NEW SIGNAL ENHANCEMENT ALGORITHMS FOR ONE-DIMENSIONAL AND TWO-DIMENSIONAL SPECTROSCOPIC DATA

by

ROD BLAINE FOIST

M.S. Electrical Engineering, University of Washington, 1989
B.S. Electrical Engineering, University of Washington, 1982

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

March 2011

©Rod Blaine Foist, 2011
Abstract

In research, the usefulness of the analytical results is crucially dependent upon the quality of the measurements. However, data measured by instruments is always corrupted. The desired information—the “signal”—may be distorted by a variety of effects, especially noise. In spectroscopy, there are two additional significant causes of signal corruption—instrumental blurring and baseline distortion.

Consequently, signal processing is required to extract the desired signal from the undesired components. Thus, there is an on-going need for signal enhancement algorithms which collectively can 1) perform high quality signal-to-noise-ratio (SNR) enhancement, especially in very noisy environments, 2) remove instrumental blurring, and 3) correct baseline distortions. Also, there is a growing need for automated versions of these algorithms. Furthermore, the spectral analysis technique, Two-Dimensional Correlation Spectroscopy (2DCoS), needs similar solutions to these same problems.

This dissertation presents the design of four new signal enhancement algorithms, plus the application of two others, to address these measurement problems and issues. Firstly, methods for one-dimensional (1D) data are introduced—beginning with an existing algorithm, the Two-Point Maximum Entropy Method (TPMEM). This regularization-based method is very effective in low-SNR signal enhancement and deconvolution. TPMEM is re-examined to clarify its strengths and weaknesses, and to provide ways to compensate for its limitations. Next, a new regularization method, based on the chi-squared statistic, is introduced and demonstrated in its ability for noise reduction and deconvolution. Then, two new 1D automated algorithms are introduced and demonstrated: a noise filter and a baseline correction scheme.
Secondly, a new two-dimensional (2D) regularization method (Matrix-MEM or MxMEM), derived from TPMEM, is introduced and applied to SNR enhancement of images. Lastly, MxMEM and 2D wavelets are applied to 2DCoS noise reduction.

The main research contributions of this work are 1) the design of three new high performance signal enhancement algorithms for 1D data which collectively address noise, instrumental blurring, baseline correction, and automation, 2) the design of a new high performance SNR enhancement method for 2D data, and 3) the novel application of 2D methods to 2DCoS.
Preface

A version of the following chapters has been previously published, or is in press, or has been accepted for publication. In all of the following papers, Dr. Turner (as co-supervisor) generally provided more supervision of the research than did Dr. Ivanov (supervisor) due to the area of specialization.

Chapter 2:

For the above paper, my contribution mainly involved participating in the research by doing extensive programming of the pre-existing algorithm (converting it from C to a C-Matlab format that facilitated rapid, user-friendly experimentation). My more intangible contributions consisted of having collaborative discussions about the work and doing a careful review/critique of the manuscript.

Chapter 3:

For the above paper, my contribution mainly involved writing a short section of the manuscript and doing a literature survey. Again, my more intangible contributions consisted of having collaborative discussions about the work and doing a careful review/critique of the manuscript.

Chapter 4:

For the above paper, my contribution mainly involved working with IT personnel to setup research software for the lead researcher and doing a literature survey. As before, my more intangible contributions consisted of having collaborative discussions about the work and doing a careful review/critique of the manuscript.

Chapter 5:

For the above paper, my contribution mainly involved testing the algorithm and writing a test report which critiqued the usefulness of the method. As before, my more intangible contributions consisted of having collaborative discussions about the work and doing a careful review/critique of the manuscript.
Chapter 6:

For the above paper, I invented the algorithm, did all of the software coding, performed the research, analyzed the data, and prepared the manuscript under the primary guidance of Dr. Schulze.

Chapter 7:

For the above paper, I did all of the software coding, conceived a novel use of 2D wavelets for this context, performed the research, analyzed the data, and prepared the manuscript under the primary guidance of Dr. Schulze.
Table of Contents

Abstract ................................................................................................................................. ii
Preface ................................................................................................................................. iv
Table of Contents ................................................................................................................ vi
List of Tables ........................................................................................................................ x
List of Figures ....................................................................................................................... xi
List of Abbreviations ........................................................................................................... xiii
Acknowledgements ............................................................................................................. xiv
Dedication ............................................................................................................................. xvi

Chapter 1: Introduction ....................................................................................................... 1
  1.1 Scientific Measurements: Silver and Dross ................................................................. 1
      1.1.1 A Wider Context and a Background Theme ....................................................... 1
  1.2 Spectroscopy: Shedding Light on the Matter ............................................................. 2
      1.2.1 Overview ............................................................................................................ 2
      1.2.2 Spectral Analysis and Two-Dimensional Correlation Spectroscopy .............. 7
      1.2.3 Measurement Problems and Issues .................................................................... 8
  1.3 Signal Processing: Separating the Silver from the Dross ......................................... 9
      1.3.1 Engineering vs. Chemistry ............................................................................... 9
      1.3.2 Signals and Noise ............................................................................................ 10
      1.3.3 Signal Enhancement Methods: Seeking a Good Estimate............................... 12
      1.3.4 Summary of Algorithms Presented .................................................................... 15
      1.3.5 Equations for Data Characterization ............................................................... 16
  1.4 Thesis Objectives ...................................................................................................... 16
  1.5 Thesis Organization and Summary of Contributions ............................................... 17

Chapter 2: TP MEM Re-examined--An SNR Enhancement and Deconvolution Algorithm ................................................................................................................................. 19
  2.1 Introduction ................................................................................................................. 19
  2.2 Theory ......................................................................................................................... 21
  2.3 Methods ..................................................................................................................... 23
  2.4 Results and Discussion ............................................................................................ 25
      2.4.1 Elevated (High Baseline-to-Noise Ratio) Baselines and Baseline Removal .... 25
      2.4.2 Elevated (High Signal-to-Noise Ratio) Peaks and Spectrum Inversion .......... 30
      2.4.3 Application to Experimental Data ................................................................. 36
      2.4.4 Readout-to-Noise Ratio Scope of Two-Point Maximum Entropy Method .... 38
  2.5 Conclusions .............................................................................................................. 44

Chapter 3: Chi-Squared-based Filters for SNR Enhancement and Deconvolution .......... 45
  3.1 Introduction ............................................................................................................... 45
Appendix 1

Chapter 8: Conclusions and Future Work

Chapter 7: Application of 2D Methods to Two-Dimensional Correlation Spectroscopy

Chapter 8: Conclusions and Future Work

References

Appendix 1—Data Characterization
Appendix 2—MxMEM Gradient Calculation (concept and coding) .......................... 194
Appendix 3—MxMEM Hardware Acceleration: Preliminary Investigations .......... 197
Appendix 4—MxMEM Implementation Details and C Code .............................. 203
Appendix 5—Fixed-Point Arithmetic: Java Code .............................................. 215
List of Tables

Table 1.1 Algorithm Descriptions

Table 2.1 TPMEM vs. SG filter: Quantitative Comparison (synthetic data)

Table 3.1 BiChi vs. SG filter: Quantitative Comparisons (synthetic data)

Table 3.2 BiChi Quantitative Results on Real Raman Data

Table 4.1 iSG Filter vs. Median Filter: Quantitative Comparison (synth. data)

Table 5.1 Auto-Baseline-Correction Method: Quantitative Results

Table 6.1 MxMEM vs. 2D TPMEM: Quantitative Comparison (CT synth. data)

Table 6.2 MxMEM vs. TPMEM: Quantitative Comparison (Raman synth. data)

Table 7.1 Intensity and Frequency Changes of Peaks (2DCoS synth. spectra)

Table 7.2 2DCoS Noise Reduction: Comparison of Three Methods

Table 7.3 Comparison of 2DCoS Maps Data

Table 8.1 Algorithm Summary and Selection Guide
List of Figures

Figure 1.1 The Electromagnetic Spectrum ................................................................. 4
Figure 1.2 Raman Spectra Examples ........................................................................ 5
Figure 1.3 Simplified Raman Spectroscopy System Diagram ......................................... 7
Figure 1.4 Raman Synthetic Spectra with Distortions .................................................... 8
Figure 2.1 TPMEM Regularization of Raman Peaks (various backgrounds) ............... 27
Figure 2.2 TPMEM Regularization with Baseline Removed Prior to Smoothing .......... 29
Figure 2.3 TPMEM Regularization of ‘Inverted’ Spectrum ........................................... 32
Figure 2.4 TPMEM Regularization vs Savitzky-Golay Smoothing ............................... 35
Figure 2.5 TPMEM vs. SG filter: Difference Spectrum of Salmon Sperm .................. 37
Figure 2.6 The Effect of RNR on Noise Discrimination ................................................ 40
Figure 2.7 SNR Enhancements as a Function of RNR: SG filter vs. TPMEM ............... 42
Figure 3.1 BiChi Filter Results across Various Initial SNR Values ............................. 54
Figure 3.2 Smoothing Results: BiChi Methods vs. SG Filter on Synthetic Data ......... 55
Figure 3.3 Typical Spectrum after BiChi Regularization (no deconvolution) ............... 57
Figure 3.4 Spectra after BiChi regularization (with deconvolution), lambda=3 .......... 58
Figure 3.5 Spectra after BiChi regularization (with deconvolution), lambda=30 ......... 59
Figure 3.6 BiChi Regularization of Real Raman Data .................................................. 62
Figure 4.1 Coefficients and Frequency Response of Iterative SG Filter .................... 72
Figure 4.2 Spectra Smoothed with iSG and Median Filters (synth. data) .................. 76
Figure 4.3 Detail of Spectra Smoothed with iSG and Median Filters ......................... 79
Figure 4.4 Real Raman Spectra Smoothed with iSG and Median Filters .................. 80
Figure 5.1 Raman Spectra Superimposed on Three Baseline Types ......................... 89
| Figure 5.2 | A Flowchart of the Automated Baseline Correction Method | 92 |
| Figure 5.3 | Automated Baseline Correction for Exponential Baselines | 94 |
| Figure 5.4 | Automated Baseline Correction for Gaussian Baselines | 95 |
| Figure 5.5 | Automated Baseline Correction for Sigmoidal Baselines | 96 |
| Figure 5.6 | Spectrum Subjected to Repeated Baseline Correction | 99 |
| Figure 5.7 | Real Raman Spectra (from tomato skin) after Baseline Correction | 102 |
| Figure 5.8 | ChiSquare-value “Error” Surface of Baseline Correction | 104 |
| Figure 6.1 | A Conceptual “Image” plus CT Data Images | 111 |
| Figure 6.2 | Raman Synthetic Data Smoothing: MxMEM vs. TPMEM | 122 |
| Figure 6.3 | Real DNA spectra smoothing: MxMEM vs. TPMEM | 130 |
| Figure 6.4 | Detail Views of Synthetic and Real Data: MxMEM vs. TPMEM | 131 |
| Figure 7.1 | Raman Synthetic Data: 1D Specta vs. 2DCoS Ideal Synch. Corr. Maps | 145 |
| Figure 7.2 | Synch. Corr. Maps: Pre- and Post-treated by 3 Methods | 151 |
| Figure 7.3 | Detail of Selected Asynch. Corr. Maps and their Source Data | 154 |
| Figure 7.4 | Synch. Corr. Coefficient Maps: Pre- and Post-treated by 3 Methods | 159 |
| Figure 7.5 | Synch. Corr. Maps Derived from Subtly Changing Synthetic Spectra | 162 |
| Figure 7.6 | Selected 2DCoS Maps Derived from Measured DNA Spectra | 166 |
| Figure A2.1 | A Conceptual Entropy-gradient Matrix | 195 |
List of Abbreviations

1D  One-Dimensional
2D  Two-Dimensional
2DCoS  Two-Dimensional Correlation Spectroscopy
ABE  Auto-Baseline-Estimate
ADC  Analog-to-Digital Converter
BCF  Bi-Chi Filter
cA1  First Level Approximation Coefficients
cA2  Second Level Approximation Coefficients
cD2  Second Level Detail Coefficients
CGM  Conjugate Gradient Method
CT  Computed Tomography
EMR  Electromagnetic Radiation
FIR  Finite Impulse Response
FTs  Fourier Transforms
FWHM  Full-Width-Half-Maximum
IPSF  Instrumental Point Spreading Function
iSG  Iterative Savitzky-Golay
KZ  Kolmogorov-Zurbenko
med3  Median Filter for Three-Dimensional Electron Tomographic Data
MM  Matrix-MEM
MWA  Moving Window Averaging
MxMEM  Matrix-MEM
NMR  Nuclear Magnetic Resonance
NRC  Numerical Recipes in C
PCA  Principal Components Analysis
PEARSC  Pearson’s Correlation Coefficient, Complemented
PHME  Peak-Height-Matching Error
RMS  Root-Mean-Square
RMSE  Root Mean Squared Error
RNR  Readout-To-Noise Ratio
RS  Raman Spectroscopy
SBR  Signal-To-Baseline Ratio
SD  Standard Deviation
SERS  Surface-Enhanced Raman Spectroscopy
SG  Savitzky-Golay
SNR  Signal-To-Noise-Ratio
TPMEM  Two-Point Maximum Entropy Method
UV  Ultraviolet
UVRRRS  Ultraviolet Resonance Raman Spectroscopy
W1D  One-Dimensional Wavelets
W2D  Two-Dimensional Wavelets
Acknowledgements

Having come through the most difficult thing that I’ve ever done—earning a PhD—I must acknowledge the significant help of others. I believe that it was by the grace of God, with the help of people, that I succeeded.

First, I thank my dear professors, Dr. André Ivanov and Dr. Robin Turner, and my dear mentor, Dr. Georg Schulze for persevering with me in this long “marathon”. I am grateful to André for inviting me to become his student, and I thank him (and wife Katherine) for several delightful times of inviting us students into their home for delicious meals and warm hospitality. Thanks also to André for granting my request to do something with “science”—even though I didn’t know what I was asking for! This led to the highly interdisciplinary nature of my research. In this, I am particularly indebted to Robin and Georg—for it was their field of study that became dominant in my work. Their support and guidance was so crucial, so enduring, and so helpful. They forced me to grow and coached me so effectively to train my mind in research. I am especially grateful for Robin’s wise counsel on several occasions, the long hours put in on my behalf (even on weekends), and the quiet excellence demonstrated in his work as my professor. In addition I am grateful for the help and guidance of the professors who were on my committee: Dr. Guy Lemieux, Dr. Lukas Chrostowski, and Dr. Hussein Alnuweiri.

I would also like to thank Dr. Andrew Jirasek and Dr. Junning Li for their special help at a very difficult time during the early stages of my research. Their assistance turned the tide when I was deep in “the valley of despair”.

Throughout the long, difficult “race”, my excellent and wonderful wife, Jean Marie Foist, continued faithfully as the tender, sensitive, mentor of my soul. She is that kind of
woman spoken of in Proverbs 31: “the heart of her husband trusts in her…she does him 
good, and not harm, all the days of her life.”

I am also thankful for the friendship, encouragement, and help of students, staff, and 
researchers in the SoC Lab and Turner group. I especially thank Roozbeh Mehrabadi, Dr. 
Roberto Rosales, Dr. Cristian Grecu, Dr. Sohaib Majzoub, Dr. Dipanjan Sengupta, Dr. Stas 
Konorov, Dr. Marcia Yu, and Dr. Chris Addison.

Also, I am deeply grateful to family and friends around the world for their 
couragement and prayers.

Finally, I would like to imitate the great composer J.S. Bach who appended each of 
his cantatas with the initials “SDG”—standing for Soli Deo Gloria (in Latin): To God alone 
the glory!
Dedication

In Loving Memory of my father, Irvin Glen Thomas Foist  (18/8/1928-5/12/1988),
And to my dear mother, Betty Jane Foist, on her 80th birthday,
And to my dear father-in-law, Byron Lee Deffinbaugh, on his 90th birthday,
And to my dear mother-in-law, Evalyn Marie Deffinbaugh.
Chapter 1: Introduction

1.1 Scientific Measurements: Silver and Dross

Measurements are vitally important in science and engineering—in order to obtain information (data) that can lead to understanding of how things work and for solving problems. In chemistry, and many other fields, this information is often acquired using a technique known as spectroscopy. However, all analytical measurements contain a combination of some desired information—known as the “signal”—and some other corrupting influences. Noise is the most common and pervasive problem.

It is therefore necessary to employ (post-measurement) signal processing methods which seek to extract the information—the signal (really only an estimate of the signal)—from the measured data. This work is about the design and application of several post-processing, or signal-enhancement, algorithms which address some of the most common and difficult problems of separating the signal from the measured data. The suite of tools presented here is primarily aimed at the field of spectroscopy, but most are useful in other application areas also.

1.1.1 A Wider Context and a Background Theme

Although this doctoral work is science-oriented, it was born out of an engineer-educator’s interest in microelectronics, science, and a passion to help engineering students get a good education and find their way in our rapidly advancing world. Chow and Gale have argued, from the context of microelectronic/microsystem design [1], that “multidisciplinary teams”—engineers working with scientists—will be increasingly important in this new age of nanotechnology. With that as inspiration, some preliminary
work, as described in Appendix 3, was done concerning a microelectronic chip design that would speed-up the processing of the algorithm presented in Chapter 6.

1.2 Spectroscopy: Shedding Light on the Matter

1.2.1 Overview

1.2.1.1 Fundamentals

Spectroscopy, in general, is the study of spectra (or frequency components of electromagnetic radiation). It is often used in physical and analytical chemistry to identify substances through the spectrum which they emit, absorb, or scatter. More precisely, spectroscopy is the study of the interaction of electromagnetic radiation (EMR) with matter [2]. The component within a sample, being analyzed in this manner, is called the “analyte”.

Atoms (and molecules) can only change their internal energies by certain discrete (quantized) amounts—depending on their state, structure, environment, etc. Based on this phenomenon, an instrument, known as a spectrometer, can be used to analyze these changes in energy (“transitions”) using EMR as a “probe”. This information is used to determine which elements (or molecules) are present, as well as other characteristics about them (e.g., structure). For these reasons, spectroscopy is very useful for the study of molecules in general [2], and also biomolecules in particular. When molecules absorb or emit photons, they change from a higher to a lower (or lower to higher) electronic energy state. This is called an electronic transition. In addition, molecules can go through energy transitions due to vibrational and rotational motion [3]. When the atoms of a molecule vibrate about their equilibrium position, they exhibit quantized energy levels corresponding to the wavelengths
of photons with which they interact. Similarly, there are quantized rotational energy levels related to certain photon wavelengths.

1.2.1.2 Types of Spectroscopy

There are many types of spectroscopy. Each type uses a different part of the electromagnetic spectrum (Figure 1.1) for investigating characteristics of a sample (analyte) [3]. More specifically, they may also use different kinds of physical interactions between the EMR and matter for probing samples. For example, some spectroscopies rely on absorption of EMR within a certain wavelength range, while others (like Raman) are based on scattering properties. Some of the major types of spectroscopy include: Atomic, Infrared (IR), Mass, Nuclear Magnetic Resonance (NMR), Raman, Ultraviolet/Visible, and X-ray [3]. The applications are far ranging. For example, Atomic spectroscopy can be used to measure Mercury content in seafood [4], Infrared spectroscopy is used to characterize arthritis [5], Mass spectroscopy has been used to identify strains of bacteria [6], NMR spectroscopy can non-invasively detect certain cancers by chemical analysis of human bile [7], UV spectroscopy has been employed for many years to study the earth’s atmosphere [8], and X-ray spectroscopy has been long used to study lead in human bones [9]. Finally, Raman spectroscopy (RS) is particularly useful in the analysis of biomolecules [10][11][12][13]. Because all of the algorithms presented in this work were developed to be useful within RS, additional detail will be given to this area.
1.2.1.3 Raman Spectroscopy

In 1930, Professor C. V. Raman received the Nobel Prize for his discovery that some of the light scattered by a molecule will undergo a change in wavelength (relative to the incident beam wavelength) [14]. Most of the incident light scatters (or reflects) in all directions. This occurs in an elastic manner, that is, with no change in frequency. But a small portion of the photons interact with the molecule due to its vibrational motion. These experience a frequency change, due to having an energy exchange with the molecule. This “Raman effect” is extremely weak—producing a light intensity typically $10^{-8}$ of the incident beam’s intensity—but is rich in information [15].

A spectrometer can be used to collect the Raman light and display it as a “spectrum”, where intensity is plotted as a function of frequency shift (relative to the incident light) [14]. Each type of molecule has its own unique set of vibrations, with corresponding frequencies. The Raman spectrum consists of a series of peaks that have been shifted according to the characteristic vibrational frequencies of the molecule. Figure 1.2 shows spectra from standard Raman spectroscopy and an advanced form known as Surface-Enhanced Raman Spectroscopy (SERS) [16]. This latter type, when the sample is adsorbed to certain metals (such as gold), can enhance the normally weak signal. Raman frequency shifts are typically
measured in “wavenumbers” (cm\(^{-1}\)). The set of peaks in the spectra are unique for this molecular mixture. On this basis, the Raman spectrum can provide a “fingerprint” of a material being studied, and reveal much about its structure and composition.

![Figure 1.2](image.png)

**Figure 1.2** Raman spectra examples: (a) SERS spectra of 1 μM 2-MPy on Au nanostars and (b) Raman spectra of 0.1 M 2-MPy. Inset shows chemical structure of 2-MPy molecule. Traces are offset in intensity for clarity. From public domain [16]

1.2.1.4 Ultraviolet Resonance Raman Spectroscopy (UVRRS)

Standard Raman spectroscopy (RS) has major drawbacks which inhibit its use for investigating biomolecules [10]. One of the major problems, particularly in the visible light range, is the presence of a large background fluorescence. Depending on the substance being investigated, this fluorescence may swamp out the weak Raman “signal” and thus make the measurement virtually impossible. However, most biological systems absorb radiation in the Ultraviolet (UV) range. In so doing, they resonate with the UV excitation, and yield highly intense spectra. In the context of Raman spectroscopy, this is known as the “resonance Raman effect”. When the excitation is in the UV range, this method is called “Ultraviolet resonance Raman spectroscopy” (UVRRS). In summary, the quality, sensitivity, and
selectivity of information available from UVRRS make it a powerful tool for a large class of biomolecule investigations [10]. Finally, for a mathematical and quantum-based explanation of Raman theory, the reader is directed to Yu [17] for a brief summary, or to Gardner and Graves for a more complete theoretical presentation [15].

1.2.1.5 Simplified UVRRS System Diagram

Figure 1.3 shows a conceptual diagram of a typical UVRRS system. A monochromatic Laser source is used to stimulate a molecular sample. Reflected (scattered) light from the Raman-effect will be slightly shifted in frequency from that of the laser source. The Filter removes the original frequency (which is also very strong). The Disperser, a prism-like component, is used to separate and disperse the remaining frequencies over the Detector’s array of channels. Each channel detects a specific frequency and generates an output voltage which is proportional to the incoming light intensity. The Analog-to-Digital Converter (ADC) then converts the voltages into a digital format readable by the Personal Computer (Signal Processor).

The actual spectrometer is represented (in simplified form) by the Disperser and Detector. In a real spectrometer, such as a monochromator, the Disperser is composed of a diffraction grating, plus entrance and exit slits, and the Detector is some sort of photodetector [10]. The slits cause an instrumental blurring, as explained below.
1.2.2 Spectral Analysis and Two-Dimensional Correlation Spectroscopy

After spectral data have been acquired from a spectroscopy system, it is necessary to perform spectral analysis in order to gain the desired information about the chemical system. In the best cases (if noise and other distortions are very slight), post-processing is not even required. If the data involves only one spectrum the analysis may be relatively simple in nature. In other scenarios, the data may involve a series of measurements (say over a range of temperatures), and if it’s derived from a very complex chemical system, then a much more sophisticated analysis method would be required, such as 2DCoS.

Two-dimensional correlation spectroscopy (2DCoS) is a powerful spectral analysis technique that has gained wide acceptance across many fields of spectroscopy in the last 20 years [18][19][20][21][22][23][24]. This mathematical/geometrical method transforms 1D data into a 2D format that helps to visualize the spectral data. A set of ordinary, one-dimensional (1D) spectra—measured under influence of an external perturbation—is used to generate two-dimensional (2D) maps of spectral intensity. These 2D maps are often able to clarify and simplify the available spectral information—especially in highly complex chemical systems—in a way not readily attainable from the original 1D spectra alone.

However, the technique’s usefulness is limited by noise and other measurement artifacts which may severely distort the true information. Consequently, some form of “pre-
treatment” of the 1D spectra is often performed before generating the 2D maps (e.g., noise reduction). The same signal enhancement methods (presented below) that are used for standard spectral analysis are usually applied in 2DCoS pre-treatment.

1.2.3 Measurement Problems and Issues

In this work we address three of the most common and difficult problems (or artifacts) that arise in spectroscopic measurements: noise, instrumental blurring, and baseline distortions/fluctuations. These are illustrated in Figure 1.4, showing (a) a synthetic Raman spectrum of seven peaks on a level background which represents an idealized measured spectrum (which includes peak broadening, but otherwise is ideal), (b) a background distortion has affected the signal, (c) a convolutional blurring has affected both the peaks and the background, and (d) noise has been added to the total spectrum.

![Figure 1.4](image)

**Figure 1.4** Raman synthetic spectra showing stages of adding distortions. (a) idealized spectrum, (b) the spectrum of (a) with a sigmoidal-shaped background added, (c) the spectrum of (b) affected by instrumental blurring, (d) the spectrum of (c) with random noise (of constant amplitude) added.

The spectrum in (d) can be modeled as
\[ m = (b + x) * p + n \]  

(1.1)

where \( m \) is a vector of measured data (from \( N \) channels of a detector); the underlying signal vector, \( x \), plus the baseline, \( b \), is convolved (*) with an instrumental blurring function, \( p \); and measurement noise, vector \( n \), is added.

Furthermore, there are two issues related to the measurement problems. First, in some applications the use of high-throughput instruments and real-time data collection are creating a great need for *automated* versions of algorithms that mitigate these measurement problems. We also address the need for automation. Second, we also address the noise problem within 2DCoS. Overall, our approach is to apply new and existing signal processing algorithms to these measurement problems and issues.

1.3 Signal Processing: Separating the Silver from the Dross

1.3.1 Engineering vs. Chemistry

Within electrical engineering, signal processing is a very broad and eclectic field concerned with modifying or analyzing signals in order to perform some useful operation on them [25][26]. The signals are electrical (analog or digital) representations of physically varying quantities [27][28]. Thus, the signals may be continuous or discrete in form.

Extracting information from the signal is only one of many important operations. Although electronic circuits may be used to implement the desired operations, in the case of *digital* signal processing, signal enhancement *algorithms* are usually employed to perform the operations of interest.

Within chemistry, and particularly analytical chemistry, signal processing is concerned with a variety of operations performed on continuous (analog) or discrete (digital) signals derived from analytical measurements. Here, the objective generally is to separate
the useful part of the signal from that which is not useful. Signal processing, whether as part of a measurement system or in post-processing, plays a critical role in ensuring the quality of analytical measurements [29][30].

1.3.2 Signals and Noise

In the most general sense, ‘noise’ is anything that obscures a desired signal. Even another signal can be noise (or “interference”). However, as is commonly done, in this work, the term ‘noise’ will refer to “random” noise [31]. And the term ‘signal’ will denote a discrete (digitized) measurement (say of a spectrum) which is composed of a pure, undistorted signal (spectrum) corrupted by noise, or noise plus other artifacts. Therefore, this work deals only with digital signal processing algorithms where the signal is a vector (1D data) or matrix (2D data) of discrete measured data.

1.3.2.1 Types of Noise

Random noise in an analytical signal can be characterized by its frequency spectrum, its (statistical) distribution, and the physical mechanism which causes it [2][29][30]. There are three main types.

1.3.2.1.1 Thermal Noise

Thermal (or Johnson) noise is caused by the thermal agitation of charged particles in the measurement instrumentation. It has a flat frequency spectrum—meaning that it has the same noise power at any frequency (up to some limit). Such noise is also called “white noise” in analogy to white light which is composed of all visible frequencies. Thermal noise follows a normal, or Gaussian, statistical distribution.
1.3.2.1.2 Shot Noise

This noise is due to (random) statistical fluctuations in electric current flow within the instrumentation. Shot noise, like thermal noise, is Gaussian and white.

1.3.2.1.3 Flicker Noise

This noise is also known as ‘1/f noise’ because it has approximately a 1/f frequency spectrum (its magnitude is inversely proportional to the frequency of the signal of interest). The causes of flicker noise are not well understood, but it becomes a significant problem at frequencies below 100 Hz [3].

Since flicker noise is a low frequency phenomenon, in this work, unless otherwise stated, noise is modeled as being white noise.

1.3.2.2 Instrumental Blurring

The spectrometer (or monochromator) described in Figure 1.3 contains a narrow entrance slit which effectively causes a convolutional (or blurring) effect on the light that passes through it. That is, the data are convolved with the transfer function of the spectrometer. This blurring was illustrated in Figure 1.4. For a given instrument, this effect can be measured (yielding a vector of data points) and is known as the Instrument Point Spread Function (IPSF). If it is known, the IPSF vector can then be used in a deconvolution process for “deblurring” the signal.

1.3.2.3 Baseline Problems

In Raman spectroscopy there is often a problem with other light interactions, within the sample, emitting a fluorescence that causes the measured spectrum to appear to sit on a broad, non-uniform ‘hump’ [32]. For example, in Raman bone tissue studies, the green
lasers that are most commonly used cause bone proteins to fluoresce, giving a background several orders of magnitude higher than the Raman signal [33].

1.3.3 Signal Enhancement Methods: Seeking a Good Estimate

In analytical measurements, the pure, underlying signal can never be truly recovered. Only an estimate can be obtained. Therefore, signal enhancement methods seek to obtain the best possible estimate. To address the measurement problems given in this work, three types of signal enhancement methods have been used. These will be introduced here, but developed further in later chapters. Also, the need for these methods, and the existing research, will be given in more detail in those chapters.

1.3.3.1 Filters—For Noise Reduction and Baseline Estimation

1.3.3.1.1 Overview

Under this topic, only digital filters are discussed because they are the most widely used in analytical chemistry. In standard definition, the filtering operation of digital filters consists of the convolution of a series of filter coefficients with the signal [28]. But this is not a universal definition. Alternately, digital filters can be classified according to their manner of operation.

1.3.3.1.2 Savitzky-Golay (SG) filters

Polynomial least-squares smoothing filters are the most commonly used digital filter in analytical chemistry and in 2DCoS for noise reduction [29][18]. Introduced into the literature by Savitzky and Golay [34], they are known as SG filters. Their functionality is equivalent to a non-recursive digital filter [28], also known as a finite impulse response (FIR)
filter. SG filters are a type of FIR low-pass filter [35]. Low-pass filters are appropriate for spectroscopic applications where a slowly varying signal is corrupted by random noise.

Unlike most digital filters, which have their properties defined in the Fourier domain, and then translated to the time domain, SG filters are derived directly in the time domain [36]. These filters were designed to be an improvement over the simpler moving window averaging (MWA) type of filter, where each data point is replaced with an average of some neighboring points. In spectroscopic applications, MWA filtering tends to distort by causing narrow spectral lines to be shortened and widened. SG filters minimize this distortion by approximating the underlying function within the window not by a constant (the average) but rather by a higher-order polynomial. At each point in the spectrum, a polynomial is least-squares fitted to include all points in the moving window. The point being smoothed is then set equal to the value of the corresponding point of the polynomial. The other points in the polynomial are not used. When the window is moved, a new polynomial is computed and fit as before. Nevertheless, a zero-order SG filter is exactly equivalent to a MWA filter (i.e., all coefficients are equal for a given window size).

SG filters are important because they are a “gold standard” in the literature (for SNR enhancement). In this thesis, they will be used for comparison. Furthermore, in two cases, they will form the basis of new designs—a noise filter (Chapter 4) and a baseline estimator (Chapter 5). Most discussions of SG filters focus only on their time-domain properties. For an excellent frequency-domain analysis of these filters, see [35].

1.3.3.2 Regularization Methods—For Noise Reduction and Deconvolution

These methods are useful for performing deconvolution (of the instrumental blurring) and/or SNR enhancement. Regularization is a technique used for solving inverse problems
Inverse problems involve determining a cause from an effect [37]. In the context of signal processing, the effect can be seen as a vector (a spectrum) of measured data (signal plus “noise”). The goal is to get to the cause of this data: conceptually, a pure signal was radiated by some chemical system and then corrupted. By solving the inverse problem one can reconstruct an estimate of the signal.

However, inverse problems (in numerical applications) are often unstable due to the corruption and incompleteness of the data. Such problems are referred to as being ill-posed, which means that they might not have a solution, or a unique solution, and/or might not depend continuously on the data. Regularization methods restore stability by adding extra information, usually in the form of smoothness to the data, and thus create an approximate solution [38].

The methods used here reconstruct an estimate of the signal by minimizing (optimizing) an objective (or cost) function that contains a smoothness and a fidelity component. This is done by using the gradient (first derivative) of the objective function to iteratively guide the minimization downhill until a “stopping criterion” is reached. 1D and 2D regularization methods are presented in this work.

1.3.3.3 Wavelets

Wavelet transforms (WTs) are widely used in engineering and chemistry for signal processing [39][40]. They are similar to Fourier Transforms (FTs) in that both can be used for “denoising” and data compression. An important advantage of wavelets, in spectroscopic applications, is the ability to model spectral line shapes with a basis function that has localized features. In contrast, the FT basis functions are limited to sines and cosines which
are periodic and continue infinitely. Both 1D and 2D wavelets are used in this work for 2DCoS denoising—in comparison to the 2D regularization method presented in Chapter 6.

1.3.4 Summary of Algorithms Presented

Table 1.1 lists the signal enhancement algorithms that are presented in this work—by name, chapter, data type, and functional capabilities. These collectively address the measurement problems and issues given above. TPMEM was presented in a previous work by our research group but is reexamined here. The other four methods shown in the table are new designs. TPMEM and the BiChi filters address noise and instrumental blurring of 1D data; the iSG filter and ABC method address automated needs for noise and baseline correction, respectively, in 1D; and MxMEM addresses noise in 2D images. All methods are useful for 2DCoS work, but only MxMEM was applied in that context (in Chapter 7). Later, in Chapter 8 (Conclusions), Table 8.1 provides a selection guide and performance summary for the algorithms.

Table 1.1 Algorithm Descriptions

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Data Type</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TPMEM</strong></td>
<td>1D</td>
<td>SNR-E, Deconv.</td>
</tr>
<tr>
<td>(Chap. 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BiChi Filters</strong></td>
<td>1D</td>
<td>SNR-E, Deconv.</td>
</tr>
<tr>
<td>(Chap. 3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>iSG Filter</strong></td>
<td>1D</td>
<td>SNR-E</td>
</tr>
<tr>
<td>(Chap. 4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ABE method</strong></td>
<td>1D</td>
<td>Baseline Correction</td>
</tr>
<tr>
<td>(Chap. 5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MxMEM</strong></td>
<td>2D</td>
<td>SNR-E</td>
</tr>
<tr>
<td>(Chap. 6)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: TPMEM = Two-Point Maximum Entropy Method; BiChi is the filter based on “dual-use” of the chi-squared function; iSG = Iterative-Savitzky-Golay; ABE = Auto-Baseline-Estimate method; MxMEM = Matrix Maximum Entropy Method

SNR-E = Signal-to-Noise-Ratio Enhancement; Deconv. = Deconvolution/Deblurring ability
1.3.5 Equations for Data Characterization

Appendix 1 contains equations for, and brief explanations of, key statistical concepts, a spectral peak model, and figures of merit (such as SNR) that are used in this work to characterize data. One additional concept/equation is highlighted here. The chi-squared statistic (or function) appears frequently—generally as a “goodness-of-fit” (or “fidelity”) functional within an algorithm’s objective function. In such cases, it is also used simultaneously as a stopping criterion (a way of deciding when to terminate an algorithm). The chi-squared general form, based on $N$ channels of a detector, is

$$\chi^2 = \sum_{i=1}^{N} \left( \frac{m_i - \hat{x}_i}{\sigma_i} \right)^2$$  (1.2)

where the $m_i$ are the measured data values (see Eq. 1.1), $\hat{x}_i$ are the estimated values of the underlying signal $x$, and $\sigma_i$ are the noise variances. Note that chi-squared essentially calculates the difference between two data distributions, and is therefore a kind of measure of “similarity”. When used with regularization methods (such as TPMEM and MxMEM), the stopping criterion value is normally chosen as approximately equal to $N$ (the number of points in the data) [36]. When comparing a processed (or reconstructed) spectrum to the original measured data, stopping at a value of $\chi^2$ that is greater than $N$ tends to result in a spectrum that is too noisy, and stopping at a value that is less than $N$ tends to result in a spectrum that is smooth but overly distorted.

1.4 Thesis Objectives

Extensive research has been done to develop signal enhancement algorithms which seek to address the widespread measurement problems caused by noise, instrumental blurring, and baseline distortions. Many of these algorithms can also be used in 2DCoS
analysis. Automation of the algorithms has also been investigated, but to a lesser extent. Yet, there is room for improvement. For example, little work appears to have been done in 2DCoS that employs 2D methods for noise reduction. Furthermore, deconvolution (or deblurring) continues to be a great need within spectroscopic and other applications.

Therefore, the thesis statement for this work is: *to investigate and apply new signal enhancement methods to spectroscopic (and other) application areas in order to address the need for noise reduction, deconvolution, baseline correction, and automation.*

Specific research objectives are as follows:

1. Explore and design high-performance noise reduction techniques for 1D and 2D data applications.
2. Apply these noise reduction methods to 2DCoS.
3. Investigate deconvolution methods coupled with the noise reduction methods for 1D and 2D applications.
4. Explore and design automation techniques for baseline correction and noise filters.

### 1.5 Thesis Organization and Summary of Contributions

Part I (Chapters 2-5) deals with 1D applications as follows:

Chapter 2 presents a re-examination of an existing regularization method, the Two-Point Maximum Entropy Method (TPMEM), in order to clarify its strengths and weaknesses, and provide ways to compensate for its limitations. TPMEM is very effective in low-SNR signal enhancement and deconvolution.

Chapter 3 introduces a new regularization method, based solely on dual application of the chi-squared statistic, and demonstrates its ability for noise reduction and
deconvolution. This design attempts to provide better performance in areas where TPMEM and Savitsky-Golay (SG) filters are weak.

Chapters 4 and 5 present new algorithms for automation of noise filtering and baseline correction, respectively. Both of these algorithms are based on SG filters and employ some of the other features used in the previous two algorithms (such as the chi-squared statistic for a stopping criterion).

Part II (Chapters 6-7) deals with 2D applications as follows:

In Chapter 6, a new 2D regularization method (Matrix-MEM), derived from TPMEM, is introduced and applied to SNR enhancement of images in two application areas.

In Chapter 7, MxMEM and 2D wavelets are applied to 2DCoS noise reduction.

Finally, Chapter 8 discusses conclusions and future work.

Appendix 1 contains a collection of equations, and their explanations, pertinent to data characterization.

Appendices 2-5 supplement Chapter 6 as follows:

Appendix 2 gives further detailed discussion of MxMEM’s gradient calculation, and how it is coded, at a high-level within the C program.

Appendix 3 summarizes our preliminary engineering investigation of hardware (microelectronic) acceleration of the MxMEM algorithm.

Appendix 4 provides more top-level explanations of how MxMEM was implemented, and a listing of the main C code.

Appendix 5 contains the Java code for our fixed-point arithmetic routines that were part of the hardware acceleration investigation (Appendix 3).
Chapter 2: TPMEM Re-examined--An SNR Enhancement and Deconvolution Algorithm *

This chapter re-examines a powerful SNR enhancement and deconvolution method for 1D data, known as the Two-Point Maximum Entropy Method. TPMEM is a regularization method which performs well on low-SNR data. It was originally developed for Raman spectroscopy but has more recently been applied to other application areas (see Chapter 6). However, TPMEM’s effectiveness is limited in the presence of high background offsets. This chapter reports on the cause of this limitation and presents solutions to minimize the effects on regularization.

2.1 Introduction

Effective discrimination between “signal” and “noise” in spectroscopy is desirable for a number of reasons: it simplifies complex information and reduces the amount of information to be stored and/or transmitted [41][42]; it permits further processing of the data such as differentiation [34], deconvolution [36], and two-dimensional spectroscopy [43][44]; it facilitates the detection and classification of features or components of interest [45]; it is used to enhance the signal-to-noise ratio (SNR) of data [41][30][46], it plays a role in quantification [34]; and it is often used to improve the presentation of data [30], amongst others. Smoothing techniques are frequently used in signal processing as a means to effect this discrimination against noise [41][34][30].

The two-point maximum entropy method (TPMEM) is a regularization method for the smoothing and deconvolution of spectra [46]. It shows superior signal-to-noise ratio (SNR) enhancement and deconvolution on low SNR spectra without significant spectral distortion [46][47]. More detailed information about TPMEM and the related maximum entropy method (MEM) can be obtained elsewhere [36][46][47]. It is not widely known, however, that the ability of TPMEM to remove noise can be significantly hampered by high background offsets (baselines) and high signals. This is due to the fact that readouts that are high relative to their noise content affect the probabilities estimated for use in the entropy calculations, as will be explained below. This limits the effectiveness of the method to spectra where high average background levels relative to noise do not occur and, depending on the measurement objectives, may contraindicate its use with high SNR spectra. In general terms then, TPMEM is not well suited to high readout-to-noise ratio (RNR; defined below) data without appropriate pre-processing. Note that a high RNR translates to a high SNR only under those circumstances when prominent baselines are not present, hence it cannot be argued that high RNR data always alleviate the need for smoothing. These cases notwithstanding, TPMEM in fact outperforms all other methods over a useful range of conditions. It is therefore important to delineate this range of efficacy and to identify procedures that may extend this range.

In this work we demonstrate the deleterious effects on TPMEM processing of high RNR data involving large offsets, prominent backgrounds, and tall peaks. First we consider the different manifestations of these effects and then we investigate the range of conditions over which they occur. We explain the origins of this susceptibility and suggest and evaluate
several potential solutions to the current limitations of TPMEM. For comparison, we use the well-known and frequently employed Savitzky-Golay filters [34][36][48].

2.2 Theory

When an underlying signal vector \( x \) is convolved (*) with an instrumental blurring function \( b \), and measurement noise, \( n \), is added, the resultant spectral measurement vector, \( m \), obtained on \( N \) detector channels can be represented as

\[
m = x * b + n
\]  
(2.1)

The purpose of regularization is then to produce an estimate or reconstruction \( \hat{x} \) of \( x \) from \( m \), or, where the effects of blurring are negligible or of no interest, a reconstruction of \( x * b \) from \( m \).

One such regularization method is TPMEM [46]. It has been used to modify the MEM [36][49][50] by replacing the negative entropy term in the cost function with a two-point negative entropy term such that the cost function, \( F_{TM} \), is given by

\[
F_{TM} = -SS_2 + \lambda \chi^2
\]  
(2.2)

where \( SS_2 \) is the sum of two-point entropies, \( \lambda \) is a Lagrange multiplier, and \( \chi^2 \) is the goodness-of-fit function. In general, the negative entropy term governs the degree of smoothness of the reconstructed spectrum while the chi-squared term governs the degree of fidelity of the reconstructed spectrum to the measured spectrum. The Lagrange multiplier is used to weight the relative importance of the “smoothing” and “fidelity” terms.

The two-point entropy is defined as

\[
S_{2i} = -\left(p_i \ln p_i + p_{i+1} \ln p_{i+1}\right)
\]  
(2.3)
where $p_i$ is the probability of event $i$ [51]. In the context of a reconstructed spectrum, the spectrum is viewed as a series of two-point probability distributions and the entropy is summed across them, i.e.,

$$SS_2 = \sum_{i=1}^{N-1} S_{2i} = \sum_{i=1}^{N-1} \left( p_i \ln p_i + p_{i+1} \ln p_{i+1} \right)$$

(2.4)

where the probabilities are determined as

$$p_i = \frac{\hat{x}_i}{\hat{x}_i + \hat{x}_{i+1}}$$

(2.5)

Equation 2.5 now permits one to see readily the effects of an offset on the smoothing term. Assume that an offset $C$ is added to the spectrum; then Eq. 2.5 becomes

$$p_i = \frac{\hat{x}_i + C}{(\hat{x}_i + C) + (\hat{x}_{i+1} + C)} \approx \frac{C}{2C}$$

(2.6)

when $C \gg \hat{x}_i$. Hence, in the case of two-point entropies, the probabilities tend toward 1/2, and in general for $j$-point entropies, tend to 1/$j$.

As pointed out in Shannon [51], any change toward equalization of the probabilities $p_i$ and $p_{i+1}$ in Eq. 2.3 increases the entropy (decreases the negative entropy). Conversely, any change away from equal probabilities, such as occurs during spectral reconstruction with TPMEM processing, will increase the negative entropy. If the negative entropy is already near its minimum, which occurs where the probabilities are nearly equal, and reconstruction within the limits imposed by the fidelity constraint cannot increase the negative entropy much, the minimization of Eq. 2.2 will proceed primarily via decreasing $\chi^2$. As a consequence, spectra with high average background levels, spectral regions with prominent baselines, and tall peaks may experience little smoothing. This analysis also immediately suggests one possible remedy: estimating the baseline and subtracting it before smoothing. If
desired, the baseline can be added back after smoothing. Several baseline correction methods with automation potential exist that could be of use for this purpose [52].

The fact that strong signal peaks themselves may experience little smoothing is also interesting. This could be of mixed benefit since, on the one hand, baseline regions are smoothed more than peaks, hence contrasting baseline and signals and preserving signal shape as measured. On the other hand, one may wish the entire spectrum to be smoothed to the same degree, for example, if subsequent deconvolution is to be attempted. Should the smoothing of peaks be of particular interest, another remedy that readily suggests itself is “inverting” the spectrum (i.e., turning the spectrum upside down) before applying the TPMEM procedure and then returning the result to the normal orientation.

2.3 Methods

In order to evaluate the sensitivity of TPMEM to high backgrounds and test the efficacy of the remedies proposed above, we created a synthetic dataset with different levels of noise and background. Briefly, we generated seven Lorentzian peaks, the first with a full width at half maximum (FWHM) of approximately 10 channels and the remainder with FWHM of 6 channels each. These peaks were then spaced at reduced intervals to generate uncongested and congested spectral regions, they were convolved with a Gaussian distribution to simulate instrumental broadening, and they were added to a sigmoid (cumulative Gaussian distribution) baseline. White noise was added to all spectra. Synthetic spectra created in this manner thus simulated 1001-point nominal Raman spectra of different SNRs (defined as the intensity of the tallest peak \([signal\ maximum - signal\ minimum]\) relative to the baseline noise) and different signal-to-background ratios (SBR, defined here as the signal intensity \([signal\ maximum - signal\ minimum]\) relative to baseline amplitude \([baseline\ amplitude - baseline\ noise]\)
maximum - baseline minimum). The tallest peak’s maximum and minimum are measured at the same point on the horizontal axis.

For ease of evaluation and continuity with our previous work, the results were also compared with those obtained from applying the commonly used Savitzky-Golay [34] filters to the same data. In a small pilot study, a number of filter parameters were evaluated in order to optimize filter performance. Two variants were found to give the best results on the synthetic data: an 11-point zero-order filter and a 21-point second-order filter. The former (SGw11n0) was chosen for further comparisons with the TPMEM due to its slightly better performance and our previous use of a zero-order filter [46].

For the calculations of SNR enhancements, Gaussian noise with a standard deviation of 1.0 was added to the spectrum consisting of seven Lorentzian peaks (the latter spectrum scaled beforehand and without baseline addition) to produce SNRs of 2, 3, 5, 10, and 100. Constant offsets of 0, 1, 3, 5, 10, 30, and 50 were added to each of these five spectra for a total of 35 spectra. The noise was estimated from channels 1 to 200 of the processed spectra and the peak maximum was taken as the maximum value between channels 201 and 400 minus the mean of channels 1 to 200. Given that no baseline was added to these spectra, the latter mean represented the base of the tallest peak.

The smoothing method and the suggested modifications were also evaluated on UV resonance Raman (UVRR) difference spectra. UVRR spectra were obtained from a sample of salmon sperm DNA (Sigma, St. Louis, MO) dissolved in distilled water to 500 μg/mL using a custom-built fiber optic probe [11][53][54]. Resonance Raman scattering was generated using 257 nm excitation with an intracavity-frequency-doubled Ar⁺ laser (Coherent, Santa Clara, CA), ~100 mW at the sample, and collected with a 0.67 m monochromator
(McPherson, Acton, MA) with a 3600 grooves/mm holographic grating, 150 μm slit widths, and 60 s integration time on a charge-coupled device (CCD) camera (PixelVision, Tigard, OR). Spectra were normalized to the 930 cm$^{-1}$ peak of NaClO$_4$ added as an internal standard to a concentration of 40 mM.

Computations were implemented using Matlab 7.0 (The MathWorks, Natick, MA) performing system calls to a custom-written C routine that in turn utilized algorithms and code from *Numerical Recipes in C* [36]. The computation platform was a 2.5 GHz dual processor PowerPC G5 running under Mac OS X 10.4.1 (Apple Computer, Santa Clara, CA). Procedures were repeated ten times with unique random noise added (Eq. 2.1) each time before TPMEM processing.

### 2.4 Results and Discussion

#### 2.4.1 Elevated (High Baseline-to-Noise Ratio) Baselines and Baseline Removal.

Figure 2.1a shows the effects of large offsets due to pronounced baseline elevation (SNR = 10 and SBR = 0.1) on spectral smoothing, and Fig. 2.1b shows the effects of less pronounced baseline elevation (SNR = 100 and SBR = 10) on spectral smoothing. The residual traces (i.e., the smoothed spectrum subtracted from the original noisy spectrum) at the bottom of each of the two panels clearly show that less smoothing occurs where spectral intensities are high. We used the $F$-statistic of variance ratios to test whether noise was equally (homoscedastic) or unequally (heteroscedastic) distributed across the residuals. The variances for the first 100 points of each of the ten residuals (i.e., from ten independent runs) were pooled and compared with those of the last 100 points for spectra of the type shown in Fig. 2.1a. The variances for the first and last 100 points were 0.01028 (numerator, df, i.e., degrees of freedom = 999) and 0.00305 (denominator, df = 999), respectively, with $F =$
33.727 and p < 0.001. When the variance of the first 100 points of the residuals was
compared with the variance of the original white Gaussian noise added, the variances were
0.01028 (numerator, df = 999) and 0.00994 (denominator, df = 9999), respectively, with F =
1.03 and p > 0.05. These results confirm that the distribution of noise in the residuals is non-
homoscedastic and that more noise was removed from the low-offset spectral regions.
Figure 2.1 Representative examples of TPMEM regularization applied to a series of simulated Raman peaks:
(a) superimposed on a strong background and with a SNR of 10; (b) the same simulated peaks superimposed on
a much weaker background and with a SNR of 100. Shown are the original simulated spectra (light gray traces),
the TPMEM-regularized spectra (black traces), and the residuals (dark gray traces). The residuals indicate that
little smoothing occurred in regions of high intensity: in (a) due to the strong background; and in (b) due to the
strong peaks. “Intensity” is in arbitrary units.

Figure 2.2 shows the results of TPMEM regularization when the baseline is removed
prior to smoothing. The baseline is estimated and subtracted from the spectrum, the spectrum

27
is processed with TPMEM, and the estimated baseline is then added back to the TPMEM-
regularized spectrum. For the baseline estimation a peak stripping method based on a 51-
point moving average is used [52]. Figure 2.2a shows that baseline estimation and removal
result in a more even distribution of noise in the residuals, as one would expect. A
comparison of variances, as before, confirms that smoothing was effected more evenly across
all regions whether they had large offsets prior to baseline adjustment or not. No significant
differences between the first and last 100 channels were observed; the variances were
0.00880 (numerator, df = 999) and 0.00940 (denominator, df = 9999), respectively, with F =
0.936 and p > 0.05. Where the baseline was perfectly removed, corresponding values of F =
0.979 and p > 0.05 were obtained. Figure 2.2b shows the averages of the reconstructed (i.e.,
smoothed) spectra compared to the spectrum that is being reconstructed (i.e., before the
addition of noise and baseline). Figure 2.2 shows that the results obtained with the removal
of an estimated baseline are comparable to those obtained with the removal of an accurately
known baseline. That is, there is a more even distribution of smoothing and there are no
significant peak distortions.
Figure 2.2 (a) Representative examples of TPMEM regularization when the baseline is removed prior to smoothing. The baseline is estimated and subtracted from the spectrum, the spectrum is processed with TPMEM, and the estimated baseline is then added back to the TPMEM-regularized spectrum. For the baseline estimation a peak stripping method based on a 51-point moving average is used. The original spectrum is shown as a light gray trace, the regularized spectrum as a black trace, and the residual (i.e., difference between them) as a dark gray trace. (b) The averages of the regularized spectra compared to the original synthetic spectrum (i.e., before the addition of noise and baseline; black trace). The figure shows that the results obtained with the removal of an estimated baseline (dark gray trace) are comparable to those obtained with the removal of an accurately known baseline (i.e., no baseline was added; light gray trace).
This interpretation is confirmed with a root-mean-square error (RMS) evaluation based on the differences between the regularized spectra and the original synthetic spectrum before baseline and noise addition: the RMS for using the baseline estimation and subtraction approach was slightly larger than the RMS for regularization when the baseline is accurately removed (i.e., never added), as shown in Table 2.1. Since baseline estimation itself is not perfect, it stands to reason that the improved regularization obtained with baseline subtraction compared to the RMS in the presence of a high baseline comes at a small cost.

The TPMEM is superior to Savitzky-Golay filtering for low SNR data [46], but the latter is not susceptible to the effects of high offsets; hence it performs better under these conditions. This is reflected by the RMS values, as indicated in Table 2.1, for the type of high baseline spectra shown in Fig. 2.1a.

Table 2.1 Means and standard errors of SNR = 10 spectra regularized with TPMEM or smoothed with an 11-point, zero-order Savitzky-Golay smoothing filter. RMS values were based on the synthetic spectra before the addition of noise. SNR values were calculated using the intensity of the tallest peak and the noise from the first 200 channels in each spectrum. For the difference spectrum, the SNR was calculated using the intensity of the tallest peak and the noise estimated from the entire spectrum using an automated noise estimation method [55].

<table>
<thead>
<tr>
<th>Method</th>
<th>TPMEM</th>
<th>Savitzky-Golay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spectra</td>
<td>RMS</td>
<td>SNR</td>
</tr>
<tr>
<td>High baseline</td>
<td>0.082±0.001</td>
<td>n/a</td>
</tr>
<tr>
<td>Removed baseline</td>
<td>0.035±0.002</td>
<td>36.174±4.6</td>
</tr>
<tr>
<td>No baseline</td>
<td>0.032±0.001</td>
<td>43.789±2.2</td>
</tr>
<tr>
<td>Inverted baseline</td>
<td>0.046±0.002</td>
<td>25.476±1.2</td>
</tr>
<tr>
<td>Difference spectrum</td>
<td>n/a</td>
<td>583.33</td>
</tr>
</tbody>
</table>

2.4.2 Elevated (High Signal-to-Noise Ratio) Peaks and Spectrum Inversion.

The lack of noise at peak locations in the residual shown in Fig. 2.1b also demonstrates the reduced negative entropy minimization that occurs for high SNR peaks. If the smoothing of peaks is of particular concern and relatively milder smoothing of the baseline could be tolerated, the spectrum could be rotated around the horizontal axis and
displaced along the vertical axis. Specifically, the spectral maximum is determined and an ‘inverted’ spectrum is obtained by subtracting the intensity in every channel from this maximum. This results in the baseline being relatively high and peaks being relatively low, as shown in Fig. 2.3a. When the result is compared to that obtained with processing in the normal orientation, Figure 2.3b, one sees that, along with greater smoothing of the peaks, their intensities are reduced. However, there is less overlap between these peaks compared to those smoothed in the normal orientation. Representative results for inverted (thicker trace) and normally oriented (thinner trace) processing are shown in Fig. 2.3b superimposed on the original noisy spectrum.
Figure 2.3 (a) TPMEM regularization of an ‘inverted’ spectrum (light gray trace), with the baseline being relatively high and peaks being relatively low, produces a result (black trace) with more noise in the baseline and less in peaks as shown by the residual (dark gray trace). (b) When the result of regularizing an inverted spectrum (dark gray trace) is compared with that processed in the normal orientation (black trace) and with an original noisy spectrum (light gray trace) one sees that, along with greater smoothing of the peaks and an increased separation between their bases, their intensities are reduced.
The RMS errors and SNR values are also reported in Table 2.1. The results in Table 2.1 show that the regularization of inverted spectra comes at the cost of increased RMS and lowered SNR values because the brunt of the regularization process affects fewer channels (i.e., primarily the inverted peaks). Since Savitzky-Golay smoothing is not susceptible to a differential effect, using the SGw11n0 filter produced similar RMS and SNR values for the normally oriented and inverted spectra.

The present comparisons between TPMEM and Savitzky-Golay smoothing are also important for what they reveal about these two methods. Firstly, in previous work we reported that TPMEM regularization increased the SNR of low SNR spectra (SNR ~ 1) approximately fivefold while a zero-order Savitzky-Golay filter did so approximately threefold. It is therefore of interest to note that the TPMEM advantage also holds for higher SNR spectra (SNR = 10) and the effect of SNR on SNR enhancement is further investigated below to determine the range of greatest effect. Secondly, the Savitzky-Golay smoothing filter, with the parameters used here, noticeably reduced peak intensities, as is evident in Fig. 2.4, hence adversely affecting SNR. It is well known however [36], that using different smoothing filter parameters can produce more accurate peak intensities, but this comes at the cost of less smoothing, hence countering the desired gain in the SNR. TPMEM regularization, on the other hand, is more effective at discriminating against noise in low-lying spectral regions while preserving peak intensities more faithfully, especially so for high SNR peaks. In other words, when TPMEM regularization is applied to baseline-corrected spectra, or spectra with mild to no baselines, what may otherwise be a drawback, as pointed out above, becomes an advantage. Specifically, this insight suggests that increasing the exponent in the chi-squared component of the cost function (Eq. 2.2) may enable one to
differentially emphasize peaks. We intend to further investigate this differential effect of smoothness versus fidelity on regularization.
Figure 2.4 A representative full-scale (a) and detail thereof (b) comparison of TPMEM regularization (dark gray) and Savitzky-Golay smoothing (11-point, zero order; medium gray) of a noisy spectrum (SNR = 10; light gray trace). The figure shows that, compared to the original synthesized spectrum without noise (dashed black trace), TPMEM regularization produces greater overall smoothness, less reduction in peak height, and less separation between peak bases than Savitzky-Golay smoothing.
Finally we note that the removal of an offset, such as baseline removal, will improve TPMEM results, but subsequent scaling or normalization of a spectrum will not since noise is scaled along with spectral peaks and hence, the RNR will remain the same.

2.4.3 Application to Experimental Data.

The advantage of TPMEM in low SNR spectra suggests that it may be particularly useful when applied to difference spectra. In other words, although one could use baseline estimation and removal before applying regularization, baseline removal can often be implemented very efficiently in difference spectra. Difference spectra also tend to contain more noise. The combined effect of these two factors, baseline reduction and noise elevation, therefore makes TPMEM well suited to SNR enhancement in difference spectra.

Figure 2.5 shows the difference spectrum of salmon sperm DNA acquired at 35 °C and 19 °C as a light gray trace. The SGw11n0 filter result (thick medium-gray trace) contains noticeably more noise than the TPMEM regularized result (black trace). The SNR values, reported in Table 2.1, are consistent with expectations of superior TPMEM performance. We found that TPMEM produced consistently better SNR values than the Savitzky-Golay filter (based on an automated noise estimation method [55]), even when a larger filter window was used. As the filter window is increased, more smoothing is obtained, but at the expense of reduced peak heights [36] and increasingly significant spectral distortion.
Figure 2.5 A difference spectrum of salmon sperm DNA acquired at 35 °C and 19 °C (light gray trace) processed with an 11-point zero order Savitzky-Golay filter (thick medium-gray trace) and with TPMEM regularization (black trace). The expectations of superior TPMEM performance are confirmed by the figure and are supported by SNR values (Table 2.1). SNR values remained in favor of TPMEM even when different Savitzky-Golay filters were used, up to the point of significant spectral distortion, to improve the SNR.

The TPMEM method produces excellent results on low SNR data [46][47], including difference spectra, especially where difference spectra contain positive or predominantly positive features. For negative or predominantly negative features, the order of subtraction can be reversed before regularization. However, where both positive and negative features of interest occur in difference spectra, it is important to realize that the noise removal obtained using TPMEM will be asymmetric. Specifically, noise will be discriminated against more strongly in negative features than positive ones. This happens because one needs to add an offset to such spectra to ensure that all spectral values are positive before the entropy can be calculated. This added offset will result in the former positive features having more of an offset than the former negative features, and less noise discrimination results where the offset
is larger. In such cases one may wish to opt for Savitzky-Golay smoothing of the difference spectrum or possibly baseline correction and TPMEM regularization of spectra before obtaining the difference spectrum.

2.4.4 Readout-to-Noise Ratio Scope of Two-Point Maximum Entropy Method.

The above results demonstrate the conditions under which the use of TPMEM is advantageous. We will now attempt to formalize the scope of TPMEM application in terms of RNR and SNR. This will permit a more detailed and useful determination of the suitability and benefits to be expected of TPMEM given the attributes of the data at hand.

The RNR at channel $i$ is now defined as

$$RNR_i = \frac{x_i + C}{\sigma_i}$$

(2.7)

in the presence of an offset, $C$. Substitution in Eq. 2.6 for the observations $x_i$ yields

$$p_i = \frac{x_i + C}{(x_i + x_{i+1} + 2C)} = \frac{RNR_i}{(RNR_i + RNR_{i+1})}$$

(2.8)

and hence we can plot $p_i$ as a function of $RNR_i$. Given that the standard deviation is expressed in terms of a variation from the mean of the observations $x_i$, the ratio $\frac{x_i}{\sigma_i}$ will remain on average relatively constant, and changes in $RNR_i$ will obtain mainly from changes in the offset $C$. Furthermore, since the entropy is calculated as $-p_i \ln p_i$, the two-point negative entropy can be plotted as a function of RNR. This has the benefit of revealing how the two-point negative entropy changes with RNR and thus those regions of RNR where optimization is likely to yield a benefit.

Figure 2.6a shows a spectrum with 10% white Gaussian noise relative to the height of the maximum peak and Fig. 2.6b shows the RNR of the same spectrum (black trace, right
Note that the major change in RNR is due to the offset; this would also be true in the presence of heteroscedastic noise since the variance of measurements is established with regard to the mean of those measurements. This observation suggests that a readout mean-to-noise ratio may be a useful alternative to readout-to-noise ratio. Also in Fig. 2.6b is plotted the noise levels of the residuals after TPMEM processing (light gray, left axis). These noise levels were calculated in a 50-point moving window and averaged over ten residuals (see also Fig. 2.1a). It is clear from Fig. 2.6b that noise levels are fairly constant initially indicating a ceiling effect, that is, most of the noise is removed from the noisy spectrum with TPMEM processing when the RNR is close to zero. Noise removal starts to decline rapidly for RNR values from 5 to 10 and by the time RNR reaches 40 to 50, little removal is effected. Figure 2.6c shows the probabilities, \( p_{x_1} \) and \( p_{x_2} \), of two hypothetical measurements \( (x_1 = 0.1 \) and \( x_2 = 0.9) \) as a function of RNR, the latter varying primarily as a function of the offset \( C \). As expected (see Eq. 2.6), the probabilities approach 0.5 as the RNR increases due to increases in the offset. The two-point negative entropy, shown in Fig. 2.6d as a function of RNR, indicates that very little change in two-point negative entropy occurs for RNR values larger than about 50, in agreement with the experimental results shown in panel Fig. 2.6b.
Figure 2.6 (a) A synthetic spectrum with SNR = 10. (b) The same spectrum as a function of RNR (black trace, right axis) along with average noise levels from 10 residual spectra, see Fig. 1(a), calculated using a 50-point moving window (gray trace, left axis). (c) The probability of two hypothetical measurements ($x_1 = 0.1$ and $x_2 = 0.9$) as a function of RNR, the latter varying primarily as a function of the offset $C$. The probabilities approach 0.5 as RNR increases due to increases in the offset. (d) The two-point negative entropy as a function
of RNR indicates that very little change (i.e., little noise discrimination) occurs for RNR values larger than about 50.

Figure 2.7 shows SNR enhancements as a function of RNR for a SGw11n0 filter (dashed lines) and TPMEM (solid lines). It is clearly evident from Fig. 2.7a that increasing offsets, represented by an increasing RNR, have relatively little effect on the SGw11n0 filter, as expected. In contrast, a pronounced decline in SNR enhancement occurs with increasing RNR, as expected from the previous results for TPMEM. From Fig. 2.7a it appears that TPMEM can be expected to excel for RNR < ~3 while SG filters are better for RNR > ~10.

Figure 2.7b shows SNR enhancement as a function of spectrum SNR for offsets of 0, 1, and 3 (means, n = 10). Error bars (standard deviations, n = 10) are shown for both methods using an offset of 0. The thicker traces in Fig. 2.7b show the means for the two methods over the three conditions (i.e., offsets 0, 1, and 3). The means for the TPMEM results are shown for comparison purposes, but it should be kept in mind that TPMEM shows a trend that varies with offset while the SGw11n0 filter does not.
Figure 2.7 (a) SNR enhancements for spectra of different SNR (2, 3, 5, 10, and 100; traces graded from black to light gray, respectively) as a function of RNR for a Savitzky-Golay filter (11-point, zero order, dashed lines) and for TPMEM (solid lines). Increasing offsets have little effect on the SGw11Inf filter but a decline occurs for TPMEM. TPMEM can be expected to excel for RNR < ~3 while SG filters are better for RNR > ~10. (b) SNR enhancement as a function of spectrum SNR for offsets of 0, 1, and 3 (means, n = 10). Error bars (standard deviations, n = 10) are shown for both methods using an offset of 0. Thicker traces show the means for the two methods over the three conditions (i.e., offsets 0, 1, and 3). TPMEM can be expected to yield good results for SNR < ~50 provided the offset is small.
Again, the SG filter’s performance (dashed lines) appears to be relatively independent of SNR over the range investigated. Given that the SNR could be seen as a specific case of the RNR, this result is to be expected. It is also clear from Fig. 2.7b that the TPMEM performs better than the SG filter over most of the SNR range for offsets of 0 and 1. With an offset of 3, its performance is very similar to that of the SGw11n0 filter. Furthermore, it is evident that TPMEM performs better on lower SNR spectra given the data examined here. In particular, Fig. 2.7b suggests that optimal noise discrimination may occur for SNRs between 5 and 50. It is interesting to note that Jirasek and coworkers [47] also found significant SNR enhancements over the SNR 5 to 20 range, although their approach was two dimensional and their examples related to X-ray computed tomography. It is yet unclear what the incremental SNR enhancement is for two-dimensional compared to one-dimensional processing and whether an optimum exists or would be found in the same SNR range.

The optimum obtained here may reflect the fact that at higher SNRs smoothing of the peak base, but not zenith, occurs, whereas for very low SNRs, both real and noise-induced spurious peaks are similarly processed. For intermediate SNRs, however, the spectral baseline (i.e., the peak bases) is smoothed more than the peak tops, although the latter are also smoothed to an appreciable extent. As pointed out above, this occurs due to the asymmetry of the $\chi^2$ component of the cost function. An interesting approach may therefore be the modification of the $\chi^2$ component of the cost function to enforce fidelity locally rather than globally, a shift in emphasis not unlike that from global MEM to two-point MEM. Calculating the $\chi^2$ value in a moving window may prove helpful in achieving this objective.
2.5 Conclusions

The work presented here confirms the major benefits of TPMEM regularization: superior SNR enhancement and faithful spectral reconstruction. In particular, spectra with SNR levels between 5 and 50 (and with background offsets less than about 3-5 times the background noise level) yield excellent results. Nevertheless, there are some circumstances under which TPMEM should be used with caution to discriminate against noise. These are: spectra with high background components (as in Fig. 2.1a); spectra with tall peaks (relative to the baseline, as in Fig. 2.1b); and difference spectra with positive and negative (i.e., derivative-like) features. The application of TPMEM can be extended, however, to give useful results under some of these conditions: for spectra riding on high backgrounds, baseline subtraction provides a good alternative; and for tall peaks, inverting the spectrum provides discrimination against noise in the peaks. Further beneficial extensions may involve the manner in which the Lagrange multiplier, $\lambda$, is adjusted between iterations. These and other possible improvements will receive attention in the future. In conclusion, the results presented here will increase the utility of TPMEM regularization by delineating the range of conditions under which it can be used effectively, revealing those conditions where its use is problematic and describing how to extend its use to the ranges in between.
Chapter 3: Chi-Squared-based Filters for SNR Enhancement and Deconvolution*

This chapter presents a new $\chi^2$-based regularization method for SNR enhancement and deconvolution. It uses the $\chi^2$ function to characterize both smoothness and fidelity (within the objective function). This Bi-chi filter (BCF) has two implementations—a recursive one (for noise filtering) and as a regularization-with-deconvolution method. For SNR enhancement, the BCF is compared to Savitzky-Golay filters and is demonstrated to be equal to or superior in performance. This design was aimed at providing better performance in areas where TPMEM (Chapter 2) and Savitsky-Golay (SG) filters are weak, and this goal has been accomplished.

3.1 Introduction

Ensemble averaging is a well-known method to enhance the signal-to-noise ratio (SNR) of spectra. Because noisy deviations from the spectrum background are uncorrelated among sampled spectra from a given system, while deviations of a signal from the background are highly correlated, addition of similar spectra leads to addition of the signal but partial cancellation of noise [30]. Alternatively, since the signal tends to be much lower frequency than the noise, averaging can also be used to obtain partial cancellation of noise in a single spectrum by averaging a number of nearby points instead. The number of points taken to contribute to the average constitutes a window that is applied sequentially, or "moved," across the spectrum to effect the desired noise reduction – hence the moving

---

average method of smoothing [41]. Every one of the points in a window contributes an equal amount to the average, but a window can be constructed such that the points are differently weighted and thus contribute differently to noise reduction. When these weights derive from the coefficients of polynomials, the Savitzky-Golay family of filters results [41][34].

Savitzky-Golay filters have become the standard against which other smoothing methods are compared [41][46][56]; they are easy to implement, fast to execute, and provide generally good results [56]. Despite their advantages however, Savitzky-Golay filters have some drawbacks. Increasing the amount of smoothness leads to a decrease in the spectral resolution [36]. Although this is generally true for any smoothing process, Savitzky-Golay filters may not represent the best trade-off between smoothness and resolution. By modeling the data locally with a polynomial, Savitzky-Golay filters minimize modeling errors, but the validity of such models remains questionable [41] and hence the choices of filter parameters remain uncertain.

It is known that Savitzky-Golay filters do not always provide the best error minimization obtainable [41]. For example, they are outperformed at low SNR by the two-point maximum entropy method (TPMEM) [46][56]. TPMEM, however, does not perform well in the presence of high backgrounds because it is dependent on differences in two-point probabilities that become dominated by the background. Savitzky-Golay filters, in contrast, do not suffer from this type of interference [56]. Taken together, these considerations suggest that a better model-free smoothing procedure may be devised.

The \( \chi^2 \) function is frequently used to provide a statistic to characterize "goodness-of-fit" between modeled and experimental data [36][57]. In essence, the statistic quantifies the difference between corresponding data points in different data sets. Extending the application
of the $\chi^2$ function to neighboring points in the same spectrum – somewhat like the 
application of the moving-window averaging described above – it then characterizes the 
difference between adjacent points. This then constitutes a measure of smoothness as 
opposed to a measure of fidelity.

We report here on a procedure that employs the $\chi^2$ function for the dual (and 
simultaneous) purposes of characterizing smoothness and fidelity. It is implemented in a 
recursive formulation, as a standard regularization method, and as a regularization-with-
deconvolution method. We have also compared these implementations to Savitzky-Golay 
filters to assess the performance of the method relative to a conventional standard. The 
results are generally superior compared to Savitzky-Golay filters.

3.2 Theory

When an underlying signal vector $x$ is convolved (*) with an instrumental blurring 
function (or instrumental point spread function, IPSF) $b$, and measurement noise, $n$, is added, 
the resultant spectral measurement vector, $m$, obtained on $N$ detector channels can be 
represented as

$$m = x \ast b + n$$

(3.1)

The purpose of regularization is to produce an estimate or reconstruction $\hat{x}$ of $x$ from $m$, or, 
where the effects of blurring are negligible or of no interest, a reconstruction of $x \ast b$ from 
$m$.

A regularization cost function $F$, based entirely on implementations of the chi-
squared function, can now be constructed such that

$$F = \chi^2_s + \lambda\chi^2_r$$

(3.2)
where $\chi^2_S$ is the chi-squared function employed in “smoothness mode”, $\chi^2_F$ is the chi-squared function employed in “goodness-of-fit mode”, and $\lambda$ is a Lagrange multiplier that governs the relative importance of the “smoothing” and “fidelity” terms. More specifically:

$$\chi^2_F = \sum_{i=1}^{N} \frac{m_i - \hat{x}_i}{\sigma_i}^2$$

and

$$\chi^2_S = \sum_{i=2}^{N} \left( \frac{\hat{x}_{i-1} - \hat{x}_i}{\sigma_i} \right)^2$$

Since Eq. 3.2 is quadratic in $\hat{x}$, the minimum can be found: expand the summation terms and set the derivative to zero. Assuming homoscedastic noise and a given $\lambda$:

$$\frac{\partial F}{\partial \hat{x}_i} = \frac{\partial}{\partial \hat{x}_i} \left\{ \sum_i \left( \frac{\hat{x}_{i-1} - \hat{x}_i}{\sigma} \right)^2 + \lambda \left[ \sum_i \frac{m_i - \hat{x}_i}{\sigma} \right]^2 \right\} = 0$$

$$= \frac{\partial}{\partial \hat{x}_i} \left[ \left( \frac{\hat{x}_{i-1} - \hat{x}_i}{\sigma} \right)^2 + \left( \frac{\hat{x}_i - \hat{x}_{i+1}}{\sigma} \right)^2 + \lambda \left( \frac{m_i - \hat{x}_i}{\sigma} \right)^2 \right]$$

$$= -\frac{2}{\sigma^2} [\hat{x}_{i-1} - \hat{x}_i + (-\hat{x}_i + \hat{x}_{i+1}) + \lambda (m_i - \hat{x}_i)]$$

Hence,

$$\left( \hat{x}_{i-1} - \hat{x}_i - \hat{x}_{i+1} + \lambda m_i - \lambda \hat{x}_i \right) = 0$$

suggestions the recursive formula

$$\hat{x}_i = \left( \frac{\hat{x}_{i-1} + \hat{x}_{i+1} + \lambda m_i}{2 + \lambda} \right)_{i \neq 1, i \neq N}$$

for smoothing.
3.2.1 Smoothness

Smoothness can be attained by using Eq. 3.2 as the objective function in a multidimensional optimization routine. A filter also can be implemented using Eq. 3.7. Equation 3.7 shows that the reconstructed signal will approach the measured one for $\lambda \gg 1$. Conversely, if $\lambda = 0$ and the formula is applied iteratively, the reconstructed signal will approach the mean of the initial estimate since every point is reconstructed to be the mean of its two flanking neighbors. Note that in the special case where the starting values for the reconstructed signal vector are identical to those of the original measured vector and with $\lambda = 1$, Eq. 3.7 implements a three-point moving average (i.e., a three-point zero-order Savitzky-Golay filter).

3.2.2 Fidelity

Equation 3.3 makes explicit that the term $\chi_f^2$ is a sum of differences between reconstructed and original spectral vectors. The smaller this sum, the greater the similarity between the vectors. A problem occurs when a large number of points are involved and it becomes possible for very good fits in some regions to compensate for poorer fits in other regions while keeping the overall sum within “acceptable” limits. This undesirable feature often presents itself as the simultaneous occurrence of good reconstruction of noise in background regions and poor reconstruction of spectral peaks. For this reason, enforcing fidelity locally by dividing the spectrum into sections and calculating Eq. 3.3 for each such section may prove advisable. The dimension of each section depends on the resolution that is desired after smoothing and it is likely to be the average full width at half-maximum (FWHM) of the signal peaks. Although local fidelity could be incorporated into the stopping criterion, incorporating it also into the objective or cost function ensures that local fidelity is
taken into account during the smoothing process. We note the above-mentioned drawback without pursuing the potential remedies at this point.

### 3.2.3 Deconvolution

Deconvolution can be incorporated into the regularization procedure if the reconstruction $\hat{x}$ is based on the underlying signal vector $x$ instead of the convolved vector $x \ast b$. The smoothness criterion $\chi_s^2$ is then applied to $\hat{x}$ and the fidelity criterion $\chi_f^2$ is determined with respect to $\hat{x} \ast b$.

### 3.2.4 Algorithms

Brief algorithms for the two regularization methods are given below.

**A** (Regularization (no deconvolution)). This method does not remove the instrumental blurring.

(i) Reconstruct $x \ast b$ by minimizing Eq. 3.2 ($\hat{x}$ represents $x \ast b$).

(ii) Assess fidelity of $\hat{x}$ using Eq. 3.3.

(iii) Repeat the above steps until the stopping criterion is satisfied ($\chi^2 \approx$ number of channels).

**B** (Regularization (with deconvolution)). This method removes the instrumental blurring.

(i) Reconstruct $x$ by minimizing Eq. 3.2 ($\hat{x}$ represents $x$).

(ii) Convolve $\hat{x}$ with the IPSF: $\hat{x} \ast b$.

(iii) Assess fidelity of $\hat{x} \ast b$ using Eq. 3.3.

(iv) Repeat the above steps until the stopping criterion is satisfied ($\chi^2 \approx$ number of channels).
3.3 Methods

The dual use $\chi^2$-based approach, as filter (bi-chi filter or BCF) and as regularization procedure, was tested on synthetic data, simulated to represent hypothetical Raman spectra, as well as real (measured) Raman spectra. Briefly, we generated seven Lorentzian peaks, the first with a FWHM of approximately 10 channels and the remainder with FWHM of 6 channels each, resulting in an average FWHM of 7 channels. These peaks were then spaced at reduced intervals to create uncongested and congested spectral regions, they were convolved with a Gaussian distribution to simulate instrumental broadening, and they were added to a sigmoid (cumulative Gaussian distribution) baseline. White noise was added to all spectra. Synthetic spectra were generated with different signal-to-noise ratios (SNR, defined as the intensity of the tallest peak [$signal\ maximum - signal\ minimum$] relative to the baseline noise).

Scaled Gaussian (white) noise with an original standard deviation of 1.0 was added to the spectrum consisting of seven Lorentzian peaks to produce SNRs of 3, 10, and 50. For SNR enhancement evaluations, the noise was estimated from the entire spectrum using an automated method [55] and the peak maximum was taken as the maximum value between channels 301 and 400. Each set consisted of 10 spectra with unique noise distributions.

For comparison, Savitzky-Golay filters were applied to the same data. In a previous pilot study [56], a number of filter parameters were evaluated in order to optimize filter performance and an 11-point zero-order filter was found to give the best results. However, we chose a slightly larger 13-point filter because the zero-order variant (SGw13n0) produced a smooth result but tended to merge the two most closely spaced peaks into one, while the second-order variant (SGw13n2) allowed discrimination between the two peaks but at the
cost of less smoothing. These two filters therefore established benchmarks for smoothness and fidelity, respectively.

The different implementations of the smoothing method were also evaluated on UV resonance Raman (UVRR) spectra and difference spectra. UVRR spectra were obtained from a sample of salmon sperm DNA (Sigma, St. Louis, MO) dissolved in distilled water to 500 μg/mL using a custom-built fiber optic probe [11][53][54]. Resonance Raman scattering was generated using 257 nm excitation with an intracavity-frequency-doubled Ar+ laser (Coherent, Santa Clara, CA), ~100 mW at the sample, and collected with a 0.67 m monochromator (McPherson, Acton, MA) with a 3600 grooves/mm holographic grating, 150 μm slit widths and 60 s integration time on a CCD camera (PixelVision, Tigard, OR). Spectra were normalized to the 930-cm⁻¹ peak of NaClO₄ added as an internal standard to a concentration of 40 mM.

Computations were implemented using Matlab 7.0 (The MathWorks, Natick, MA). The computation platform was a 2.5 GHz dual processor PowerPC G5 running under Mac OS X 10.4.1 (Apple Computer, Santa Clara, CA). Procedures were repeated 10 times with unique random noise added each time before BCF processing.

3.4 Results and Discussion

3.4.1 Recursive BiChi Implementation

Figure 3.1a shows the result, after 5 iterations with a lambda value of 0.003, of the recursive BCF applied to a synthetic spectrum with different SNR values. The dependence of performance on the number of iterations and value of lambda from different runs (i.e., spectra with unique noise distributions) is shown in Figs. 3.1b-3.1d. Note that for iterations in
excess of 5, the SNR continues to improve, but both the root mean square (RMS) error and the chi-squared statistic deteriorate. In practical terms, the spectra become smoother with continued iterations but at the cost of losing resolution, peak height, and generally, fidelity to the original. As expected from Eq. 3.7, smoothness increases with decreasing lambda.
Figure 3.1 (a) Detail of the 5 most congested of the 7 simulated peaks - showing typical spectra of varying SNR smoothed with the recursive biChi filter (5 iterations, lambda = 0.003); these are superimposed on the original noise-free spectrum for comparison; (b) RMS error; (c) SNR enhancement; and (d) $\chi^2$-values of the residual (smoothed spectrum – noisy starting spectrum) for the recursive biChi filter on spectra with original SNR = 10 as a function of iteration for varying values of lambda.
Figure 3.2 shows a comparison of the BCF with two variants of the Savitzky-Golay filter. A number of observations deserve mention. First, the SGw13n0 is already near optimal as established in the pilot study; larger windows lead to rapid spectral distortion despite a continued SNR enhancement. Here smoothness is gained at the expense of fidelity and the SGw13n0 filter represents the smoothness benchmark. Second, the SGw13n2 filter exhibits a lower SNR enhancement (cf. SGw13n0), but with less spectral distortion. This variant of the filter represents the fidelity benchmark. Third, the performance of the recursive biChi filter is superior to both Savitzky-Golay filters, with minimal spectral distortion evident compared to the noise-free original and with the residual free of spectral remnants. The results are quantified in Table 3.1.

![Figure 3.2](image)

**Figure 3.2**  Detail showing a typical spectrum (SNR = 10) smoothed with each of the recursive biChi (5 iterations; biChi lambda = 0.003), SGw13n0, and SGw13n2 filters; these are superimposed on the noise-free spectrum for comparison. The residual is the biChi-smoothed spectrum minus the noisy original spectrum.
Table 3.1  Comparative results (mean ± standard error) for the biChi recursive and regularization (in deconvolution mode, lambda = 30) methods show the biChi results to be superior to the Savitzky-Golay filters. Note that, due to deconvolution, the RMS and $\chi^2$-values for the biChi-d spectra are not informative.

<table>
<thead>
<tr>
<th>Figure-of-merit</th>
<th>biChi (recursive)</th>
<th>biChi-d</th>
<th>biChi-c</th>
<th>SGw13n0</th>
<th>SGw13n2</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMS</td>
<td>0.00110 ± 0.00002</td>
<td>0.00298 ± 0.00008*</td>
<td>0.00107 ± 0.000002</td>
<td>0.00110 ± 0.00002</td>
<td>0.00132 ± 0.00002</td>
</tr>
<tr>
<td>SNR</td>
<td>501 ± 8</td>
<td>339 ± 9</td>
<td>668 ± 11</td>
<td>131 ± 1</td>
<td>91 ± 1</td>
</tr>
<tr>
<td>$\chi^2$ (residual)</td>
<td>879 ± 16</td>
<td>1375 ± 32*</td>
<td>889 ± 17</td>
<td>964 ± 18</td>
<td>811 ± 16</td>
</tr>
</tbody>
</table>

*Not informative.

3.4.2 BiChi Regularization

An advantage of the biChi method is that it can be implemented as a regularization method that can optionally incorporate deconvolution, as mentioned above. Figure 3.3 shows the results of implementing the biChi regularization method, Eq 3.2, on SNR = 10 spectra with lambda = 0.03 without implementing deconvolution. The chi-squared acceptance criterion was reached after six iterations. Note that some spectral distortion is evident despite reaching this limit. Relaxing this limit provides an opportunity to tune the trade-off between smoothness and fidelity [47].
Figure 3.3 Detail showing a typical spectrum (SNR = 10) after biChi regularization (6 iterations; \( \lambda = 0.03; \chi^2 < \) number of channels). The original noisy and noise-free spectra are shown for comparison. The bottom trace shows the residual obtained by subtracting the noisy starting spectrum from the regularized spectrum. Note some spectral distortion is evident in the regularized spectra (e.g., reduced peak amplitudes, loss of resolution).

Figure 3.4 shows the results of regularization using the biChi method with deconvolution where some underlying ideal spectrum has been reconstructed. This reconstruction is then convolved with an instrumental point spread (broadening) function and the convolution is compared to the original noisy spectrum using the chi-squared statistic. In other words, the smoothness constraint is applied to the reconstruction before convolution with the IPSF and the fidelity constraint is applied after, as explained in the Theory section. As expected, the convolved spectrum is smoother but exhibits loss of resolution.
The trade-off between smoothness and fidelity can also be adjusted with the Lagrange multiplier $\lambda$. Figure 3.5 shows that a relatively high value of $\lambda$ increases the fidelity of the regularized spectrum after convolution (biChi-c). Agreement between the original noise-free spectrum and biChi-c is now excellent (compare to Fig. 3.4). The “deconvolved” regularized spectrum (i.e., before the convolution step; biChi-d) exhibits enhanced resolution; specifically, the overlapping peaks near channel 650 are now clearly resolved. Although these peaks are not resolved in biChi-c, the results suggest that an even larger value of $\lambda$ (compared to the ones investigated here) is likely to further enhance resolution in both regularized spectra. However, determining the optimal value for $\lambda$ fell outside the scope of our investigation.
Figure 3.5 Detail showing typical spectra (SNR = 10) after biChi regularization using the deconvolution option (6 iterations; lambda = 30; $\chi^2 < $ number of channels). The original noisy and noise-free spectra are shown for comparison. The bottom trace (Residual-c) shows the residual obtained by subtracting the noisy starting spectrum from the regularized spectrum convolved with the instrumental blurring function (biChi-c; biChi-d = biChi-c before convolution). Note the excellent agreement between the convolved regularized spectrum and the noise-free original, also reflected in the residual, and the improved resolution of the overlapping peaks centered near channel 650 in the ‘deconvolved’ regularized spectrum.

Note that the $\chi^2$-function consists of a sum over all channels; it is therefore possible for excessively close fitting in some regions to be compensated for by a lack of fitting in others. Both effects are undesirable: close fitting means that noise is incorporated into the reconstructed spectrum while a lack of fitting means that spectral features of importance are neglected, and thus distorted. In practical terms this often means too much noise in the baseline, along with the truncation of peaks or poor resolution of overlapping peaks. Enforcing smoothness and fidelity locally with a segmented or piece-wise application of the $\chi^2_S$ and $\chi^2_F$ functions may address this problem. Again, pursuing this line of enquiry is outside the current scope of our work.
Another interesting observation arises from Eqs. 3.3 and 3.4. In these two equations, deviations between chosen points are scaled relative to the standard deviations of those points. Since the square is taken, deviations that exceed the relevant standard deviation are comparatively emphasized and those less than this standard deviation comparatively suppressed. This suggests that this scaling factor could be manipulated to become differentially more sensitive to outliers (i.e., spectral features not modeled accurately). The limit of detection (LOD) is often defined as the concentration that yields an analyte signal that is three times the standard deviation of the background intensity [58][59], hence using $6\sigma$ instead of $\sigma$ in Eq. 3.3 will suppress the modeling of noise, within a $3\sigma$-band on either side of the mean, and emphasize the modeling or reconstruction of signal. In a like manner, smoothness may be emphasized in Eq. 3.4 by reducing the denominator. In previous work developing an automated method to estimate noise in spectra, we found that subtracting two successive shifts of a spectrum from the original (i.e., an approximation of the second derivative of a spectrum) discriminated well against baseline and signal and lead to a variance in the difference spectrum six times greater than that of the variance in the starting spectrum [55]. Dividing $\sigma$ in Eq. 3.4 by $\sqrt{6}$ thus provides a good starting point.

Table 3.1 shows a quantitative comparison of the results for the biChi methods to Savitzky-Golay filters. A major advantage of the biChi method is the deconvolution option; in other respects the performance is similar to or superior to that of the Savitzky-Golay filters.

### 3.4.3 Real Spectra

The biChi regularization method was tested on a UVRR difference spectrum (DNA at 45 °C - DNA at 35 °C). Difference spectra often present particular difficulties of
interpretation due to high noise levels and derivative-like features. The SNR of the difference spectrum was estimated, using an automated method to determine noise in spectra [55], to be about 9. Qualitative results shown in Figure 3.6 and quantitative results shown in Table 3.2 confirm the utility of biChi regularization in a real-world application: a substantial improvement in SNR is obtained while the chi-squared statistic of the residual remains acceptable (i.e., $\chi^2 < \text{number of channels}$). The reconstruction of the deconvolved spectrum clearly resolves two adenine vibrational modes near 1310 cm$^{-1}$ and 1340 cm$^{-1}$, respectively, the latter appearing derivative-like due to a temperature-induced shift in peak position.
Figure 3.6 (a) Superimposed UV resonance Raman spectra of salmon sperm DNA obtained at 35 °C (dark gray trace) and 45 °C (light gray trace); (b) the same spectra shown in (a) but with the abscissa expanded to enlarge the region around 1300 cm$^{-1}$; (c) the difference spectrum of those shown in (a) processed with biChi
regularization (“Original”; salmon sperm at 45 °C - salmon sperm at 35 °C; SNR=9; iterations = 6; lambda = 30); (d) enlargement for the region around 1300 cm⁻¹ shows higher resolution obtained with the ‘deconvolved’ biChi result (biChi-d). Panels (c, d) exemplify some of the typical attributes of difference spectra where features of interest are reduced compared to the level of noise present.

Table 3.2 The quantitative results of a UV resonance Raman difference spectrum (salmon sperm at 45 °C - salmon sperm at 35 °C; SNR=9) processed with biChi regularization in deconvolution mode (resulting in convolved and deconvolved regularized spectra).

<table>
<thead>
<tr>
<th>Figure-of-merit</th>
<th>biChi-c (i = 6)</th>
<th>biChi-d (i = 6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNR</td>
<td>592</td>
<td>226</td>
</tr>
<tr>
<td>( \chi^2 ) (residual)</td>
<td>966</td>
<td>1238*</td>
</tr>
</tbody>
</table>

*Not informative.

3.5 Conclusion

The \( \chi^2 \) function is often used to determine fidelity or goodness-of-fit by generating a metric of variance based on corresponding points between a modeled data set and a measured data set. It can also be used to determine smoothness when applied to adjacent points in the same data set. We demonstrated that this dual role can be exploited to design a digital filter where the \( \chi^2 \) function is implemented recursively for noise-reduction or for regularization with multidimensional optimization methods. The recursive implementation reduces to a three-point zero-order Savitzky-Golay filter as a special case. We have also implemented a regularization-with-deconvolution algorithm that yields excellent results. In order to compare the \( \chi^2 \)-based noise reduction methods to established standards, two Savitzky-Golay filters were also implemented: a 13-point zero-order variant and a 13-point second-order variant. The former was optimized for smoothness and the latter produced less smooth results but maintained better spectral resolution. It has been demonstrated that the \( \chi^2 \)–based approach produces results superior to those obtained with the Savitzky-Golay filters. The methods
presented here thus offer spectrosocists a family of excellent new tools for spectral noise reduction and resolution enhancement that afford attractive computational efficiency through the use of the same function to govern processing for both smoothness and fidelity.
Chapter 4: A Fully-Automated Iterative Savitzky-Golay Filter*

This chapter presents a new fully-automated, high performance noise reduction filter that is based on an iterative three-point zero-order Savitzky-Golay (SG) filter. Specifically, it is based on the very familiar moving window average concept (and thus very intuitive), is fast, and is easy to implement. The “iSG filter” is compared to an iterative median filter and a standard SG filter, and generally has superior results. This new filter employs some of the features used in the algorithms of previous chapters (such as the chi-squared statistic for a stopping criterion and an automated way of calculating the variance).

4.1 Introduction

The increasing prevalence of high-throughput and real-time procedures in a variety of applications is creating mounting pressures to automate data processing and data analysis in many and diverse fields. For example, efforts are being made to automate at least some aspects of the analysis of cardiovascular function [60], EEG signals to detect seizure activity [61], living human skin [62], microorganisms and their marker proteins [63], mineral and soil types [64][65], and NMR data [66][67][68]. Automation offers the ability to process large numbers of spectra, at high speed, in remote, inaccessible or hostile conditions, and with high precision and consistency. In addition, some data reduction often accompanies digital processing of large raw data sets, thus simplifying/reducing storage and/or transmission requirements.

* A version of this chapter has been published.
Manual processing generally produces superior results by objective measures of accuracy and precision when compared to a range of automated baseline correction methods [69][52], and this is likely to be the case for other automated spectral processing methods as well. Hence the demands for automated data processing and analysis are also influenced by the need to achieve a quality comparable to or higher than that obtainable via human intervention.

Spectroscopists are repeatedly confronted with the problem of separating their signals from low frequency background and high frequency noise (more below) and this need is often reflected in the methods advanced for automation: background removal [62][52][70], spike removal and noise estimation [55], and periodic signal reconstruction with concomitant noise discrimination [71]. In many, but not all cases [72], background and noise rejection are also necessary to perform further analyses such as two-dimensional correlation spectroscopy [73][43][44] and spectral classification [74]. Although several methods exist for automated baseline removal [52][70] and the field continues to evolve [62], filter design requires the specification or selection of filters and their parameters [41][75][29][76], and hence depends on human intervention, thus making noise removal more difficult to automate.

A promising development in the direction of automatic noise reduction is the three-pass iterative median filter (med$^3$) for three-dimensional (3D) electron tomographic data [77]. This simple and rapid filter appears to be well suited to 3D electron tomographic data due to the specific noise characteristics of such data, but its performance on one-dimensional (1D) spectra and especially on spectra originating from other methods, e.g., vibrational spectroscopies, NMR, etc. where the noise characteristics are different from that of electron tomography, is not known. Furthermore, according to the recommended protocol, the noise
reduction process is terminated after three passes since, in this specific application, the bulk of noise reduction occurs within the first three iterations. Additional iterations, possibly until the stationary “fixed point” is reached, may result in further noise reduction but thereafter no further filtering is achieved [77]. This leaves open the possibility of proceeding with iterated filtering until the fixed point is reached or an unacceptable degree of signal distortion occurs (more below).

Alternatives that we consider to have potential for automated noise reduction are the well-known least squares or Savitzky-Golay filters [34] and regularization methods such as the two-point maximum entropy method (TPMEM) [46]. While TPMEM excels at smoothing spectra with low signal-to-noise ratios (SNR), i.e., SNRs around 5 or less [46][47][56], it suffers from interference by high backgrounds [56]. The Savitzky-Golay filters are not susceptible to the same type of background interference as TPMEM, but they are known to produce some signal distortion [29][77][36]. Moreover, their iterative application using optimal parameters simply leads to more signal distortion [77], though a variant implementation, known as a Kolmogorov-Zurbenko filter [78][79], and using a few iterations [80][81], has proved useful. Other methods exist [82][83][84][85] and some require only minor parameter determination or constraint specification. However, these are still not fully automatable and their performance on spectroscopic data has not been assessed.

Savitzky-Golay filters have become the standard against which other smoothing methods are compared [41][46][56]; they are easy to implement, fast to execute, and provide generally good results [56]. By modeling the data locally with a polynomial, Savitzky-Golay filters minimize modeling errors, hence the blurring of signal edges [77] and peaks [36] may be mitigated with the use of higher order polynomials. Due to their simplicity, their
advantages as mentioned above, and their wide acceptance, the automation potential of the Savitzky-Golay filters may merit further exploration.

We have investigated their automation potential and found these filters to have excellent performance on both simulated and experimental Raman spectra. Specifically, we report on the use of a zero-order filter and small smoothing window to preserve signal integrity and the $\chi^2$-statistic to automatically terminate iteration before any significant signal distortion becomes apparent. Since iterations start with the smallest possible symmetric window (three-point), the lowest order filter (zero), and proceed until the stopping criterion is reached, there are no parameters to set and the entire noise-reduction procedure can be performed easily, swiftly, and completely without user intervention giving highly satisfactory results as we report below.

4.2 Theory

4.2.1 The Noise Reduction Problem

Measured data obtained on N detector channels can be represented as

$$m = x \ast b + n$$  \hspace{1cm} (4.1)

where the pure or underlying signal vector $x$ is convolved ($\ast$) with an instrumental blurring function $b$, and measurement noise, $n$, is added. The purpose of noise reduction, given our current objective, is to produce an estimate $y$ of $x \ast b$ from $m$.

4.2.2 The Automation Problem

Filter design requires the selection of a number of different filter parameters as mentioned above. In the case of Savitzky-Golay filters, this is particularly uncomplicated since there are only two parameters to select – these being the number of points in the local
modeling (i.e., smoothing) window and the polynomial order used for local modeling of the
data – but trial and error and intuition are nevertheless required in selecting them, thus
making automation difficult [36].

In general, a bigger window leads to more noise rejection but also to more signal
distortion [36]. As also pointed out above, the iterative application of (optimized) Savitzky-
Golay filters leads to increased signal distortion [77], suggesting that it amounts to increasing
the smoothing window as will be shown below. Taken together, these observations suggest
that starting the smoothing procedure with the smallest window possible (three-point) and
iterating subsequently would result in a gradual increase in the effective smoothing window
despite keeping the specified smoothing window fixed over all iterations (see below). This
procedure would result in the least amount of smoothing and the least signal distortion
initially, but both would increase upon further iteration. The problem now is to terminate the
iterations before signal distortion becomes unacceptable.

The $\chi^2$-statistic is widely used as a measure of similarity between different
distributions and often an acceptable value for this statistic is taken as $N$ [36]. Thus, when the
$\chi^2$-statistic exceeds this value, the signal distortion becomes unacceptable and, as is common
in regularization methods [36], we employ the $\chi^2$-statistic as stopping criterion here. The
statistic is given by:

$$\chi^2 = \sum_{i=1}^{N} \frac{(y_i - m_i)^2}{\sigma^2}$$  \hfill (4.2)

where the $y_i$ are filtered data points, the $m_i$ are the original input data points, and $\sigma$ is the
(homoscedastic) noise of the input data. Of key importance here is that $\sigma$ can be obtained
from a simple automated noise estimation procedure [55], thus, user intervention is
unnecessary throughout the filtering procedure. This noise estimation procedure, conceptually, consists of shifting a spectrum and subtracting it from the original non-shifted spectrum, thus doubling the variance of the original spectrum and stripping most of its peaks [55].

4.2.3 Implementation of an Iterative Savitzky-Golay Filter

The zero-order Savitzky-Golay filter, also known as a moving average filter, has coefficients that are identical and equal to $1/wp$ where $wp$ is the number of points in the smoothing window. Consider, by way of illustration, a spectrum consisting of five points being smoothed with a three-point Savitzky-Golay filter. Each point is subscripted: the first subscript refers to the serial position of the datum and the second to the iteration in progress.

\begin{align*}
  x_{21} &= \frac{x_{10}}{3} + \frac{x_{20}}{3} + \frac{x_{30}}{3} \\
  x_{31} &= \frac{x_{20}}{3} + \frac{x_{30}}{3} + \frac{x_{40}}{3} \\
  x_{41} &= \frac{x_{30}}{3} + \frac{x_{40}}{3} + \frac{x_{50}}{3}
\end{align*}

(4.3a) \hspace{1cm} (4.3b) \hspace{1cm} (4.3c)

In order to avoid edge effects and to demonstrate the principles involved, we proceed with the central point of the spectrum where the second iteration yields

\begin{align*}
  x_{32} &= \frac{x_{21}}{3} + \frac{x_{31}}{3} + \frac{x_{41}}{3} \\
  &= \frac{1}{3} \left( \frac{x_{10}}{3} + \frac{x_{20}}{3} + \frac{x_{30}}{3} \right) + \frac{1}{3} \left( \frac{x_{20}}{3} + \frac{x_{30}}{3} + \frac{x_{40}}{3} \right) + \frac{1}{3} \left( \frac{x_{30}}{3} + \frac{x_{40}}{3} + \frac{x_{50}}{3} \right) \\
  &= \frac{x_{10}}{9} + \frac{2x_{20}}{9} + \frac{3x_{30}}{9} + \frac{2x_{40}}{9} + \frac{x_{50}}{9}
\end{align*}

(4.4a) \hspace{1cm} (4.4b) \hspace{1cm} (4.4c)

Eq. 4.4c shows how repeated iterations lead to an increase in the window size – on the second iteration, the window increases implicitly by two points. This increase in window
size by two points is a general feature of the iteration process. Note also that as the window increases in size, the coefficients of the leading and trailing edges diminish leading to a more subtle filter response to the increase in window size when compared to a similar sized zero-order filter.

The coefficients that obtain after the first and second iterations are readily recognized as those from the first and second levels of the trinomial expansion, or Pascal’s pyramid, normalized with regard to their sum. Therefore one could determine the coefficients of an equivalent filter if one wishes to terminate iterations after a set number instead of proceeding until the $\chi^2$ stopping criterion is reached. Such an approach will reduce the computation time, especially if the first spectrum in a series is used to determine the number of iterations to use and a single iteration is used for the remaining spectra, but with equivalent filter coefficients. Figure 4.1(a) shows the changes in the shape of the filter with the first ten iterations, i.e., the shape of the filter effectively used if that many iterations of a three-point zero-order Savitzky-Golay filter were applied. As the number of iterations increase, the filter coefficients approximate those of a Gaussian distribution more closely; hence, a Gaussian filter is approximated as shown for the corresponding normalized frequency responses in Figure 4.1(b). The frequency response or gain of this filter is:

$$gain = \left[ \frac{1}{3^2} \frac{\sin^2(3\nu)}{\sin^2(\pi\nu)} \right]^{3/5}$$

(4.5)

where $\nu$ is the normalized frequency (i.e., the frequency normalized to the Nyquist frequency or $f/f_N$) and $k$ is the number of iterations. Advantages of Gaussian filters include reduced ringing [86] and near-optimality [87].
Figure 4.1 The iterative application of a three-point zero-order Savitzky-Golay filter produces filters that increasingly resemble a Gaussian distribution. (a) Equivalent filter coefficients (see text) for up to ten iterations of a three-point zero-order Savitzky-Golay filter. With every iteration of the filter, the effective filter window increases. The filter coefficients are centered for ease of viewing. There are only three identical coefficients on the first iteration (broken line) but twenty-three coefficients (thick solid line) on the tenth iteration. Coefficients are connected to aid in viewing. (b) Frequency response (transfer function; Bode plot) for the iterative three-point zero-order Savitzky-Golay filter from zero (broken line) to ten (thick solid line) iterations. The frequency response is normalized to the Nyquist frequency ($f_N$). The frequency response approaches that of a Gaussian filter as the number of iterations increase.

The amount of noise reduction can also be estimated with the use of the equivalent filter coefficients if the noise in the data set is approximately normally distributed. The noise reduction is then given by the relationship [29]:

$$\frac{\sigma^2_{filtered}}{\sigma^2_{unfiltered}} = \sum_{n=1}^{wp} c_n^2$$  \hspace{1cm} (4.6)
Although this relationship suggests that one may determine in advance how many iterations to perform, given a desired amount of noise reduction, it is not clear that one can know in advance whether this level of noise reduction would produce an unacceptable degree of signal distortion. We recommend therefore that the iterative approach be used, at least on the first spectrum in a series as discussed above.

As pointed out above, the iterative application of a zero-order Savitzky-Golay filter is also known as a Kolmogorov-Zurbenko filter [78]. This filter is denoted as KZ\((m, k)\), with the parameter \(m\) specifying the size of the moving average window and \(k\) specifying the number of iterations applied [78]. Our fully automated filter is therefore a special case of the KZ filter with \(k\) left free to increase until the \(\chi^2\) stopping criterion is reached and (always) with \(m = 3\), thus explaining the factors 3 and \(3^2\) in Eq. 4.5. Thus, with our approach, the specification of \(m\) and \(k\) through user intervention [80][81] is never required (although it remains possible), and this serves to differentiate, in a crucial practical respect, our implementation from the KZ filter. Indeed, our filter also differs from the Savitzky-Golay filter, in addition to being iterative, because the order of the filter and its window size do not have to be specified.

### 4.3 Methods

The iterative Savitzky-Golay filter (iSG) was tested on synthetic data, simulated to represent Raman spectra, and real Raman spectra. Briefly, we generated seven Lorentzian peaks, the first with a FWHM of approximately 10 channels and the remainder with FWHM of 6 channels each, resulting in an average FWHM of 7 channels. Spaced at reduced intervals, these peaks then constituted uncongested and congested spectral regions. They were convolved with a Gaussian distribution to simulate instrumental broadening, and were
added to a sigmoid (cumulative Gaussian distribution) baseline. Synthetic spectra of 1001 points each were generated with a given signal-to-noise ratio (SNR, defined as the intensity of the tallest peak \([signal \ maximum - signal \ minimum]\) relative to the baseline noise).

Scaled Gaussian (white) noise with an original standard deviation of 1.0 was added to the spectrum consisting of seven Lorentzian peaks to produce spectra with SNRs of 10. For SNR enhancement evaluations, the noise was determined from the first 150 points of the spectrum and the peak maximum was taken as the maximum value between channels 301 and 400. The set consisted of 10 spectra with unique noise distributions. The spectra were mirrored around their ends to provide padding (of filter window size) in order to mitigate potential edge effects from filtering.

For comparison, the med\(^3\) filter and two variants of the Savitzky-Golay filter with optimal parameters were applied to the same data. In a previous pilot study [56], a number of filter parameters were evaluated in order to optimize filter performance and an 11-point zero-order filter (SGw11n0) was found to give the best results, closely followed by a 21-point second order filter (SGw21n2).

The different implementations of the automated smoothing method were also evaluated on UV resonance Raman (UVRR) spectra and difference spectra. UVRR spectra were obtained from a sample of salmon sperm DNA (Sigma, St. Louis, MO) dissolved in distilled water to 500 \(\mu g/mL\) using a previously described custom-built fiber optic probe [11][53][54]. Resonance Raman scattering was generated using 257 nm excitation with an intracavity frequency-doubled \(Ar^+\) laser (Coherent, Santa Clara, CA), \(~100\) mW at the sample, and collected with a 0.67 m monochromator (McPherson, Acton, MA) with a 3600 G/mm holographic grating, 60 s integration time and 150 \(\mu m\) slit widths on a CCD camera.
(PixelVision, Tigard, OR). Spectra were normalized to the 930-cm$^{-1}$ peak of NaClO$_4$ added as an internal standard to a concentration of 40 mM.

Matlab 7.0 (The MathWorks, Natick, MA) was used to implement the filters. The computation platform was a 2.5 GHz dual processor PowerPC G5 running under Mac OS 10.4.1 (Apple Computer, Santa Clara, CA).
Figure 4.2 Typical examples of SNR 10 spectra smoothed with fully automated filters and their residuals (med3 – thick black; iSG – thick gray). (a) Smoothed spectra and residuals after both were iterated until the $\chi^2$ stopping criterion was reached for the iSG filter (approximately 40 iterations). (b) Smoothed spectra and residuals after 10 iterations show that most noise was removed by the iSG filter after 10 iterations. (c) Detail of (b) for channels 500 to 700 with the original noisy spectrum (thin gray) and noise-free spectrum (thin black) added shows that signal distortion was minimal for the iSG filter. (d) As in (c), but showing the results from the application of Savitzky-Golay filters with optimized parameters (SGw1n0 – thick black; SGw2n2 – thick gray).
4.4 Results and Discussion

4.4.1 Simulated Spectra

Figure 4.2 shows typical results comparing the automated smoothing methods as well as results from two optimized Savitzky-Golay filters. Statistical comparisons are reported in Table 4.1. In Fig. 4.2a, simulated spectra smoothed with the automated methods are shown along with their residuals. The figure shows clearly that much better smoothing is obtained with the iSG filter than the three-point med$^3$ filter. It should be noted that the results shown in the figure are obtained after the $\chi^2$ stopping criterion was reached for the iSG filter, meaning both filters were iterated the same number of times (approximately 40). This deviates from the 3 iterations suggested for the med$^3$ filter, but increases the amount of smoothing [77].

Since the med$^3$ filter was originally devised for 3D data and intended to be applied as a three-point filter constituting a 3x3x3 volume, its extension to a one-dimensional (1D) spectrum is not without difficulty: in order to benefit to the same extent from noise reduction a twenty-seven point window has to be used on 1D data but this comes at the cost of considerable signal distortion, especially flattening of the peaks (data not shown).

Although the residual does not show any noticeable artifacts as would be expected with signal distortion, the iSG filter does reduce peak height. This effect can be much mitigated by arbitrarily stopping the process after 10 iterations and results for both filters are shown in Fig. 4.2b. The residual of the iSG filter shows that most noise was removed at this stage. The choice of stopping after 10 iterations is in the same spirit as stopping the med$^3$ filter after 3 iterations, but nevertheless represents a parameter that needs to be set or optimized as discussed further below.
Figure 4.2c shows a detail from Fig. 4.2b for channels 500 to 700. The results of the smoothed spectra are here superimposed on the original noisy and noise-free spectra for comparison. Greater fidelity to the noise-free spectrum is very much in evidence for the spectrum smoothed by the iSG filter. The superior performance of the iSG filter is also evident when compared to Savitzky-Golay filters using optimal parameter settings; spectra smoothed by the latter two filters (SGw11n0 and SGw21n2) are shown in panel Fig. 4.2d.

The demonstration of qualitative results in Figure 4.2 is confirmed with the quantitative data shown in Table 4.1 for the same filters on similar spectra. The results show that the iSG automated filter performs significantly better than the med$^3$ automated filter as one would expect given the noise characteristics of the simulated data. The iSG filter also performs as well as or better than Savitzky-Golay filters with optimized parameters.

It should be noted that spectra smoothed with the iSG filter show less point-to-point variation than those smoothed by any of the other filters, although this is not captured by the metrics reported in Table 4.1 but is evident in the detail of a representative example presented Figure 4.3. Filtering time for a 1001-point spectrum was approximately 0.545 seconds (mean for 10 spectra). It should also be relatively uncomplicated to extend this filter to the processing of higher-order (e.g., 2D) data sets.

**Table 4.1** Comparative results (mean ± standard error) for the automated smoothing methods, i.e., the iterative three-point Savitzky-Golay (iSG) filter and the iterative three-point median filter (med$^3$) after the $\chi^2$ stopping criterion was reached and after 10 iterations, as well as for single iterations of Savitzky-Golay filters with optimized parameters (11-point zero-order and 21-point second-order, respectively). These filters were applied to a set of 10 spectra with unique distributions of Gaussian noise.

<table>
<thead>
<tr>
<th>Figure-of-merit</th>
<th>iSG ($\chi^2$)</th>
<th>iSG (10)</th>
<th>med$^3$ (10)</th>
<th>SGw11n0</th>
<th>SGw21n2</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMS</td>
<td>0.042 ± 0.001</td>
<td>0.034 ± 0.001</td>
<td>0.063 ± 0.001</td>
<td>0.034 ± 0.001</td>
<td>0.033 ± 0.001</td>
</tr>
<tr>
<td>SNR</td>
<td>40 ± 4</td>
<td>30 ± 1</td>
<td>16 ± 1</td>
<td>33 ± 2</td>
<td>32 ± 2</td>
</tr>
<tr>
<td>$\chi^2$</td>
<td>1009 ± 1</td>
<td>845 ± 12</td>
<td>616 ± 7</td>
<td>962 ± 17</td>
<td>919 ± 16</td>
</tr>
</tbody>
</table>
4.4.2 Real Spectra

Difference spectra can be much degraded by noise because peaks are subtracted but noise is additive, i.e., the opposite of (ensemble) averaging occurs. In such cases smoothing may be a valuable aid for the presentation and interpretation of spectra. An example of a difference spectrum (salmon sperm at 45 °C – salmon sperm at 35 °C) is shown in Figure 4.4. Superimposed on the difference spectrum are the spectra obtained by smoothing with the automated filters, shown in Fig. 4.4a, and those obtained from smoothing with Savitzky-Golay filters using optimized parameters, shown in Fig. 4.4b. The iterations for both automated filters were terminated upon reaching the \( \chi^2 \) stopping criterion for the iSG filter, thus both filters were iterated the same number of times. These results confirm the prior observations on simulated spectra and extend them to real-world data.
Figure 4.4 A UV resonance Raman difference spectrum (salmon sperm at 45 °C - salmon sperm at 35 °C, SNR~9; thin gray trace). (a) Processed with the two fully automated filters: an iterative three-point median filter (med³, thick gray); and an iterative three-point Savitzky-Golay filter (iSG, thick black). Iterations continued until the $\chi^2$ stopping criterion for the iSG filter was reached (iterations ~40); (b) Processed with Savitzky-Golay filters with optimal parameter settings (11-point zero-order, thick black; 21-point second-order, thick gray). The results show that excellent smoothing is obtained with the fully automated iSG filter.

4.4.3 Stopping Criteria and Parameter Settings

Although the automated iSG filter we present here does not require the user to specify any parameters, it is not true that the filter is parameter-free. We have devised a simple, fully automated filter by taking advantage of one extreme of parameter settings: the lowest possible order (zero-order) and the smallest possible symmetric window (three-point).
Repeated application of this filter results in the natural but implicit evolution of these parameters since the window expands and the “order” increases (although not in integer amounts). Because we use the $\chi^2$ stopping criterion, another (though widely accepted) “parameter” is implicitly pressed into service. The med$^3$ filter also uses an implicit parameter since it is applied for only three iterations [77]. From a practical point of view though, these parameters are either easy or unnecessary to determine and these filters thus represent an advantage when compared to more sophisticated filters where specialized design knowledge is required. The TPMEM filter should be included in this small group of automated filtering techniques since it has only two parameters to set and these are also easily specified [46].

It is worthwhile mentioning that exceptionally good results can be obtained with an automated filter based on a second-order Savitzky-Golay filter and starting the iteration process with a five-point window (data not shown). The higher order polynomial permits a more accurate local fitting of the data [77][36] but tends to require more iterations. However, we are not able to justify the selection of the order and window size on an a priori basis as we explained above for our current implementation of the automated (three-point, zero-order) Savitzky-Golay filter. Nevertheless, this higher-order variant may be of value to many readers.

In the present case, we have chosen the $\chi^2$ stopping criterion as justified above, but one could also terminate iterations using different criteria. For example, if noise reduction follows an exponential-like decay as suggested by the reduction in root mean square error observed for the med$^3$ filter [77], then one may wish to terminate iterations when the noise in the smoothed spectrum has decreased to 1/e of its level in the original spectrum. An investigation of alternate stopping criteria is set aside for possible future pursuit.
4.5 Conclusions

High throughput measurement methods are creating vast quantities of data and consequently generate a pressing need for automated methods of analysis. Due to the fact that the parameters of smoothing filters require optimization [41][29][77] and that few fully automated smoothing techniques exist, there is presently a bottleneck in the post processing of data [77][85]. We are aware of only one “fully” automated filter, the iterative median filter, constructed specifically to take advantage of 3D tomographic data [77]. For many spectroscopic techniques, e.g., the vibrational spectroscopies, an automated filter does not seem to exist. Compared to progress in automated background removal [52], progress in automated smoothing methods is scant indeed. Toward this end, we have introduced a novel implementation of a version of Savitzky-Golay smoothing that is suitable for full automation.

Results on spectroscopic data show that the iSG filter improves on the iterative median filter and is comparable to or better than Savitzky-Golay filters with optimized parameters. We expect this fully automated smoothing method to be of immediate and practical use to researchers requiring automated smoothing of spectral data.

This chapter presents a fully-automated spectral baseline removal/correction algorithm. Based on a Savitsky-Golay filter—but in a model-free manner—it uses a large-window moving average to estimate the baseline. A peak-stripping method is used in conjunction to remove spectral peaks (until the baseline is corrected). Based on multiple passes (or iterations), the method starts with the largest possible window and gradually reduces it until acceptable baseline correction, based on the $\chi^2$-statistic, is achieved. After processing, the baseline-corrected spectrum should yield a flat baseline. Results, obtained on both simulated and measured Raman data, are presented and discussed. This automated method employs aspects of Chapter 4’s automated filter design, such as the use of a zero-order SG filter, the chi-squared stopping criterion, and an automated method for calculating the variance.

5.1 Introduction

Due to an increase in high-throughput instrumentation and otherwise improved and accelerated collection efficiencies, the automated processing of large data spectral collections is increasingly sought-after [66][61][63][64][67][68][60][65][62]. One of the common processing needs—and one of the most difficult to automate—is the removal of spurious background signals [62][88][52] which, in general, may interfere with further processing [73][43][89], complicate quantification [62][89][90], or hinder the presentation and

---

*A A version of this chapter is in press.
visualization of relevant data [91]. For example, when raster scanning a plant leaf to obtain information about the distribution of its chemical constituents using Raman microspectroscopy [92][93] over an area of 70 µm x 100 µm with 5-µm resolution, over 300 spectra with steeply sloping baselines are generated. In the absence of an automated method, these spectra need to be corrected by hand. Clearly, this can require a considerable amount of time and can quickly lead to fatigue accompanied by increased likelihood of operator error/bias. For example, based on a variety of spectra, we have found the average time to correct a baseline manually to be about 100 s [52]. Thus, correcting 300 spectra would take in excess of 8 h to correct manually. In addition to combating fatigue and maintaining consistency, it is furthermore desirable to perform automated baseline correction with the same or better speed and accuracy than attainable manually by appropriately trained operators [52][69].

Despite a number of near-fully-automated methods being devised [62][52][89][94][95][96][97] since the advent of electronic computers, with the exception of a second derivative-based method by Rowlands and Elliott [98] requiring low noise spectra, a truly model-free, fully-automated procedure remains elusive. Extant methods often require filter coefficient settings, wavelet selection, polynomial specification, stopping criterion specification, and so on. One possible approach to this problem is to use a method that can be implemented with its parameters initially at their extremes and to iterate subsequently to their optima or to acceptable values. The latter thus requires a suitable stopping criterion. We have previously used this approach to fully automate a Savitzky-Golay smoothing filter with initial parameters a zero order and a three-point window and iterating until meeting the widely used $\chi^2$ stopping criterion [99]. We follow a similar approach here to create a model-free baseline
removal method that can be implemented in a fully-automated and ‘parameter-free’ manner, the latter in the sense that user intervention is not required.

Savitzky-Golay filters are widely known filters and have become the standard against which other smoothing methods are compared [41][46][56]; they are easy to implement, fast to execute, and provide generally good results [56]. The zero-order Savitzky-Golay filter is also known as a moving average filter. A general characteristic of smoothing filters is that lower frequencies are passed and that the passband of low frequencies narrows as the window size increases [41]. Thus a very large window zero-order Savitzky-Golay filter could be used to pass the baseline but attenuate high frequency noise and, to a lesser extent, the signals of interest. When combining this approach iteratively with peak stripping, effective baseline estimation can be obtained. We have therefore investigated the potential of implementing the zero-order Savitzky-Golay filter, in conjunction with peak stripping and the Pearson $\chi^2$ stopping criterion as further explained below, to produce a truly model-free and fully-automated baseline removal method.

5.2 Theory

5.2.1 The Baseline Estimation Problem

Measured data obtained on $N$ detector channels can be represented as

$$ m = (b + x) \ast p + n $$

$$ = b \ast p + x \ast p + n $$  \hspace{1cm} (5.1)$$

where the underlying signal vector, $x$, plus the baseline, $b$, is convolved (*) with an instrumental blurring function, $p$, and measurement noise, $n$, is added. The purpose of
baseline estimation is to produce an estimate of \( b \) from \( m \). We use a large window zero-order Savitzky-Golay filter (i.e., a moving average filter) to estimate the baseline, where the estimated baseline is given by the filter output. Baseline estimation is improved by stripping signals above the baseline from the spectrum. This process is iterated until no more signal is removed, that is until the signal-to-noise ratio (SNR) of the remaining signals reaches a specified value, or until some termination threshold is reached [52].

5.2.2 The Automation Problem

Filter design requires the selection of a number of different filter parameters as mentioned above. In the case of Savitzky-Golay filters, this is particularly uncomplicated since there are only two parameters to select – these being the number of points in the local modeling window and the polynomial order used for local modeling of the data. In the present case, this is further simplified by using the zero-order filter since it has the best low pass characteristics among polynomial filters [41][36][29]. Nevertheless, selection of the window size is required, making automation difficult [29]. In general, a bigger window leads to more noise rejection and also to more signal distortion [29]. However, signal distortion is not problematic because we employ signal stripping as part of the baseline estimation procedure. Thus, we overcome the problem of window size selection by starting with the largest possible window and incrementing its size as needed.

Instead of the criteria above regarding when to terminate peak stripping, one could take an alternative approach by asking when the baseline is estimated well enough. The answer to this is simple: the baseline should be, within reason, flat. When baseline correction is applied to the ‘corrected’ spectrum, the estimated baseline should be flat after peak stripping. Moreover, flatness can be assessed with the \( \chi^2 \)-statistic. Importantly, if the baseline
is not flat, the estimate obtained on the second correction (i.e., on the ‘corrected’ spectrum) will reflect regions of overcorrection or undercorrection of the first correction. Thus a better estimate of the baseline is arrived at by summation of the successive estimates on prior results. As soon as an estimate is flat (i.e., the baseline is zero), further estimates will all be flat (disregarding accumulation of small errors due to the peak stripping operations) and no further improvements accrue in the overall estimate. Estimation should therefore be terminated when an acceptably flat baseline is obtained. Thus

\[
\hat{b} = \hat{b}_1 + \hat{b}_2 + \ldots + \hat{b}_i, \quad \hat{b}_i \neq 0
\]

(5.2)

where \( \hat{b}_i \) indicates the \( i \)-th successive estimate of the baseline \( \hat{b} \), not de novo, but on the spectrum after termination of the previous correction sequence.

If the window is too large and baseline correction cannot be effected, there should be a failure of convergence for the \( \chi^2 \)-statistic (between a flat line and \( \hat{b}_i \)) with increasing \( i \).

Thus, in the absence of a decreasing \( \chi^2 \)-statistic, the window size is reduced

\[
\text{windowsize} = N-n, \quad n = 1, 2, 3 \ldots N-3
\]

(5.3)

and the process is started anew. In the results reported here, though, we reduced the window size to speed computations according to

\[
\text{windowsize} = \text{integer}(N/n), \quad n = 1, 2, 3 \ldots N/2
\]

(5.4)

The \( \chi^2 \)-statistic is widely used as a measure of similarity between different distributions and often an acceptable value for this statistic is taken as \( N \) [36]. When the \( \chi^2 \)-statistic exceeds this value, the baseline of the \( i \)-th corrected spectrum is not acceptably flat and further iterations are required if there is evidence of convergence. Thus, as is common in regularization methods [36], we employ the \( \chi^2 \)-statistic as stopping criterion here. Since
calculating the latter is based on a very simple automated noise estimation procedure [55], user intervention is not required.

5.3 Methods

The automated baseline estimation (ABE) method was tested on synthetic data (simulated to represent Raman spectra) and real Raman spectra. Briefly, we generated seven Lorentzian peaks, the first with a FWHM of approximately 10 channels and the remainder with FWHM of 6 channels each, resulting in an average FWHM of 7 channels. Spaced at reduced intervals, these peaks then created uncongested and congested spectral regions. They were convolved with an instrumental point spread function approximated by a Gaussian distribution (5 channels to 1 \( \sigma \)) to simulate instrumental blurring, and were added to a baseline. Three types of baseline were generated: (i) an exponential baseline, (ii) a Gaussian baseline, and (iii) a sigmoidal baseline (cumulative Gaussian) to investigate the method’s performance on a variety of dissimilar baselines. Baselines had signal-to-baseline ratios (SBR, defined as the intensity of the tallest peak \([\text{signal maximum} - \text{signal minimum}]\) relative to the baseline height \([\text{baseline maximum} - \text{baseline minimum}]\)) of 1, 0.1, and 0.01. Synthetic spectra of 1001 points each were generated with a signal-to-noise ratio (SNR, defined as the intensity of the tallest peak \([\text{signal maximum} - \text{signal minimum}]\) relative to the baseline noise) of 10 by adding a constant level of Gaussian (white) noise with an original standard deviation of 1.0 to the spectrum consisting of seven Lorentzian peaks. We simulated spectra with a constant but reasonably challenging level of noise, irrespective of baseline intensities, in order to keep the number of variables tractable. Each of the nine sets (three baseline types with three SBRs each) consisted of 10 identical spectra but with unique noise distributions. Spectral ends were extrapolated before baseline correction with a second order polynomial to
provide padding (of window size) in order to mitigate potential edge effects from baseline correction. Typical examples of spectra from these sets are shown in Fig. 5.1.

![Figure 5.1](image)

**Figure 5.1** Sample simulated Raman spectra superimposed on three baseline types: (a) an exponential baseline; (b) a Gaussian distribution baseline; and (c) a sigmoidal baseline. Baselines of all three types also had three signal-to-baseline ratios each, for example approximately 0.01 (a), 0.1 (b), and 1 (c). All nine sets of spectra contained 10 identical spectra but with independent, normally distributed noise to give a signal-to-noise ratio (tallest peak/baseline noise) of ~10.

Overall, the baseline correction approach consists of three nested sets of iterations. The innermost routine performs peak stripping, the middle routine checks for baseline flatness and invokes the inner routine using the current window size as long as further improvements in baseline flatness (of the already processed spectrum) can be obtained, and
the outer routine applies the stopping criterion or adjusts the window size with subsequent invocation of the middle routine. Details follow below.

Starting with the largest possible window (i.e., of the same size as the spectrum), a baseline is estimated by convolving the spectrum with a zero-order Savitzky-Golay filter. After baseline estimation, all spectral regions above the baseline are removed. More specifically, peaks, above the estimated baseline plus twice the noise standard deviation (as determined from the starting spectrum), are reduced to the level of the baseline. Thereafter, zero-mean random Gaussian noise, estimated from the starting spectrum [55], is optionally added (discussed below) to those positions where peak stripping occurred. Thus, although noise in the Raman spectra may be heteroscedastic for high SNR spectra, the spectrum after progressive stripping of its peaks retains noise characteristics similar to those of the original spectrum. Noise injection is aimed at diminishing possible distortions arising from sharp discontinuities between stripped and intact regions. The baseline estimation followed by stripping is then repeated on the resulting, partially stripped spectrum. It is terminated when the maximum number of iterations is exceeded, no more stripping occurs, or aggressive stripping causes the estimated baseline to become negative. The estimated baseline, final stripped spectrum, corrected spectrum (i.e., the starting spectrum - estimated baseline), and number of stripping iterations required are returned as outputs from the peak stripping (i.e., the innermost) routine.

A well-corrected baseline is expected to be flat and of zero intensity. If the $\chi^2$-statistic between a flat line and the estimated baseline continues to decrease, the corrected spectrum is again subjected to the peak stripping-baseline estimation procedure. These baseline-testing iterations provide successive intermediate baseline estimates and are recorded along with
their $\chi^2$-values and their sequence number. If not, the process is terminated and a concluding baseline estimate is made by autosmoothing [99] the final stripped spectrum. An overall baseline estimate is then made by summing the successive intermediate estimates obtained from $\chi^2$-based testing. This baseline is subtracted from the original spectrum to obtain a baseline-corrected spectrum.

If the minimum $\chi^2$-value recorded above exceeds the $\chi^2$ stopping criterion, the window size is decreased according to Eq. 5.4 and the entire procedure is restarted using the original spectrum. The process is terminated when the $\chi^2$ stopping criterion is met and when repeated subtractions no longer provide an improvement in $\chi^2$-values. At this point, a good guess of the window size and the number of estimates required for baseline flattening is available. However, due to noise in the spectrum and noise added when stripping peaks, there is some variability in the parameters obtained: repeated applications of the procedure to exactly the same spectrum leads to slightly different parameter outcomes. Therefore parameter space is then doubled and a search is made for the parameters corresponding to the best $\chi^2$-value. These are then used to obtain the final result. Along with $\chi^2$-values, the root-mean-square (RMS) error for final estimated baselines, vis-à-vis synthetic baselines, is determined as a figure-of-merit. A flow chart is given in Fig. 5.2.
To assess the performance of the ABE procedure on real data, Raman spectra were obtained from samples of solid triacontanol and triacontanoic acid as well as a tomato skin. The tomato skin was mechanically removed from the fruit. Raman spectra were collected on a Raman microscope with a CCD detector (RM 1000, Renishaw, Gloucestershire, UK). A
50x/0.75 NA (Numerical Aperture) objective (Leica Microsystems, Wetzlar, Germany) was used resulting in power at the focal point from the 785 nm diode laser of ~35-45 mW. Spectra were collected over a 350-2000 cm\(^{-1}\) range with collection times of 10-30 s, one to two accumulations, and using a 100 µm slit width.

Matlab 7.0 (The MathWorks, Natick, MA) was used to implement the ABE procedure. The computation platform was a 2.5 GHz dual processor PowerPC G5 running under Mac OS X 10.4.1 (Apple Computer, Santa Clara, CA). Code for the ABE procedure is available from the authors upon request.

### 5.4 Results and Discussion

#### 5.4.1 Simulated Spectra

The sample spectra in Fig. 5.1 suggest that manual correction of such spectra may be difficult. Where the SBR is very low (e.g., Fig. 5.1a) signal peaks are hardly noticeable. In other cases, even with moderate SBR (e.g., Fig. 5.1b), strong baseline curvatures and overlapping peaks make the estimation of the baseline problematic. With high SBR (e.g., Fig. 5.1c), the relatively low SNR complicates baseline location. We therefore consider the simulated data non-trivial from the perspective of performing baseline correction on them.

Figures 5.3, 5.4, and 5.5 show results for flattening the exponential, Gaussian, and sigmoidal baselines, respectively. Statistical results and parameters values are reported in Table 5.1. Spectra recovered from baselines with SBRs of 1, 0.1, and 0.01, are shown in panels a, c, and e, respectively (gray traces, superimposed black traces are those of the set of blurred Lorentzians) while panels b, d, and f show the corresponding estimated (gray) and true (black) baselines. Remarkably, even with low SBRs, a spectrum can still be recovered, albeit at the cost of some distortion.
Figure 5.3 Spectra (gray traces), after automated baseline correction, for exponential baselines with signal-to-baseline ratios of (a) 1, (c) 0.1, and (e) 0.01 with the “ideal” (noiseless and baseline-free) spectrum superimposed thereon (black). The estimated (gray) and true baselines (black) for the corresponding signal-to-baseline ratios are shown in panels (b), (d), and (f), and the differences between them as black traces negatively offset for clarity in panels (a), (c), and (e).
Figure 5.4 Spectra (gray traces), after automated baseline correction, for Gaussian baselines with signal-to-baseline ratios of (a) 1, (c) 0.1, and (e) 0.01 with the “ideal” (noiseless and baseline-free) spectrum superimposed thereon (black). The estimated (gray) and true baselines (black) for the corresponding signal-to-baseline ratios are shown in panels (b), (d), and (f), and the differences between them as black traces negatively offset for clarity in panels (a), (c), and (e).
Figure 5.5 Spectra (gray traces), after automated baseline correction, for sigmoidal baselines with signal-to-baseline ratios of (a) 1, (c) 0.1, and (e) 0.01 with the “ideal” (noiseless and baseline-free) spectrum superimposed thereon (black). The estimated (gray) and true baselines (black) for the corresponding signal-to-baseline ratios are shown in panels (b), (d), and (f), and the differences between them as black traces negatively offset for clarity in panels (a), (c), and (e).

Smoothing the concluding baseline estimate (i.e., the remnant spectrum after peak stripping) has the advantage of reducing correlated noise in the final flattened baseline.
Therefore, if the final corrected spectrum is also smoothed, some degree of denoising of the baseline is accomplished because both high and low frequency noise in the flat baseline are then reduced. For example, the corrected spectrum in Fig. 5.3a shows the flattened baseline with low frequency correlated noise largely absent while the presence of such noise in a baseline would appear similar to the slight artifact near channel 150 in Fig. 5.3e.

Baselines with greater curvature are harder for the ABE procedure to deal with, thus the Gaussian and sigmoidal baselines generally require smaller windows than the exponential baselines to be estimated effectively (e.g., Table 5.1). Since the method starts with the largest possible window, relatively more computational time is needed to reach the required smaller window sizes. Furthermore, using smaller windows for baseline estimation results in the erosion of peaks in areas of peak congestion (e.g., channel 500 to channel 700 in Fig. 5.1c). Thus, more of the peak intensities are partitioned into the baseline as is evident when comparing Fig. 5.4e and 5.5e to Fig. 5.3e where the differences between actual and estimated baselines are shown as black traces offset below the recovered spectra.
Table 5.1 Comparative results (mean ± standard deviation) for the automated baseline-correction method on three sets of simulated Raman spectra with very high, moderately high, and low baselines as indicated by the signal-to-baseline ratios (SBR). Every spectrum in each set of 10 spectra had a unique distribution of Gaussian noise giving a signal-to-noise ratio (SNR) of 10. Upon automated termination, the termination parameters (zero-order Savitzky-Golay smoothing window size, number of successive baseline-testing iterations needed) for each spectrum were recorded. The termination $\chi^2$-value and the root mean square (RMS) error for every estimated baseline were also recorded. Their means and standard deviations are given in the table.

<table>
<thead>
<tr>
<th>Baseline</th>
<th>SBR (approximate)</th>
<th>Window mean ± sd</th>
<th>Iterations mean ± sd</th>
<th>$\chi^2$ terminal mean ± sd</th>
<th>RMS mean ± sd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponential 0.01</td>
<td>254 ± 64</td>
<td>9 ± 2</td>
<td>13.62 ± 8.95</td>
<td>4.84 ± 2.34</td>
<td></td>
</tr>
<tr>
<td>Exponential 0.1</td>
<td>451 ± 209</td>
<td>7 ± 1</td>
<td>8.63 ± 5.23</td>
<td>3.28 ± 0.87</td>
<td></td>
</tr>
<tr>
<td>Exponential 1</td>
<td>801 ± 258</td>
<td>7 ± 1</td>
<td>2.93 ± 1.19</td>
<td>1.41 ± 0.15</td>
<td></td>
</tr>
<tr>
<td>Gaussian 0.01</td>
<td>159 ± 53</td>
<td>9 ± 2</td>
<td>19.61 ± 12.68</td>
<td>4.29 ± 0.51</td>
<td></td>
</tr>
<tr>
<td>Gaussian 0.1</td>
<td>449 ± 383</td>
<td>7 ± 2</td>
<td>79.86 ± 112.01</td>
<td>24.27 ± 33.90</td>
<td></td>
</tr>
<tr>
<td>Gaussian 1</td>
<td>688 ± 341</td>
<td>6 ± 2</td>
<td>12.14 ± 6.05</td>
<td>4.29 ± 2.93</td>
<td></td>
</tr>
<tr>
<td>Sigmoidal 0.01</td>
<td>434 ± 391</td>
<td>9 ± 3</td>
<td>277.53 ± 436.93</td>
<td>29.52 ± 44.30</td>
<td></td>
</tr>
<tr>
<td>Sigmoidal 0.1</td>
<td>214 ± 34</td>
<td>7 ± 1</td>
<td>6.43 ± 3.48</td>
<td>2.46 ± 0.91</td>
<td></td>
</tr>
<tr>
<td>Sigmoidal 1</td>
<td>518 ± 183</td>
<td>6 ± 2</td>
<td>3.58 ± 1.60</td>
<td>1.87 ± 0.59</td>
<td></td>
</tr>
</tbody>
</table>

Sharp curvatures and large windows also generate the opposite problem – here, sharp curves where the baseline is underestimated are subtracted from the spectrum and thus become partitioned into the peak “space”. Figure 5.6a shows examples of repeated baseline correction (more below) on the same spectrum originally with a SBR 0.01 exponential background (e.g., Fig 5.1a). Pronounced artifacts due to curvature are evident in some of these results centered near channels 20 and 180. The greatest errors coincide with the use of the largest windows and with more baseline correction iterations using these large windows. However, there is no correspondence between the terminal $\chi^2$-values (i.e., terminal flatness) achieved and the presence or severity of artifacts.
Figure 5.6 A spectrum with exponential baseline and a signal-to-baseline ratio of 0.01 repeatedly (10 times) subjected to baseline correction (a, 10 grayscale traces) shows that automated baseline estimation with noise injection at the position of stripped peaks can produce artifacts (e.g., near channels 20 and 200) but often with reduced peak base erosion. When these spectra are averaged, the average (b) can be baseline-corrected to reduce the presence of artifacts and erosion in peak bases (c, light gray trace) and optionally smoothed with an automated procedure (c, dark gray trace). An alternative is to multiply the 10 spectra, set all baseline values below a threshold to zero, and get the 10-th root of the product spectrum (d, light gray trace) with optional subsequent smoothing bases (d, dark gray trace). In both (c) and (d) the “ideal” (noiseless and baseline-free) spectrum is superimposed in black for comparison.

The fact that repeated processing of the same spectrum does not yield identical outcomes is due to the peak stripping procedure where noise, estimated from the spectrum, is injected into the stripped spectrum at those positions formerly occupied by peaks. In the absence of this intervention, baseline correction on the same spectrum mentioned above (i.e., originally with a SBR 0.01 exponential background), produces no artifacts and identical results when repeated 10 times. Values, corresponding to the first entry in Table 5.1, are 112
(window), 5 (baseline-testing iterations), 5.58 ($\chi^2_{\text{terminal}}$), and 3.89 (RMS). Where the same approach (i.e., no noise injection during peak stripping) is taken on 10 identical spectra but each with independent noise, these values are 106.50 ± 5.80 (window), 5.10 ± 0.32 (baseline-testing iterations), 6.12 ± 0.84 ($\chi^2_{\text{terminal}}$), and 3.94 ± 0.17 (RMS) giving means ± standard deviations. Likewise, there were no artifacts.

The absence of artifacts, however, comes at the cost of increased peak erosion (i.e., some of the peak is partitioned into the baseline, as in Fig. 5.4e vs. Fig. 5.3e). Thus, where artifacts do not overlap with peaks, an interesting option arises if one were able to discriminate against or ameliorate the effects of artifacts. One possibility is to do a number of repeated corrections on the same spectrum (e.g., 10 times, Fig. 5.6a) and then to average these (Fig. 5.6b) before baseline correction (Fig. 5.6c, light gray trace) and smoothing (Fig. 5.6c, dark gray trace) of the average spectrum. An alternative, based on previous work [100], is to multiply all the spectra (corresponding channels). Since the baseline correction leads to a near-flat baseline with approximately zero mean, the baseline of the multiplied spectrum is much reduced compared to the peaks. Spectral intensity values less than a threshold (e.g., three times the standard deviation of the noise in the original spectrum, raised to the power of the number $n$ of spectra multiplied together) are set to zero (Fig. 5.6d, light gray trace) and the $n$-th root ($n$ here is 10) of the resulting spectrum (Fig. 5.6d, dark gray trace) is obtained.

### 5.4.2 Real Spectra

Real Raman spectra obtained from solid triacontanol and triacontanoic acid, plant growth promoters [101][102], and from the skin of the tomato fruit have moderate to pronounced sloping baselines as shown by the gray traces (scaled and offset for ease of viewing) in Fig. 5.7. The black traces in Fig. 5.7 show the results after correction.
Application of the ABE method to the tomato skin spectrum produced the result shown in Fig. 5.7a where a maximum window size of 1651 wavenumbers and 12 baseline-testing iterations were used. This result was not completely satisfactory. However, when started with an initial window half the size (i.e., \( n = 2 \) in Eq. 5.4), convergence occurred to a window of size 826 wavenumbers (i.e., \( n = 2 \) in Eq. 5.4) and 9 baseline-testing iterations at this window size were used. The result is shown in Fig. 5.7b. A similar problem arose for the other two spectra. The triacontanol spectrum, after baseline correction, is shown in Fig. 5.7c where a maximum window size of 1651 wavenumbers and 11 baseline-testing iterations were used. When started with an initial window half the size (i.e., \( n = 2 \) in Eq. 5.4), convergence occurred to a window of size 332 wavenumbers (i.e., \( n = 5 \) in Eq. 5.4) and 4 baseline-testing iterations at this window size were used (Fig. 5.7d). Fully-automated baseline correction of the triacontanoic acid produced the spectrum shown in Fig. 5.7e where convergence to the maximum window size of 1651 wavenumbers occurred and 9 baseline-testing iterations were needed. Likewise, when started with an initial window half the size, convergence occurred to a window of size 332 wavenumbers (i.e., \( n = 5 \) in Eq. 5.4) and 3 baseline-testing iterations at this window size were used and the spectrum in Fig. 5.7f was obtained.
Figure 5.7 Real Raman spectra from tomato skin (a, b), triacontanol (c, d), and triacontanoic acid (e, f) shown as gray traces and as black traces after baseline correction. Fully-automated baseline correction of the tomato skin spectrum starting with the maximum window size (= spectrum size) leads to incomplete baseline removal (a) but a corrected spectrum can be obtained by starting with a smaller (i.e., half the maximum) initial window (b). As in (a) and (b), respectively, but for triacontanol (c), (d). As in (a) and (b), respectively, but for triacontanoic acid (e), (f).
We also observed some artifacts as a consequence of ABE, especially at spectral edges (e.g., right edge, Fig. 5.7b). Thus attempts to pad the spectra by extrapolating beyond their edges were not entirely successful and require additional attention. Furthermore, the trade-off with ‘denoising’ of correlated noise is the loss of small peaks from corrected spectra (cf. Fig. 5.7c to Fig. 5.7d near 1400 cm\(^{-1}\)). In addition to these problems, we noted that repeated processing resulted in nearly-identical results for both triacontanol and triacontanoic acid, thus the methods proposed above (i.e., averaging, multiplication) for discriminating against artifacts induced by ABE could not be applied to them. Nevertheless, despite these difficulties, good performance was obtained on real Raman spectra where peaks of varying width as well as different baseline features, such as exponential- and linear-like features, were present within the same spectrum.

5.4.3 Stopping Criteria and Parameter Settings

Our aim is to devise a model-free baseline removal procedure that can track arbitrary baselines adequately as well as one that is “parameter-free” and can be implemented without recourse to user intervention. In practice, however, whenever the choice of a parameter is circumvented, conditional statements seem to emerge in compensation. Thus the question is: what tradeoff – between parameter selection, parameter space search, and the invocation of thresholds – can be implemented that is effective and reasonable?

On theoretical grounds, as explained above, the search of parameter space starts at the one extreme of employing the largest possible window and using it in a zero-order Savitzky-Golay filter to estimate the baseline repeatedly. If the filter window is too large, the success in estimating the baseline is limited and repeated applications can begin to generate distortions in the baseline causing it to become progressively less flat. The trend is captured
by testing with the $\chi^2$-statistic (latest baseline estimate compared to a flat line using the noise level of the original spectrum) and when it starts to increase, it suggests that no further improvements could be obtained (Fig. 5.8, bottom rows). This establishes the optimal number of successive baseline removal iterations and the minimum $\chi^2$-value for a window of given size. Furthermore, if the minimum $\chi^2$-statistic obtained is in excess of the number of channels, the result is not deemed acceptable [36].

**Figure 5.8** The $\chi^2$-value “error” surface of baseline correction (no noise injection at the position of stripped peaks) on a spectrum with exponential baseline and a signal-to-baseline ratio of 0.01. Lower $\chi^2$-values are darker; $\chi^2$-values above 100 were truncated to 100 and these are white. The surface indicates that large windows (e.g., number of channels/window size = 1, 2) are not effective in baseline estimation, that smaller windows are more effective, but require more iterations (e.g., number of channels/window size = 3, 4, 5), and that much smaller windows produce aggressive peak erosion with less low $\chi^2$-values (e.g., number of channels/window size = 16+). Thus the objective is to find a local minimum with the largest window size (e.g., number of channels/window size = 8, 9, 10). The automated procedure reported here accomplishes this by starting at the large extreme of the window size parameter and searches until the local $\chi^2$-minimum is reached, i.e., when $\chi^2$-values increase with increasing window size and increasing repeated estimations.

Thus, the window size is reduced and the search continues with a smaller window that is better able to fit the baseline. Even though, at some point, the $\chi^2$-statistic falls below the threshold for acceptability, further improvements can be obtained and should be sought. This is because, if baseline correction is completely successful, an estimated baseline on a
corrected spectrum should be completely flat (the success condition). In that case, the $\chi^2$-statistic would be zero. Therefore, the lower the $\chi^2$-statistic, the better the result (Fig. 5.8, middle rows).

With smaller windows, however, some amount of peak erosion begins to occur and continued baseline removal appears to aggravate this. Consequently, the $\chi^2$-statistic again starts to increase. With even smaller windows, peak erosion is more aggressive and less small $\chi^2$-values are obtained (Fig. 5.8, top rows). Taken together then, the objective is to find a local $\chi^2$-value minimum with the largest possible window size. We mention here a local minimum since the global minimum is likely to occur with the smallest window size, i.e., where both peaks and baseline are removed. The results shown in Fig. 5.8 were generated when correcting the simulated spectrum with exponential SBR 0.01 baseline.

Conceptually, our ABE procedure is based on a parameter space that is progressively simplified. First, by selecting a Savitzky-Golay filter to estimate the baseline since it has only two parameters (order and window size) and can model any arbitrary baseline. Second, by selecting a zero-order filter because this produces the greatest amount of smoothing. Third, by initiating a search for the optimal value of the remaining parameter, starting at its large extreme, and iterating to an expected nearby local $\chi^2$-value minimum. If the local $\chi^2$-value minimum is also acceptable from a statistical point of view, the solution is in hand. Note that one could claim that repeated baseline corrections simply introduce a new parameter (number of repeats or iterations needed) to replace filter order, now fixed at a constant value, as a parameter; however, even when retaining the filter order as a full parameter, repeated baseline corrections would have utility.
Even if this method is not used for automated baseline correction, it provides other benefits. For example, baselines estimated with the method presented here can be used for modeling purposes (e.g., with the commonly-used polynomial fitting approaches) [62][52][95][97]. Additionally, the same peak stripping and $\chi^2$-testing approach can be used, but based on polynomial modeling instead of the zero-order Savitzky-Golay filter. Even though a polynomial of a given order is then used, an essentially model-free method results because the baseline is repeatedly estimated and corrected. The same approach is likely adaptable to other parameter-sparse methods, such as the median filter method [103], as well. Finally, one could use the results (window size, iterations) of the ABE method to guide the subsequent selection of baseline-estimation filter parameters for a variety of other methods.

5.5 Conclusions

High throughput measurement methods are creating vast quantities of data and consequently generate a pressing need for automated methods of analysis. These include the correction of sloping baselines in spectra. We have presented here a fully-automated and model-free baseline estimation procedure capable of baseline correction without, or at most with minimal, user intervention. This ABE method is based on a zero-order Savitzky-Golay filter and the quality of the corrected baseline is assessed with the $\chi^2$-statistic which thereby serves as a computational stopping criterion. Because we start with the largest possible window, and progressively decrease it, the method is inherently robust with regard to wide and/or overlapping peaks.

Although the proposed ABE method is not restricted to applications involving Raman spectroscopy, simulated and real Raman spectra were used here for evaluation purposes. Our
results show that the ABE method consistently produces baseline-flattened spectra of high quality without user intervention. However, the method is not entirely error-free. Where baselines are very prominent with strong curvatures, correction requires the use of smaller-than-ideal filter windows causing peak erosion. In other cases, an undesirable local minimum occurs at the maximum window size leading to incomplete baseline removal. The latter problem can be overcome by starting the iteration process with a smaller initial window (i.e., half the maximum) as discussed above. Ideally though, we wish to avoid manually setting such ‘arbitrary’ thresholds. Minor problems such as edge artifacts also occur. Nevertheless, we believe that the major benefits of none or minimal user intervention, the ability to handle large data sets with great consistency, and the high quality of baseline-corrected spectra will be of great interest to many spectroscopists with substantial baseline-correction needs.
Chapter 6: Matrix–Based 2D Regularization for SNR Enhancement *

This chapter presents a new two-dimensional (2D) regularization-based spectral image processing algorithm for SNR enhancement, called Matrix-MEM (or MxMEM). This new method is an extension of TPMEM (Chapter 2) into two dimensions. Furthermore, just as in the algorithms presented in earlier chapters, MxMEM employs the chi-squared statistic as a stopping criterion. MxMEM is tested and compared in two application areas and found to be superior in both—in x-ray computed tomography versus the so-called 2D TPMEM algorithm, and with Raman spectra (both simulated and real) in comparison to 1D TPMEM.

6.1 Introduction

Traditional approaches to signal-to-noise (SNR) enhancement of Raman spectroscopy data operate on an individual spectrum. These include filtering approaches, such as Savitzky-Golay filters [41][99], chi-squared-based filters [104], and regularization methods, such as the maximum entropy method (MEM) [32][15] and TPMEM [46]. One spectrum at a time is passed into the algorithm for processing.

The guiding principle in these one-dimensional (1D) methods is the use of some form of averaging of neighboring points, either by calculating a moving window average or by converging toward the average of neighboring points. As will be shown, this principle can be easily and effectively extended to two dimensions.

If multiple measurements of an analyte are made across, for example, a range of temperatures, a time sequence, or some other perturbation or sampling protocol, then a set of related spectra will be generated. Although the noise components of the spectra are uncorrelated, the underlying signals, though changing due to a perturbation, are highly correlated. If the spectra are arranged into a matrix, then every point within a spectrum now has a relationship with not only its two adjacent elements, but also the corresponding points of the two adjacent vectors (and beyond). This relationship among vectors allows additional (inter-vector) information to be used for the reconstruction of each individual spectrum. That is, the averaging can now include points from neighboring vectors as well as those within a given vector. Thus, an SNR enhancement method that treats this collection of spectra as a two-dimensional (2D) “image” could take advantage of the additional information and (potentially) produce a better reconstruction of individual spectra as compared to traditional 1D methods.

Savitzky-Golay (SG) filters [34] have long been the standard of comparison for traditional 1D smoothing techniques because of their ease of implementation, fast execution times, and reasonably good results. An alternative method, which is somewhat more complex in implementation but which performs considerably better in comparison to SG filters, is the two-point maximum entropy method (TPMEM), developed by Greek, et al [46]. TPMEM is a regularization method, based upon the maximum entropy method (MEM) [32][15] and performs well on low-SNR data [56]. As the name denotes, every two adjacent points within a vector are used in the calculation of the algorithm’s objective function. It was originally developed for SNR enhancement and deconvolution of Raman spectroscopy data
but has recently been applied successfully in other fields, such as signal enhancement of magnetoencephalography data [105][106].

A 2D form of TPMEM was recently developed by Jirasek, et al., and was demonstrated to perform extremely well when applied to image smoothing of X-ray computed tomography (CT) data of radiosensitive polymer materials [47]. Although the overall algorithm processes a true 2D image, it does so by smoothing individual vectors. As explained below, TPMEM is still used as the underlying 1D “smoothing engine”, but applied recursively within an overall 2D method. However, as shown later, some limitations are encountered by processing vectors instead of a complete matrix.

There are many examples of 2D regularization, both MEM-based and non-MEM-based, such as for image processing [107] and restoration [108] and edge detection [109]. Methods using matrices also exist, for example, Birkinshaw [110][111] and Wells and Birkinshaw [112] use a matrix to hold the detector response curves of a detector array, but only process one spectrum at a time. However, none of these are derived from TPMEM, and thus none use order restriction for 2D entropy calculations. In this work, we introduce the matrix maximum entropy method (MxMEM), a new spectral image-processing regularization algorithm that is a matrix-based extension of TPMEM into two dimensions. MxMEM iteratively and two-dimensionally smoothes the entire matrix by exploiting the relationship among data within a two-dimensional four-point “box”— two in each adjacent vector (Fig. 6.1a)—and then simultaneously processes all 4-point boxes across the image in the calculation of the algorithm’s objective function.
Figure 6.1 (a) A conceptual “image” (or matrix) of 4 rows, 8 columns, and 32 elements (or pixels). MxMEM calculates and sums the entropies of all adjacent 4-point “boxes” on all adjacent vectors (e.g., the 4 black dots of rows 1 and 2 form a box). Thus, as shown, any two adjacent rows contain 7 adjacent boxes, and any two adjacent columns contain 3 adjacent boxes. In calculation of Eq. 6.9, the 4-point box combines two points \((\hat{x}_i, \hat{x}_{i+1})\) from vector a, and two \((\hat{x}_{i+2}, \hat{x}_{i+3})\) from vector b. (b) A synthetic “cylinder” image (noise free) used to emulate aspects of X-ray computed tomography data. (c) Image of (b) with noise added to produce SNR = 5. (d) Image of (c) reconstructed by MxMEM. (e) Image of (c) reconstructed by TPMEM (T1D), one vector at-a-time. (f) Comparison of MxMEM (MM) and T1D reconstructed profiles from center of image, showing superiority of 2D methods; noisy and noise-free profiles also shown.
We demonstrate the utility of MxMEM for SNR enhancement in two application areas. First, we compare it to 2D TPMEM in smoothing CT data used by Jirasek et al. [47], and we show an improvement (e.g., in terms of root mean squared error and SNR) over their published results. Second, we apply MxMEM to 2D Raman “images”. As a comparison, we employ the original (1D) TPMEM on the same data. The results show that 2D processing of related spectra generally yields better SNR enhancement than 1D processing of the same spectral data. Overall, we demonstrate that MxMEM has several advantages over 2D TPMEM for general image processing and TPMEM for smoothing of Raman spectra.

6.2 Theory

Here, we first present a brief review of the fundamentals of Greek’s TPMEM [46] algorithm and Jirasek’s [47] extension into 2D TPMEM. Then we present the underpinnings of MxMEM.

6.2.1 One-Dimensional Two-Point Maximum Entropy Method (T1D)

In Raman spectroscopy, a measured spectrum can be modeled as

\[ m = x \ast b + n \]  

(6.1)

where \( m \) is a vector of measured data (from \( N \) channels of a detector); \( x \) is the underlying signal vector, which is convolved (*) with an instrumental blurring function, vector \( b \); the convolution pair is then combined with additive white noise, vector \( n \). A single value of \( b \) is assumed (implying a uniform blurring) in order to simplify the discussion that follows. In the signal reconstruction problem, an estimate of \( x \), which will be called \( \hat{x} \), is sought by solving the inverse problem using a regularization approach as follows:

The objective function of T1D to be minimized is
\[ F = -S + \lambda \chi^2 \]  

(6.2)

where \( S \) is the two-point information theoretic entropy introduced by Shannon [51], \( \lambda \) is a LaGrange multiplier, and \( \chi^2 \) is the chi-squared distribution function [36][57]. Note that \(-S\) (negative entropy) operates as the measure of smoothness, which is added to the \( \lambda \)-weighted fidelity function, \( \chi^2 \). Optimum smoothness occurs when \( S \) is maximized (but \(-S\) minimized).

The two-point entropy function is defined [51] as

\[ S = \sum_{i=1}^{N-1} -\left( p_i \ln p_i + q_i \ln q_i \right) \]  

(6.3)

where

\[ p_i = \frac{\hat{x}_i}{\hat{x}_i + \hat{x}_{i+1}} \]  

(6.4a)

and

\[ q_i = \frac{\hat{x}_{i+1}}{\hat{x}_i + \hat{x}_{i+1}} \]  

(6.4b)

The \( \hat{x}_i \) and \( \hat{x}_{i+1} \) pairs, formed by every two adjacent values of \( \hat{x} \), represent two-point probability distributions. The chi-squared statistic (when the data is a vector) is defined as

\[ \chi^2_V = \sum_{i=1}^{N} \left( \frac{m_i - \hat{x}_i}{\sigma^2_V} \right)^2 \]  

(6.5)

where \( \sigma^2_V \) is the vector’s noise variance, \( m_i \) is the measured data from detector channel \( i \), and \( \hat{x}_i \) is the \( i \)th value of \( \hat{x} \) (i.e., the reconstructed estimate of the spectrum). The variance \( \sigma^2_V \) was calculated using a simple noise estimation procedure that, conceptually, consists of
shifting a spectrum and subtracting it from the original non-shifted spectrum, thus doubling the variance of the original spectrum and stripping most of its peaks [55].

The T1D technique, using a conjugate-gradient method [36][113] as an optimizer, solves for the spectral estimate vector $\hat{x}_v$, and $\lambda$, which minimize $F$, subject to the (stopping criterion) constraint of $\chi^2 \cong N$.

### 6.2.2 Two-Dimensional Two-Point Maximum Entropy Method (T2D)

T2D uses all of the above equations, plus $\chi^2$ for the image (or matrix, $M$), which is used to calculate the stopping criterion ($\chi^2 \cong N\times N$):

$$\chi^2_M = \sum_{i=1}^{N\times N} \frac{(m_i - \hat{x}_i)^2}{\sigma^2_M}$$

(6.6)

where $\sigma^2_M$ is the noise variance for the image, $m_i$ are the data points for the entire image ($N\times N$ points), and $\hat{x}_i$ are the values of the reconstructed spectral estimate matrix. The top-level 2D algorithm inputs a data image (matrix) and outputs a reconstructed image: the spectral estimate matrix, $\hat{M}$. In operation, T2D smoothes one vector at a time (from the data image) by calling the T1D “smoothing engine”. Vectors are selected in row-column-row-column (etc.) order, until one entire pass of the image has been sent to, and processed by, T1D. That is, first row 1 is smoothed, then column 1, then row 2, then column 2, etc. After each pass of the image, $\lambda$ is updated, and the stopping criterion for the image, Eq. 6.6, is tested.

### 6.2.3 Matrix Maximum Entropy Method (MxMEM)

This new algorithm’s objective function has the same form as Eq. 6.2 above:
\[ F_{\text{num}} = -S_{\text{num}} + \lambda \chi^2_{\text{num}} \]  (6.7)

However, in contrast to T1D’s entropy function, which is based on every two adjacent data points, MxMEM’s entropy function is based on *four* neighboring points across two vectors (Fig. 6.1a). This is the simplest symmetrical 2D extension of TPMEM. For efficiency, we write our entropy equation at two levels. First, the lower level, which we call the “kernel”, calculates the entropy \( S_k \) for one “box” of four elements. Comparing to Eqs. 6.3 and 6.4, and following Shannon [51] for four points, the entropy kernel is of this form:

\[ S_k = -\left( p_i \ln p_i + q_i \ln q_i + r_i \ln r_i + s_i \ln s_i \right) \]  (6.8)

and the 4-point probability distributions are:

\[ p_i = \frac{\hat{x}_i}{\hat{x}_i + \hat{x}_{i+1} + \hat{x}_{i+2} + \hat{x}_{i+3}}, \ldots, s_i = \frac{\hat{x}_{i+3}}{\hat{x}_i + \hat{x}_{i+1} + \hat{x}_{i+2} + \hat{x}_{i+3}} \]  (6.9)

Second, the higher level sums the entropies of all adjacent 4-point boxes within the entire matrix/image (where \( x_{\text{box}} \) is a 4-point vector):

\[ S_{\text{num}} = -\sum_{i=1}^{M^{1/N}-1} S_k(x_{\text{box}}) \]  (6.10)

The calculation of the chi-squared term in the objective function—and MxMEM’s stopping criterion—are identical to T2D’s Eq. 6.6, (except that the limit is \( M^*N \)):

\[ \chi^2_{\text{num}} = \sum_{i=1}^{M^{1/N}} \frac{(m_i - \hat{x}_i)^2}{\sigma^2_M} \]  (6.11)

MxMEM, analogous to T1D, uses the same conjugate-gradient method [36][113] as an optimizer, but solves for the spectral estimate matrix \( \hat{x}_M \), and \( \lambda \), which minimize \( F \)—subject to the (stopping criterion) constraint of \( \chi^2 \equiv M^*N \). Following Jirasek et al. [47],
we include an extra multiplier ($X$) that allows “tuning” of the result for more smoothness or more fidelity (where $X > 1$ increases smoothness, $X < 1$ increases fidelity). Thus,

$$\chi^2 \equiv X \ast M \ast N$$

(6.12)

6.2.4 Gradient Matrix Calculation

Because all three algorithms use a conjugate gradient method [36][113] (CGM) for optimizing the objective function, a gradient calculation is required in order to form the gradient matrix (or vector, in TPMEM’s case). A gradient matrix (or vector) is a collection of individual partial first derivatives (also called directional derivatives). That is, each point within the gradient matrix is the gradient (directional derivative) of $F$ at that point, and is derived from the same corresponding point of the spectral estimate matrix, $\hat{X}_M$ and the input data matrix, $m$.

Here we present only MxMEM’s gradient function. For simplicity, we calculate its gradient as two separate components—entropy and fidelity—and then add them together. Thus, using the kernel approach, the gradient equation of the entropy kernel component (a 4-point box) is:

$$\frac{\partial S_k}{\partial x_j} = \sum_{i=1}^{4} \left( 1 + \ln \frac{\hat{x}_i}{x} \right) \left( \frac{\hat{x}_i}{x^2} \right) - \left( 1 + \ln \frac{\hat{x}_j}{x} \right) \left( \frac{1}{x^2} \right)$$

(6.13)

where $j$ also equals 1 to 4 and $\hat{x}_j = \hat{x}_i + \hat{x}_{i+1} + \hat{x}_{i+2} + \hat{x}_{i+3}$. An individual entropy gradient value ($\frac{\partial F_1}{\partial x_i}$) is calculated by applying Eq. 13 to the 4-point boxes which surround that point.

Next, the entire entropy gradient matrix ($g_{ent}$) is formed by calculating, and accumulating, all of the adjacent 4-point box gradients across the image. For the fidelity gradient matrix ($g_{fid}$),
the equation for one point is not based on the 4-point box but only on corresponding points of $m_i$ and $\hat{x}_i$:

$$\frac{\partial F_2}{\partial \hat{x}_i} = \frac{2\lambda}{\sigma_M^2} (m_i - \hat{x}_i) \quad (6.14)$$

Finally, the total gradient matrix equation, for one point, is:

$$g(i) = \frac{\partial F_{mn}}{\partial \hat{x}_i} = \frac{\partial F_1}{\partial \hat{x}_i} + \frac{\partial F_2}{\partial \hat{x}_i} \quad (6.15a)$$

Alternately, the total gradient matrix equation can be expressed as:

$$g = g_{ent} + g_{ld} \quad (6.15b)$$

Additional detail on the gradient calculation (implementation concept and C coding) is provided in Appendix 2.

6.2.5 MxMEM Algorithm Flow (Simplified)

All three algorithms have closely analogous flows, but MxMEM minimizes a matrix-based objective function, as follows:

1. Initialize spectral estimate matrix ($\hat{x}_M$) to the measured noisy data matrix
2. Initialize $\lambda_{high}$ and $\lambda_{low}$ values (e.g., 1 and $10^{-10}$, respectively)
3. Set $\lambda = \lambda_{high}$
4. Minimization loop (optimizes $\lambda$ to find the minimum of all the minimums of Eq. 6.7)
   (a) Reconstruct $\hat{x}_M$ by minimizing Eq. 6.7
   (b) If $\chi^2 < X^*M^*N$, set $\lambda_{high} = \lambda$, else set $\lambda_{low} = \lambda$
   (c) Set $\lambda = (\lambda_{high} + \lambda_{low})/2$. 
(d) Repeat the above three steps until the stopping criterion is satisfied (within a tolerance).

6.2.6 Optimization and the Two-Dimensional Advantage

In this discussion, for simplicity, we begin with T1D as the example (Eqs. 6.3-6.5). Optimization of an objective function requires, by implication, the optimization of its component functions (smoothness and fidelity). Under some conditions, their joint optimization is mutually exclusive, requiring a trade-off in order to optimize the overall objective function. For smoothness, -S is minimized when the probabilities \( p_i \) and \( q_i \) are equal, a condition that is (practically) satisfied for any number such that \( \hat{x}_i \) and \( \hat{x}_{i+1} \) are identical. Certainly this will hold true for \( \hat{x}_i \) and \( \hat{x}_{i+1} \) being equal to the mean \( \bar{m} \) of \( m_i \) and \( m_{i+1} \), or both \( \hat{x}_i \) and \( \hat{x}_{i+1} \) being equal to either \( m_i \) or \( m_{i+1} \).

For fidelity, \( \chi^2 \) is minimized when \( \hat{x}_i = m_i \) and \( \hat{x}_{i+1} = m_{i+1} \); however, the smoothness component requires the estimates to be equal for optimality. Furthermore, it is known that the sum of squared deviations of points from their mean is less than the sum of squared deviations from any other point [57]. Because the fidelity component relies on a sum of squared deviations, identical estimates that are equal to the mean will minimize this sum.

Consequently, the concurrent requirement to optimize both fidelity and smoothness suggests that the optimal estimates \( \hat{x}_i \) and \( \hat{x}_{i+1} \) are bracketed on one side, respectively, by the measurements \( m_i \) and \( m_{i+1} \) and on the other side by the mean \( \bar{m} \). As an aside, this analysis suggests that a good starting value for any estimate is either the raw spectrum (i.e., \( \hat{x}_i = m_i \)), or a two-point moving average.
The optimization problem therefore resides in determining how far to adjust the estimate from its starting value at one end of the bracket toward the other end of the bracket to minimize $F$. If, for example, the raw spectrum is used as an initial estimate, subsequent estimates have to be adjusted away from the measurement values toward the mean values. The precise extent of the adjustment required will depend on the value of $\lambda$ and the local gradients of the two components of $F$. Both these components are parabolic in shape but differ in steepness around the minimum. Thus, starting out at the fidelity minimum (i.e., $\hat{x}_i = m_i$), and increasing the estimate by some small amount toward the joint mean $\bar{m}$, is likely to increase the error due to the $\chi^2$ “parabola” by a small amount, but decrease the error due to the -$S$ “parabola” by a larger amount. For a $\lambda$ of unity, the optimum should therefore be reached at that point where the estimate corresponds to equal gradients of the two sub-functions (i.e., smoothness, fidelity) of $F$.

It is also known from statistical inference that better estimates of the true mean are obtained by considering more observations [57]. This can be done in one of two ways or in combination. First, within the same vector, if several points adjacent to a given measurement $m_i$ are highly correlated, these adjacent points can be included in the estimate for $\bar{m}$ to improve it. Second, as previously mentioned, if a set of related spectra are combined into a matrix to create an “image”, then the adjacent points will also occur in the neighboring vectors.

Hence, all the adjacent points in a 2D neighborhood (both the intra- and inter-vector points) can potentially be used to improve the estimate $\bar{m}$, provided that these points are all highly correlated except for independent noise. Therefore, one could argue that a 2D approach provides a better estimate of the true local value of a spectrum and provides
reduced point-to-point variation. This is because the one endpoint of the range confining the optimal value of a given point-estimate fluctuates less between adjacent points in a spectrum. Thus, estimates calculated on tighter ranges would provide, ultimately, better smoothing of noisy spectra. Larger boxes or rectangles could therefore also be used, but rectangles need to have their long dimension aligned with the directions in which adjacent points are most highly correlated (exclusive of noise). Put differently, narrow peaks require smaller boxes to reduce smoothing-induced distortions, an issue that is worth considering when processing data from different spectroscopic methods.

6.2.7 Summary of MxMEM versus 2D TPMEM

T2D functions similarly to MxMEM but, because T1D is the underlying smoothing engine, has four noticeable differences. First, due to the row-column sequencing, T2D’s optimization of point \( m_i \) only involves adjacent points in the same row and the intersecting column, but MxMEM also includes the effect of optimizing the points that are diagonal to \( m_i \) (Eq. 6.13). Second, T2D’s row-column sequencing creates a (slightly) non-symmetrical reconstruction, since some points are smoothed more than others, as Jirasek acknowledges [47]. Third, this sequencing also produces different results if the image is transposed. In contrast, MxMEM, which smoothes all 4-point boxes simultaneously, produces a symmetrical reconstruction and is impervious to image transposition. Fourth, T2D only processes square images, whereas MxMEM can process square or rectangular ones. Padding the data would allow T2D to process rectangular images, but at the cost of much increased memory storage and computational time (e.g., a 1000x10 Raman image would be converted to 1000x1000).
6.3 Methods

The MxMEM algorithm was tested by simulating it with two types of application data—simplified CT data (synthetic only), and Raman data (synthetic and real, measured Raman spectra). For the CT data, three images identical to those reported in Jirasek’s work [47] were used: U-shaped, wedge, and cylinder images (only the cylinder image is shown here, Fig. 6.1b). Ten copies of each image type, each with unique noise (e.g., Fig. 6.1c), were generated at SNRs of 5 and 20, resulting in a total of 60 images.

Raman synthetic data were created as follows. First, a “base” spectrum of seven Lorentzian peaks, across 1000 points, was generated (Fig. 6.2a, inset). Peaks were spaced at progressively reduced intervals in order to create uncongested and congested spectral regions. The first peak has a full width at half-maximum (FWHM) of approximately ten channels. The remainders have FWHM of six channels each, resulting in an overall average of FWHM of seven channels. Second, ten copies of the base spectrum were combined into a matrix to form a 1000x10 “image” and modified such that the peak heights varied across the set of ten spectra (Fig. 6.2a). This collection of spectra simulates intensity changes due to measurements at ten different temperatures. Peak heights were made to vary across the ten spectra at various rates: one sigmoidal, one exponential, four linear, and one constant. For example, peak 1 varied sigmoidally (downhill) from spectrum 1 through spectra 10. Third, each of the 10 spectra was convolved with a Gaussian-shaped instrumental point spreading function (IPSF) in order to simulate instrumental broadening. Fourth, each spectrum was added to a sigmoid (cumulative Gaussian distribution) baseline. Fifth, scaled Gaussian (white) noise, with an original standard deviation of 1.0, was added to each image. Finally, ten images, each with unique noise, were generated at SNRs of 3, 5, 10, and 20, resulting in
40 “Raman images” total. For comparison purposes, T1D was also tested with the same collections of Raman spectra (but processed as individual vectors in the traditional method).

Figure 6.2 (a) Detail of Raman synthetic data—base “image”, consisting of 10 spectra made from the spectrum shown in the inset, and modified such that peak intensities vary, at different rates, across the 10 spectra. (Inset) Detail of Raman synthetic data—a “base” spectrum of seven Lorentzian peaks, across 1000 points, used as a building block to create an artificial Raman “image”; only channels 200-800 shown. (b) Detail of a typical Raman spectrum, original SNR of 5, after regularization by MxMEM (MM), as part of an image, and T1D (as an individual spectrum), showing superior smoothing by MxMEM—as evidenced by the peak shape, “peak-height-matching”, and relatively close matching of the peak backgrounds. The noise-free and noisy original spectra are shown for comparison.
In addition, MxMEM and T1D were evaluated on ultraviolet resonance Raman (UVRR) spectra (and their derived difference spectra) measured across 12 temperatures (15 °C - 70 °C). Standard desalted Deoxyribonucleic Acid (DNA) polyadenine pentamers were obtained from IDT Technologies (Coralville, IA). Pentamers were centrifuged prior to use and then dissolved to a sample concentration of 60 µM in a buffer solution consisting of 10 mM phosphate buffer (pH 7.0), 1 M NaCl and 1 mM EDTA. Concentrations were quantified by ultraviolet-visible (UV-Vis) spectroscopy using the 260 nm absorption band of the bases. The 1005 cm\(^{-1}\) ring trigonal bend mode (\(v_{23}\)) [114] of residual benzamide from the synthesis process was used to align the spectra. UVRR spectra were generated with 244-nm excitation light from a frequency-doubled Ar\(^+\) ion laser (Innova 90C FreD, Coherent Inc., Santa Clara, CA) and collected with a custom fiber-optic probe [11][53][54]. Power at the samples was approximately 20 mW. Samples were stored in a 500-µL centrifuge tube, placed in a rotating holder, and spun during measurement to minimize photodegradation. A dielectric interference filter (Barr Associates Inc., Westford, MA) was used to remove Rayleigh scattering. Raman scattered light was dispersed with a 1.0-m-focal-length monochromator (Model 2061, McPherson Inc., Chelmsford, MA) using a 3600 grooves/mm holographic grating (Model 8358-1004-0, McPherson Inc., Chelmsford, MA). A liquid-nitrogen-cooled charge-coupled device (CCD) detector operating at -120 °C (Spec-10 400B, Roper Scientific, Trenton, NJ) was used for acquisitions of 10 s from each sample.

All algorithms were implemented primarily in the C programming language, but interfaced to Matlab R2009a (The MathWorks, Natick, MA), and utilized C routines from \textit{Numerical Recipes in C} (NRC) [36]. The computation platform was a DELL Inspiron 1545
personal computer with a 2.0 GHz Intel Pentium Dual-Core T4200 CPU, 3 GBytes of RAM, running under Windows XP (with Service Pack 3).

6.4 Results and Discussion

6.4.1 Computed Tomography Data

MxMEM and T2D were compared against simulations of synthetic CT data. Following Jirasek [47], we used the root mean squared error (RMSE), a complementary Pearson’s correlation coefficient (PEARSC), and SNR as metrics.

We compared the algorithms at the multiplier values of \( X = 0.9, 1.0, 1.1, 1.2, 1.4 \) and using the stopping criterion, Eq. 6.12. Then, we used the lowest value of RMSE (across all the values of \( X \)) to define “best fidelity”. As a result, MxMEM’s performance was better in most, but not all, cases. However, this is based on the arbitrary choice of \( X \) values.

An alternate way to assess the performance of the two methods and reveal their fundamental capabilities is to capture the true RMSE minimum point (“min-point”) independent of the stopping criterion. This is done by running the simulations and calculating the three metrics at every iteration. This analysis reveals that both algorithms’ min-points will sometimes occur one iteration before or after the stopping criterion point.

Figure 6.1d shows MxMEM’s reconstruction, at the min-point, of the noisy cylinder image of Fig. 6.1c (original SNR of 5). Table 6.1 provides a statistical comparison of MxMEM versus T2D, based on true min-point, for the cylinder image. MxMEM showed a consistently better performance than T2D on all three images and all metrics, but with less margin of improvement on the wedge and U-shape images (data not shown). All \( t \)-test \( p \)-values were \( p < 0.05 \) except for the U-shape image at SNR5, RMSE: \( p = 0.78 \).
Table 6.1 Comparative results, using the RMSE “min-point”, for MxMEM vs. T2D, on the CT Cylinder image, using 10 unique-noise images, at two different initial SNR values (5, 20). Thus the mean ± standard deviation of 10 reconstructed images are shown for the given metrics. Note that for RMSE and PEARSC a lower number represents a better reconstruction. The t-test values of probability $p$ are also given.

<table>
<thead>
<tr>
<th>CYLINDER image</th>
<th>SNR 5</th>
<th>SNR 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure of Merit</td>
<td>MxMEM</td>
<td>T2D</td>
</tr>
<tr>
<td>RMSE</td>
<td>$0.0169 \pm 0.0002$</td>
<td>$0.0189 \pm 0.0003$</td>
</tr>
<tr>
<td>$p &lt; 0.01$</td>
<td>$p &lt; 0.01$</td>
<td></td>
</tr>
<tr>
<td>PEARSC</td>
<td>$0.00146 \pm 3.86 \times 10^{-5}$</td>
<td>$0.00186 \pm 5.07 \times 10^{-5}$</td>
</tr>
<tr>
<td>$p &lt; 0.01$</td>
<td>$p &lt; 0.01$</td>
<td></td>
</tr>
<tr>
<td>SNR</td>
<td>$52.4 \pm 3.3$</td>
<td>$41.6 \pm 2.3$</td>
</tr>
<tr>
<td>$p &lt; 0.01$</td>
<td>$p &lt; 0.01$</td>
<td></td>
</tr>
</tbody>
</table>

The superior performance of MxMEM over T2D is derived from the fact that, as explained earlier, it processes an image in a symmetrical, true matrix-based manner, and utilizes adjacent and diagonal points in its “optimization window”. In contrast, T2D processes in a slightly non-symmetrical manner and does not use diagonal points in its optimization window. Additionally, the superiority of 2D versus 1D processing is seen by comparing Fig. 6.1e (reconstructed by T1D one vector at a time) and Fig. 6.1d, as well as a profile comparison (from image center) in Fig. 6.1f.

The computational time for processing a 299 x 299 image for T2D is approximately 15 seconds; for MxMEM the time is approximately 41 seconds.

6.4.2 Simulated Raman Spectra

MxMEM and T1D were compared against simulations of synthetic and real Raman data. The $X$ multiplier, Eq. 6.12, was found to be useful with MxMEM when applied to Raman data. T1D directly reconstructs individual spectra and does not need to use an $X$
multiplier (essentially it’s fixed at $X = 1$). Its stopping criterion is based on the $\chi^2$ of the reconstructed spectrum. However, MxMEM, when applied to Raman “images”, indirectly reconstructs spectra. Its stopping criterion is based on the image $\chi^2$. Therefore, as shown below, sometimes the best spectral results are obtained when the image multiplier, $X$, is slightly greater than or less than one.

For Raman synthetic data, four metrics were used: the three mentioned above plus one that we call “peak-height-matching error” (PHME). In Raman signal reconstruction, smoothness is important, but so is accurate estimation of peak height, or intensity. Since MxMEM treats the Raman spectra collectively as an image, it is possible that it may smooth the overall image well but severely distort the peak intensities. Hence, we use PHME to evaluate MxMEM’s ability to reconstruct this important aspect of the signal. This is done by computing the RMSE for a region that brackets each of the seven peaks. PHME is the sum of those seven individual peak RMSE values (and a lower value of PHME means better reconstruction of peaks).

Figure 6.2b (a detail view) presents typical results comparing MxMEM and T1D for SNR 5 data. It shows the excellent smoothing of MxMEM in signal areas, and the algorithm’s clearly superior peak-height matching ability. As shown, MxMEM’s reconstructions are more accurate in the baseline, but slightly less smooth than T1D which typically has wider extremes in its baseline reconstruction. Table 6.2 reports the statistical comparisons for SNRs 3 and 5 showing MxMEM’s superior performance in all metrics versus T1D. Similar results were obtained for SNRs 10 and 20 (data not shown), except that T1D’s reconstructed SNR values were (progressively) slightly better than MxMEM’s. All $t$-test $p$-values were $p < 0.001$. 
Although both the 1D and 2D methods use $\chi^2$ as a stopping criterion, different results were obtained for the reconstructions of individual spectra. The $\chi^2$ value is essentially a sum of deviations and all points may not contribute equally to the sum. Thus some parts of a spectrum contribute more to the sum and this is compensated for by other parts of a spectrum contributing less. It is known [56] that T1D causes more smoothing of lower-lying points in a spectrum (i.e., baseline if there is no background present) than higher-lying points (i.e., tall peaks in the absence of a prominent background); thus, by virtue of how the sum is derived at, there is room for solutions of different quality to coexist given the constraint of a single threshold. Likewise, the entropy component of the objective function is based on a sum and this optimized sum, even when identical for different algorithms processing the same data, can consist of different addends as generated by these different algorithms. This consideration is furthermore complicated by the fact that in the 1D case, the thresholds (or entropy calculations) are applied to individual spectra, while in the 2D case, they are ensemble-wide. Therefore, in the latter case, there is more room for compensation and the entropy maximization process is not identical for the 1D and 2D cases.
Table 6.2  Comparative results for the two algorithms on Raman synthetic spectra showing MxMEM to be superior to TPMEM (T1D). The data were composed of four different initial SNR values (3, 5, 10, 20)—formed as Raman “images”—where each image contained 10 spectra (data for SNRs 10, 20 not shown). Ten unique-noise images were used at each SNR value. Thus the mean ± standard deviation of 100 reconstructed spectra are shown for the given metrics. Note that for peak-height-matching error, RMSE, and PEARSC a lower number represents a better reconstruction. The sets of 10 spectra were processed by MxMEM as an image (with the image multiplier, X, as shown), but were processed as individual spectra by T1D. The t-test values of probability p are also given.

<table>
<thead>
<tr>
<th>Figure of Merit</th>
<th>SNR 3</th>
<th>SNR 5</th>
<th>SNR 3</th>
<th>SNR 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MxMEM (X = 1.1)</td>
<td>T1D</td>
<td>MxMEM (X = 1.0)</td>
<td>T1D</td>
</tr>
<tr>
<td>PHME</td>
<td>4.96 ± 2.26</td>
<td>6.78 ± 1.58</td>
<td>2.86 ± 0.95</td>
<td>4.96 ± 1.24</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.399 ± 0.0779</td>
<td>0.703 ± 0.0605</td>
<td>0.274 ± 0.0480</td>
<td>0.462 ± 0.0480</td>
</tr>
<tr>
<td>PEARSC</td>
<td>0.0170 ± 0.0052</td>
<td>0.0532 ± 0.0102</td>
<td>0.0083 ± 0.0025</td>
<td>0.0232 ± 0.0050</td>
</tr>
<tr>
<td>SNR</td>
<td>35.3 ± 7.6</td>
<td>18.6 ± 6.6</td>
<td>42.0 ± 9.6</td>
<td>32.1 ± 9.7</td>
</tr>
</tbody>
</table>

Finally, responses to peak shifting were explored as shown in Figs. 6.4a through 6.4c, for later comparison to measured data (Figs. 6.3 and 6.4d through 6.4f). The same data of Figure 6.2b were altered such that peaks 1 and 2 were shifted right by one channel across all ten spectra and peak 3 was shifted left by one channel across all ten spectra. Fig. 6.4a is a detail of the ten ideal spectra (no noise), all overlaid on the same plot. Only peak 1, whose intensity changes at a sigmoidal rate, is shown. MxMEM and T1D reconstructions of the ten spectra (peak 1 only) are given in Figs. 6.4b and 6.4c, respectively. Although both algorithms distort peak heights, MxMEM tended to preserve relative peak heights and relative peak shifting in all three peaks (only peak 1 is shown). T1D tended to distort relative peak heights and convey some uncertainty about peak shifting. For example, compared to the ideal peak 1, the mean absolute difference (mean ± standard deviation of the mean) in
reconstructed peak positions were 0.6 ± 0.22 and 0.8 ± 0.20 for MxMEM and T1D, respectively. The computational time for processing a 1000 x 10 image for MxMEM is approximately 4 seconds; T1D processes the same ten spectra (individually) in approximately one second.
Figure 6.3 (a) Twelve measured DNA spectra overlaid on the same plot; spectra were aligned using the benzamide peak at 1005 cm$^{-1}$; spectra 1-12 were taken at temperatures 15 °C-70 °C, respectively. (b) Image of (a) reconstructed by MxMEM. (c) Image of (a) reconstructed by TPMEM (T1D). After peak alignment, MxMEM’s reconstruction shows better matched sections of the baseline bracketing the peak at 1609 cm$^{-1}$ than does T1D’s.
Figure 6.4 (a-c) Synthetic data of Figure 6.2b were altered such that peak 1 (near channel 340) was shifted right by one channel across all 10 spectra. (a) Detail of the 10 ideal spectra (no noise), all overlaid on the same plot, peak 1 only; highest intensity value is in spectrum 1, lowest in spectrum 10, changing at a sigmoidal rate. (b) Detail of MxMEM reconstruction of the 10 spectra (peak 1 only). (c) Detail of T1D reconstruction of the 10 spectra (peak 1 only). (d-f) Real DNA data of Figure 6.3 focusing on peak near 1609 cm\(^{-1}\) with all 12 spectra overlaid on each plot (spectra 1-12 correspond to temperatures 15 °C-70 °C, respectively). (d) Detail of noisy data (original SNR of 45 approx.). (e) Detail of MxMEM reconstruction of the 12 spectra, using \(X = 1.2\) multiplier value. (f) Detail of T1D reconstruction of the 12 spectra. Comparing the synthetic data of (a-c) to
real data of (d-f), MxMEM shows better smoothing and resolution of individual spectra than T1D. MxMEM also shows qualitative trends more clearly than T1D, such as the apparent peak shift in real DNA data and the sigmoidal rate of intensity changes across temperature.

6.4.3 Measured Raman Spectra

The two algorithms were also tested on a set of 12 UVRR DNA spectra that were obtained over 12 temperatures, as mentioned above, and then were aligned to the position and intensity of the benzamide peak at 1005 cm\(^{-1}\). Alignment caused the baselines on either side of the group of three overlapping peaks centered around 1610 cm\(^{-1}\) to be relatively well-matched; therefore, changes in peak heights as a function of temperature were unlikely to have been the result of changes in background as a function of temperature. Figure 6.3a shows the 12 measured (and aligned) DNA spectra overlaid on the same plot. MxMEM and T1D’s reconstructions of the same data, also overlaid on one plot, are given in Figs. 6.3b and 6.3c, respectively. When comparing Fig. 6.3b to Fig. 6.3c, it is clear that the MxMEM-processed spectra possessed greater between-spectra consistency.

Figures 6.4d through 6.4f present the DNA data of Fig. 6.3, but focusing on the peak near 1609 cm\(^{-1}\), with all 12 spectra overlaid on each plot. Fig. 6.4d is a detail of the measured data with original SNR of 45, approximately. Reconstructions by MxMEM and T1D are given in Figures 6.4e and 6.4f, respectively. By comparing the synthetic data (Figs. 6.4a through 6.4c) to the real data (Figures 6.4d through 6.4f), we see that MxMEM shows better smoothing and resolution of individual spectra than T1D. MxMEM also reveals qualitative trends more clearly than T1D, such as the apparent peak shift in real DNA data and the sigmoidal rate of intensity changes across temperature. Interestingly, MxMEM’s reconstruction caused a declining intensity-ordering of 1609 cm\(^{-1}\) peaks that matched exactly their increasing-temperature sequence, but differed from the sequence of peak intensities in
the raw spectra. Indeed, MxMEM has a tendency to make adjacent spectra more equal and may mask more subtle between-spectra variations in real peak heights. Therefore, just as smoothing a 1D spectrum may cause the loss of subtle within-spectrum features, smoothing in the orthogonal direction could do the same to between-spectrum features.

6.4.4 Difference Spectra

For Raman 1D spectra, the underlying signal normally changes gradually from point to point. However, with Raman spectra organized as a 2D image, there is the potential that adjacent peaks (or features), in the orthogonal direction, may have non-gradual intensity changes, whether slight or dramatic in extent. Difference spectra are particularly difficult to smooth because peaks are subtracted but the noise is additive, and there may be derivative-like features also if peak shifting occurs in successive spectra. Furthermore, adjacent features in the orthogonal spectra may be radically different. From simulations of the 11 difference spectra (data not shown), T1D in general provided very good smoothing on all of the spectra, but MxMEM performed very poorly overall. This limitation in MxMEM’s ability was predicted in the earlier discussion on optimization (and in the discussion of measured Raman spectra). The peaks in difference spectra have reduced correlations, even though they come from a set of spectra with correlated peaks. That is, the peaks in adjacent spectra have weakened relationships to one another, reducing the benefits of MxMEM reconstruction. Consequently, the adjacent vector optimizing will create distorted peak estimates, but will still generate good baseline estimates. Therefore, a trade-off exists between the distortion of peak estimates and the reduction in overall noise in peaks and baselines and how they contribute to SNR enhancement. Whether MxMEM is useful in
reconstructing this kind of data set will depend on the effects of this trade-off on SNR enhancement.

6.5 Hardware Acceleration of MxMEM

As a multi-disciplinary team, some of our research interests involve investigating ways to speed up algorithms by using microelectronics and parallel processing. A secondary theme of this Thesis was the exploration of the potential for hardware-based acceleration of MxMEM. A summary of this investigation is provided in Appendix 3.

6.6 Conclusions

In this work we have introduced a new spectral image processing algorithm, Matrix-MEM, for SNR enhancement of data in multiple application areas. It is regularization based, performs true matrix processing, and is relatively fast in processing. We demonstrated the performance advantage of MxMEM over 2D TPMEM using computed tomography data. Furthermore, we found excellent results when processing 2D spectroscopic data (both synthetic and real) in a 2D manner compared to the traditional 1D technique of TPMEM. As in the 1D case, 2D smoothing entails the risk that subtle spectral features could be lost. Overall, however, MxMEM should be a useful tool in a wide range of application areas for researchers who need to perform high quality SNR enhancement on their data.
Chapter 7: Application of 2D Methods to Two-Dimensional Correlation Spectroscopy

This chapter reports on the application of two-dimensional (2D) methods for noise reduction in Two-Dimensional Correlation Spectroscopy (2DCoS). MxMEM (from Chapter 6) and 2D wavelets are tested and compared against each other and 1D wavelets. The algorithms are applied in both a “pre-treatment” (before map generation) and “post-treatment” (after map generation) manner.

7.1 Introduction

Two-dimensional correlation spectroscopy (2DCoS) is a spectral analysis technique that Noda proposed [115] in 1986 and then put into a general form [116] in 1993. It has gained widespread spectroscopic use as a “universal spectroscopic tool” [117]. In 2DCoS, the spectral peaks are spread over a second dimension in order to simplify the visualization of complex spectra that have overlapping peaks [18]. That is, spectral resolution is increased because overlapped peaks are spread onto one plane, instead of one line (of a raw spectrum) [118]. The method starts with a set of ordinary, one-dimensional (1D) spectra which are measured under the influence of an external perturbation. From these spectra, synchronous and asynchronous two-dimensional (2D) spectral intensity maps are generated. Within this 2D domain, spectral intensity is now a function of two independent spectral variables. This allows dynamic features correlated in the perturbation domain to be more effectively investigated (e.g., intensity changes of one peak compared to another, band shifts, and

* A version of this chapter has been accepted for publication (with minor revision).

changes in band shape). An important characteristic of the method is the fact that the synchronous and asynchronous correlation maps are orthogonal components which individually carry distinct and useful information [18] not readily discerned in the original 1D spectra [117].

However, despite the many benefits of 2DCoS, noise is a significant problem because it distorts and obscures real features while generating artifactual ones. Some form of noise reduction is necessary to improve the utility of the maps. In this regard 2DCoS affords two options for noise reduction—it can be applied to the data before performing 2DCoS or afterwards to the resulting maps. The latter option obviously is not available without performing 2DCoS. Consequently, the question arises as to whether specific advantages could be had by favoring one or the other approach and whether pre- or post-processing would reveal differential performances among noise reduction techniques. Thus, although pre-treatment is the more common approach [18] we were interested in examining both. Indeed, very little work has been done in post-treatment noise reduction. Jung [119] cites four examples [117][118][120][43] of attempts to filter out the noise from 2D correlation maps, but only for asynchronous maps, and concludes that none of these were truly satisfactory. De Weijer et al. used 2DCoS covariance maps (more below) as input to an artificial neural network (ANN) with the aim of providing improved peak detection of highly overlapped peaks [121]. However, this was not really a form of post-treatment (since noise reduction was not required), but rather simply a use of map data to guide the ANN.

Information is generally lost in noise reduction processes. Therefore, it is preferred to build 2DCoS maps from spectra that do not require signal enhancement. When this is not possible, due to excessively noisy spectra, post-treatment is conceptually the preferred
method since information is theoretically only lost or altered once (assuming purely white noise). With pre-treatment, there are two stages of altering the information—the initial noise reduction of the raw spectra and then the conversion of these 1D spectra to 2D maps that will further modify/amplify the errors due to noise reduction and thus cause additional distortion to the true information. It is therefore unclear, and of interest, to know whether 2D methods and post-treatment have inherent noise reduction advantages for 2DCoS.

Amongst the most common noise reduction techniques are Savitsky-Golay (SG) smoothing filters [34][122]. Other techniques include Fourier filters [43][123], wavelets [124][125][126], and various forms of Principal Components Analysis (PCA) [119][127][128]. Except for PCA, all these methods apply noise reduction in a 1D manner. That is, one spectrum at a time is processed. Wavelets have been shown to outperform Savitsky-Golay (SG) and Fourier transform filters in terms of signal enhancement [129]. From a systematic and comparative study of noise reduction effects on 2DCoS performance, it is known that wavelets are superior to both 1D median filter smoothing and PCA based on the Carbo similarity metric and visual comparisons of maps [124]. PCA is closely related to, and sometimes known as, eigenvector reconstruction [124]. Although PCA does use multiple vectors in the noise reduction processing, it utilizes embedded information differently from the 2D methods that we present. Furthermore, there are a number of 2D methods [130][131] that can perform noise reduction, but we are not aware of any being used within 2DCoS as presented here.

In this work, we apply a relatively new 2D smoothing algorithm, the Matrix Maximum Entropy Method (MxMEM) [132], for pre-treatment and post-treatment of the 2DCoS maps. MxMEM is a two-dimensional regularization algorithm that performs well
We used 1D wavelets (W1D), applied to individual spectra, for comparison. First, we apply all three algorithms to regular signal enhancement of synthetic Raman spectra, over a range of signal-to-noise-ratio (SNR) values. This step is equivalent to the pre-treatment step, but compares the noise-reduction results on the spectra (whether processed individually or as an image) before creating any 2DCoS maps. The results show that both MxMEM and W2D outperform W1D and that MxMEM and W2D are very similar (e.g., in terms of root mean squared error and signal-to-noise ratio). Second, 2DCoS maps are evaluated from both pre- and post- treatment methods by all three algorithms using synthetic and real Raman data. The results show that the 2D methods generally produce superior (easier-to-interpret) maps. Also, although pre-treatment usually produced better maps, in some cases, post-treatment yields the best. Overall, we demonstrate that 2D has several advantages over 1D for smoothing spectra and for noise reduction in 2DCoS maps.

7.2 Theory

7.2.1 Two-dimensional Correlation Spectroscopy

When collecting spectra successively during the physical perturbation of an analyte, such as raising its temperature, pH, etc., some of the peaks in those spectra may be affected and change in intensity or position in subtle but meaningful ways. In short, 2DCoS reveals whether and how these changes are correlated using the covariance and correlation coefficient metrics [133].
With Raman spectra as an example, the spectral intensity variation across the set of spectra (e.g., for a peak) is therefore a function of the wavenumber \( \nu_1 \) and the perturbation \( t \):

\[
y = y(\nu_1, t)
\]  

(7.1)

These intensity changes can be considered as intensity profiles (composed of the vectors orthogonal to the spectra) represented as:

\[
y_1 = y(\nu_1) = \begin{bmatrix} y(\nu_1, t_1) \\ y(\nu_1, t_2) \\ \vdots \\ y(\nu_1, t_j) \end{bmatrix}
\]  

(7.2)

The 2D correlation spectra and 2D correlation maps (which we present below) are, generally speaking, simply the covariances and correlation coefficients derived from these profiles [133]. Thus, assuming that the sampling increment within \( t \) is constant, a synchronous 2D correlation spectrum is now created by calculating the covariance of every profile with every other profile in the set of spectra. Hence, any point within the synchronous correlation map, \( \Phi_{\text{cov}}(\nu_1, \nu_2) \), derives from the covariance of the two profiles situated at \( \nu_1 \) and \( \nu_2 \), respectively:

\[
\Phi_{\text{cov}}(\nu_1, \nu_2) = \text{cov}(y_1, y_2)
\]  

(7.3a)

where

\[
\text{cov}(y_1, y_2) = \sum_{i=1}^{j} \frac{[y(\nu_1, t_i) - \bar{y}_1] [y(\nu_2, t_i) - \bar{y}_2]}{j-1}
\]  

(7.3b)

and \( \bar{y}_1 \) and \( \bar{y}_2 \) are the respective means of \( y_1 \) and \( y_2 \). The synchronous 2D correlation coefficient map is constructed analogously, but using the correlation coefficient:

\[
\Phi_{\text{cc}}(\nu_1, \nu_2) = \text{cor}(y_1, y_2)
\]  

(7.4a)
where

\[ \text{cor}(\mathbf{y}_1, \mathbf{y}_2) = \frac{\text{cov}(\mathbf{y}_1, \mathbf{y}_2)}{\sigma_1\sigma_2} \]  

(7.4b)

and \( \sigma_1 \) and \( \sigma_2 \) are the respective standard deviations of \( \mathbf{y}_1 \) and \( \mathbf{y}_2 \).

Because spectral changes may be related, but lagged in time, an asynchronous 2D correlation spectrum can be used to show these relationships. The asynchronous correlation map is generated as before, but requires the use of the Hilbert-Noda transformation matrix \( \mathbf{N} \), which is given by [18]

\[ N_{ij} = \{0, \text{if } j = i\} \text{ or } N_{ij} = \{1/\pi(j-i), \text{ otherwise}\} \]  

(7.5a)

or:

\[
\mathbf{N} = \frac{1}{\pi} \begin{bmatrix}
0 & 1 & \frac{1}{2} & \frac{1}{3} & \ldots \\
-1 & 0 & 1 & \frac{1}{2} & \ldots \\
\frac{1}{2} & -1 & 0 & 1 & \ldots \\
\frac{1}{3} & \frac{1}{2} & -1 & 0 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{bmatrix}
\]  

(7.5b)

However, in this work we use \( \overline{\mathbf{N}} \), the mean-centered Hilbert-Noda transformation matrix, which is obtained from \( \mathbf{N} \) by subtracting the mean of each row \( n \) of \( \mathbf{N} \) as follows:

\[ \overline{N}_{ij} = N_{ij} - \frac{1}{n} \sum_{j=1}^{n} \left( \frac{1}{\pi(j-i)} \right) \]  

(7.5c)

\( \overline{\mathbf{N}} \) is more flexible than \( \mathbf{N} \) because it does not require that \( \mathbf{y} \) be formulated as a "dynamic spectrum" [18], i.e., \( \tilde{\mathbf{y}} = \mathbf{y} - \bar{\mathbf{y}} \), as does \( \mathbf{N} \). The asynchronous correlation map is then given by:

\[ \Psi_{\text{cov}}(\nu_1, \nu_2) = \text{cov}(\mathbf{y}_1, \mathbf{z}_2) \]  

(7.5d)

where \( \mathbf{z} \), the orthogonal perturbation profile, is derived through the following simple linear transformation:

\[ \mathbf{z} = \overline{\mathbf{N}}\mathbf{y} \]  

(7.5e)
Likewise, the asynchronous 2D correlation coefficient map is given by:

\[ \Psi_{cc}(v_1, v_2) = cor(y_1, z_2) \] (7.5f)

### 7.2.2 Noise Reduction Algorithms

#### 7.2.2.1 Matrix Maximum Entropy Method (MxMEM)

MxMEM is a regularization method that reconstructs an estimate of the noise-free data matrix subject to the minimization of a function consisting of a fidelity component and a smoothness component and has been described in detail elsewhere [132]. Briefly, the method seeks to minimize the general objective function

\[ F = -S + \lambda \chi^2 \] (7.6)

where the fidelity component is a goodness-of-fit component based on the chi-squared distribution function [36][57], \( \chi^2 \), while the smoothness component is based on the negative information theoretic entropy [51], \(-S\), with \( \lambda \) being a LaGrange multiplier. Both the fidelity and smoothness components are adapted for application to data in matrix form. Regularization is terminated when the reconstruction, initialized with the noisy measured data, is acceptably smoothed as determined by a chi-squared based stopping criterion. Generally, when the \( \chi^2 \)-statistic is greater than \( N \), the number of data points in the set, the signal is deemed too smooth and the distortion due to noise reduction, compared to the original, is unacceptable. Nevertheless, strict adherence to the statistical limits may not be desirable and some leeway can be introduced to these limits by employing an extra multiplier \( X \) that allows “tuning” of the result for more smoothness or more fidelity (where \( X > 1 \))
increases smoothness, $X < 1$ increases fidelity). Thus, for an $M^*N$ image, regularization is terminated when

$$\chi^2 \approx X * M * N \tag{7.7}$$

### 7.2.2.2 Discrete Wavelet Transform (DWT)

Wavelet Transform theory was extensively developed in the 1980s and became popular in many fields of science and engineering after the publication of two important papers [40]: Daubechies’ work (1988) on compactly supported orthonormal wavelets [134], and Mallat’s work (1989) on a fast calculation algorithm [135]. The discrete wavelet transform (DWT), owing to the above work, is an efficient and fast implementation of the generalized wavelet transform [136] and can be applied to both 1D and 2D denoising [137].

Wavelet analysis, like Fourier analysis, decomposes signals into basis functions described by a set of coefficients [138]. For wavelets, separate sets of ‘approximation’ and ‘detail’ coefficients are generated. The decomposition process can be done at multiple levels (something like a binary tree), where the first level of approximation coefficients (“cA1”) are further decomposed into a second level of approximation and detail coefficients (“cA2”) and (“cD2”), and so on. Unlike Fourier decomposition with its limited basis functions of indefinitely repeating sines and cosines, wavelet basis functions have localized features that are better suited for the modeling of signals. The method also decomposes signals into time and scale components rather than frequency components [39]. Wavelets furthermore permit the use of thresholding in the denoising process to remove selected coefficients. A drawback of wavelet-based denoising is that an appropriate wavelet has to be selected [124].

Following Berry and Ozaki [124], we chose the coiflet-2 wavelet for 1D and applied it to
W2D as well (except in one case as noted in the Discussion). W2D denoising is easily implemented using Matlab’s Wavelets Toolbox [137], requiring only two lines of code:

\[
[\text{thr, sorh, keepapp}] = \text{ddencmp('den','wv', noisy_data)};
\]

\[
\text{denoised_data} = \text{wdencmp('gbl', noisy_data, wavelet, decomp_level, thr, sorh, keepapp)};
\]

Our W1D implementation, from the same Toolbox, is as follows:

\[
\text{scal} = 'mln';
\]

\[
\text{denoised_data}_\text{1d} = \text{wden(noisy_data}_\text{1d}, 'heursure', 's', scal, decomp_level, wavelet);
\]

### 7.3 Methods

The three algorithms were tested and compared by applying them to the following cases: 1) 1D noise reduction of synthetic Raman data, 2) pre- and post-treatment of 2DCoS maps derived from synthetic data, and 3) pre- and post-treatment of 2DCoS maps derived from real Raman DNA data.

Raman synthetic data were created, to represent a complex chemical system, as follows. First, a “base” spectrum of seven Lorentzian peaks, across 1000 points, was generated (inset, Fig. 7.1a). Peaks were spaced at progressively reduced intervals so as to create uncongested and congested spectral regions. The first peak has a full-width-half-maximum (FWHM) of approximately 10 channels. The remainders have FWHM of 6 channels each, resulting in an overall average of FWHM of 7 channels. Second, ten copies of the base spectrum were combined into a matrix to form a 1000x10 “image” and modified such that the peak heights varied smoothly across the set of ten spectra (Fig. 7.1a), and peaks 1-3 shifted position, as described in Table 7.1. This collection of spectra simulates 10 consecutive measurements of peak intensity and position changes that are spaced over the
course of a perturbation, such as an applied increase in temperature. Third, each of the 10 spectra was convolved with a Gaussian-shaped instrumental point spreading function (IPSF) so as to simulate instrumental broadening. Fourth, each spectrum was added to a sigmoid (cumulative Gaussian distribution) baseline. Fifth, scaled Gaussian (white) noise, with an original standard deviation of 1.0, was added to each image (Fig. 7.1b). Finally, 10 images, each with unique noise distributions, were generated with SNRs of 3, 5, 10, and 20 resulting in 40 “Raman images” total. SNR here was defined by taking the signal as the largest peak within an image (set of 10 spectra). From these sets, 2DCoS maps were generated (Fig 7.1c).

Table 7.1 Description of intensity and frequency changes of peaks shown in Figure 7.1a.

<table>
<thead>
<tr>
<th>Peak</th>
<th>Intensity: rate change</th>
<th>Intensity: direction change</th>
<th>Intensity: change (%)</th>
<th>Position: direction change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sigmoidal</td>
<td>down</td>
<td>~50</td>
<td>right, 10 channels</td>
</tr>
<tr>
<td>2</td>
<td>Exponential</td>
<td>down</td>
<td>~40</td>
<td>right, 10 channels</td>
</tr>
<tr>
<td>3</td>
<td>Linear</td>
<td>down</td>
<td>~40</td>
<td>left, 10 channels</td>
</tr>
<tr>
<td>4</td>
<td>Linear</td>
<td>down</td>
<td>~15</td>
<td>No change</td>
</tr>
<tr>
<td>5</td>
<td>No change</td>
<td></td>
<td></td>
<td>No change</td>
</tr>
<tr>
<td>6</td>
<td>Linear</td>
<td>down</td>
<td>~10</td>
<td>No change</td>
</tr>
<tr>
<td>7</td>
<td>Linear</td>
<td>up</td>
<td>~10</td>
<td>No change</td>
</tr>
</tbody>
</table>
Figure 7.1 (a) Synthetic Raman data set consisting of 10 spectra using the spectrum shown in the inset as basis. The basic spectrum was modified to have peaks vary, at different rates, across the 10 spectra as summarized in Table 7.1. (Inset) The seven Lorentzian peaks (channels 200-800) of the 1000-channel basic synthetic Raman spectrum. (b) A Raman spectrum (original SNR = 5), from a typical set of 10, after 2D processing by MxMEM (MM) and W2D (as part of a set) and by W1D (as an individual spectrum). The 2D methods produced better noise reduction as can be seen in the figure and reported in the text. The noise-free and noisy original spectra are shown for comparison. (c) The synchronous correlation map (noise-free), derived from inner 400 channels of data in (a). (d) Detail of (c) focusing on the 9 large peaks (derived from spectral peaks 1-3) which are elongated due to peak shifts and tapered due to intensity changes.
The real Raman data consisted of ultraviolet resonance Raman (UVRR) spectra of single-stranded DNA homo-oligomers measured across 12 temperatures (15 °C-70 °C). Standard desalted deoxyribonucleic acid (DNA) polyadenine pentamers were obtained from IDT Technologies (Coralville, IA). Pentamers were centrifuged prior to use and then dissolved to a sample concentration of 60 µM in a buffer solution consisting of 10 mM phosphate buffer (pH 7.0), 1 M NaCl and 1 mM EDTA. Concentrations were quantified by UV-Vis spectroscopy using the 260 nm absorption band of the bases. The 1005 cm\(^{-1}\) ring trigonal bend mode (\(\nu_{23}\)) [114] of residual benzamide from the synthesis process was used to align the spectra. UVRR spectra were generated with 244 nm excitation light from a frequency-doubled Ar\(^+\) ion laser (Innova 90C FreD, Coherent Inc., Santa Clara, CA) and collected with a custom fiber-optic probe [13][11][53]. Power at the samples was approximately 20 mW. Samples were loaded into a 500 µL centrifuge tube, placed in a rotating holder and spun (at approximately 300 RPM) during measurement to maximize sample exchange within the measurement volume and thereby minimize photodegradation. Temperature (the perturbation variable) was controlled using a DNA Thermal Cycler 480 (Perkin Elmer, Norwalk, CT, USA). A dielectric interference filter (Barr Associates Inc., Westford, MA) was used to remove Rayleigh scattering. Raman scattered light was dispersed with a 1.0 m focal length monochromator (Model 2061, McPherson Inc., Chelmsford, MA) using a 3600 groove/mm holographic grating (Model 8358-1004-0, McPherson Inc., Chelmsford, MA). A liquid nitrogen cooled CCD detector operating at -120 °C (Spec-10 400B, Roper Scientific, Trenton, NJ) was used for 10 s duration acquisitions from each sample.
MxMEM was implemented primarily in the C programming language, but interfaced to Matlab R2009a (The MathWorks, Natick, MA), and utilized C routines from *Numerical Recipes in C* (NRC) [36]. W2D and W1D were implemented entirely in Matlab. The computational platform was a DELL Inspiron 1545 personal computer with a 2.0 GHz Intel Pentium Dual-Core T4200 CPU, 3 GBytes of RAM, running under Windows XP (with Service Pack 3). The post-treatment computational time to process a 400x400 map for W2D was approximately 5 seconds; for MxMEM the time was approximately 52 seconds.

Noise reduction in spectra was assessed using four metrics: the root mean squared error (RMSE); a complementary Pearson’s correlation coefficient (PEARSC); SNR; and a RMSE restricted by bracketing peaks only, i.e., a “peak-height matching error” (PHME). In Raman signal reconstruction, smoothness is important, but so is accurate estimation of peak height, or intensity. Since the 2D methods treat the Raman spectra collectively as an image, it is possible that they may smooth the overall image well, but severely distort the peak intensities. Furthermore, peak shapes in 2DCoS maps are affected by synchronous and asynchronous changes in peak position and therefore may contain important information [117][133][139]. Hence, we use PHME to evaluate an algorithm’s ability to reconstruct this important aspect of the signal. Noise reduction in maps used the RMSE, PHME, and the Carbo similarity metric [124][140]. The Carbo metric is given by:

$$ C_{AB} = \frac{\sum P_A P_B}{\sqrt{(\sum P_A^2)(\sum P_B^2)}} $$

(7.8)

where $P_A$ is a point from the ideal (noise-free) map and $P_B$ is the corresponding point from a non-ideal map. Identical maps would yield a value of 1, and highly correlated maps would yield a value of 0.95 or greater. The Carbo and RMSE metrics were used on all maps, while
the PHME was applied only to the covariance (correlation maps) data and are reported for the 9 (2D) peaks shown in the synchronous correlation map (and in the asynchronous correlation map below).

Results were evaluated with a multiple comparison [141][142][143] (MC) procedure (Tukey; pair-wise $\alpha = 0.5$, d.f. = 18) to reveal the existence of all significant pair-wise differences per method, per metric, and across all SNR cases.

7.4 Results and Discussion

7.4.1 Simulated Raman Spectra

Comparisons were first made on the noise-reduction ability of the algorithms on the spectral data set itself. Figure 7.1b presents a detail view comparing typical results obtained following MxMEM, W2D, and W1D processing for SNR 5 data. It shows the excellent smoothing of the 2D methods in signal areas, and their clearly superior peak-height matching ability vs. W1D. Table 7.2 reports the statistical comparisons for SNRs 3 and 20. Similar results were obtained for SNRs 5 and 10 (data not shown). MxMEM and W2D were superior to W1D in all metrics (except for W2D and W1D being equivalent on PHME at SNR 10). In general, MxMEM and W2D were not significantly different in terms of RMSE and PEARSC, but W2D was superior to the other methods in terms of PHME and MxMEM was superior in terms of SNR. These results suggested that maps generated from data processed with the 2D methods ought to be of better quality than those generated from 1D methods.
Table 7.2 Comparative noise reduction results for the three algorithms on Raman synthetic spectra. The data consisted of sets of 10 simulated Raman spectra each. 10 multiples of these sets, each with unique white noise added to give SNRs of 3, 5, 10, and 20, were created. The spectral sets were processed as individual spectra (W1D) or as a 2D ensemble (MxMEM and W2D; X and DL refer to the MxMEM image multiplier and wavelets decomposition level, respectively). The results show the mean ± standard deviation of (100) processed spectra per metric and original SNR (only SNR 3 and 20 results are shown here). A lower number represents better noise reduction on the PHME, RMSE, and PEARSC metrics. A multiple comparison test (α = 0.5) showed MxMEM and W2D to produce better noise reduction than W1D on all metrics (except one case, see text). Generally, MxMEM and W2D produced equivalent results using the RMSE and PEARSC metrics, but W2D tended to be better on the PHME and MxMEM on the SNR metrics.

Table 7.2a

| Figure of Merit | SNR 3 | | | | | | |
|-----------------|-------|-------|-------|-------|
|                 | MxMEM | W2D   | W1D   |
| PHME            | 5.88 ± 2.26 | 4.72 ± 1.57 | 7.24 ± 1.08 |
| RMSE            | 0.436 ± 0.093 | 0.446 ± 0.070 | 0.835 ± 0.062 |
| PEARSC          | 0.0208 ± 0.0073 | 0.0218 ± 0.0057 | 0.0737 ± 0.0136 |
| SNR             | 42.5 ± 9.0 | 33.6 ± 5.9 | 17.3 ± 4.4 |

Table 7.2b

| Figure of Merit | SNR 20 | | | | | | |
|-----------------|--------|-------|-------|-------|
|                 | MxMEM  | W2D   | W1D   |
| PHME            | 1.52 ± 0.58 | 1.17 ± 0.40 | 2.11 ± 0.30 |
| RMSE            | 0.122 ± 0.037 | 0.129 ± 0.026 | 0.194 ± 0.010 |
| PEARSC          | 0.0017 ± 0.0009 | 0.0019 ± 0.0007 | 0.0042 ± 0.0004 |
| SNR             | 137.1 ± 27.6 | 106.8 ± 21.3 | 75.7 ± 15.5 |

7.4.2 2DCoS Maps (from synthetic data)

Figure 7.1c shows the noise-free synchronous correlation map derived from the data set of Fig. 7.1a, with detail map view in 7.1d focusing on the 9 large peaks. As the ideal map in Figure 7.1c shows most clearly, three large correlation peaks appeared along the diagonal. Covariances emphasize large absolute changes in peaks at the expense of small absolute changes even if the latter are proportionally equal to or better than the former. Thus, spectral
peaks 1-3 were prominent and peaks 4, 6, and 7 were weak (see Table 7.1). The three pairs of off-diagonal peaks, all being positive, indicated that the spectral peaks were changing intensity in the same direction. Furthermore, elongated shapes of the peaks stem from the position changes of spectral peaks 1-3, and they are tapered due to corresponding peak intensity changes.

Figure 7.2 shows the synchronous correlation maps generated from the same data set but with noise added to an initial SNR of 20. All seven peaks of Figure 7.1a were used as input data, but the 1000-point spectra were reduced to the inner 400 channels for clarity in presentation. In Figure 7.2, the maps are (a) W1D pre-treatment, (b) W1D post-treatment, (c) MxMEM pre-treatment, (d) MxMEM post-treatment, (e) W2D pre-treatment, and (f) W2D post-treatment. The 9 large covarying peaks were clearly visible in the maps of all 6 panels in Fig. 7.2. The top and side panels contain the first and last spectrum of the set, exactly as used for generating the maps, respectively.
Figure 7.2 Detail of six typical synchronous 2D correlation spectroscopy maps (effectively covariance maps) generated from the set of 10 synthetic Raman spectra of Fig. 7.1(a) after noise addition to give an SNR of 20. (a) W1D (pre-treatment) map. (b) W1D (post-treatment) map. (c) MxMEM (pre-treatment) map. (d) MxMEM (post-treatment) map. (e) W2D (pre-treatment) map. (f) W2D (post-treatment) map. Compared to the other
methods, superior visualization of the data was obtained by using the 2D MxMEM method on the simulated noisy spectra before performing 2DCoS (pre-treatment) as shown in panel (c). This result was consistent with the high Table 7.3a reported Carbo metric of similarity. In general for this type of map, pre-treatments produced better results than post-treatments and 2D methods better results than the 1D method.

Table 7.3a reports the statistical comparisons of the maps for SNR 20 input data (W1D post-treatment was excluded). The results indicated that MxMEM pre-treatment has the best overall map similarity to the ideal (Carbo metric), but the worst PHME and RMSE (this divergence is explained below). Visual comparison of the six maps to the ideal in Fig. 7.1d confirms that the MxMEM pre-treatment map has the best reconstruction of the 9 peaks (e.g., in terms of smoothly rounded (not squared), elongated peaks and the dual cross-peak “wings” of each main peak). The 2D algorithms, and especially MxMEM, tend to make adjacent vectors the same (i.e., peak heights within a profile tend to be “compressed” toward equal values). This is actually the same effect seen in all 1D noise reduction methods—neighboring points within a spectrum tend to converge toward the same value as part of the “smoothing” process. The 2D methods cause this effect to occur in the two (orthogonal) directions.
Table 7.3 Comparative results when applying noise-reduction to the synchronous correlation and asynchronous correlation maps generated from SNR 20 data, using the 3 pre-treatment and 2 post-treatment approaches. Ten unique-noise sets of spectra (see caption Table 7.2) were used to generate these (ten) synchronous and (ten) asynchronous maps. Thus, the means ± standard deviations of 10 maps are shown for the given metrics in each table. A lower number for the RMSE and PHME metrics represents better noise reduction. A multiple comparison test ($\alpha = 0.5$) showed that MxMEM pre-treatment produced a better result on the Carbo metric than W1D and W2D, but that the other two methods were better on the PHME and RMSE metrics for the synchronous correlation map. Importantly, the Table 7.3b data show that W1D produced the best asynchronous correlation maps on all metrics. See text for further details.

Table 7.3a

<table>
<thead>
<tr>
<th>Figure of Merit</th>
<th>Synchronous Correlation Map</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MxMEM (X = 0.9)</td>
</tr>
<tr>
<td>Carbo</td>
<td>0.9880 ± 0.0012</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.1263 ± 0.0134</td>
</tr>
<tr>
<td>PHME</td>
<td>2.2526 ± 0.2728</td>
</tr>
</tbody>
</table>

Table 7.3b

<table>
<thead>
<tr>
<th>Figure of Merit</th>
<th>Asynchronous Correlation Map</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MxMEM (X = 0.9)</td>
</tr>
<tr>
<td>Carbo</td>
<td>0.8652 ± 0.0118</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.0404 ± 0.0021</td>
</tr>
<tr>
<td>PHME</td>
<td>0.3298 ± 0.0241</td>
</tr>
</tbody>
</table>
Figure 7.3 Detail of simulated spectral sets (spectral peaks 1-3 only) showing overlaid (a) noise-free, (c) W1D processed and (e) MxMEM processed sets. Corresponding asynchronous 2D correlation spectroscopy maps are shown for the noise-free case in (b), the W1D pre-treatment case in (d), and the MxMEM pre-treatment case in (f). In contrast to the results shown in Fig. 7.2 and Table 7.3a, the 1D method produced better visual (e.g. panel d) and quantitative (Table 7.3b) results for asynchronous 2D correlation spectroscopy maps.
Figure 7.3 presents detail views of selected asynchronous correlation maps and the source data (spectral peaks 1-3 only). In (a), (c), and (e), spectral plots are given with all 10 spectra overlaid on the same plot, for the ideal, W1D, and MxMEM, respectively. The corresponding asynchronous correlation maps are given in (b), (d), and (f) and reveal 9 sets of asynchronous peaks which correspond to the synchronous peaks of Figure 7.1d. The Table 7.3b results indicate that 1) W1D produced the best asynchronous correlation maps for all metrics, 2) the Carbo values for the other four cases were not statistically different, 3) the two post-treatment methods were not significantly different in terms of any of the metrics, and 4) the MxMEM and W2D values were significantly different from each other as well as from the other methods in terms of RMSE and PHME. As previously described, and now shown in Fig. 7.3c and 7.3e, W1D’s pre-treated spectra have peaks with a wider span (across their profiles), though less accurate in shape. In contrast, although compressed in height, MxMEM-derived maps are more accurate in shape. Both lead to fairly clear map peaks of similar shape, but MxMEM’s map peaks are lower as evidenced by the higher PHME value. It is interesting to note that the spectral valleys in (e) have lower levels of noise but with a higher frequency than those in (c). Higher frequency means more asynchronous events. Since MxMEM’s spectral peaks were compressed, this tended to make the asynchronous events relatively stronger and thus produced more noise “stripes” in the map of (f) compared to W1D’s map in (d), and a higher RMSE value.

Differences in MxMEM’s pre- and post-treatment values of Table 7.3a and 7.3b stem from the order of processing. MxMEM pre-treatment smoothes first (and thereby compresses relative to the noise-free spectra) and then performs the dot product calculation of Eq. 7.3b to create a map. MxMEM post-treatment reverses these steps. In the former
case, profiles compressed relative to the noise-free profiles mean smaller numbers are multiplied, yielding smaller products. This equates to reduced map peak heights of the 9 primary peaks relative to the noise-free case. In addition, the total map surface, along with its other smaller positive and negative features, is compressed. In the former case, the PHME is increased and in the latter case, the RMSE is increased if the compression of real features outweighs the compression of artifacts. In the post-treatment case, the dot product is performed first on noisy spectra with peak heights that tend to cluster around the noise-free spectral peak heights giving products that are similar to those of the noise-free case.

MxMEM then smoothes (and compresses) these map features, resulting in more accurate peak heights (a lower PHME) as well as a less-compressed total map surface (a lower RMSE if the lack of compression of real features outweighs the lack of compression of artifacts), but a noisier map (lower Carbo metric) vs. pre-treatment, as shown in Table 7.3a. Thus, pre-treatment tends to compress (spectral data) away from the noise-free intensities, while retaining high similarity of the overall surface otherwise, whereas post-treatment, although noisier, tends to retain intensities comparable to those of the noise-free case. What is still unclear is the extent to which compression occurs, whether it is proportional to peak height thus preserving relative peaks heights, and whether the compression can be compensated for after the fact if the nature of the compressions can be fully characterized; these are topics beyond the scope of our current investigation.

Figure 7.4 (a)-(f) present synchronous correlation coefficient maps as follows: (a) ideal, (b) W1D pre-treatment, (c) MxMEM pre-treatment, (d) MxMEM post-treatment, (e) W2D pre-treatment, (f) W2D post-treatment. The data were slightly modified (for ease of presentation) such that the only change in Table 7.1 is that peaks 1-3 do not change position.
The correlation coefficient maps are normalized versions of the covariance maps (since profiles are divided by their standard deviations) and thus are sensitive to peak changes, however subtle. As shown in (a), peaks 1-6 have a similar correlation value despite the drastic differences in changes of spectral peaks 1-3 vs. 4-6. Peak 7, the only peak with a positive (though subtle) intensity change (Table 7.1) has the opposite correlation value. In Figure 7.4b, the weakness of the 1D (pre-treatment) methods is evident—the W1D spectra (see Figure 7.3c) have noisy valleys that produce numerous artifacts rendering the map of little use. In contrast, both 2D methods applied as pre-treatments produce smoother results (cf. Fig. 7.3e) and thereby provide synchronous correlation coefficient maps with fairly clear information content; W2D is the best as seen in 7.4(e). An interesting result occurred with the 2D post-treatment methods (d, f)—a map was generated that contained individual auto- and cross-peaks reminiscent of a synchronous correlation map but these occur for the non-subtle and subtle spectral peak changes alike. The diagonal now contains three large auto peaks (corresponding to spectral peaks 1-3), but also clearly reveals correlation peaks derived from spectral peaks 4, 6, and 7 and none for peak 5 which is not changing. Furthermore, the off-diagonal cross-correlation peaks are all present, and of proper polarity, for all 6 active peaks including the anti-correlation of peak 7. This is due to the fact that, in regions where the signal intensity is low relative to the noise, numerous spurious positive and negative cross correlations of comparably scaled variances are produced that are subsequently evened out by the smoothing algorithms. We infer that where the signal is higher relative to the noise, the local probability of positive correlations tend to increase and covariance-like peaks emerge when the map is smoothed. However, these are still scaled by their respective profile variances, thus changes in small peaks are not inherently discriminated against. They appear
weaker because of their relatively low SNR and the greater relative effect of the spurious correlations described above. On the other hand, because white noise also has low frequency noise components, the latter may act like correlated features (i.e., spectral peaks) and, because they are variance-scaled, attain prominence relative to real peaks, especially in spectral valleys. These low frequency-generated artifacts then correlate/anti-correlate to the real peaks.
Figure 7.4 Detail of six typical synchronous 2D correlation coefficient maps generated from the set of 10 synthetic Raman spectra of Fig. 7.1(a) after noise addition (except for panel a) to give an SNR of 20. The noise-free map is shown in (a), the map produced with W1D pre-treatment in (b), the map produced with MxMEM
pre-treatment in (c), the map produced with MxMEM post-treatment in (d), the map produced with W2D pre-treatment in (e), and with W2D post-treatment in (f). For these maps, the data were slightly modified (for ease of presentation) such that the only change in Table 7.1 was that peaks 1-3 did not change position. The 2D methods produced better data visualization than the 1D method. Surprisingly, good data visualizations were obtained when using the 2D methods for noise reduction after performing 2D correlation spectroscopy (d, f).

The asynchronous correlation coefficient maps (not shown) in pre-treatment were all severely affected by noise and were not so useful to guide interpretation, but the post-treatment asynchronous correlation coefficient maps, like the synchronous correlation coefficient ones, had a similar covariance-like result and were useful. In post-treatment, although the correlation coefficient maps are first created from raw, noisy data, the (normalized) peak-change information may be fairly well-preserved. In such cases, as demonstrated, important map features are present (within the noise) and can be effectively denoised/smoothed by the 2D methods.

In Figure 7.5, only data with relatively subtle peak intensity changes are used to create synchronous correlation maps (note the slight spectral differences of the top vs. side panels). Here, only spectral peaks 4-7 are processed, with peaks 4 and 6 diminishing, peak 7 increasing, and peak 5 not changing (Table 7.1). In this case, the two post-treatment methods, (d) and (f), produced superior results. With subtle peak changes, the slight cascading of peaks in a profile may be masked by noise creating random height differences along the profile and thereby hide the subtle change after W1D filtering. Noise may also cause false changes in stationary peaks, like peak 5. Both of these weaknesses are more likely to occur in a 1D method, since adjacent vectors are not involved in the processing and therefore cannot provide a constraint that may preserve relative peak heights as with 2D methods. Hence, Fig. 7.5b, compared to the ideal in (a), does not clearly reveal any of the 3 peaks on the diagonal, nor any cross peaks, and even misleads in that peak 5 is not changing. In contrast, the 2D pre-treatment results tend to preserve relative peak heights (along a
profile) even when undergoing subtle changes. Thus as shown in (c) and (e), despite distortions, it is reasonably clear that there are three peaks on the diagonal, peak 5 has no peak on the diagonal, and by the cross peaks we can discern that peak 7 is moving in an opposite direction to peaks 4 and 6. The post-treatment images in (d) and (f) tend to supplement and clarify the interpretations of (c) and (e) and overall are remarkably clear.

Thus in (b) we see that significant information was lost in pre-treatment, but in (d) and (f) much information was preserved despite creating maps with noisy raw data and then performing post-treatment.
Figure 7.5 Six synchronous correlation maps generated from the mildly-changing peaks 4-7 of the set of 10 synthetic Raman spectra of Fig. 7.1 (original SNR = 20): (a) map from 10 noise-free raw spectra; (b) W1D pre-treatment map; (c) MxMEM pre-treatment map; (d) MxMEM post-treatment map; (e) W2D pre-treatment map;
and (f) W2D post-treatment map. With relatively subtle peak intensity changes (and peak 5 not changing), the two post-treatment methods, (d) and (f) produced remarkably clear data visualizations. As before, the W1D map produced poor data visualization.

As a result of these findings, we believe MxMEM post-treatment may provide a novel and valuable approach to take advantage of the key feature of 2DCoS because the visualization of complex spectra that have overlapping peaks is clearly enhanced by this method. Furthermore, the W2D post-treatment map of Fig. 7.5f was generated using the daubechies-8 wavelet because it gave a superior result as compared to the coiflet-2 wavelet (which is used in all other cases in this study). Although we in general adopted the use of the coiflet-2 wavelet [124] (because of its similarity to vibrational spectral bands), in the case of filtering map features, we now infer that other wavelets may provide a better approximation (though detailed comparison of different wavelets is beyond the scope of this study).

7.4.3 2DCoS Maps (from measured data)

Figure 7.6a displays a set of 12 UVRR DNA spectra that were obtained over 12 temperatures (15 °C-70 °C), as described above, all overlaid on the same plot before baseline removal. The inset is a detail of overlaid peaks in the 1609 cm⁻¹ region after baseline removal. The results (not shown) generated from the spectra shown in the inset by each pre-treatment method exhibited an apparent perturbation-induced peak shift of the 1609 cm⁻¹ peak. These spectra, derived from a recent DNA oligomers study [144], provide a conceptually simple perturbation experiment by way of temperature variation. Interestingly, commercial (single-stranded) DNA oligomers were found to be contaminated with trace amounts of benzamide from the synthesis that was not removed by the standard desalting purification step. UV-vis analysis [144] suggested that the benzamide was hydrogen-bonded to the DNA oligomer. Therefore, over the temperature perturbation defined above, we
expected that ‘melting’ would occur and potentially give rise to small intensity and frequency changes as the benzamide dissociates. In the 2DCoS maps shown in Figure 7.6, we can easily visualize the subtle frequency shift in the strongly resonance enhanced 1609 cm\(^{-1}\) peak (attributed to a ring breathing mode of benzamide) consistent with a down-shift associated with breaking of hydrogen bonds. The intensity shift is consistent with thermally-induced broadening, but may also involve other contributing factors, so the discussion below focuses on the frequency shift.

In Figures 7.6 (b-f), the benefit of revealing subtle changes is clearly demonstrated with real Raman DNA data. Focusing on the 1609 cm\(^{-1}\) peak, all of the synchronous correlation maps and asynchronous correlation maps were informative but only the following selected ones are given in the figure: (b) MxMEM (post-treatment) synchronous correlation coefficient map, (c) W1D (pre-treatment) synchronous correlation map, (d) W1D (pre-treatment) asynchronous correlation map, (e) MxMEM (pre-treatment) synchronous correlation map, (f) MxMEM (pre-treatment) asynchronous correlation map. The maps show the temperature–induced subtle frequency shift in the 1609 cm\(^{-1}\) peak that occurred during the perturbation as an elongated shape near 1609 cm\(^{-1}\) in the synchronous correlation maps and the modified “butterfly pattern” shown in the lower-left region of the asynchronous correlation maps. In the synchronous correlation coefficient map of (b) there may be further confirmation of this peak shift as seen in the elongation of the large correlation peak near 1609 cm\(^{-1}\). This shift is extremely difficult to identify in the raw spectra with any degree of confidence without the use of 2DCoS analysis. Although the shifts are qualitatively discernable in the non-processed maps (data not shown), the noise-reduced maps provide a
much clearer view and, provided the smoothing associated distortions can be corrected for (as alluded to above), may offer the possibility of quantitative 2DCoS analysis.
Figure 7.6  (a) 12 measured DNA spectra (SNR ~45) overlaid on the same plot before baseline removal; spectra 1-12 were taken at temperatures 15 °C-70 °C, respectively. (Inset) Detail of peak near 1609 cm\(^{-1}\) after baseline removal. (b-f) Selected maps generated from real Raman data shown in the inset. (b) MxMEM post-treatment synchronous correlation coefficient map. (c) W1D pre-treatment synchronous correlation map. (d) W1D pre-treatment asynchronous correlation map. (e) MxMEM pre-treatment synchronous correlation map. (f) MxMEM pre-treatment asynchronous correlation map. The large correlation peak near 1609 cm\(^{-1}\), being elongated in varying degrees among all synchronous correlation maps, appears to reflect a subtle perturbation-induced peak shift. This is further confirmed in the asynchronous correlation maps by the modified “butterfly pattern” shown.
in the lower-left region of the maps, especially clear in (f). The synchronous correlation coefficient map of (b) also provides confirmation of this peak shift as seen in the elongation of the large correlation peak near 1609 cm$^{-1}$.

### 7.5 Conclusions

In this work, we compared the noise reduction performance of 1D and 2D methods for generating 2DCoS maps, and this was done using both a pre-treatment and post-treatment approach.

For general noise reduction of related spectra, we demonstrated that 2D methods were superior: both MxMEM and W2D outperformed W1D. For map generation, when peak intensities undergo large changes, we demonstrated that all methods, including pre- and post-treatment, provide information which aids interpretation in varying degrees, and the techniques often complement one another. However, the pre-treatment methods were superior in general, with MxMEM best in the synchronous correlation maps (Carbo metric), but W1D best in asynchronous correlation maps (all metrics). This distinction between MxMEM and W1D was based on how the maps in question were affected by the tendency to compress or widen the peak spans, respectively. In summary, the 2D methods improve upon existing noise reduction of 2DCoS map generation, enable effective use of post-treatment, and altogether help to enhance interpretability.

We also made a novel observation that 2D post-treatments produced covariance-like maps that also included peaks resulting from subtle changes that are normally difficult to visualize in 2D correlation spectra. Furthermore, we discovered that the wavelet producing the best denoising of spectral features may not necessarily produce the best denoising of map features because map features and spectral features may have different attributes due to synchronous and asynchronous peak changes. Both of the latter findings deserve further
investigation. In particular, it would be interesting to see if the 2D post-treatments are of any benefit in dealing with the narrow peaks and peak congestion often present in NMR spectra. Unfortunately, pursuing these findings are beyond the scope of the present study.

Overall, we recommend W2D as the primary denoising algorithm for 2DCoS due to its ease of use, computational speed, excellent 2D denoising, and its versatility as both a pre- and post-treatment algorithm. However, MxMEM provided slightly better noise reduction of related spectra for most metrics and also demonstrated very good (and sometimes superior) results in map generation for both pre- and post-treatment modes.
Chapter 8: Conclusions and Future Work

The goal of this research was to investigate and apply new signal enhancement methods so as to improve the quality of analytical measurements and data analysis for spectroscopic applications, and where possible, for other applications. Specifically, the goal was to provide noise reduction, deconvolution, baseline correction, and automation methods for 1D and 2D applications. All aspects of this goal were achieved except for deconvolution in 2D data. In this chapter, the main findings and contributions of the previous chapters are presented.

8.1 Conclusions

In our re-examination of the Two-Point Maximum Entropy Method (TPMEM), we demonstrated its excellent performance for SNR enhancement and faithful spectral reconstruction of low-SNR data, but for the first time we quantitatively defined the performance boundaries for its application. The circumstances where it performs less effectively at noise discrimination were shown to be when spectra have 1) high background components, 2) tall peaks, or 3) derivative-like features which result from difference spectra. Ways to extend TPMEM’s usefulness in some of these circumstances were given, as follows: baseline subtraction can provide a good alternative for handling high backgrounds; for tall peaks, processing the inverted spectrum can provide discrimination of noise in the peaks. The usefulness of these results actually extends beyond spectroscopy, since TPMEM was recently applied to signal enhancement of magnetoencephalography data [105][106]. Finally, TPMEM also served as the inspiration and basis for developing the 2D SNR enhancement method, MxMEM, presented in Chapter 6.
The digital Bi-chi filter (BCF) design was inspired by considering 1) TPMEM’s limitation in the presence of high backgrounds and 2) SG filters sometimes not providing the best error minimization. By employing the chi-squared function to characterize both smoothness and fidelity, an improved design was created. The BCF is versatile—it can be implemented as a recursive noise filter and as a regularization-with-deconvolution algorithm. Its spectral noise reduction ability was demonstrated to be superior to SG filters, and its deconvolution ability showed excellent resolution enhancement. However, this resolution enhancement was shown only in two qualitative examples—one with synthetic data and one with real Raman data. Our work would have been strengthened if we had applied the deconvolution on synthetic data and used metrics to provide some quantitative results. In addition, comparison to another deconvolution algorithm would much more clearly establish BCF’s performance ability.

There is a pressing need for automated smoothing (noise reduction) techniques, but very few “fully” automated methods exist. The iterative Savitzky-Golay (iSG) filter was developed to address this need. It is based on the very familiar moving window average concept (and thus very intuitive), is fast, and is easy to implement. For spectroscopic data, the iSG filter demonstrated improved performance over the iterative median filter (e.g., a 30% reduction in RMS error and a 200% improvement in SNR), and demonstrated results that were comparable to or better than standard SG filters with optimized parameters.

There is also a pressing need for automated baseline removal/correction schemes. Much more research has been done on these compared to automated smoothing techniques. Nevertheless, prior to this work, a truly model-free, fully-automated baseline removal method remained elusive (with the exception of a second derivative-based method by Rowlands
and Elliott [98] which requires low noise spectra, as previously noted. We presented an auto-
baseline-estimate (ABE) method that is fully-automated and model free. It is based upon a
zero-order SG filter and uses the chi-squared statistic as a stopping criterion (within an
iterative process). The ABE method is capable of baseline correction without user
intervention. Simulated and real Raman spectra were used for evaluation purposes. Results
showed that the ABE method consistently produced baseline-flattened spectra of high
quality—without user intervention. However, this was not the case for all baseline shapes.
When baselines are very prominent with strong curvatures, user intervention may be required
in terms of selecting window size, but this also resulted in some peak distortion. In other
cases, there was incomplete baseline removal. Nevertheless, the overall benefits make this a
useful tool—because a large number of baseline types can be corrected with this method in a
fully-automated manner.

There are many opportunities to utilize 2D spectral smoothing as an advantage, but
little research seems to have explored this concept. Based on our literature survey, and to the
best of our knowledge, we believe that we have introduced a novel approach by treating a set
of related Raman spectra as an artificial “image”. Our hypothesis was that better noise
reduction would result from a 2D method applied to the image vs. a 1D method applied to the
individual spectra (one-at-a-time). The MxMEM spectral image processing algorithm, for
SNR enhancement, was developed to investigate this idea. We demonstrated its superior
performance in two application areas: first, as a general image smoothing algorithm in
computed tomography vs. the 2D TPMEM algorithm, and second, with Raman spectral
“images” (both simulated and real) in comparison to 1D TPMEM. For example, with the CT
“cylinder” image at SNR 5 (Table 6.1), MxMEM had a 10% reduction in RMS error and
25% improvement in final SNR vs. 2D TP MEM. And on Raman spectra at SNR 3 (Table 6.2), MxMEM out-performed TP MEM with an approximate 40% reduction in RMS error and nearly double improvement of final SNR. Finally, because MxMEM is a regularization method, based on TP MEM, it should be conceptually capable of performing deconvolution. Unfortunately, time did not permit this feature to be explored, and it will be left for future work (as mentioned below).

Another important and interesting opportunity for 2D spectral smoothing is in the area of Two-Dimensional Correlation Spectroscopy (2DCoS). MxMEM and 2D wavelets were applied in both a “pre-treatment” and “post-treatment” manner and compared with 1D wavelets. The 2D methods proved to be superior in most cases (e.g., the metrics of Table 7.2 and Table 7.3a, and in the qualitative map results of Figures 7.2, 7.4, 7.5, 7.6). We believe that this was a novel and first time use of 2D wavelets for smoothing in 2DCoS.

Very little work has been done to explore the post-treatment denoising of 2D spectral maps. As was demonstrated, the 2D methods really enabled the effective use of post-treatment. In some cases, post-treatment noise-reduction was superior to pre-treatment noise-reduction (e.g., Figure 7.5). Thus, the two important findings of this work—the superiority of 2D noise reduction over 1D and that in some cases post-treatment is superior to pre-treatment—are significant for improving the performance of 2D correlation spectroscopy. Finally, because 2DCoS is a “universal spectroscopic tool” [117], the notion of using 2D methods—which enable the complementary-supplementary benefits of post-treatment—may be helpful to a wider audience. In this regard, our novel application of 2D methods to 2DCoS may be the contribution, within this thesis, that has the greatest impact.
8.2 Algorithm Selection Guide

Table 8.1 provides a summary of the algorithms and serves as a selection guide for choosing when to use a given method. The five primary algorithms presented in this work appear first (with names in bold), followed by a selected set of three which were used for comparison (the standard Savitzky-Golay filter and two forms of wavelets). Column 2 indicates the data types (one- or two-dimensional). Column 3 shows the available functionality: SNR enhancement, Deconvolution, and Baseline Correction. As column 4 shows, only the iSG filter and ABE methods are automated, but the other three primary algorithms have a high potential for being automated (see Future Work, below, for MxMEM automation).

Table 8.1 Algorithm Summary and Selection Guide

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Data Type</th>
<th>Functions</th>
<th>Automated</th>
<th>SNR-E Quality</th>
<th>Computation Time(^1)</th>
<th>2DCoS Useable</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPMEM (Chap. 2)</td>
<td>1D</td>
<td>SNR-E, Deconv.</td>
<td>(high</td>
<td>Very good</td>
<td>~ 1 second</td>
<td>yes</td>
</tr>
<tr>
<td>BiChi (Chap. 3)</td>
<td>1D</td>
<td>SNR-E, Deconv.</td>
<td>(high</td>
<td>Good</td>
<td>&lt; 1 sec. (+d =5.6 sec.)</td>
<td>yes</td>
</tr>
<tr>
<td>iSG Filter (Chap. 4)</td>
<td>1D</td>
<td>SNR-E</td>
<td>yes</td>
<td>Good</td>
<td>&lt; 1 sec.</td>
<td>yes</td>
</tr>
<tr>
<td>ABE (Chap. 5)</td>
<td>1D</td>
<td>Baseline Correction</td>
<td>yes</td>
<td>N/A</td>
<td>(1.6 sec.)(^3)</td>
<td>yes</td>
</tr>
<tr>
<td>MxMEM (Chap. 6, 7)</td>
<td>2D</td>
<td>SNR-E</td>
<td>(high</td>
<td>Best</td>
<td>~ 4 seconds</td>
<td>yes</td>
</tr>
<tr>
<td>SG Filter (Chap. 2,3,4)</td>
<td>1D</td>
<td>SNR-E</td>
<td>N/A</td>
<td>Good to moderate</td>
<td>&lt; 1 sec.</td>
<td>yes</td>
</tr>
<tr>
<td>2D wavelets (Chap. 7)</td>
<td>2D</td>
<td>SNR-E</td>
<td>N/A</td>
<td>Similar to MxMEM</td>
<td>&lt; 1 sec.</td>
<td>yes</td>
</tr>
<tr>
<td>1D wavelets (Chap. 7)</td>
<td>1D</td>
<td>SNR-E</td>
<td>N/A</td>
<td>Good</td>
<td>&lt; 1 sec.</td>
<td>yes</td>
</tr>
</tbody>
</table>

Notes: TPMEM = Two-Point Maximum Entropy Method; BiChi is the filter based on “dual-use” of chi-squared function; iSG = Iterative-Savitzky-Golay; ABE = Auto-Baseline-Estimate method; MxMEM = Matrix Maximum Entropy Method; SG = Savitzky-Golay

SNR-E = Signal-to-Noise-Ratio Enhancement; Deconv. = Deconvolution/Deblurring ability
N/A = Not Applicable
1. Times given in this column (except for ABE) represent the total time to process 10 spectra (1000x10) such that the 1D methods process them one-at-a-time and the 2D methods process them as an “image”.

2. For BiChi, by adding deconvolution with SNR enhancement (“+d”), the average time to process one spectrum (from a batch of 10 spectra, each with unique noise) is shown.

3. This time is the average time to perform baseline correction of one spectrum (from a batch of 10 spectra, each with unique noise), where SBR = 1.0, SNR = 10, baseline = sigmoidal (see Figure 5.1c).

Column 5 states the quality of SNR enhancement, for applicable methods, based on quantitative data presented in the chapters (plus other results that were not shown). The computational times given in column 6 are based on noise reduction processing of a set of 10 Raman spectra (1000x10) such that the 1D methods process them one-at-a-time (total time to process 10 is given in the table) and the 2D methods process the 10 spectra as an “image”.

However, the ABE method’s time given in this column is an exception—it simply provides an example time for processing one spectrum for baseline correction (see table note 2 above). The computational platform used for these timing tests is equivalent to the one stated at the end of section 6.3. All of the algorithms in the table can be used within 2DCoS (at least as a form of “pre-treatment”, per Chapter 7), but only MxMEM and wavelets were so applied.

Algorithm selection might go as follows (assuming that all methods are available to the user):

1) for baseline correction, use ABE;

2) for deconvolution + SNR enhancement of spectra with low background (see Chap. 2), use TPMEM, else with high background, use BiChi.

3) for automated SNR enhancement (of a large set of spectra), use iSG;

4) for the best SNR enhancement of a set of related spectra (or an image in general), use MxMEM, else for unrelated spectra (with low background), use TPMEM.
5) for the best SNR enhancement of images in general (such as CT data), use MxMEM if the size is near 300x300, else use 2D wavelets for larger sizes to improve computational time.

6) for SNR enhancement in 2DCoS, MxMEM is most preferred in general, except for generating Asynchronous maps (as discussed in Chap. 7)—where 1D wavelets is preferred.

### 8.3 Future Work

#### 8.3.1 MxMEM Regularization With Deconvolution

In medical applications that use computed tomography (CT) scanning, such as for detection of lung cancer, there is a need for resolution enhancement (or deconvolution) of images. Because MxMEM is a regularization method, derived from TPMEM, deconvolution of (2D) images should be conceptually possible. As mentioned above, this development was not explored. However, both TPMEM and the Bi-Chi filter (BCF) are able to perform deconvolution. The BCF’s algorithm for this process was given in the Theory section of Chapter 3. Similar steps can be adapted for use by MxMEM, as follows:

**MxMEM Regularization With Deconvolution**

\( \hat{x}_M \) represents \( x_M \)

1. Initialize \( \hat{x}_M \) (with either flat plane or measured noisy data)
2. Convolve \( \hat{x}_M \) with the IPSF: \( \hat{x}_M * b \)
3. Apply \( \hat{x}_M \) to entropy component, but \( \hat{x}_M * b \) to chi-sqr component of Eq. 6.7
4. Reconstruct image \( x_M \) by minimizing MxMEM’s objective function (Eq. 6.7).
5. Assess fidelity using \( \chi^2 \) (Eq. 6.11)
(vi) Repeat steps (ii)-(v) until the stopping-criterion is satisfied ($\chi^2 \equiv M * N$).

Note that the IPSF could have two configurations. First, as in Chapter 6, a Raman artificial “image” actually consists of a collection of related spectra (combined into a matrix). In this case, one and the same IPSF vector would be convolved with each of $N$ spectra (simultaneously) in step (ii) of the algorithm. Second, for a true image, such as derived from a CT scanner, the IPSF would itself be a matrix, and a matrix convolution would be required in step (ii).

8.3.2 Exploration of Best Wavelets Family for 2DCoS Post-treatment Maps

As noted in Chapter 7, an interesting discovery was made. A wavelets post-treatment map was generated using the daubechies-8 wavelet because it gave a superior result as compared to the coiflet-2 wavelet. Although we in general adopted the use of the coiflet-2 wavelet (because of its similarity to vibrational spectral bands), in the case of filtering map features, we inferred that other wavelets may provide a better approximation.

As Berry and Ozaki observed [124], wavelets denoising is most effective when the wavelet is similar in shape to the target feature being reconstructed. In the case of vibrational spectra, this would be the peaks. In the case of Raman spectra, these would be Lorenztian in shape. Overall, there are not too many shape variations for spectra. In contrast, 2DCoS map features (when performing post-treatment denoising), may come in many varieties. For example, in Chapter 7, the synchronous and asynchronous covariance maps exhibited a wide variety of shapes.

Further exploration of this topic could begin by performing, as Berry and Ozaki did [124], a careful trial and error approach to find a wavelet that would be most suitable for map features. However, unlike vibrational spectral shapes, 2DCoS map features do not fall into
one category. Rather, it is anticipated that there would to be a large variety of 2D shapes/features that could be created. With that in mind, perhaps a set of wavelets could be determined, each of which would more closely match its own group of map features.

**8.3.3 Testing the Various Algorithms with other Spectroscopy Data Sets**

Generally, all of the algorithms in this thesis were tested only with Raman data. To extend the usefulness of this work, the spectra of other spectroscopy types should be tested against the algorithms. This would give other spectroscopists greater confidence in these methods.

A specific example for further exploration was suggested in section 7.5: “In particular, it would be interesting to see if the 2D post-treatments are of any benefit in dealing with the narrow peaks and peak congestion often present in NMR spectra.”

Furthermore, it would be valuable and interesting to explore the application of the *automated* iSG noise filter (of Chapter 4) to other areas such as infrared spectroscopy, UV-Vis, and diagnostic imaging techniques.

**8.3.4 Automation of MxMEM (and TPMEM)**

MxMEM has a high potential for being extended into a fully-automated 2D SNR enhancement algorithm. Only one parameter requires user selection: the $X$ multiplier (see Eq. 6.12, section 6.2.3). As seen in Table 6.2, MxMEM requires a higher $X$ value for noisier data ($X=1.1$ for SNR 3), but a lower value for less noisy data ($X=1.0$ for SNR 5, and $X=0.9$ for SNR 20, data not shown). In order to automate the selection of $X$, the noise level (specifically the standard deviation, $\sigma$) of the image must be determined. In addition, the calculation of chi-squared (Eq. 6.11) requires the calculation of variance, $\sigma^2$. Both of these can be calculated using the method described in Chapter 4 (section 4.2.2) for automating the

iSG filter. That is, “…σ can be obtained from a simple automated noise estimation procedure [55], thus, user intervention is unnecessary…” In this manner, through the automated calculation of σ, the value of X can be programatically chosen as a function of the determined noise level, and thus the entire MxMEM algorithm will be fully-automated.

Note that by implication, TPMEM can also be automated, even more easily, since it has no X multiplier, but only the need to have an automated calculation of σ for its chi-squared function.

8.3.5 Hardware Implementation/Acceleration of Algorithms

As briefly introduced in Chapter 6 and discussed in Appendix 3, our multi-disciplinary research included (as a minor theme), preliminary investigations into hardware implementations of our algorithms (to speed-up computations and for future “micro-spectrometer” systems on a chip which will require embedded/automated signal processing algorithms). Based on the above analysis of automating MxMEM, this method is now ready for hardware implementation using the design concepts recommended in Appendix 3.

8.3.6 Exploration of N-point TPMEM/MxMEM

As reviewed in Chapter 6, TPMEM is based on using every two adjacent points for its entropy calculation. MxMEM, as presented in Chapter 6, is an extension of TPMEM into 2D, and uses a 4-point (box) entropy calculation—because this is the simplest symmetrical 2D format. The question arises—how would these two methods perform with a larger number of points, respectively, in the entropy calculations? Three-point and four-point MEM versions of TPMEM were informally explored (data not shown) using Raman data similar to that used in Figure 6.2. The general outcome was that greater smoothing resulted (compared to TPMEM), but also greater peak distortion. This result would seem to be
analogous to using a moving-window-average filter and comparing a 3-point window to larger window sizes—more smoothing results with larger windows but also more peak distortion, since more points are being averaged.

Larger point sizes were not investigated for MxMEM. It seems likely that the above phenomenon may result in 2D also. However, it would be interesting to explore rectangular as well as square “window” patterns with a larger number of points (e.g., 3x3 or 2x4, etc.). Similar to the statement in section 8.3.3 above, these other window sizes and orientations might provide superior SNR enhancement for NMR and other spectroscopies.

8.3.7 Investigation of Cosmic Spike Removal

This thesis addressed three significant measurement problems: noise, instrumental blurring, and baseline distortion. A fourth measurement problem in spectroscopy is the occasional presence of cosmic spikes. None of the noise reduction algorithms in this thesis attempt to address the spike problem, and in their present implementations, none are capable of spike removal. Therefore, it would be interesting to explore whether the noise reduction algorithms can be modified to address this problem.
References


Proceedings of the 22nd Canadian Medical and Biological Engineering Society Conference, Charlottetown, PEI, 1996, pp. 78-80.


Appendix 1—Data Characterization

A1.1 Statistics Concepts

Several statistical concepts and equations are used throughout this work [145][146]. A summary of them is presented here. These equations can be formulated for a population (divide by N), as given below, or for a sample (divide by N-1).

A1.1.1 Mean

The average value of the data, in statistical terms, is called the mean (indicated by the lower case Greek \( \mu \)):

\[
\mu = \frac{\sum_{i=1}^{N} X_i}{N} \tag{A1.1}
\]

A1.1.2 Variance

Variance (indicated by the lower case Greek \( \sigma^2 \) squared) is one of several ways to describe a distribution of data. It is a measure of how far each value in the data set is from the mean. It is defined as the sum of the squared distances of each term in the distribution from the mean, divided by the number of terms in the distribution (N).

\[
\sigma^2 = \frac{\sum_{i=1}^{N} (X_i - \mu)^2}{N} \tag{A1.2}
\]

A1.1.3 Standard Deviation and Standard Error

The standard deviation (SD) is simply the square root of the variance. It also shows how much variation there is from the mean. A low SD value indicates that the data points tend to cluster close to the mean, whereas a high value indicates that the data is spread out over a large range of values. In addition, SD is used as a way to measure confidence in
statistical conclusions from experimental data (as will be seen in many of the data tables presented in later chapters).

\[
\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (X_i - \mu)^2}
\]

(A1.3)

The standard error (SE) (or standard error of the mean) is also used as a way to measure confidence in statistical conclusions from experimental data [36]. Specifically, for comparison purposes, it is a measure of how much the means of different samples, drawn from the same population, vary. In contrast, the standard deviation is a measure of how much data points within a single sample vary from the mean of that sample. The SE is simply the SD divided by the square root of the number of terms in the data set:

\[
SE = \frac{\sigma}{\sqrt{N}}
\]

(A1.4)

A1.1.4 Covariance

Covariance [147][148] is a measure of how two variables change (or co-vary) together. More specifically, it measures the degree to which pairs of points systematically vary about their respective means. Note that variance is a special case of covariance when the two variables are identical.

\[
Cov_{xy} = \sum_{i=1}^{N} \frac{(X_i - \mu_x)(Y_i - \mu_y)}{N}
\]

(A1.5)

A1.1.5 Correlation and Pearson’s Correlation Coefficient (PCC)

Correlation is a measure of the relation between two variables. Correlation coefficients provide a numerical measure of the degree of correlation. Pearson’s is the most
commonly used and provides values between +1 and -1, inclusive. Note that PCC is simply the covariance divided by the standard deviations of the two variables.

$$\rho_{xy} = \frac{Cov_{xy}}{\sigma_x \sigma_y}$$  \hspace{1cm} (A1.6)

### A1.2 Simulated Data and Figures of Merit

#### A1.2.1 Spectral Lorentzian peaks

In this work, the spectroscopic signals are often modeled (and simulated) as having Lorentzian-shaped peaks which are defined \([10]\) by

$$x_i = \frac{c}{1 + (i - i_0)^2/\Gamma}$$  \hspace{1cm} (A1.7)

where \(x_i\) is the \(i\)th bit (channel) of a spectrum, with peak amplitude \(c\), channel number \(i\), peak center channel number \(i_0\), and \(\Gamma\) is the half-width-at-half-maximum (HWHM) of the peak.

#### A1.2.2 Root-mean-squared error (RMSE)

The RMSE is used with simulated data to make a point-by-point (\(N\) points) comparison of an ideal spectrum (\(X_1\)) and a reconstructed spectrum (\(X_2\)).

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (X_{1,i} - X_{2,i})^2}$$  \hspace{1cm} (A1.8)

When two algorithms are being compared in their ability to reconstruct (or estimate) a signal, RMSE values are calculated from each algorithm’s output (in comparison to the ideal). These two values are then used as “figures of merit” to numerically indicate which algorithm’s reconstruction was closest to the ideal spectrum. A lower RMSE value indicates a better estimate.
Similarly, the Pearson’s Correlation Coefficient, given above, is often used in conjunction with the RMSE as an additional figure of merit. The PCC is a “similarity” measure which is insensitive to offset. That is, if a vector’s shape is identical to another vector’s, but has an offset, PCC will equal 1 (perfect correlation), whereas RMSE will be sensitive to the offset and tend to give a high (poor) value. A good reconstruction would have a low RMSE and a high PCC value. However, in Chapter 6, for ease of comparison with RMSE, we use the complementary form of PCC such that a good reconstruction has a low value.

A1.2.3 Signal-to-Noise Ratio (SNR)

In this work, SNR is generally defined as the intensity of the tallest peak \( [signal \ maximum - signal \ minimum] \) relative to the baseline noise. The noise value can be obtained by calculating the standard deviation of a relatively flat section of the spectrum’s baseline.
Appendix 2--MxMEM Gradient Calculation (concept and coding)

MxMEM’s gradient calculation was introduced in Chapter 6 (equations 6.13-6.15). Additional details are given here—relating to those equations and how they are implemented in the C code (see Appendix 3).

Entropy gradient calculation

For efficiency, Eq. 6.13 is repeated here:

$$\frac{\partial S_k}{\partial x_j} = \left[ \sum_{i=1}^{4} \left( 1 + \ln \frac{\hat{x}_i}{\hat{x}_s} \left( \frac{\hat{x}_i}{\hat{x}_s} \right) \right) \right] - \left[ \left( 1 + \ln \frac{\hat{x}_j}{\hat{x}_s} \right) \left( \frac{1}{\hat{x}_s} \right) \right]$$

(Eq. 6.13)

The above equation is used for calculating the entropy gradient values of one 4-point box (or kernel) within an image, by utilizing the $\hat{x}_i$ values from the spectral estimate matrix. Note that four values are calculated, one for each value of $j$, and that these correspond to the four elements within a given box, such as box B1 (dashed-lines) given below in Figure A2.1. Also note from the equation that each $j$-point’s gradient value is based on all 4 points ($i = 1$-$4$) within the box.
Figure A2.1 (a) A conceptual entropy-gradient matrix of 4 rows, 8 columns, and 32 elements. (b) Detail of (a) showing that all points of the matrix (image), except for edge points, are surrounded by 4 boxes, hence the gradient calculation for any one point is based on the adjacent boxes which surround it.

**Entropy gradient calculation via “sliding window”**

In order to calculate the entire entropy gradient matrix, a “sliding window” is used as follows, beginning with rows 1 and 2 (and showing the first 3 locations of the window, indicated by highlight boxes, B1-B3, Fig. A2.1a). Imagine that lying underneath the matrix of Fig. A2.1a is the spectral estimate matrix/image, $\hat{\mathbf{M}}$. Eq. 6.13 is first applied to the box contained within B1 in the figure (utilizing the corresponding 4 points of the “underlying” $\hat{\mathbf{x}}_M$). This results in four gradient values held in the gradient matrix elements within B1. Next, the window slides to adjacent box B2 (solid lines), Eq. 6.13 is applied again (based on the underlying $\hat{x}_i$ values of B2), and four new gradient values are obtained. Note that two of these values (the ones in column 2) must be added (accumulated) to those which were obtained in the B1 calculation. Next, the window slides to B3 and so on, until the end of the row. Then, the window moves up to cover rows 2 and 3, beginning at columns 1 and 2. At each stop of the window, four gradient values are calculated and then accumulated to any previously existing values within that box. This process continues, upward by adjacent rows (and left to right), until every point’s gradient...
value has been calculated/accumulated. Finally, note that, as in Fig. A2.1b, every $\hat{x}_i$ point
(except edge-points) has a gradient that is based on 9 points, because each point is
surrounded by 4 boxes.
Appendix 3—MxMEM Hardware Acceleration: Preliminary Investigations

Hardware acceleration of large images

As a multi-disciplinary (engineering-science) team, our research also includes investigations into novel methods of speeding up computational times for algorithms when processing large images and/or data sets which require high-throughput. This involves the design of custom electronic “hardware” which permits parallel processing (doing many computations simultaneously) vs. the limited ability of standard computers which use sequential software methods. Furthermore, since this hardware is based on microelectronic chips, our research [149] also anticipates helping to pave the way for future “microspectrometer” systems on a chip [150][151] (which will require an on-chip “signal processor” that implements algorithms like MxMEM).

Exposing Parallelism of the Algorithm

In order to exploit parallelism within an algorithm, modifications might need to be made which slightly reduce the quality of results but allow for significant speed-up. With this in mind, we discovered a novel modification of MxMEM that “exposes the parallelism”. An earlier version of MxMEM processes only two rows at-a-time (but still uses the 4-point box as a small matrix kernel). Its performance was slightly inferior to the design presented in Chapter 6. It processes rows 1 and 2, then 2 and 3, etc. In this sequence, parallelism is not exposed because the algorithm must wait for the previous two rows to finish before beginning the next two rows. However, a simple modification, which we call “adjacent rows toggling” (ART), potentially allows all of the rows (or columns) in an image to be processed
simultaneously, and produced comparable results (to its predecessor) when simulated. In ART, pairs of rows are processed as follows (assuming 8 rows for ease of discussion): rows 1-2, 3-4, 5-6, 7-8. Then “toggle” and process rows 2-3, 4-5, 6-7. Then toggle back to the first sequence, etc. Note that the outermost rows may receive less smoothing. In a parallel processing environment, all of the rows in the sequence before the toggle may be smoothed simultaneously, then after the toggle, all of the rows in the second sequence may be handled simultaneously, then repeat until the stopping criterion (Eq. 6.12) is reached.

In Chapter 6, the final version of MxMEM, which might be called “full-matrix-MEM” (fMM), was actually designed after we had begun exploring the hardware acceleration of the modified-MxMEM (mod-MM) version. We have since realized that fMM already has the parallelism exposed, and it too could be implemented in parallel form using the ART procedure.

**Multi-Processor Platforms Explored**

Two multi-processor platforms (or approaches) were explored for parallel processing implementation of the above mod-MM algorithm:

1) A Xilinx Field Programmable Gate Array (FPGA) with embedded processors, and

2) An Ambric Am2045 Massively Parallel Processor Array (MPPA).

Based on a preliminary analysis, we determined that to simplify the design (using either of these multi-processor platforms), each individual processor must run its own copy of the mod-MM algorithm, but operate on its own unique sub-set of data (i.e., some portion of the overall image/matrix). Within the algorithm, it is already natural to process two rows at a time (since 4-point boxes lie across two rows). Further, as a “benchmark” image size,
we estimated that parallel processing would become significantly faster, vs. sequential processing, for a 1000x1000 image (such as an expanded version of Figure. 6.1c, which is only a 300x300 image).

**FPGA Investigation**

Our donor, Canadian Microsystems Corporation (CMC), provided an excellent research vehicle in the form of an Amirix AP1000 PCI card [152][153] with a Xilinx Virtex II ProT FPGA (XC2VP100) [154] on board for implementing/investigating embedded systems. The FPGA contained two PowerPC “hard” processors and also allowed for numerous MicroBlaze “soft” processors to be implemented as desired. Furthermore, CMC later made available a “Multi-MicroBlaze” design (downloadable into the card/FPGA) that implemented a PowerPC running Linux and connected to 4 MicroBlaze processors [152]. This embedded system allowed for exploration of parallel processing with 4 processors (under control of the PowerPC). We were able to successfully boot Linux in the PowerPC and command the 4 processors to run simple C programs (modified from design examples supplied by CMC). A lot of time was invested in learning this hardware/software system (and all of the associated design tools). However, although it is an interesting and powerful platform, it proved to be very difficult to use (managing only 4 processors), and did not appear to have a large enough capacity for running even say 50 MicroBlaze processors. Placement, layout and routing, synthesis, and potential timing issues were anticipated (as the design moved from 4 to N processors). Therefore, this platform was deemed not suitable for Mod-MM implementation.
MPPA Investigation

Ambric/Nethra’s Am2045 MPPA contains 336 32-bit Reduced Instruction Set Computer (RISC) processors in a partly-programmable standard cell design, running at 300 MHz [155]. We were able to prototype our designs in the Ambric’s PCI Express-based Am2045 GT Development Board (which contains one MPPA), as well as in the simulator. The aDesigner tool set proved to be considerably easier to use than the FPGA tools (and managing Linux, etc.), but still was moderately complicated. However, there were two additional major challenges: 1) all of our C code would have to be converted to Java, and 2) there is no floating point support, so we would have to create our own integer arithmetic fixed-point (software) design for mathematical operations (the most difficult being division and logarithm routines). Here is a summary of our accomplishments in creating a partial preliminary design:

1. A single processor design that implemented (one-dimensional) TPMEM was successfully ported from C to Java and simulated correctly (in standard Java).
2. We determined that in general we should use a Q24 fixed-point format (1 sign bit, 7 integer bits, 24 fractional bits).
3. A fixed-point division routine was successfully designed based on Suh et al’s work [156]. This was first done in C, and then correctly ported/tested in Java. Our division routine ran successfully in the Ambric simulator and hardware (Am2045 GT Development Board).
4. We adopted and successfully implemented Andreas Lauschke’s fixed point logarithm routine in Java [157]; it ran identically in the Ambric simulator and hardware.
5. Based on a research paper published by Ambric engineers [155], we adopted the “work farm” architecture as the basis for implementing mod-MM.

6. A 4-processor design (representing a pre-cursor to a simple work-farm) was implemented and ran successfully in the hardware (mainly passing 32-bit data correctly into the chip, through the processors, into and out of on-card memory, and then out of the chip to the PCI bus).

7. A stand-alone chi-squared calculation block, equivalent to Eq. 6.11, was implemented (using our division routine, plus the built-in 32-bit multiply routine) and ran successfully in the Ambric simulator and identically in the hardware.

In conclusion, a large amount of time was also invested in this MPPA investigation. Overall, Ambric’s MPPA platform (based on their carefully thought out programming model, hardware architecture, and software tools) appears to be very well designed and explicitly intended for parallel processing applications. Therefore, it is potentially quite suitable for mod-MM implementation.

A small challenge we encountered was that coding the fixed-point commands is rather time consuming and tedious work. However, the biggest issue, according to our analysis, is that there is not sufficient local memory for each processor to hold its subset of the data vectors partitioned from the overall data matrix (when the size is 1000x1000). Standard MxMEM requires memory space for 5 matrix variables (measured data \( m \), spectral estimate \( \hat{x} \), gradient \( g \), plus two copies of \( g \) required by the conjugate gradient routine). For mod-MM, the same 5 are required, but only in 2-row sections. Thus, each hardware processor (running its own copy of mod-MM) must provide memory for \( 5 \times 2 \times 1000 = 10k \) 32-bit words. However, each Ambric (Am2045) core unit (known as a bric) has 4 processors
and 4k 32-bit words of Random Access Memory (RAM)—2k for processor Instructions and 2k for Data. So, 2k data words must be shared among 4 processors. Although the RAM can be allocated from neighboring brics, the design becomes more complicated. On a positive note, it appears that the simulator can simulate any memory size, for the sake of research, but the present MPPA hardware chip does not have sufficient memory for our target image size of 1000x1000. Future versions of the Ambric MPPA will probably increase the amount of RAM per bric and thereby enable our desired implementation size.
Appendix 4—MxMEM Implementation Details and C Code

**Implementation details.** MxMEM (from Chapter 6) is implemented with a Matlab “front-end” that prepares the data and then passes it to C code (organized in a Matlab “mex” format) which performs most of the processing. A version of MxMEM composed of 100% Matlab code was also successfully built and used for analysis, but takes approximately 10 times longer to run (e.g., 20 minutes vs. 2 minutes in C code for a 300x300 image). The C code employs the following utilities from NRC [36], which compose the conjugate gradient method (CGM) section of the algorithm: frprmn.c, linmin.c, brent.c, mnbrak.c, and f1dim.c. These were converted to using the C data type “double” (instead of “float”) so as to most accurately interface to Matlab, which uses “double” data types by default.

For the discussion that follows, it is important and interesting to discuss one key aspect of how the optimization works [36]. The CGM minimizes an objective function, $F$, by iteratively adding a multiple of the gradient vector to a trial copy ($\hat{x}_{trial}$) of the current spectral estimate vector, and then evaluating $F$, with the trial copy, to see if it has reached a minimum. Within $N$-dimensional space, the gradient vector “points downhill”. The key equation has this form:

$$\hat{x}_{trial} = \hat{x} + mul \ast g$$  \hspace{1cm} (A3.1)

where $mul$ is a multiplier and $g$ is the gradient vector. Note that if the gradient has large negative values (which it often does early in the simulation) these will be added into $\hat{x}_{trial}$.

One limitation of MxMEM (as well as T1D and T2D) is the use of the natural log function (“ln”) in the entropy calculation (Eq. 6.8). Consequently, negative numbers cannot be allowed to pass into any log calculations—or else Matlab and C will generate a NaN (“not
a number”) for invalid input operands. This generally will cause the simulation to “hang” or go awry in some other way. We prevent negative values from reaching the entropy calculation in two steps: 1) data is normalized before processing (such that the absolute value of \( \text{data}_{\text{max}} - \text{data}_{\text{min}} \) is 1), and then restored in post-processing; 2) a small offset is added to the normalized data. Normalization limits the gradient vector to having only small negative numbers. This along with the offset insures that \( \hat{x}_{\text{trial}} \) will have only positive numbers.

Since the above-mentioned NRC code requires the use of vectors, in our Matlab front-end we disassemble the entire 2D data matrix by rows, and then concatenate those rows into one long vector like a train of cars (row 1, row 2, etc.). Note that Matlab stores matrices exactly in an analogous manner in computer memory—except that it concatenates columns, not rows. Thus, although the entire image is passed as one long vector, every mathematical operation performed on the image is done in a true matrix manner. For example, in doing a 4-point entropy kernel calculation (Eq. 6.8), the algorithm must manage the assembly of a “box” by taking 2 points from row 1 (for example) and the corresponding 2 points from row 2, and do this in a sliding-window manner across those two rows (and all other rows), analogous to the gradient calculation given in Appendix 2. After processing, the reconstructed image—in the form of a (long) output vector—is reassembled by Matlab into a 2D matrix (image), so as to do post-processing and plotting.

**MxMEM C Code**

The following code represents only the core of the MxMEM design. In addition, three Matlab files (not shown) are used which perform the highest level control and implementation of the algorithm, and also manage data into/out of the C code. Furthermore,
several C files (not shown) from NRC are also compiled, along with this C code, which compose the CGM as mentioned above. See Appendix 2 for a high-level explanation of how some of the following C code works.

/* fmx_4pmem_mex.c  (Full-Matrix, 4-point Maximum Entropy Method, version 0)
2.2.09  by Rod Foist (for the King and for the Kingdom!)
1. This C code collection is accessed by calls from Matlab into/out of
   * the gateway function at the very end of this file:
   *   fmx_4pmem_mex(Num_rows_in, lambda, variance, data_vec, recon_vec_in,
   *      recon_vec_out, Num_cols_in, EP_in, grad_variance);
   *
   *  2. Note that the matrix which is processed here is input (from Matlab) in
      *  the form of a long concatenated vector

COMPILE: in Matlab, run "compile_fmx_4pmem.m" in this folder
*
*
CODE TYPE:  C program source file--"Main" Program, but configured for calling by MatLab; thus "main"
changed to become a function prototype

DESCRIPTION: Mainly implements "frprmn" minimization as a mex file. This file will be called by Matlab
which together are implementing the Full-matrix, 4-point MEM algorithm

*/
/*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#include <string.h>
#define NRANSI
#include "nr_d.h"       // Using double version
#include "nrutil.h"
#include "TWOMEM_rod.h"
#include "mex.h"        // required to use MatLab functions
#endif FTOL 1e-6 /* fractional tolerance for frprmn minimization */
#define FTOL 1e-8

#include <time.h>

#include "mex.h"       // required to use MatLab functions

#include <stdio.h>

// ...... global variables .....
// Pointers used in entropy and gradient calculations
double *xe, *xpe, *xlogpe;
double *xg, *xp, *xpp, *xlogp, *g;
double *x_vec, *x_vec_g, *df_ent_vec;

/* Lagrange multiplier, noise variance, X2 target*/
double lam, var, dtot;
double grad_var;
int EP;
int Num_rows, Num_cols;

// function calculates chi-squared statistic between reconstructed and measured vals
// (entire matrix concatenated together, thus NP = Num_rows*Num_cols)
double chisqr(double recon_vec_long[])
{
    int i;
    double result;
    result=0.0;
    for(i=1;i<=Num_rows*Num_cols;i++)
        result+=SQR(data[i]-recon_vec_long[i])/var;
    return result;
}

// function calculates one 4-point, 2-vector entropy value (one "box" of 4 points)
double entropy_base(double x[])
{
    // globals used here: EP, xpe, xlogpe, xe, (x[] = xe[])
    int i, j;
    double xs, ent_box;

    // *** Calculate EP-element entropy (2-point, 3-point, etc. depending on EP)
    xs = 0; ent_box = 0;
    for(i = 1; i <= EP; i++)
    {
        xs += x[i];               // scalar result
    }
    for(i = 1; i <= EP; i++)
    {
        xpe[i] = x[i]/xs;          // vector result
        xlogpe[i] = log(xpe[i]);    // vector result
        ent_box += -(xpe[i]* xlogpe[i]);  // scalar result (directly to Matlab, via pointer *f)
    }
    return ent_box;
}

// function calculates sum of 4-point, 2-vector entropies
double fmx_entropy_2vec_sum(double x_2vec[])
{
    // globals used here: xe, Num_cols
double S_sum;
int i,j;

S_sum = 0;
// x_2vec contains latest 2 rows concatenated = 2*Num_cols long, but we operate on the 2 halves, each row is
// Num_cols long (but Num_cols-1 pairs of adjacent points in each row)
for (i = 1; i <= Num_cols-1; i++) {
  // for each i, load xe[] with 4 bits (i->4)
  for(j = 1; j <= 4; j++){ //concatenate together 4 bits, 2 from each vector-half, in adjacent-windowing
    if(j<3){
      xe[j] = x_2vec[i+(j-1)]; // load 2 bits from left-half of x_2vec (1->Num_cols)
    } else{
      xe[j] = x_2vec[i+(j-3)+ Num_cols]; // load 2 bits from right-half of x_2vec (Num_cols+1->2Num_cols)
    }
  }
  S_sum += entropy_base(xe);
}
return S_sum;

double fmx_entropy_sum(double recon_vec_long[])
{
  // globals used here: Num_rows, Num_cols, x_vec[])
  /*
  Sums "full-matrix" of entropy calc's, by adding all adjacent 2-vector sums (per MxMEM)
  Uses fmx_entropy_2vec_sum (= MxMEM's f_4p2v_entropy_sum)--but here done repeatedly for all 2-vec sets
  Equation is of form: Smx = S_12 + S_23 + ... S_mn
  (Thus, sum of Entropy boxes of rows 1-2, rows 2-3, ... + rows N-1->N )
  */
  int i,j;
  double S_mx_sum, S_2vec_sum;
  S_mx_sum = 0;
  S_2vec_sum = 0;
  for (i = 1; i <= Num_rows-1; i++) {
    // recon_vec_long[] is a looong vector holding entire matrix concatenated
    // in row order
    for (j = 1; j <= 2*Num_cols; j++) {
      // one row is Num_cols long, 2 rows are 2*Num_cols long
      // global x_vec contains latest 2-rows concatenated, ripped from recon_vec_long
      // Sequence is: rows 1-2, 2-3, etc.
      x_vec[j] = recon_vec_long[i+(j-1)*Num_cols]; // verified in: test_2rows_from_longvec.m
    }
    S_2vec_sum = fmx_entropy_2vec_sum(x_vec);
    S_mx_sum = S_mx_sum + S_2vec_sum;
  }
  return S_mx_sum;
}

/* function func calculates Sum4pMem+lam*chisquared for entire matrix;
this Num_rows*Num_cols dimensional function is minimized by frprmn */
double func(double recon_vec_long[])
{
    double result;

    result = -1*fmx_entropy_sum(recon_vec_long) + lam*chisqr(recon_vec_long);

    return result;
}

// function calculates 4-point, 2-vector Entropy gradient (for one "box" of 4-points)
double ent_gradient_base(double x[])
{
    // globals used here: EP, xp, xlogp, xpp, g, (x[] = xg[])
    int i;
    double xs, g1;

    // *** Calculate EP-point gradient (2-point, 3-point, etc. depending on EP)...EP = 4 for 4-point box
    xs = 0; g1 = 0;
    for(i = 1; i <= EP; i++){
        xs += x[i];               // scalar result
    }
    for(i = 1; i <= EP; i++){
        xp[i]    = x[i]/xs;          // vector result
        xlogp[i] = log(xp[i]);    // vector result
        xpp[i]   = xp[i]/xs;          // vector result
        g1 += ((1+xlogp[i]) * xpp[i]);  // scalar sum
    }
    for(i = 1; i <= EP; i++){
        g[i] = g1 - (1+xlogp[i])/xs;    // vector result
    }
}

// function calculates 2*Num_cols dimensional Entropy gradient (via 4-point, 2-vector "boxes") of func at point
x_2vec and
// puts the results in vector df_ent_vec; this function is only called by frprmn
void fmx_2vec_ent_gradient(double x_2vec[])
{
    // globals used here: Num_cols, xg, g, lam, dtot, (recon_vec_long[] = recvec[])

    // NOTE: This function works just like MxMEM's gradient_4p2v, except Chisqr gradient
    // removed (since added at higher level)
    // a. Sums 4-point Entropy gradient calc's, from 2 vectors concatenated, using ent_gradient_base(x)

    int i,j,k;

    // Calculate gradient 2-row-vector for Entropy only (2 rows = 2*Num_cols points are calculated)
    // Initialize df_ent_vec to zeroes
    for(k = 1; k <= 2*Num_cols; k++){
        df_ent_vec[k] = 0.0;
    }
    // x_2vec contains latest 2 rows concatenated = 2*Num_cols long,
    // but we operate on the 2 halves, each row is Num_cols long (but Num_cols-1 pairs of adjacent points)
    for (i = 1; i <= Num_cols-1; i++) {
        // for each i, load xg[] with 4 bits (i->4)
for(j = 1; j <= 4; j++)  // concatenate together 4 bits, 2 from each vector-half, in adjacent-windowing
     // manner
    if(j<3){
      xg[j] = x_2vec[i+(j-1)]; // load 2 bits from left-half of x_2vec (1->Num_cols)
    }
    else{
      xg[j] = x_2vec[i+(j-3)+ Num_cols]; // load 2 bits from right-half of x_2vec (Num_cols+1-
                // >2*Num_cols)
    }
}

// Pass xg[] to ent_gradient_base, then accumulate resultant vector values into df_ent_vec
// Note: df_ent_vec is "loaded", in 2 halves, by sliding EP-wide window along, (1->Num_cols;
// Num_cols+1->2*Num_cols)
for(k = 1; k <= 4; k++){
    ent_gradient_base(xg);              // output pointed to by g[], which is 4-bits long
    if(k<3){ // accumulate (via sliding window) values into df_ent_vec[]
        df_ent_vec[i+(k-1)] += g[k]; // load/accum 2 bits into left-half of df_ent_vec (1->Num_cols)
    }
    else{
        df_ent_vec[i+(k-3)+ Num_cols] += g[k]; // load/accum 2 bits into right-half of df_ent_vec
                // (Num_cols+1->2*Num_cols)
    }
}
return;

// Function calculates the gradient for the entire matrix
void fmx_mx_gradient_sum(double recon_vec_long[], double df_lvec[])
{
    // globals used here: Num_rows, Num_cols, xg, g, lam, dtot, data[], df_ent_vec[]
    // ** malloc for df_lvec[] allocated in frprmn

    /*
     Calculates total 4P-rows gradient matrix...Now for entire matrix: every 2-adjacent vectors
     Equation form: Fcost = -(S_12 + S_23 + ... + Smn) + Lambda * Chisqr(full-matrix)
     a. Sums 4-point Entropy gradient calc's, from 2 vectors concatenated, using ent_gradient_base(x)
     b. Does the above for every two-adjacent rows within the matrix, thus creating an entropy-gradient matrix
     c. Adds in ChiSqr gradient calcs for entire matrix
     */
    int i, j, k;
    double variance;

    // 1a. Calculate Entropy gradient-matrix values (two adjacent vectors at a time)
    // df_lvec is a long vector containing the entire gradient matrix, with all rows concatenated
    for(k = 1; k <= Num_rows*Num_cols; k++)  // Initialize gradient long-vector to 0's
        df_lvec[k] = 0.0;
    for (i = 1; i <= Num_rows-1; i++)  // recon_vec_long[] is a looong vector holding entire matrix concatenated
        // in row order
            // 1) Assemble current 2-row vector, in adjacent row manner
            for (j = 1; j <= 2*Num_cols; j++)  // one row is Num_cols long, 2 rows are 2*Num_cols long
                // x_vec_g contains latest 2-rows concatenated, ripped from recon_vec_long

// Sequence is: rows 1-2, 2-3, etc.
    x_vec_g[j] = recon_vec_long[j+(i-1)*Num_cols];  // verified in: test_2rows_from_longvec.m

// 2) Calc entropy-grads for current 2 rows (just like in MxMEM)
    fmx_2vec_ent_gradient(x_vec_g, Num_cols);  // **result into global: df_ent_vec (= gradient vector, 2
    // rows long)

// 3) Accumulate current 2 rows of gradient values into df_lvec
    // (long gradient vector holding all rows of matrix); df_ent_vec[] = global, calc'd above by
    // fmx_2vec_ent_gradient
    // Sequence is: rows 1-2, 2-3, etc.
    for (j = 1; j <= 2*Num_cols; j++){
        df_lvec[j+(i-1)*Num_cols] = df_lvec[j+(i-1)*Num_cols] + df_ent_vec[j]; // verified in:
            // test_2rows_from_longvec.m
    }

// 1b. Negate entropy-grad values (per Fcost = -S ...)
    for(k = 1; k <= Num_rows*Num_cols; k++){
        df_lvec[k] = -df_lvec[k];
    }

// 2. Add/accumulate gradient values for Lambda*ChiSqr (for entire matrix = Num_rows*Num_cols points)
    variance = 1.0;  // Recall that we are setting variance = 1 in gradient, but not in Fcost calc.

    // These 3 long vectors should all be aligned: df_lvec, recon_vec_long, data
    for(i = 1; i <= Num_rows*Num_cols; i++){
        //df_lvec[i] = df_lvec[i] + (lambda/variance)*(2*recvec(i) - 2*data(i));
        df_lvec[i] += (lam/variance)*(2*recon_vec_long[i] - 2*data[i]);
    }
    return;

/*/ .................. "Main" routine ................. */

void fmx_4pmem_mex(double Num_rows_in, double lambda, double variance, double *data_vec,
    double *recon_vec_in, double *recon_vec_out, double Num_cols_in, double EP_in,
    double grad_variance)
{
    int i, j, count;
    int iter;  // for frpmn output
    double fret,xacc,root,posa,wida,rmser,fitcor,prob,z;

    clock_t start, end;
    double elapsed;

    start = clock();

    // pass in parameter values via MatLab
    Num_rows = Num_rows_in;
    Num_cols = Num_cols_in;

    //NP = Num_rows*Num_cols;  // NP = number of points in entire matrix
    EP = EP_in;

lam = lambda;
var = variance;
grad_var = grad_variance;

// *** Allocate memory via NumRec's "dvector" (which calls malloc) ***

// Num_rows*Num_cols-long vectors are entire matrix concatenated (length = Num_rows*Num_cols)
data = dvector(1,Num_rows*Num_cols); // measured data (with noise)
recvec = dvector(1,Num_rows*Num_cols); // reconstructed data

// Malloc calls for dvectors used in entropy calc
xe = dvector(1,EP);
xpe = dvector(1,EP);
xlogpe = dvector(1,EP);
x_vec = dvector(1,2*Num_cols);

// Malloc calls for dvectors used in gradient calc
xg = dvector(1,EP);
xp = dvector(1,EP);
xpp = dvector(1,EP);
xlogp = dvector(1,EP);
g = dvector(1,EP);
x_vec_g = dvector(1,2*Num_cols);
df_ent_vec = dvector(1,2*Num_cols);

// Load local vectors with vectors from Matlab--vectors are Num_rows*Num_cols long (entire matrix concatenated)
for (j = 1; j <= Num_rows*Num_cols; j++) {
    data[j] = *(data_vec + (j-1));
    recvec[j] = *(recvec_in + (j-1));
}

// *** Initialization of recvec is now done in Matlab calling routine //

// ........... Main algorithm computation .............

// minimize equation "func" (= Ftm)
// also uses globals: lam, var (set above);

// NOTE: Num_rows*Num_cols = number of points in entire matrix!
frprmn(recvec,Num_rows*Num_cols,FTOL,&iter,&fret,func,fmx_mx_gradient_sum);

//printf("n C code: NP = %d, EP = %d, var = %lf, lam = %lf \n", NP, EP, var, lam);
//printf("n C code: data[1,NP,2NP] = %lf, %lf, %lf \n", data[1], data[NP], data[2*NP]);
//printf("n C code: recvec[1,NP,2NP] = %lf, %lf, %lf \n", recvec[1], recvec[NP], recvec[2*NP]);
//printf("n C code: data[1->4] = %.5lf %.5lf %.5lf %.5lf \n", data[1],data[2],data[3],data[4]);
//printf("n C code: data[NP+1->4] = %.5lf %.5lf %.5lf %.5lf \n", data[NP+1],data[NP+2],data[NP+3],data[NP+4]);

// Pass recovered vectors to MatLab (vectors = entire mx of rows concatenated = Num_rows*Num_cols)
for (j = 1; j <= Num_rows*Num_cols; j++) {
    *(recon_vec_out + (j-1)) = recvec[j];
} /* output results to screen and log file */

    // free memory (using Num. Recipes function, "free_dvector")
    free_dvector(recvec,1,Num_rows*Num_cols);
    free_dvector(data,1,Num_rows*Num_cols);
    free_dvector(xe,1,EP);
    free_dvector(xpe,1,EP);
    free_dvector(xlogpe,1,EP);
    free_dvector(x_vec,1,2*Num_cols);
    free_dvector(xg,1,EP);
    free_dvector(xp,1,EP);
    free_dvector(xpp,1,EP);
    free_dvector(xlogp,1,EP);
    free_dvector(x_vec_g,1,2*Num_cols);
    free_dvector(df_ent_vec,1,2*Num_cols);

    end = clock();
    elapsed = ((double) (end - start)) / CLOCKS_PER_SEC;
    //printf("\n CPU run Time = %f seconds\n", elapsed);
    //printf("\n C code: grad_var = %f \n", grad_var);
    return;
}

/************* MatLab "gateway" function *************/
void mexFunction(
    int nlhs,       mxArray *plhs[],
    int nrhs, const mxArray *prhs[])
{
    // This mexFunction code based on MatLab mex-file example, "xtimesy.c"

    int num_rows, num_cols;
    double Num_rows_in, lambda, variance, Num_cols_in, EP_in, grad_variance;
    double *data_vec, *recon_vec_in, *recon_vec_out;  // pointer to inputs and output vectors

    /* Check for proper number of arguments. */
    if (nrhs != 8) {
        //printf("\n ....... nrhs = %d", nrhs);
        mexErrMsgTxt("8 inputs required.");
    }
    if (nlhs != 1) {
        //printf("\n ....... nlhs = %d", nlhs);
        mexErrMsgTxt("1 outputs required.");
    }

    /* Check to make sure first 3 arguments + last are scalars. */
    // Check for proper number of arguments. */
    if (nrhs != 8) {
        //printf("\n ....... nrhs = %d", nrhs);
        mexErrMsgTxt("8 inputs required.");
    }
    if (nlhs != 1) {
        //printf("\n ....... nlhs = %d", nlhs);
        mexErrMsgTxt("1 outputs required.");
    }
    /* Check to make sure first 3 arguments + last are scalars. */
if (!mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) ||
    mxGetN(prhs[0])*mxGetM(prhs[0]) != 1) {
    mexErrMsgTxt("Input 'Num_rows_in' must be a scalar.");
}
if (!mxIsDouble(prhs[1]) || mxIsComplex(prhs[1]) ||
    mxGetN(prhs[1])*mxGetM(prhs[1]) != 1) {
    mexErrMsgTxt("Input 'lambda' must be a scalar.");
}
if (!mxIsDouble(prhs[2]) || mxIsComplex(prhs[2]) ||
    mxGetN(prhs[2])*mxGetM(prhs[2]) != 1) {
    mexErrMsgTxt("Input 'variance' must be a scalar.");
}
if (!mxIsDouble(prhs[5]) || mxIsComplex(prhs[5]) ||
    mxGetN(prhs[5])*mxGetM(prhs[5]) != 1) {
    mexErrMsgTxt("Input 'Num_cols_in' must be a scalar.");
}
if (!mxIsDouble(prhs[6]) || mxIsComplex(prhs[6]) ||
    mxGetN(prhs[6])*mxGetM(prhs[6]) != 1) {
    mexErrMsgTxt("Input 'EP_in' must be a scalar.");
}
if (!mxIsDouble(prhs[7]) || mxIsComplex(prhs[7]) ||
    mxGetN(prhs[7])*mxGetM(prhs[7]) != 1) {
    mexErrMsgTxt("Input 'grad_variance' must be a scalar.");
}

/* Get the scalar inputs */
Num_rows_in = mxGetScalar(prhs[0]);
lambda = mxGetScalar(prhs[1]);
variance = mxGetScalar(prhs[2]);
Num_cols_in = mxGetScalar(prhs[5]);
EP_in = mxGetScalar(prhs[6]);
grad_variance = mxGetScalar(prhs[7]);

/* Create a pointer to input vector*/
data_vec = mxGetPr(prhs[3]);

/* Create a pointer to the input evec variance vector*/
recon_vec_in = mxGetPr(prhs[4]);

/* Create a pointer to the input spectrum matrix . */
//data_mx = mxGetPr(prhs[4]);

/* Get the dimensions of the matrix (or vector) input. */
num_rows = mxGetM(prhs[3]);
um_cols = mxGetN(prhs[3]);

/* Set the output pointer to the output matrix. */
plhs[0] = mxCreateDoubleMatrix(num_rows, num_cols, mxREAL);

/* Create C pointer to a copy of the output matrix. */
recon_vec_out = mxGetPr(plhs[0]);

/* Do the actual computations in a subroutine */
fmx_4pmem_mex(Num_rows_in, lambda, variance, data_vec, recon_vec_in, recon_vec_out, Num_cols_in, EP_in, grad_variance);
return;

} // void mexFunction
Appendix 5—Fixed-Point Arithmetic: Java Code

Java Code for Division and Square-root Fixed-point Routines
(Introduced in Appendix 2)

/*
 * fixNumbers_v2.java
 *
 * 31.7.09 by Roozbeh Mehrabadi (and Rod Foist)
 *
 */

package org.fixNumbers;

import ajava.lang.Math;

/* This is part of the package to define the fixNumbers class and their arithmetic
operations. */

public class fixNumbers_v2 {
    // so what is a fixNumber?

    // ....................................
    private int Intvalue;          // lets start by setting it to zero

    //private final int decimal_point = 24;   // if your decimal point changes, change it here
    private int decimal_point = 0;

    private final int Word_Length = 32;     // for Ambric, otherwise change it here

    // Note: use this function if you want to change the decimal_point_in from 24.
    public void setNumber (int val, int decimal_point_in) {
        decimal_point = decimal_point_in;
        //System.out.format("setNumber: decimal_point =in %d ,local %d \n", decimal_point,
        //decimal_point_in);

        if ( val >= 0x7FFFFFFF ) {
            //System.out.print ("Input value is too large, overflow: ");
            //System.out.println (val);
            Intvalue = val;
            return;
        }
    }

    public void fixNumbers_r1_Amb() {
        // empty number
        Intvalue = 0;
        decimal_point = 24;       // by default set it to something that makes sense e.g. Q24
        return;
    }

    public int getNumber () {
        return Intvalue;
    }
}
public void printNumber () {
    //System.out.format("%08X",Intvalue);
    return;
}

public void add( fixNumbers_v2 in) {
    // add in to Intvalue, assumes same decimal_point
    Intvalue += in.Intvalue;
    return;
}

public void subtract(fixNumbers_v2 in) {
    Intvalue -= in.Intvalue;
    return;
}

public void multiply(fixNumbers_v2 in) {
    // check on this; Must use Ambric's method (multHi/Low etc.)
    Intvalue *= in.Intvalue;
    return;
}

// the result of division will be in Intvalue
public void divide (int x, int y, int decimal_point_in) {
    decimal_point = decimal_point_in;
    int sign = (x >> 31) ^ (y >> 31);  // keep the sign to use at the end
    //x = Math.abs(x);            // standard Java
    //y = Math.abs(y);            // Ambric Java
    x = Math.addabs(x, 0);
    y = Math.addabs(y, 0);

    int rem = x, quo = 0, y_org = y;
    if (x >= y ) {
        x -= y;
        while ( x >= y ){
            x -= y; y = y << 1;  // double y
        }
    }
    for (;;) {
        if (rem >= y) {
            rem -= y; quo++;
        }
        if ( y == y_org) { break; }
        quo = quo << 1;  // double quo
        y = y >> 1;      // half y
    }
    // rem has remainder (s) and quo has the quotient (ci).
    rem = rem <<1;    // shift it left one more than y
int theR = y;    // we don't need to shift y by zero and it already went through ABS
int cf = 0;            // initialize the floating part

for (int i=0; i < decimal_point; i++) {
    cf = cf << 1;
    rem = theR;
    if (rem < 0) { rem += theR; }
    else { cf = cf | 1 ; }

    rem = rem << 1;       // s = s<< 1
}

//System.out.format("divide: decimal_point = %d\n", decimal_point);
quo = quo << decimal_point;

if (sign != 0 ) { Intvalue = - (quo | cf); }
else { Intvalue = ( quo | cf ); }

return;
} // end of divide (int, int)

// to simplyfy and take 2 ints and return an int
public int retdiv (int x, int y, int decimal_point_in) {
    //divide (x,y);
    divide (x,y, decimal_point_in);

    return Intvalue;
}

// the result of Intvalue/in.Intvalue will be returned in Intvalue
// has to check for division for zero: add it
public void mydiv ( fixNumbers_v2 in ) {

    int y = in.getNumber();
    int sign = (Intvalue >> 31) ^ (y >> 31);
    //y = Math.abs(y);
    y = Math.addabs(y, 0);
    //Intvalue = Math.abs(Intvalue);
    Intvalue = Math.addabs(Intvalue, 0);

    int rem = Intvalue, quo = 0;
    int y_org = y;

    if (Intvalue >= y ) {
        Intvalue -= y;
        while ( Intvalue >=y ){
            Intvalue -= y; y = y << 1;       // double y
        }
    }
    for (;;) {
        if (rem >= y) {
            rem -= y; quo++;
        }
        if ( y == y_org) { break; }
    }
}
quo = quo << 1;       // double quo
y = y >> 1;             // half y
}
// rem has remainder (s) and quo has the quotient (ci).
rem = rem << 1;        // shift it left one more than y
//int theR = Math.abs (y); // we don't need to shift y by zero
int theR = Math.addabs (y, 0);

int cf = 0;           // initialize the floating part
for (int i=0; i < decimal_point; i++) {
    cf = cf << 1;
    rem -= theR;
    if (rem < 0) { rem += theR; }  
    else { cf = cf | 1 ; }  

    rem = rem << 1;       // s = s<< 1
}
quo = quo << decimal_point;

if (sign != 0 ) { Intvalue = -(quo | cf); }
else { Intvalue = ( quo | cf ); }
return;

// note: x is assumed positive and hence the rest should work fine.
public void sqrt(int x)
{
    int  root, remHi, remLo, testDiv=0, count, tmp;

    if ( x < 0 ) {
        //System.out.println ("*** Negative argument given for sqrt, will return INT_MAX ***\n")
        Intvalue = 0x7FFFFFFF;
        return;
    }

    root = 0;         /* Clear root */
    remHi = 0;        /* Clear high part of partial remainder */
    remLo = x;        /* Get argument into low part of partial remainder */
    count = 27;       /* Load loop counter ; somehow it works best with 27 */
    do {
        // printf ("1= In FFracSqrt: count: %d, remHi: %8X, remLo: %8X, testDiv:%8X, root: %8X,\n",  
        //         count, remHi, remLo, testDiv, root);
        if ( remLo < 0 ) {
            tmp = remLo >> 1;
            tmp &= 0x7FFFFFFF;  // 0 sign bit to make int shift to work like unsigned shift
            remHi = (remHi<<2) | (tmp >>29);  
        } else {
            remHi = (remHi<<2) | (remLo>>30);
        }

    remLo <<= 2;                       /* get 2 bits of arg, extra bits fall out */
    root <<= 1;                       /* Get ready for the next bit in the root */
    testDiv = (root << 1) + 1;         /* Test radical */
if (remHi >= testDiv) {
    remHi -= testDiv;
    root++;
}

// printf ("2= End of loop: count: %d, remHi: %8X, remLo: %8X, testDiv:%8X, root: %8X."n",

//          count, remHi, remLo, testDiv, root);

} while (count-- != 0);

Intvalue = root;
return;
} // end of sqrt(int)

/*
public void sqrt () {
    sqrt (Intvalue);
    return;
} // end of sqrt()
*/

} // end of class fixNumbers