iterations reached \n' )
                break
        end
        waitbar(counter/MaxIterations, hw)
 end
close(hw)
%
% Re-matricise the final estimate of t to create transformation matrix T
[K, N] = size(V_bar);
T=zeros(N);
for n=1:N
        T(:,n) = t((n*N)-(N-1):(n*N));
end
%
end
% END OF SUBFUNCTION 'NGLM'
%
%*************************************************************************
% Subfunction VAKF_Opt                                                   *
%*************************************************************************
function re = VAKF_opt(Z, V_bar, t, VAKFISO_options)
% Assign function handles. This improves performance when functions are
% called repeatedly.
% fh_VAKF=@VecAdaptiveKF;
fh_VAKF=@VAKF;
fh_CWRM=@CWRM;
%
% Extract relevant options values from options structure 'VAKFISO_options'
W=VAKFISO_options.W;
Rmin=VAKFISO_options.Rmin;
p_norm=VAKFISO_options.p_norm;
alpha1=VAKFISO_options.alpha1;
alpha2=VAKFISO_options.alpha2;
alpha3=VAKFISO_options.alpha3;
```

```
alpha4=VAKFISO_options.alpha4;
%
%Create options structure for Vectorised Adaptive Kalman filter
KF_options.Rmin=Rmin;
KF_options.Xin=[];
KF_options.Plotting=0;
%
% Need to re-matricise the vector t to create the transformation matrix T
[K, N] = size(V_bar);
T=zeros(N);
for n=1:N
        T(:,n) = t((n*N)-(N-1):(n*N));
end
% disp(T)
% Use current estimate of transformation matrix to calculate S_hat
S_hat = V_bar * T;
%
% Normalise S_hat
for n=1:N
        S_hat(:,n)=S_hat(:,n) ./ norm(S_hat(:,n), p_norm);
end
%
% Apply Vectorised Adaptive Kalman filtering to Z using current estimate of
% S_hat
VAKF_output=feval(fh_VAKF, S_hat', Z, W, KF_options);
X=VAKF_output.X;
V=VAKF_output.V;
Pf=VAKF_output.Pf;
%
 % Call the subfunction CWRM using the function handle 'fh_CWRM' to
% calculate the weighted residual matrix E
re=feval(fh_CWRM, S_hat, VAKF_output, alpha1, alpha2, alpha3, alpha4);
%
% Plot results
subplot(4,1,1)
plot(X, 'linewidth' , 2); axis tight
title( 'State parameters' , 'fontweight' , 'bold' );
xlabel( 'Sample number' );
ylabel( 'Value (Arb.)' );
subplot(4,1,2)
plot(S_hat, 'linewidth' , 2); axis tight
title( 'Estimated reference measurement functions' , 'fontweight' , 'bold' );
xlabel( 'Variable number' );
ylabel( 'Intensity (Arb.)' );
subplot(4,1,3)
plot(V'); axis tight
title( 'Innovations' , 'fontweight' , 'bold' );
xlabel( 'Sample number' );
ylabel( 'Intensity (Arb.)' );
subplot(4,1,4)
bar(diag(Pf));
title( 'State estimate variance' , 'fontweight' , 'bold' );
xlabel( 'Component number' );
ylabel( 'Intensity^2 (Arb.)' );
drawnow
end
```

```
% END OF SUBFUNCTION 'VAKF_opt'
%
%*************************************************************************
% Subfunction CWRM                                                       *
%*************************************************************************
function re = CWRM(S_hat, VAKF_output, alpha1, alpha2, alpha3, alpha4)
% Extract relevant options values from output structure 'VAKF_output'
X=VAKF_output.X;
V=VAKF_output.V;
%Rk=VAKF_output.Rk;
Pf=VAKF_output.Pf;
%
[K, N]=size(S_hat);
[J]=size(X, 1);
%
% Calculate PI
PI=trace(Pf);
%
% Calculate SIGMA
OMEGA=(abs(S_hat)-S_hat)./2;
SIGMA = (trace(OMEGA'*OMEGA))./ (trace(S_hat'*S_hat));
SIGMA = SIGMA ./ N;
%
% Calculate XI
THETA=(abs(X)-X)./2;
XI = (trace(THETA'*THETA))./ (trace(X'*X));
XI = XI ./ N;
%
% Calculate weighted residual matrix
E=(alpha1.*V) + (alpha2.*PI.*V) + (alpha3.*SIGMA.*V) + (alpha4.*XI.*V);
A = (alpha1.*V);
B = (alpha2.*PI.*V);
C = (alpha3.*SIGMA.*V);
D = (alpha4.*XI.*V);
%
RMS_A=sqrt((trace(A'*A))./ (J*K));
RMS_B=sqrt((trace(B'*B))./ (J*K));
RMS_C=sqrt((trace(C'*C))./ (J*K));
RMS_D=sqrt((trace(D'*D))./ (J*K));
%

fprintf( 'RMS of (Alpha1 .* V) = %6.4E \n' , RMS_A)
fprintf( 'RMS of (Alpha2 .* PI .* V) = %6.4E \n' , RMS_B)
fprintf( 'RMS of (Alpha3 .* SIGMA .* V) = %6.4E \n' , RMS_C)
fprintf( 'RMS of (Alpha4 .* XI .* V) = %6.4E \n' , RMS_D)
fprintf( '---------------------------------\n' )
% Vectorise E for NGL/M calculations
re=E(:);
end
% END OF SUBFUNCTION 'CWRM'
```