

Chemometrics for Ordered and Disordered Data Sets

Stephen Gerard Hughes

Submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University

Halifax, Nova Scotia

December, 1995

© Copyright by Stephen Gerard Hughes, 1995



National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services Branch

Direction des acquisitions et
des services bibliographiques

395 Wellington Street
Ottawa, Ontario
K1A 0N4

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file *Votre référence*

Our file *Notre référence*

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-612-15833-0

Canada

Name _____

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

Chemistry

SUBJECT TERM

0486

SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture 0729
 Art History 0377
 Cinema 0900
 Dance 0378
 Fine Arts 0357
 Information Science 0723
 Journalism 0391
 Library Science 0399
 Mass Communications 0708
 Music 0413
 Speech Communication 0459
 Theater 0465

EDUCATION

General 0515
 Administration 0514
 Adult and Continuing 0516
 Agricultural 0517
 Art 0273
 Bilingual and Multicultural 0282
 Business 0688
 Community College 0275
 Curriculum and Instruction 0727
 Early Childhood 0518
 Elementary 0524
 Finance 0277
 Guidance and Counseling 0519
 Health 0680
 Higher 0745
 History of 0520
 Home Economics 0278
 Industrial 0521
 Language and Literature 0279
 Mathematics 0280
 Music 0522
 Philosophy of 0998
 Physical 0523

Psychology 0525
 Reading 0535
 Religious 0527
 Sciences 0714
 Secondary 0533
 Social Sciences 0534
 Sociology of 0340
 Special 0529
 Teacher Training 0530
 Technology 0710
 Tests and Measurements 0288
 Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language
 General 0679
 Ancient 0289
 Linguistics 0290
 Modern 0291
 Literature
 General 0401
 Classical 0294
 Comparative 0295
 Medieval 0297
 Modern 0298
 African 0316
 American 0591
 Asian 0305
 Canadian (English) 0352
 Canadian (French) 0355
 English 0593
 Germanic 0311
 Latin American 0312
 Middle Eastern 0315
 Romance 0313
 Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy 0422
 Religion
 General 0318
 Biblical Studies 0321
 Clergy 0319
 History of 0320
 Philosophy of 0322
 Theology 0469

SOCIAL SCIENCES

American Studies 0323
 Anthropology
 Archaeology 0324
 Cultural 0326
 Physical 0327
 Business Administration
 General 0310
 Accounting 0272
 Banking 0770
 Management 0454
 Marketing 0338
 Canadian Studies 0385
 Economics
 General 0501
 Agricultural 0503
 Commerce-Business 0505
 Finance 0508
 History 0509
 Labor 0510
 Theory 0511
 Folklore 0358
 Geography 0366
 Gerontology 0351
 History
 General 0578

Ancient 0579
 Medieval 0581
 Modern 0582
 Black 0328
 African 0331
 Asia, Australia and Oceania 0332
 Canadian 0334
 European 0335
 Latin American 0336
 Middle Eastern 0333
 United States 0337
 History of Science 0585
 Law 0398
 Political Science
 General 0615
 International Law and Relations 0616
 Public Administration 0617
 Recreation 0814
 Social Work 0452
 Sociology
 General 0626
 Criminology and Penology 0627
 Demography 0938
 Ethnic and Racial Studies 0631
 Individual and Family Studies 0678
 Industrial and Labor Relations 0629
 Public and Social Welfare 0630
 Social Structure and Development 0700
 Theory and Methods 0344
 Transportation 0709
 Urban and Regional Planning 0999
 Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture
 General 0473
 Agronomy 0285
 Animal Culture and Nutrition 0475
 Animal Pathology 0476
 Food Science and Technology 0359
 Forestry and Wildlife 0478
 Plant Culture 0479
 Plant Pathology 0480
 Plant Physiology 0817
 Range Management 0777
 Wood Technology 0746
 Biology
 General 0306
 Anatomy 0287
 Biostatistics 0308
 Botany 0309
 Cell 0379
 Ecology 0329
 Entomology 0353
 Genetics 0369
 Limnology 0793
 Microbiology 0410
 Molecular 0307
 Neuroscience 0317
 Oceanography 0416
 Physiology 0433
 Radiation 0821
 Veterinary Science 0778
 Zoology 0472
 Biophysics
 General 0786
 Medical 0760

EARTH SCIENCES

Biogeochemistry 0425
 Geochemistry 0996

Geodesy 0370
 Geology 0372
 Geophysics 0373
 Hydrology 0388
 Mineralogy 0411
 Paleobotany 0345
 Paleocology 0426
 Paleontology 0418
 Paleozoology 0985
 Palynology 0427
 Physical Geography 0368
 Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences 0768
 Health Sciences
 General 0566
 Audiology 0300
 Chemotherapy 0992
 Dentistry 0567
 Education 0350
 Hospital Management 0769
 Human Development 0758
 Immunology 0982
 Medicine and Surgery 0564
 Mental Health 0347
 Nursing 0569
 Nutrition 0570
 Obstetrics and Gynecology 0380
 Occupational Health and Therapy 0354
 Ophthalmology 0381
 Pathology 0571
 Pharmacology 0419
 Pharmacy 0572
 Physical Therapy 0382
 Public Health 0573
 Radiology 0574
 Recreation 0575

Speech Pathology 0460
 Toxicology 0383
 Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences
 Chemistry
 General 0485
 Agricultural 0749
 Analytical 0486
 Biochemistry 0487
 Inorganic 0488
 Nuclear 0738
 Organic 0490
 Pharmaceutical 0491
 Physical 0494
 Polymer 0495
 Radiation 0754
 Mathematics 0405
 Physics
 General 0605
 Acoustics 0986
 Astronomy and Astrophysics 0606
 Atmospheric Science 0608
 Atomic 0748
 Electronics and Electricity 0607
 Elementary Particles and High Energy 0798
 Fluid and Plasma 0759
 Molecular 0609
 Nuclear 0610
 Optics 0752
 Radiation 0756
 Solid State 0611
 Statistics 0463
 Applied Sciences
 Applied Mechanics 0346
 Computer Science 0984

Engineering
 General 0537
 Aerospace 0538
 Agricultural 0539
 Automotive 0540
 Biomedical 0541
 Chemical 0542
 Civil 0543
 Electronics and Electrical 0544
 Heat and Thermodynamics 0348
 Hydraulic 0545
 Industrial 0546
 Marine 0547
 Materials Science 0794
 Mechanical 0548
 Metallurgy 0743
 Mining 0551
 Nuclear 0552
 Packaging 0549
 Petroleum 0765
 Sanitary and Municipal 0554
 System Science 0790
 Geotechnology 0428
 Operations Research 0796
 Plastics Technology 0795
 Textile Technology 0994

PSYCHOLOGY

General 0621
 Behavioral 0384
 Clinical 0622
 Developmental 0620
 Experimental 0623
 Industrial 0624
 Personality 0625
 Physiological 0989
 Psychobiology 0349
 Psychometrics 0632
 Social 0451



DALHOUSIE UNIVERSITY

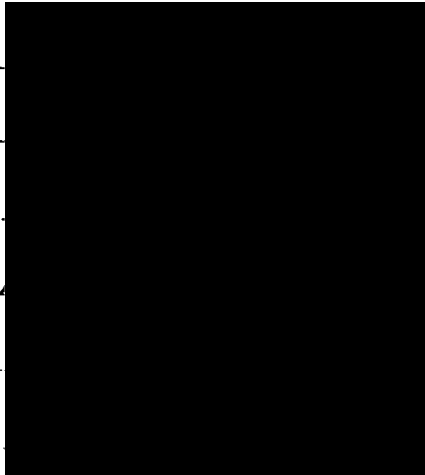
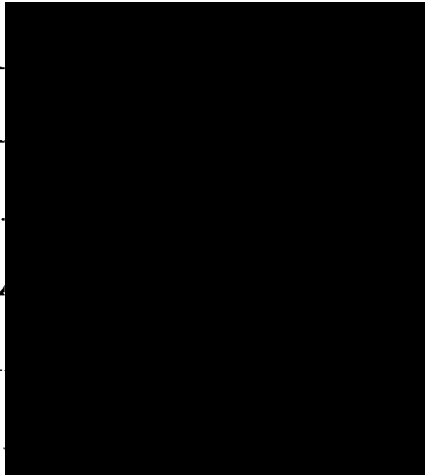
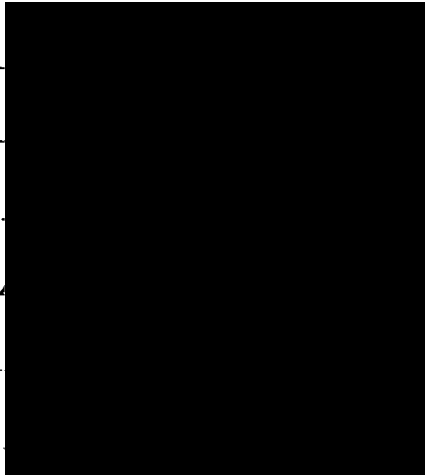
FACULTY OF GRADUATE STUDIES

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "Chemometrics for Ordered and Disordered Data Sets"

by Stephen Gerard Hughes

in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dated December 12, 1995

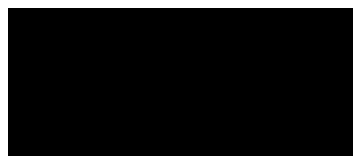
| | |
|---------------------|--|
| External Examiner |  |
| Research Supervisor |  |
| Examining Committee |  |

DALHOUSIE UNIVERSITY

December 20, 1995

Author: Stephen Gerard Hughes
Title: Chemometrics for Ordered and Disordered Data Sets
Department: Chemistry
Degree: Ph.D.
Convocation: Spring, 1996

Permission is herewith granted to Dalhousie University to circulate and have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

A solid black rectangular box redacting the author's signature.

Signature of Author

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

The author attests that permission has been obtained for the use of any copyrighted material appearing in this thesis (other than brief excerpts requiring only proper acknowledgment in scholarly writing) and that all such use is clearly acknowledged.

To my father,
J. Alfred Hughes (1928 - 1993).

Table of Contents

| | | |
|----------|--|-------|
| | Table of Contents | v |
| | List of Figures | x |
| | List of Tables | xiv |
| | Abstract | xv |
| | Notation | xvi |
| | Abbreviations and Symbols | xvii |
| | Acknowledgements | xxiii |
| 1 | Introduction | |
| | 1.1 Chemometrics | 1 |
| | 1.2 Types of Analysis | 4 |
| 2 | Ordered Sets 1: Multicomponent Systems | |
| | 2.1 Multicomponent Systems | 7 |
| | 2.2 Evolving Projection Analysis | 9 |
| | 2.3 Advanced EPA Considerations | 18 |
| | 2.4 Experimental | 22 |
| | 2.5 Results | 24 |
| | 2.6 Conclusions | 37 |
| 3 | Ordered Sets 2: Peak Purity Analysis in the Presence of Non-ideal Detector Response | |
| | 3.1 Peak Purity Analysis | 39 |

| | | |
|----------|--|----|
| 3.2 | EPA in Nonideal Systems | 39 |
| 3.3 | Nonlinearities | 44 |
| 3.4 | Heteroscedasticity | 51 |
| 3.5 | Experimental | 55 |
| 3.6 | Results and Discussion | 56 |
| 3.7 | Conclusions | 74 |
| 4 | Disordered Sets 1: Conductivity Prediction | |
| 4.1 | From Order to Disorder | 75 |
| 4.2 | Background | |
| 4.2.1 | Theoretical and Semi-Empirical Modelling Techniques | 77 |
| 4.2.2.1 | Empirical Modelling Techniques | 82 |
| 4.2.2.2 | Multiple Linear Regression | 84 |
| 4.2.2.3 | Principal Component Regression | 85 |
| 4.2.2.4 | Partial Least Squares | 86 |
| 4.2.2.5 | Continuum Regression | 87 |
| 4.2.2.6 | Neural Networks | 88 |
| 4.3 | Experimental | |
| 4.3.1 | Chemical Analysis | 91 |
| 4.3.2 | Computational Aspect | 92 |
| 4.4 | Results and Discussion | |
| 4.4.1 | Semi-Empirical Models | 95 |
| 4.4.2 | Empirical Models | 98 |

| | | |
|----------|--|-----|
| 4.5 | Conclusions | 106 |
| 5 | Disordered Sets 2: Term Selection | |
| 5.1 | Introduction | |
| 5.1.1 | Term Selection | 108 |
| 5.1.2 | Optimization Methods | 111 |
| 5.1.3 | Genetic Algorithms | 114 |
| 5.2 | Experimental | 126 |
| 5.3 | Results and Discussion | |
| 5.3.1 | Variable Number of Terms | 126 |
| 5.3.2 | Analysis of GA Results for the 40-term Model | 131 |
| 5.3.3 | Fixed Number of Terms | 140 |
| 5.3.4 | Comparison with Iterative Searching | 154 |
| 5.4 | Conclusions | 158 |
| 6 | Ordering Disordered Data Sets | |
| 6.1 | Introduction | 159 |
| 6.1.1 | The Ordering Problem | 160 |
| 6.1.2 | Problem Types | 162 |
| 6.1.3 | The Objective Function | 166 |
| 6.1.4 | Ordering Modes | 172 |
| 6.1.5 | Genetic Algorithms | 175 |
| 6.2 | Experimental | 180 |
| 6.2.1 | Integer Sorting | 181 |

| | | |
|------------|--------------------------------------|-----|
| 6.2.2 | Cluster Analysis - Simulated Data | 181 |
| 6.2.3 | Cluster Analysis - Experimental Data | 181 |
| 6.2.4 | Simulated Chromatographic Data | 182 |
| 6.2.5 | The Clock Reaction | 182 |
| 6.2.6 | Pyrocatechol Violet | 183 |
| 6.2.7 | St. Louis Data. | 184 |
| 6.2.8 | Computational Aspects | 186 |
| 6.3 | Results and Discussion | |
| 6.3.1 | Integer Sorting | 186 |
| 6.3.2 | Cluster Analysis - Simulated Data | 187 |
| 6.3.3 | Cluster Analysis - Experimental Data | 193 |
| 6.3.4 | Simulated Chromatographic Data | 195 |
| 6.3.5 | The Clock Reaction | 197 |
| 6.3.6 | Pyrocatechol Violet | 201 |
| 6.3.7 | St. Louis Data Set | 204 |
| 6.4 | Conclusions | 206 |
| 7 | Conclusions | 210 |
| | References | 213 |
| | Appendices | 220 |
| Appendix A | Program listing for multkf.m | 221 |
| Appendix B | Program listing for henosim.m | 229 |

| | | |
|------------|--|-----|
| Appendix C | Program listing for kfcorr.m | 242 |
| Appendix D | Program listing for fenemp.m | 251 |
| Appendix E | Program listing for mlrfinal.m | 266 |
| Appendix F | Program listing for crfinal.m | 269 |
| Appendix G | Program listing for gabin.m | 275 |
| Appendix H | Program listing for gaperm.m | 283 |
| Appendix I | Program listing for gaorder.m | 291 |

List of Figures

| | | |
|------|---|----|
| 1.1 | Glucose calibration curves.. | 3 |
| 2.1 | Elution profiles and A^2 plots for chromatographic mixtures. | 11 |
| 2.2 | Pictorial representation of Evolving Projection Analysis. | 14 |
| 2.3 | Spectrochromatogram of the simulated two-component system. | 16 |
| 2.4 | EPA results for a simulated two-component system. | 17 |
| 2.5 | PCA results for simulated data sets. | 19 |
| 2.6 | Spectrochromatogram, for the simulated four-component mixture. | 26 |
| 2.7 | EPA of the data in Figure 2.6. | 27 |
| 2.8 | Maximum fit error obtained for the three-component model applied to a simulated four-component mixture. | 30 |
| 2.9 | Data from the spectrophotometric titration of pyrocatechol violet. | 31 |
| 2.10 | Evolving projection analysis of the data in Figure 2.9. | 32 |
| 2.11 | Experimental spectrochromatograms for the three- and four-component mixtures. | 34 |
| 2.12 | Spectra for the four components employed in the chromatography experiments. | 35 |
| 2.13 | Results of evolving projection analysis for the data in Figure 2.12. | 36 |
| 3.1 | Effect of various experimental factors on A^2 plots. | 43 |
| 3.2 | Calibration curves predicted by the equation of Dose and Guichon. | 47 |

| | | |
|-------------|--|-----|
| 3.3 | Figure showing the influence of stray light on calibration curves. | 49 |
| 3.4 | Measured errors in absorbance for the diode array detector. | 52 |
| 3.5 | Detector response curves measured at various wavelengths for <i>p</i> -xylene showing nonlinearities. | 59 |
| 3.6 | Realistic calibration curve obtained for <i>p</i> -xylene at 278 nm. | 60 |
| 3.7 | Noise (standard deviation) characteristics of the DAD. | 62 |
| 3.8 | Spectrum of PrCl ₃ indicating the wavelengths used for the simulation. | 65 |
| 3.9 | Effect of nonidealities on EPA for simulated data. | 66 |
| 3.10 | Effect of a two-component mixture on various EPA models. | 69 |
| 3.11 | Effect of nonidealities on EPA for experimental data (3% PrCl ₃). | 71 |
| 3.12 | Illustration of limitations of EPA. | 73 |
| 4.1 | Neural Network topology. | 89 |
| 4.2 | Comparison of measured and predicted conductivity for data using Rossum model. | 96 |
| 4.3 | Comparison of measured and predicted conductivities using EMRM. | 99 |
| 4.4 | Comparison of measured and predicted conductivities using MLR models. | 101 |
| 4.5 | Prediction error surface for continuum regression. | 103 |
| 4.6 | Variation of calibration and prediction errors during typical training of neural networks with number of hidden nodes. | 105 |
| 5.1 | Graphical representation of optimization. | 112 |
| 5.2 | String representation of the types of problems a GA can handle and a graphical representation of a population. | 117 |

| | | |
|-------------|---|-----|
| 5.3 | A flow chart of a typical GA. | 123 |
| 5.4 | Convergence of a typical GA run for the 20 term case. | 128 |
| 5.5 | SEP versus generation using the 40 term model. | 130 |
| 5.6 | Characteristics of representative solution spaces. | 133 |
| 5.7 | Characteristics of 40 term model. | 134 |
| 5.8 | Two examples to show how autocorrelation functions work. | 136 |
| 5.9 | Autocorrelation function for the search space in the two conductivity regions employed. | 138 |
| 5.10 | GA convergence for the simulated set. | 141 |
| 5.11 | New mutation operator. | 143 |
| 5.12 | GA convergence for models of fixed size n in the two conductivity regions. | 145 |
| 5.13 | Plot of SEP versus number of terms for the two conductivity regions. | 146 |
| 5.14 | Comparison of search landscapes for the 10 term case. | 148 |
| 5.15 | MDI as function number of terms for the fixed size subset selection for the 10 term case. | 151 |
| 5.16 | Comparison of search landscapes for the 40 term case. | 153 |
| 5.17 | Comparison of the iterative and GA term selection solutions. | 156 |
| 6.1 | A pictorial representation of the ordering of samples. | 161 |
| 6.2 | Diagrams illustrating two common cluster analysis techniques. | 163 |
| 6.3 | Illustration of the <i>double sum</i> discrimination ability for disordered and ordered arrangements. | 167 |
| 6.4 | The clustering example to illustrate the new objective function. . . | 169 |

| | | |
|-------------|---|-----|
| 6.5 | An outline of the new objective function. | 170 |
| 6.6 | Ordering modes. | 174 |
| 6.7 | Flow system for pyrocatechol violet study. | 185 |
| 6.8 | A typical run for the integer example. | 188 |
| 6.9 | Illustration of how the GA convergence time and the number of solutions depend on n . for different values of n | 189 |
| 6.10 | GA results for the simulated set. | 191 |
| 6.11 | Connection diagram for the simulated set. | 192 |
| 6.12 | Classification of mushroom data based on PCA and connection diagram for the mushroom data. | 194 |
| 6.13 | Dendrogram of the mushroom data. | 196 |
| 6.14 | Results of GA ordering of the simulated chromatographic data. | 198 |
| 6.15 | Spectra obtained from the clock reaction. | 199 |
| 6.16 | Results of GA sequencing of clock reaction as illustrated by the absorbance at 444 nm. | 200 |
| 6.17 | Spectra obtained from the randomized pyrocatechol violet data. | 202 |
| 6.18 | Results of GA ordering of the pyrocatechol violet data set illustrated using the corresponding pH values. | 203 |
| 6.19 | Elemental profiles at the receptor site as a function of time. | 205 |
| 6.20 | Results of GA ordering of the St. Louis data set. | 207 |

List of Tables

| | | |
|-------------|---|-----|
| 4.1. | List of symbols and units related to conductivity measurements | 80 |
| 4.2. | Statistical summary of water quality parameters. | 93 |
| 4.3. | Summary of modified Rossum models. | 97 |
| 4.4. | Standard errors of prediction for models investigated. | 97 |
| 5.1 | The bit-term assignments for the term selection problem given by their bit location and chemical symbol. | 119 |
| 5.2 | A simple example of the number of possible combinations | 149 |
| 5.3 | Comparison of the number of possible evaluations for iterative and GA. | 155 |
| 5.4 | Best models found by the iterative method and the GA for the all samples region. | 157 |
| 6.1 | Summary of the GA configuration used for the integer example. | 180 |
| 6.2 | The six sets analyzed by the GA sequencing method. | 208 |
| 6.3 | Summary of windowing GA parameters for the considered in the six sets considered in Table 6.2. | 209 |

Abstract

Solutions to modern problems in analytical chemistry often require the use of multivariate data sets in which the measurements for individual samples consist of a vector or matrix. While present day instruments make the acquisition of such data relatively simple, the challenge has become presenting it in an informative and logical manner. Chemometrics is the branch of analytical chemistry that addresses this challenge.

When treating multivariate data sets, the method of data analysis depends greatly on the underlying structure of the data set. It is found, for example, that the presence of "order" in data sets can greatly simplify analysis. An ordered data set is one in which the component contributions change systematically in accordance with some underlying ordinal variable (e.g. time, pH). In this work, a method for the analysis of such ordered data sets, called evolving projection analysis (EPA), is extended to mixtures of more than two-components and to systems with nonideal detector response. Applications to liquid chromatography and spectrophotometric titrations are considered.

In contrast, the sequence of samples in disordered data sets is essentially random. A number of standard methods have been devised for these kinds of data sets and this work considers an interesting case involving conductivity prediction of non-brine water samples using a variety of calibration methods. Improvements to these methods and the limitations of this approach are discussed.

Finally, methods were sought to search for order in disordered data sets, working from the hypothesis that many data sets that appear disordered are actually ordered, if only the proper ordinal variable could be identified. It was found that the ordering algorithm, implemented using the Genetic Algorithms could be successfully applied to problems involving cluster analysis and evolving data sets. A novel study of environmental receptor modelling data showed that this approach can reveal unique information hidden in the disordered data set.

Notation

The discussions that follow will obey these conventions:

x is a scalar value (i.e. a number) represented by a lower case letter, except for established symbols, like A for absorbance, listed on the following page.

$\mathbf{x} = [x_1 \ x_2 \ x_3]$ is a vector, represented by a bold lower case letter.

\mathbf{X} is a matrix of dimensions $m \times n$ as shown below and is represented by a bold upper case letter.

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & - & - & x_{1n} \\ x_{21} & - & - & - & - \\ - & - & - & - & - \\ - & - & - & - & - \\ x_{m1} & - & - & - & x_{mn} \end{bmatrix}$$

The superscript p represents a predicted quantity, e.g. x^p .

The superscript $^{-1}$ indicates the inverse of a matrix, e.g. \mathbf{X}^{-1} .

The superscript T indicates the transposed matrix or vector, e.g. \mathbf{X}^T .

$$\mathbf{X}^T = \begin{bmatrix} x_{11} & x_{21} & - & - & x_{m1} \\ x_{12} & - & - & - & - \\ - & - & - & - & - \\ - & - & - & - & - \\ x_{1n} & - & - & - & x_{nm} \end{bmatrix}$$

Abbreviations and Symbols

| | |
|-----------------|---|
| A | absorbance |
| A_i | absorbance at wavelength i |
| A_{ideal} | ideal absorbance |
| $A_{observed}$ | observed absorbance |
| A_{stray} | calculated absorbance with stray light incorporated |
| A_i^p | predicted absorbance at wavelength i |
| A/D | analog to digital |
| ANN | Artificial Neural Networks |
| AU | absorbance units |
| a_1 | local spectral first derivative (concentration dependent) |
| b | sample pathlength |
| b_i | model coefficient |
| b | vector of model coefficients |
| c_j, c_+, c_- | equivalent concentration if an ion, cation and anion respectively |
| C | concentration |
| C | matrix of concentrations |
| C_s | child string |
| CAPS | C3-cyclohexylamino-1-propane sulfonic acid |
| CR | Continuum Regression |
| d_{ij} | distance between sample i and sample j |
| D | dielectric constant |

| | |
|--------------------------|---|
| e | proton charge |
| EFA | Evolving Factor Analysis |
| EPA | Evolving Projection Analysis |
| EMRM | Extended Modified Rossum Model |
| ERM | Extended Rossum Model |
| f | fitness |
| F | Faraday constant |
| FE | fit error |
| FWEFA | Fixed Window Evolving Factor Analysis |
| FT-IR | Fourier Transform - Infra-Red spectroscopy |
| G | measured conductivity |
| $G^{\circ+}, G^{\circ-}$ | ionic conductivities in a many electrolyte solution for all cations and anions respectively |
| G^p | predicted conductivity |
| GA | Genetic Algorithms |
| GC/MS | gas chromatography/mass spectrometry |
| H | objective function for ordering problem (variance method) |
| H' | objective function for ordering problem (standard deviation method) |
| HCA | Hierarchical Clustering Analysis |
| HELP | Heuristic Evolving Latent Projections |
| HPLC/DAD | high performance liquid chromatography /diode-array detection |
| I_d | dark current intensity |

| | |
|--------------------|---|
| I_r | intensity of reference |
| I_s | intensity of sample |
| I_{stray} | intensity of stray light |
| ITTFA | Iterative Target Transformation Factor Analysis |
| k | number of parameters |
| K | natural logarithm of 10 $\ln(10)$ |
| LC | liquid chromatography |
| m_{cal} | number of calibration samples |
| m_{pred} | number of prediction samples |
| MLR | Multiple Linear Regression |
| MRM | Modified Rossum Model |
| MDI | Model Distinction Indicator |
| o | desired output |
| pH | $-\log[H^+]$ |
| P_1 | first parent string |
| P_2 | second parent string |
| PCA | Principal Component Analysis |
| PC_1 | first principal component |
| PC_2 | second principal component |
| PCR | Principal Component Regression |
| PE | prediction error |
| PLS | Partial Least Squares |

| | |
|--------------------|--|
| r_s | fraction of stray light |
| r | autocorrelation function |
| R | gas constant |
| R_s | spectral resolution |
| RMS | root mean square |
| RMS(PE) | root mean square of prediction error |
| RMS(FE) | root mean square of fit error |
| s_A | standard deviation in absorbance |
| s_{A1} | standard deviation in absorbance at wavelength 1 |
| s_{A2} | standard deviation in absorbance at wavelength 2 |
| s_{A1}^2 | variance in absorbance at wavelength 1 |
| s_{A2}^2 | variance in absorbance at wavelength 2 |
| s_b | standard deviation in baseline |
| s_{eff} | effective standard deviation |
| s_{eff}^2 | effective variance |
| s_s | standard deviation in sample |
| SEC | standard error of calibration |
| SEP | standard error of prediction |
| SFE | scaled fit error |
| SMCR | Self-Modelling Curve Resolution |
| T | transmittance |
| T_i | scores on the i th principal component |

| | |
|--|---|
| uv | ultraviolet |
| w | weight |
| z_+, z_- | ionic charge of cations and anions respectively |
| Z_+, Z_- | effective charge in many electrolyte solution for cations and anions respectively |
| Δ_d | bandpass of the detector |
| ϵ_0 | molar absorptivity for the analyte/vacuum permittivity |
| ϵ_1 | local spectral first derivative (concentration independent) |
| η | solution viscosity |
| $\lambda^{\circ}_+, \lambda^{\circ}_-$ | ionic equivalent conductance at infinite dilution for cations and anions respectively |
| λ'_+, λ'_- | limiting ionic equivalent conductance for cations and anions respectively |

Acknowledgements

This research would not have been possible without the support from many individuals. My most sincere appreciation goes to my supervisor Dr. Peter D. Wentzell. His support and patience kept me motivated to complete this work. I would also like to thank past and present members of the research group who provided support over the three years: Dr. Stephen J. Vanslyke, Dr. Carlos B. Lucasius, Deborah Ford, Beth Taylor, Tara Paton, Darren Andrews, Mitch Lohnes and Stephanie Mehlman.

I would like to thank other members of the chemistry department that provided assistance, Dr. Robert D. Guy, Dr. Louis Ramaley, Dr. J. C. T. Kwak and Dr. Roger Stephens. Also other students who made working at Dalhousie more enjoyable, Dr. Pierre Losier, Dr. Kevin Thurbide, Dr. Allison Perrot, Dr. Bogumila Lysenko and Peter Delijser.

The work was not possible without the financial support from the Natural Science and Engineering Research Council. In addition I would like to thank Dalhousie University and Sumner Foundation for their support as well.

I would to thank my family for their continuing support. Finally and most importantly I thank Heather. Without her support, especially over the last few months, I would have never have finished.

1 Introduction

1.1 Chemometrics

To validate a hypothesis one needs to gather supporting evidence. This evidence can take many forms, for example questionnaires, visual images, or instrumental measurements. In chemistry, probably the most common form of evidence is instrumental measurements. Once the measurements have been collected, the challenge is to obtain the necessary information from these measurements. This thesis examines methods for extracting information from multivariate data sets in chemistry, although many of the techniques described can be extended to other disciplines as well. The branch of chemistry that deals with the extraction of information from chemical data is known as chemometrics [1-4]. Massart defines chemometrics as the application of mathematical, statistical, and formal logic to provide the maximum relevant information from chemical data [1].

To illustrate how chemometrics can be used to enhance information from chemical measurements, a simple example will be considered. The objective in this case is to reliably monitor glucose levels in blood plasma using infra-red spectroscopy [5-8]. Although spectrophotometric measurements of glucose in the infrared region of the spectrum are possible, such measurements at physiological levels are hampered by low sensitivity and relatively high noise. This is a consequence of the short pathlength resulting from the use of attenuated total

reflectance (ATR) techniques. Such techniques are necessary for aqueous samples such as blood plasma because of high background absorbance, and measurements are further complicated by the presence of numerous interferences. As a result, quantitative measurements based on absorbance at a single wavelength are generally unreliable. This is illustrated in Figure 1.1a. Here, glucose is determined through a calibration step which uses the wavelength of maximum absorbance, 1038 cm^{-1} . To generate Figure 1.1a, 26 samples of known concentration were employed. Of these samples 25 were used for calibration, and the concentration of the remaining sample was predicted using the calibration model. This process was repeated 25 times, each time leaving a different sample out for prediction. The procedure is known as cross-validation and is one of the most reliable methods for evaluating a model's predictive ability. The results in Figure 1.1a show the predicted versus actual concentrations of the cross-validation samples. Perfect agreement is represented by the line with unity slope. It is clear from the figure that there is a high degree of uncertainty associated with this approach.

In contrast to measurements made at one wavelength (*i.e.* univariate calibration), multiple wavelengths can be used to improve the predictive ability of the model (*i.e.* multivariate calibration). A multivariate calibration procedure called principal components regression (PCR) was applied to the data in this case. PCR and related techniques are discussed in more detail in Chapter 4. The effect of PCR is analogous to weighted averaging - information from all wavelengths is

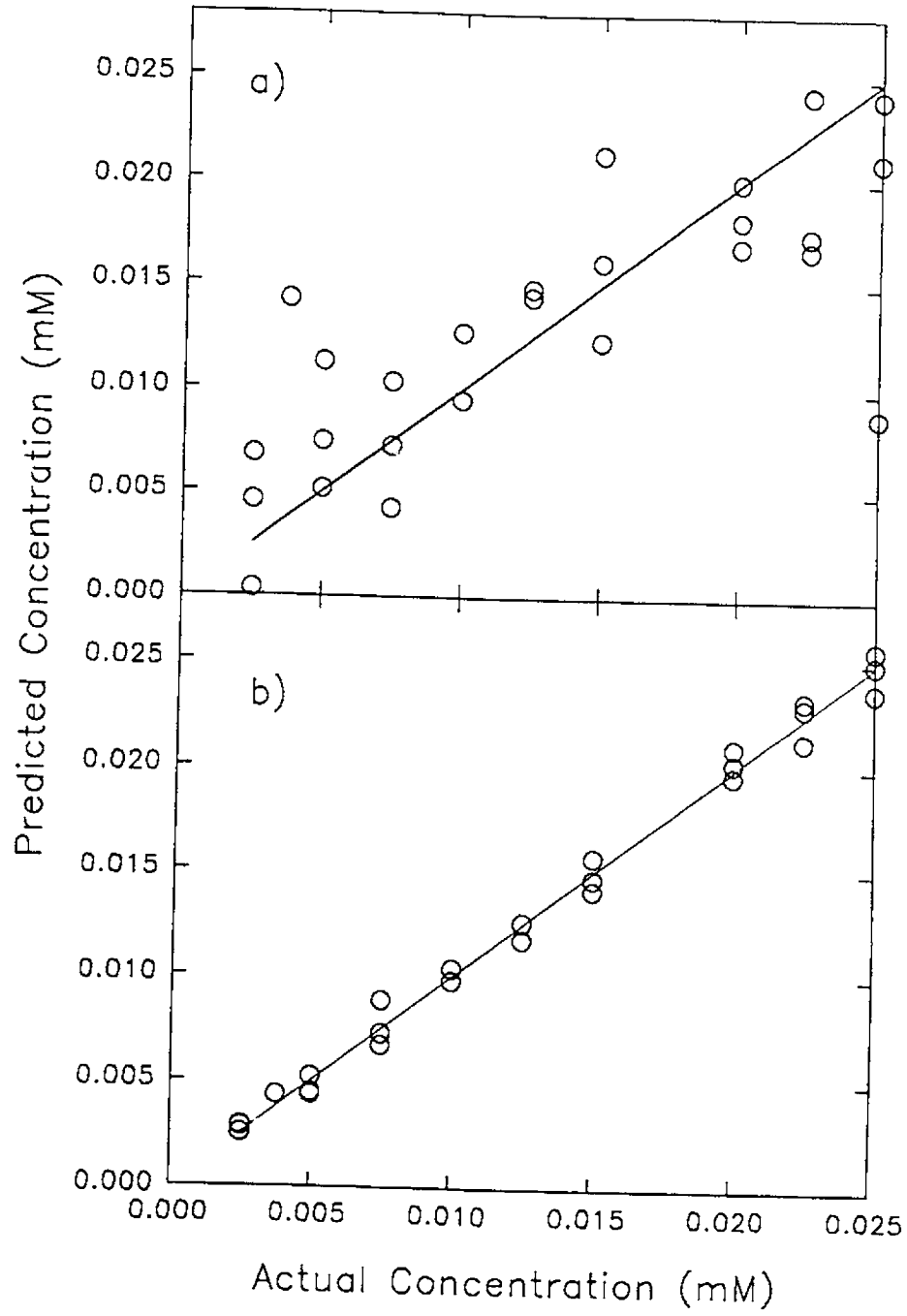


Figure 1.1 a) Glucose calibration using one wavelength. b) Glucose calibration using many wavelengths.

used to reduce the overall noise in the model. The results of PCR are shown in Figure 1.1b. Note that the predictive ability of PCR is greatly improved over that of the univariate model. This is just one illustration of how chemometric techniques can be used to improve the information yield of analytical measurements.

1.2 Types of Analysis

Measurements made on chemical systems can be described as zero-order (one measurement per sample), first-order (a vector of measurements per sample, such as a spectrum) or second-order (a matrix of measurements per sample). All of the systems studied in this work involve multiple measurements per sample. The measurements or data can be arranged into a matrix, \mathbf{X} , of m samples and n measurements, i.e.,

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & - & - & x_{1n} \\ x_{21} & - & - & - & - \\ - & - & - & - & - \\ - & - & - & - & - \\ x_{m1} & - & - & - & x_{mn} \end{bmatrix} \quad (1.1)$$

The rows of this matrix represent different samples and the columns represent measurements of different variables. For the glucose study described above, the rows are samples at different concentrations of glucose and the columns are absorbance measurements at different wavelengths.

In applying chemometrics to a data set, the *structure* of the data set is

important, since it may influence the method of analysis used. In this work, two different kinds of structure are considered: *ordered* and *disordered*. An ordered data set is one in which the component contributions change systematically in accordance with some underlying ordinal variable, such as time. In this case, the ordered structure of the data set can be exploited to yield information that might otherwise be difficult to extract. A disordered data set, on the other hand, is one in which there is no apparent relationship among the samples, or at least none which can be used to advantage.

Chapters 2 and 3 of this thesis deal with some well-known cases of ordered data sets in chemistry. The case of chromatography with first-order (*i.e.* a diode array spectrometer) detection is considered most extensively. In this case the ordering variable is time, since mixture components have a defined elution order. Spectrophotometric titrations, in which the ordering variable is pH, are also considered, and extensions to other systems, such as kinetic studies, are also possible. All of these systems have been examined in the literature by methods which take advantage of their order. In this work, the focus is on a technique called evolving projection analysis (EPA) which seeks to determine the number of components in unresolved mixtures. This method has been previously applied to mixtures of two components, and Chapter 2 considers the extension of the method to mixtures of more than two components. In Chapter 3, modifications to the EPA algorithm necessary to adapt it to practical situations involving nonideal detector response are examined. Specifically, it is shown that the method can be modified

to deal with nonlinear detector responses and non-uniform (heteroscedastic) noise, situations which are especially problematic for other methods.

Disordered data sets are more common in chemistry than ordered data sets, and Chapters 4 and 5 consider a particular problem of this type. The problem considered is the prediction of conductivity in non-brine water samples based on the concentrations of ten predominant ions in natural waters, as well as other water quality parameters. This problem is of practical importance for quality assurance in analytical laboratories and more generally for the prediction of conductivity. In Chapter 5, improvements to multivariate modelling using term selection via genetic algorithms (GAs) are investigated. In the course of this investigation, some general observations are made on the applicability of GAs to term selection problems.

Chapter 6 considers an important question thus far ignored in the literature involving the relationships between ordered and disordered data sets. It is postulated that some disordered data sets are not inherently disordered, but are simply ordered data sets for which an ordinal variable has not been found. It is further postulated that ordering of these disordered data sets may reveal important information that is not otherwise apparent. A method for ordering based on GAs is described and used to reveal hidden information of practical importance in widely studied data sets.

In the final chapter, a brief discussion summarizing the conclusions of this work is presented.

Ordered Sets 1: Multicomponent Systems

2.1 Multicomponent Systems

Determining the number, identity and concentration of chemical components in a mixture is a difficult problem in analytical chemistry. Often one tries to physically separate the mixture using methods such as chromatography. In many cases the separation is not complete and one or more partially overlapped peaks are present. Mathematical separation of these overlapped peaks may be possible, but before the identity and concentration of each component can be established, the number of components in the mixture must be determined. This chapter will deal with the determination of the number of components in multicomponent mixtures. The methodology presented is applicable to ordered data sets, such as those found in chromatography, spectrophotometric titrations and kinetics. The discussion that follows uses chromatography as an example, but the arguments can be extended to the other cases as well, such as the study of equilibria and kinetics.

In 1982, Rosenthal [15] showed that the occurrence of overlapping components is more common than previously thought. Although manufacturers of liquid chromatography columns give the number of theoretical plates as approximately a couple of thousand per meter, and this translates into a possible separation of few hundred peaks, this separation represents a maximum possible

separation rather than an experimental figure of merit. Based on statistical analysis Davis and Giddings [16] showed that, for the separation of a 50 component mixture and on a column with a possible separation of 100 components, total separation was not attainable in most cases. On average there would be 18 one-component peaks, 7 two-component peaks, 3 three-component peaks, 1 four-component peak and 1 peak with five or more components. Of course, in many analytical problems there may be fewer than 50 components and the chances of multicomponent peaks would diminish accordingly.

Much of the past work on the mathematical separation of unresolved peaks has focussed on determining the identity and concentration profiles of components in a mixture. In 1971, Lawton and Sylvestre [17] developed a technique for separation of two-component systems called self-modelling curve resolution (SMCR). SMCR has been successfully applied to gas chromatography-mass spectrometry (GC/MS) [18] and high performance liquid chromatography-diode array detection (HPLC/DAD) [19]. Extending SMCR to three-component systems is difficult and authors have had only limited success [20-22]. Other techniques such as iterative target transformation factor analysis (ITTFA) [23-25], have also been applied to the problem with some success, but again have difficulties as the number of components is increased. The success of such methods can be greatly improved with the availability of prior knowledge regarding: (1) the number of components present, and (2) the nature of the elution profiles, especially the location of pure component regions which can be used to obtain individual spectra.

A number of methods have been developed for this purpose, including evolving factor analysis (EFA) [26-31], fixed-window evolving factor analysis (FWEFA) [32-36] and heuristic evolving latent projections (HELP) [37-40].

Another mathematical separation technique that attempts to determine the number of components in ordered sets and the characteristics of their concentration profile was previously developed in this laboratory. This technique, called *evolving projection analysis* (EPA), has been applied to one- and two-component systems [32]. In the work presented here, the adaptation of the algorithm to systems of more than two components was developed and tested. Before describing the extension of EPA to systems of more than two components, a brief overview of the algorithm will be presented for one and two component chromatographic mixtures.

2.2 Evolving Projection Analysis

Fundamentally EPA examines the responses observed for a chemical system to determine if it can be modelled by a given number of components. To do this, the algorithm models data based on expected component behaviour, and monitors the fit error or prediction error to determine the number of components present. In this work the technique will be applied to a multiwavelength absorbance detector, but it could also be applied to other first-order detectors such as a mass spectrometer. In the case of multiwavelength spectrometry, absorbance measurements at various wavelengths are used as response

measurements. In order to see how one can model various systems it is instructive to consider plots of absorbance at one wavelength against absorbance at another wavelength. These plots, which are referred to as A^2 plots, are critical to EPA. Parts b and d of Figure 2.1 show typical A^2 plots for one- and two-component chromatographic systems. A_1 and A_2 are the absorbance values at two wavelengths and the numerals represent various stages of a separation. It is important to note that, for a one-component system, if Beer's Law holds, the shape of the spectrum is not important since the spectrum will only change the slope of the line not its linearity. It is also important to note how various stages of the separation correspond to points on the A^2 plots (See Figure 2.1). The plot in Figure 2.1b could be modelled by:

$$A_2^P = \alpha A_1 \quad (2.1)$$

where A_1 represents the absorbance at wavelength 1, α is a parameter to be determined and A_2^P is the absorbance predicted at wavelength 2. Although the above equation will be valid for the one-component system, it obviously wouldn't hold for the two-component system if the two components have different spectral characteristics at the two wavelengths (See Figure 2.1d). However the equation that would fit this case is:

$$A_3^P = \alpha A_1 + \beta A_2 \quad (2.2)$$

where A_1 and A_2 are the absorbances at wavelengths 1 and 2, α and β are parameters to be determined, and A_3^P is the absorbance predicted at wavelength

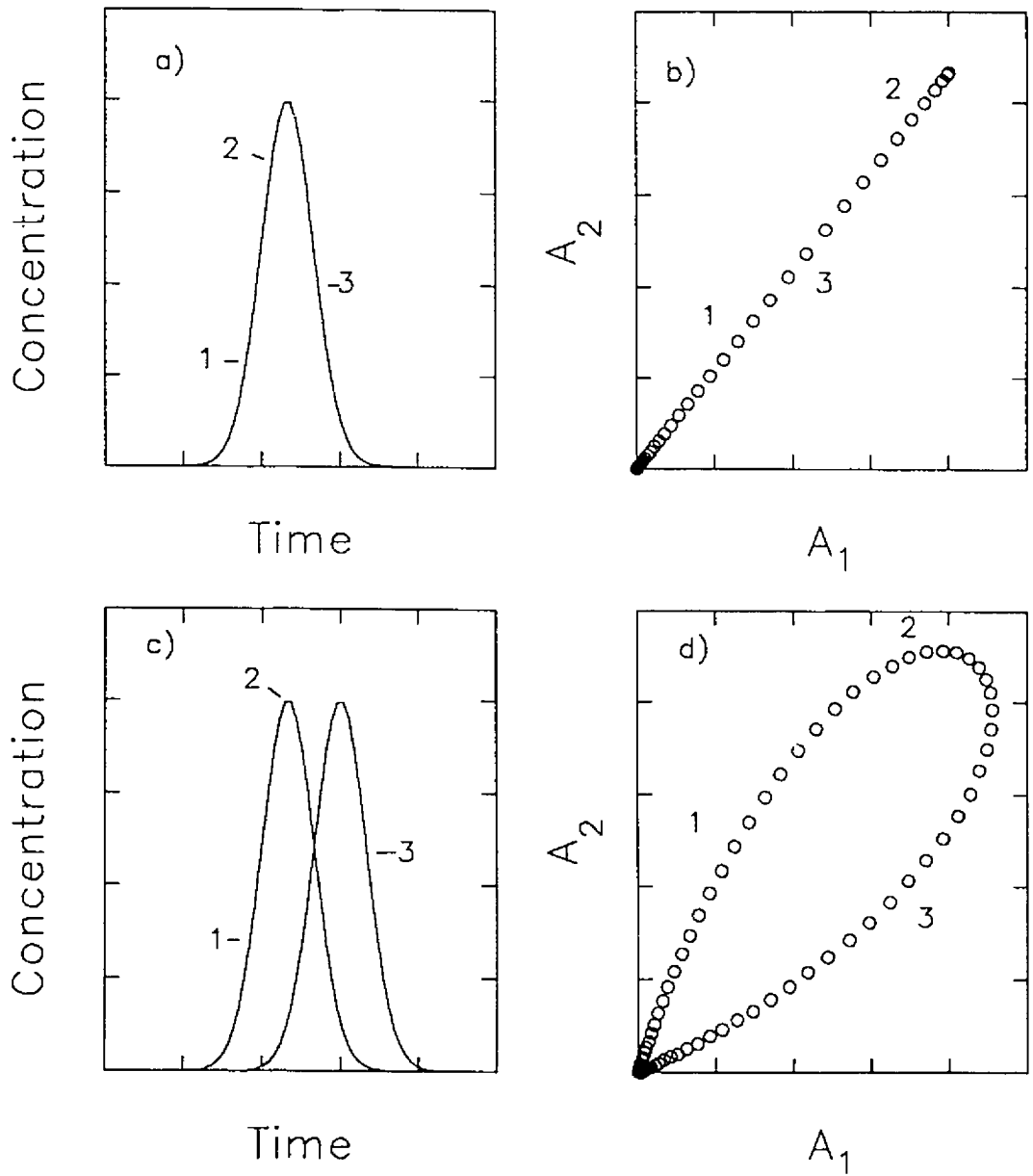


Figure 2.1 Elution profiles for a) the one-component and c) two-component chromatographic mixtures, and their corresponding A^2 plots, b) and d).

3. EPA models the data according to equation 2.1 and 2.2 and monitors the quality of the fit to assess the rank of the chemical system. In essence, EPA exploits the fact that a one-component system can be modelled, within experimental error, by a straight line (a one-dimensional model) in any A^2 space. Likewise, a two-component system can be described by a planar equation (a two-dimensional model) in any A^3 space, but is likely to be poorly approximated by a linear model in at least some A^2 spaces. This argument can be extended to higher dimensions with the use of hyper-planar models.

For convenience, EPA uses an algorithm called the Kalman filter to model the system. The Kalman filter was developed in the communications industry some 30 years ago [42], but has been used only recently in chemistry [43-46]. The Kalman filter can be considered to be an algorithm for the recursive estimation of parameters from a series of noisy measurements in accordance with a linear system model [47]. For the one-component model, for example, the Kalman filter provides updates of α as each spectrum is acquired. In the usual implementation of the Kalman filter, the validity of the model is evaluated using the prediction error (PE), or innovation. The prediction error is,

$$\begin{aligned} (PE)_k &= (A_2)_k - (A_2^A)_k \\ &= (A_2)_k - \alpha_{k-1} (A_1)_k \end{aligned} \quad (2.3)$$

This is the difference between the value of A_2 measured for a given time (k) and the value predicted by the model before that measurement is incorporated in the

model. In this work, for reasons that will be given in Chapter 3, it was found to be more effective to use the fit error (FE), which is the difference between the measured value A_2 and the predicted value after incorporating the current measurement. The fit error is,

$$\begin{aligned} (FE)_k &= (A_2)_k - (A_2^p)_k \\ &= (A_2)_k - \alpha_k (A_1)_k \end{aligned} \quad (2.4)$$

In either case, note that, unlike least squares regression which uses all of the points to evaluate model validity, this approach only considers points up to and including the most recent one, and is therefore sensitive to instantaneous model deviations.

Figure 2.2 shows how EPA would work for a one-component model applied to a one-component system. Consider a data set of n responses measured m different times. EPA analyzes the data as each time slice is acquired. Initially only noise is acquired and the EPA program is only activated when the signal is greater above the baseline (in this work, a level six times greater than the baseline standard deviation was found to work well). Then an independent wavelength must be selected and an obvious choice is the wavelength of maximum response which will be designated as A_1 . For the one-component model and n responses, $n-1$ models are monitored (e.g. A_2 vs A_1 , A_3 vs A_1 , etc). The fit error for each of these models is combined in a root mean square sum; that is for the i th time slice the root mean square (RMS(FE)) would be:

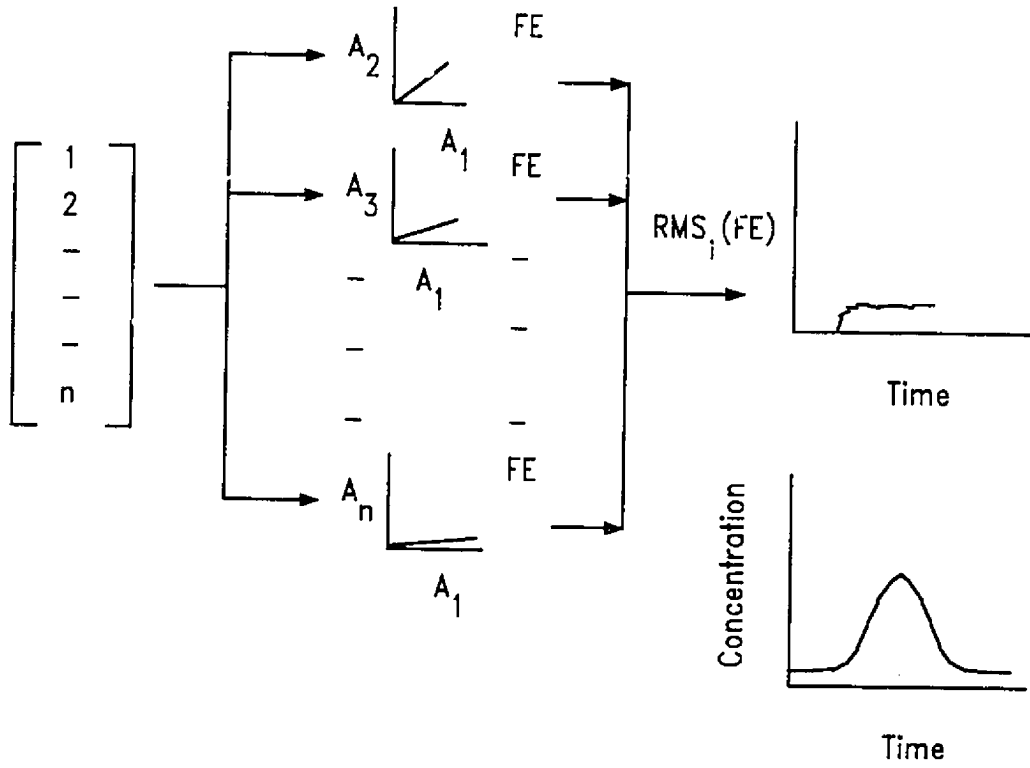


Figure 2.2 Pictorial representation of Evolving Projection Analysis showing the application of parallel one-component models to one-component systems.

$$RMS(FE) = \sqrt{\frac{\sum_{j=1}^n (FE)_j^2}{n-1}} \quad (2.5)$$

Note that the trace of RMS(FE) in Figure 2.2 remains essentially flat throughout the chromatogram since it reflects only model deviations due to experimental noise. Such a flat trace indicates that there is only one observable in the mixture.

Typically more than one type of model is analyzed by EPA to determine the number of components present. Figure 2.3 shows the spectrochromatogram for a simulated two component mixture. In this case at least two types of models need to be monitored, namely the one- and two-component models. The top panel of Figure 2.4 shows the noise-free concentration profiles for a two-component chromatographic system and the bottom panel shows the results using EPA for this system. The analysis by EPA starts from the left side and moves along the data set until it reaches the right side. As each time slice becomes available, EPA updates model estimates using data from this new slice and all the previously collected data. One sees that in the early stages of separation, only one component is present and both the one- and two-component models fit the data. This is because both linear and planar models can adequately represent a single component. The RMS(FE) shows only random fluctuations and reflects the magnitude of the experimental noise. However, when the second component begins to elute, a systematic deviation occurs indicating the failure of the one-component model, but the two-component model continues to hold. The two-

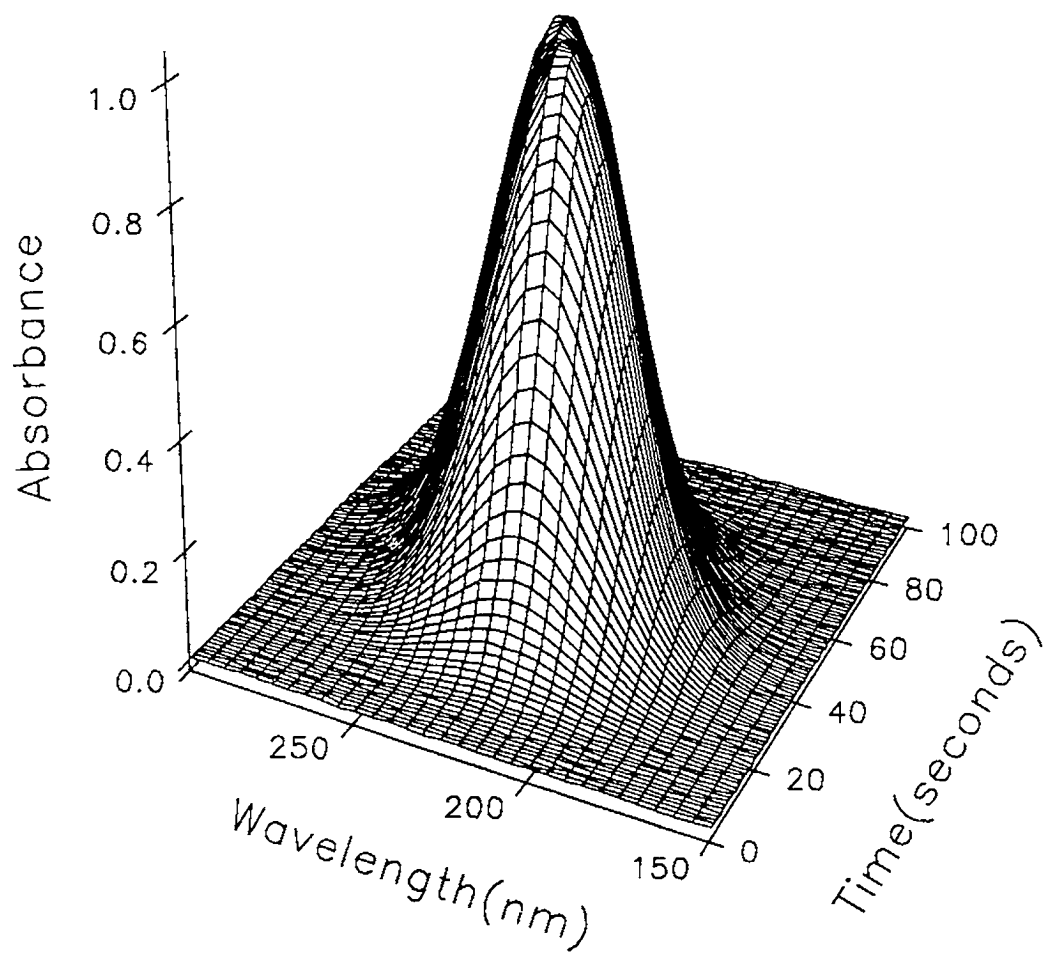


Figure 2.3 Spectrochromatogram of the simulated two-component system.

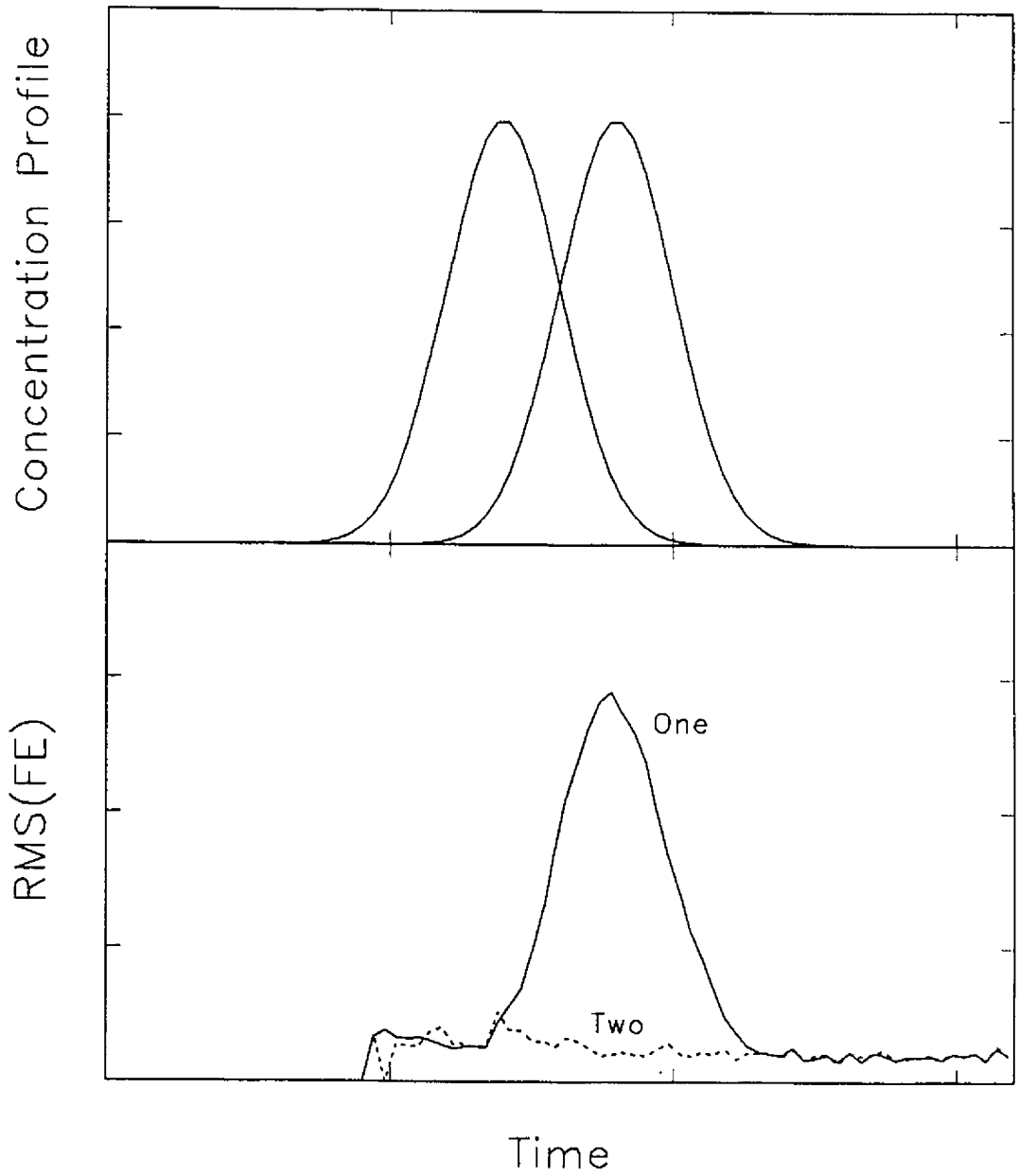


Figure 2.4 a) Concentration profiles for a simulated two-component system. b) Error sequences for one-component and a two-component model.

component model is valid throughout the separation since the system contains only two components. By analyzing the failure of models of progressively increasing dimensionality, EPA allows the number of components present in the mixture to be determined and can help to identify where each component begins to elute. Note that EPA can also be applied in the reverse direction (right to left) to give complementary information, although this can't be done in real time.

2.3 Advanced EPA Considerations

Many of the methods mentioned previously (*i.e.* SMCR, ITTFA, FWEFA and EFA) were based on a technique called principal component analysis (PCA). EPA is very similar to PCA, but with some important differences. Both EPA and PCA estimate the number of components by observing how well an n -component model describes a multicomponent system. However EPA and PCA differ in how they model the data. EPA models the data using equations 2.1 and 2.2 and similar equations for more than 2 components, whereas PCA models the data by generating a set of orthogonal basis vectors. PCA determines the number of components by the number of these vectors it needs to describe the data within experimental error. How PCA selects these vectors is described below.

Figure 2.5 shows A^2 plots for the one- and two-component systems. In PCA, the first principal component (PC_1) is chosen to pass through the points in the direction with the greatest variability in A_1 and A_2 . The second principal component (PC_2) is chosen to be at right angles to PC_1 and to account for the

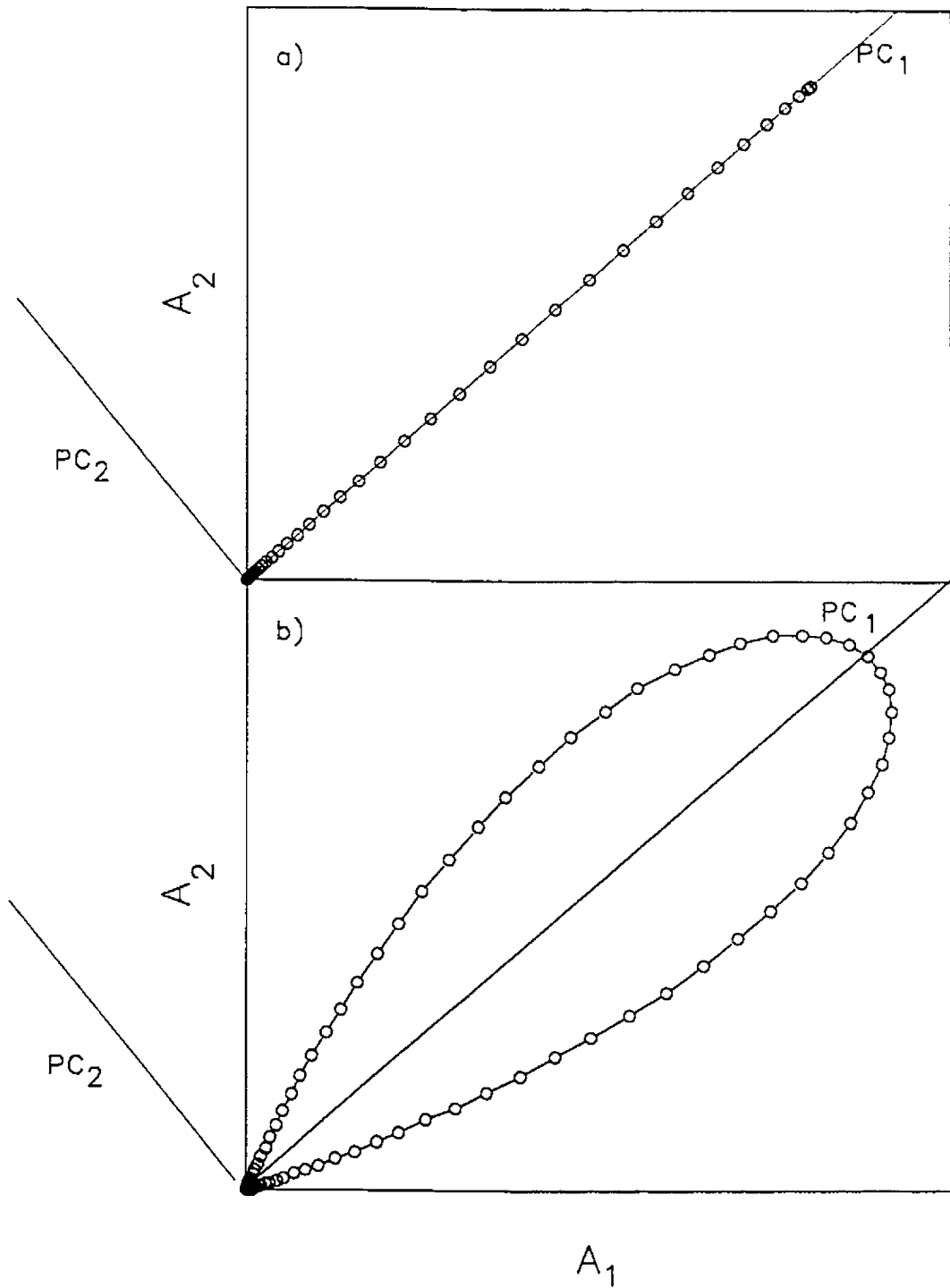


Figure 2.5 Results of PCA for simulated a) one-component and b) two-component data sets.

remaining variability. Note that in the A^2 space, PC_2 is fixed by the choice of PC_1 , but this will not be true in the more general case of spaces of higher dimensionality (A^n). Since PC_2 is essentially modelling random noise in Figure 2.5a, there is effectively only one component, or the data set is rank 1. In Figure 2.5b, however, both PC_1 and PC_2 would be needed to describe the data within experimental error, indicating that this data set has a rank of two. In general the number of PC's needed to describe the data is indicative of the number of observable components. In both of these cases EPA and PCA would give similar results, but EPA has certain advantages. First EPA is an evolutionary method, meaning that it exploits the ordering of the data set to provide an indication of where the rank changes. Although PCA can be applied in the same fashion, as is done with EFA, the least-squares formulation of EPA and its recursive implementation with the Kalman filter makes it more efficient and allows it to be implemented in real-time. Additionally, by using the fit error, EPA provides a point of reference for rank estimation (*i.e.* the level of experimental noise). Other advantages of EPA for the two component systems, such as the linear concentration dependence of the fit error and its ability to estimate peak shapes have been described elsewhere [48].

One aspect of EPA that needs to be addressed when moving to higher dimensions is the selection of wavelengths to be used. For a three-component model, the general model would be:

$$A_i = \alpha_i A_p + \beta_i A_q + \gamma_i A_r \quad (i \neq p, q, r) \quad (2.6)$$

The indices of the wavelengths corresponding to the independent axes in the model have been designated as $p, q,$ and r . The selection of these independent axes is very important. For numerical stability, absorbances at these wavelengths should exceed baseline values, and should not exhibit a high degree of collinearity (*i.e.* not dependent on each other). Also, they should be selected to optimally distinguish among the components. For the one- and two-component cases the selection of wavelengths is more straightforward than it is for models of higher dimensionality. It was found [49] that the selection of wavelengths is important for determining the success of the algorithm.

Several approaches to the problem of wavelength selection were attempted. In one approach, a wavelength for the one-component model was first selected (*e.g.* maximum absorbance as discussed earlier) and that model was applied to the data set. When the one-component model failed, the wavelength showing the greatest deviation was then selected as the second independent axis, and the two-component model was applied. This procedure was repeated until the dimensionality of the model satisfied the data set. Unfortunately, the success of this approach was found to be very dependent on the nature of the data set. For example, the second wavelength, while effective for distinguishing between the first two components, may not be as effective for the second and third components.

A successful alternative approach was found to be pretreatment of the data set with PCA. The first k principal components (orthogonal vectors) were then

used to represent the data; *i.e.* the scores on the first k eigenvectors were used instead of the absorbances at n wavelengths. Typically k was chosen to be large enough to ensure that it exceeded the rank of the data set, yet small enough so some of the noise was excluded. A value of $k=10$ was found to be satisfactory for this work. Since the eigenvectors are ordered according to the proportion of variance they encompass, and since they are by definition orthogonal, the independent axes were simply selected as the first j eigenvectors, where j is the dimensionality of the model. Therefore, the three component model is defined in a manner analogous to Equation 2.6:

$$T_i = \alpha_i T_1 + \beta_i T_2 + \gamma_i T_3 \quad (i=1,2,3) \quad (2.7)$$

where T_i represents the score on principal component i . Using this data pretreatment step had several advantages. First, it provided a reliable method of variable selection that resulted in consistent performance of the algorithm. Second, it increased sensitivity to model deviations by reducing the contribution of those wavelengths containing little or no information. Finally, because the algorithm only needed to deal with k variables rather than n wavelengths, computation time was reduced. For all the results presented here, preprocessing via PCA was used and the first ten principal components were retained.

2.4 Experimental

The multicomponent EPA algorithm was applied to three problems: 1) a simulated chromatographic separation (4 components), 2) the spectrophotometric

titration of pyrocatechol violet (4 components) and 3) the liquid chromatographic separation of aromatic hydrocarbons (3 and 4 components). All chemicals used in this work were analytical reagent grade, except for the pyrocatechol violet which was indicator grade. Spectra were obtained on a HP 8452A diode array spectrometer (Hewlett-Packard, Palo Alto, CA) with 2 nm resolution. Spectra for the spectrophotometric titration were taken with a standard 1 cm cuvette, whereas detection for the chromatographic studies was facilitated by a 30 μ L flow cell with a 1 cm path length (Hellma cells, Jamaica, NY).

Gaussian profiles were used in both the spectral and chromatographic domains for the simulated chromatographic data set. The spectrum and elution profile of each component were shifted by one standard deviation from the preceding one. The effective concentration ratio (components 1 to 4) was 1:2:1:4. The maximum absorbance was set to unity and gaussian noise was added at a level of 0.1% of the maximum.

The spectrophotometric titration of pyrocatechol violet (Aldrich, Milwaukee, WI) was carried out by first placing 100 mL of a 0.01 M solution of pyrocatechol violet in a beaker and adding dilute hydrochloric acid to adjust the pH to 3.00. The pH was recorded with a model 3D pH electrode and a model 119 pH meter (Fisher Scientific, Toronto, ON) which was calibrated with buffer standards. Spectra were obtained between pH 3.00 and 12.50 at intervals of 0.25 pH units by adding small amounts of dilute sodium hydroxide. Spectra were obtained between 266 and 818 nm at 4 nm intervals with an integration time of 1 s.

Chromatographic data from three- and four-component mixtures were used to evaluate the EPA algorithm. The three-component mixture consisted of toluene (11.48 mM), naphthalene (0.42 mM) and m-xylene (8.12 mM) in a 7:1 (v:v) methanol:water solvent (analytes are given in order of elution). The effective concentration ratio (*i.e.* compensating for the maximum molar absorptivity of each compound) is *ca.* 1.3:1:1.1. The four component mixture consisted of toluene (11.54 mM), naphthalene (0.42 mM), m-xylene (7.97 mM) and biphenyl (0.22 mM), with an effective concentration ratio of 1.3:1:1.1:2. These mixtures were analyzed on a chromatographic system consisting of a Shimadzu LC-6A pump (Shimadzu, Columbia, MD), an injection valve with 50 μ L loop (Rheodyne model 5020, Cotati, CA), a Partisphere C₈ column (12.5 cm x 4.6 mm with 5 μ m packing) (Whatman, Hillsboro, OR). Isocratic elution was used with 7:1 (v:v) methanol:water as the mobile phase. These conditions ensured sufficiently poor resolution to test the algorithm. Spectra were obtained at 0.5 s intervals with an integration time of 0.4 s. Data were acquired between 61 s and 169 s after injection for the three-component mixture, and up to 183.5 s for the four-component mixture.

2.5 Results

Initial investigations into the application of EPA to multicomponent chromatographic systems were carried out using simulations. This permitted an examination of the algorithm in the absence of certain experimental artifacts that can complicate real chromatograms. It also allowed a direct comparison to be

made between the times at which components begin to elute and the fit error traces for EPA. Figure 2.6 shows the simulated spectrochromatogram for a four-component mixture. Gaussian profiles were used in both the spectral and chromatographic domains. Figure 2.7a shows the noise free chromatographic elution profiles for the four components in the mixture, Figure 2.7b and c show the fit error traces after processing the data with the EPA algorithm in the forward and reverse directions, respectively. Focussing first on Figure 2.7b, it can be seen that the fit error trace for the one-component model shows deviations from the baseline shortly after the appearance of the second component. Likewise, the two- and three-component models fail with the elution of the third and fourth components. As expected, the RMS(FE) for the four-component model remains flat throughout the chromatogram. Figure 2.7c, which is obtained by presenting the chromatographic data to the algorithm in the reverse order, provides complementary information to that in Figure 2.7b. The results confirm the presence of four components and indicate where the elution of each component is complete. A knowledge of the regions where each component elutes can be useful in curve resolution for the individual components. In earlier work with two-component mixtures, it was demonstrated that the fit errors for the one-component model closely matched the chromatographic elution profile of the second component [48]. With more than two components, the shape of the fit error profiles is more complex, arising from a combination of elution profiles for multiple components. The precise shape of the trace, including the position of the

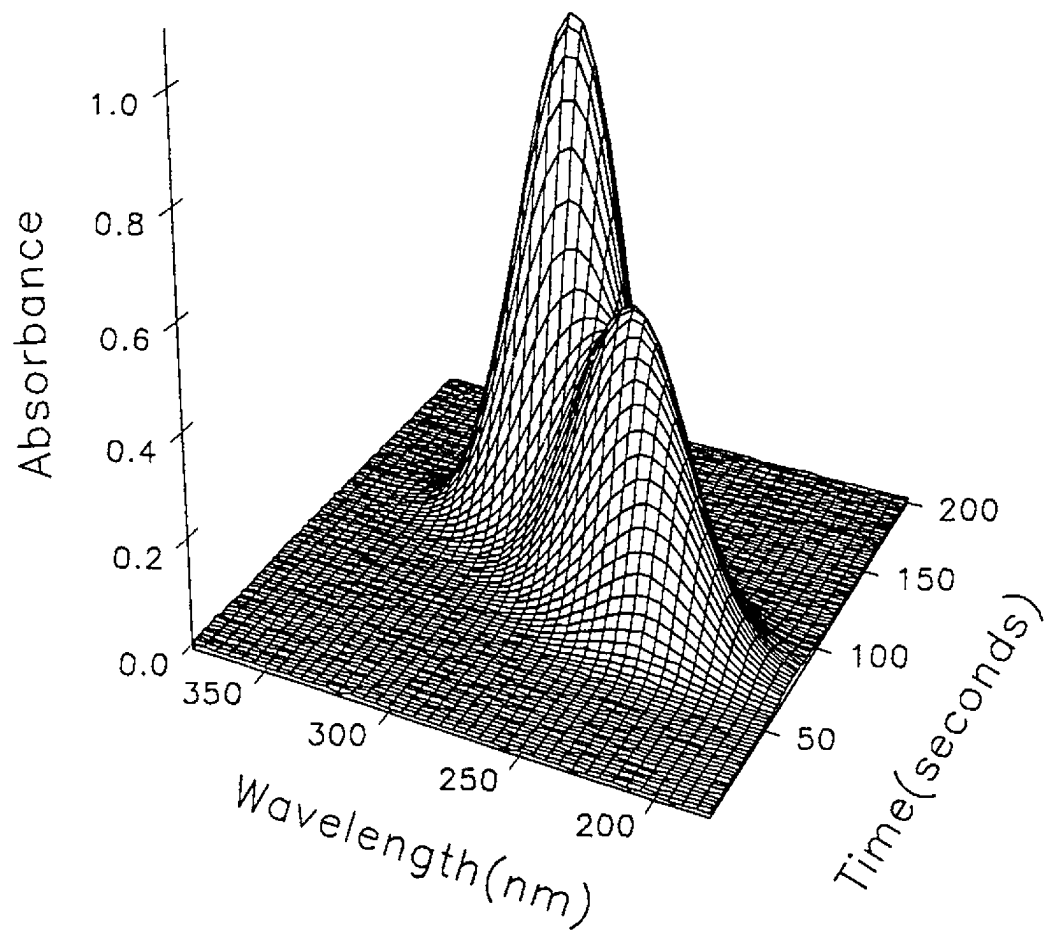


Figure 2.6 Spectrochromatogram for the simulated four-component mixture.

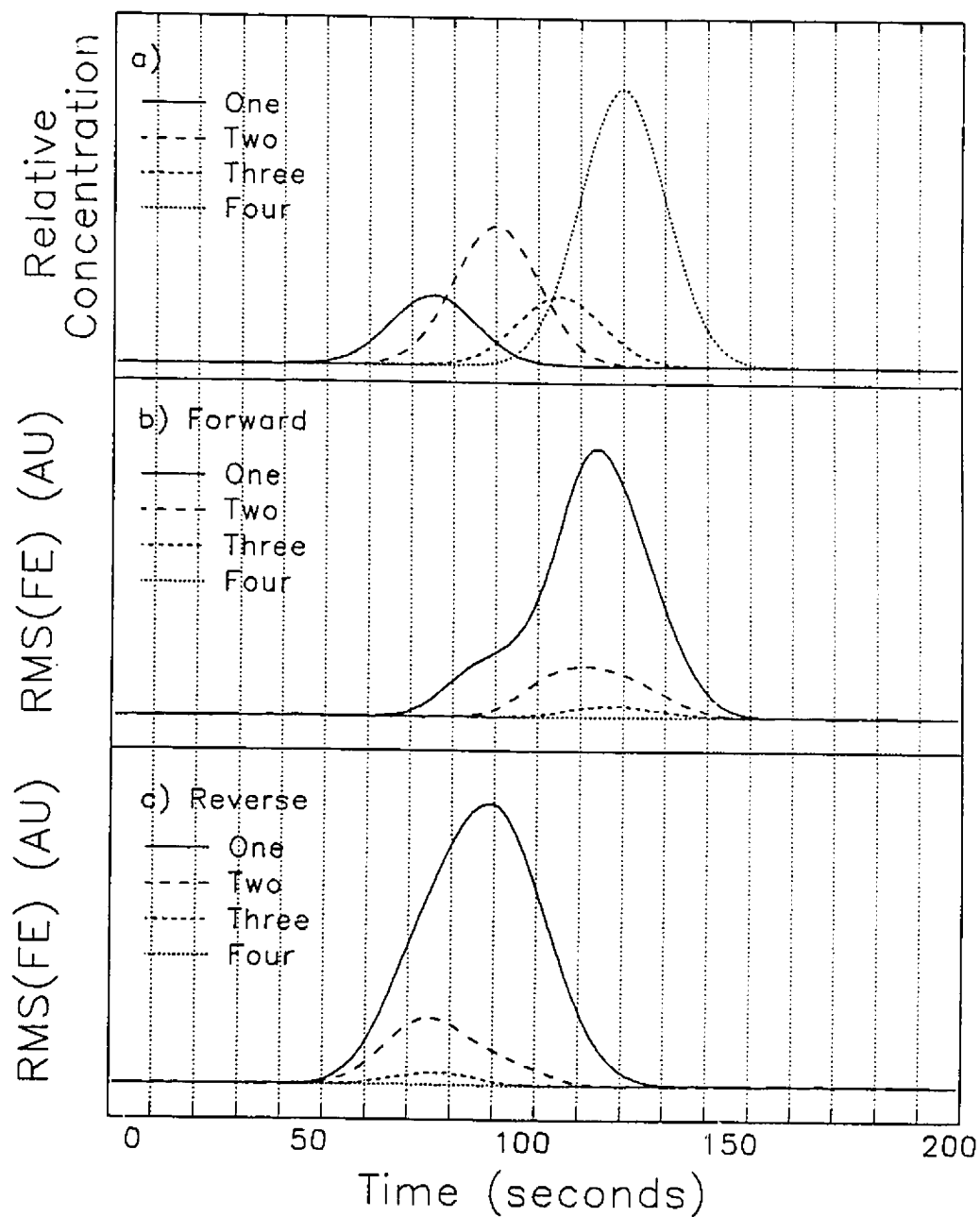


Figure 2.7 EPA of the data in Figure 2.6: **a)** chromatographic elution profiles for each component, **b)** fit error traces obtained by processing the data with the EPA algorithm in the forward direction and **c)** fit errors traces obtained by processing the data with the EPA algorithm in the reverse direction.

maximum, will depend on a number of factors, including the order of the model, the number of components present, spectral and chromatographic resolution, and relative concentration ratios. However, it appears that the trace for the final component (*i.e.* the $(n-1)$ th order model for an n -component mixture) matches the elution profile for that component well. Therefore, it is possible to obtain an indication of the elution profiles for the first and last components with this approach.

Another feature of the RMS(FE) traces is that they tend to decrease in maximum amplitude as the order of the model increases. This is expected, since deviations from high-order models will be smaller than for the low-order models, which are less capable of describing the multicomponent mixture. Again, absolute magnitudes will depend on chromatographic resolution and spectral correlation of the overlapped components, as well as their concentrations and elution order. These factors, along with the level of measurement noise, will ultimately determine the sensitivity of this method to minor components in the mixture. While the sensitivity of this model has been studied for two-component mixtures [38], the case of multicomponent mixtures is more difficult because of the large number of factors influencing performance. Nevertheless, a simple study to test the limitations of the method was undertaken. In this study, a four-component mixture with Gaussian spectral and chromatographic profiles was simulated, with equal peak separation in the chromatographic and spectral domains. The first three components had a maximum absorbance of 0.25, while the fourth component

varied from 0 to 0.025. The maximum amplitude of RMS(FE) was then measured for the three-component model running in the forward direction. This should represent a stringent test, since the last component will produce the smallest trace. Results from this study are plotted in Figure 2.8 for peak separations of 0.75, 1, and 1.25 standard deviations. As expected, larger model deviations are observed as the relative concentration and resolution (spectral and chromatographic) of the fourth component increase. As found to be the case for peak purity detection [50] the detectability of the fourth component will depend on the level of experimental noise. For example, if noise at a level of 1×10^{-3} A.U. were added to the simulated sets, the fourth component will not be detectable when its effective concentration falls below 0.8% and 1.7% of the total for chromatographic/spectral separations of 1.25 and 1σ , respectively.

The spectrophotometric titration of pyrocatechol violet was employed as further application of EPA. Although this is not an exceptionally challenging case for rank analysis since the spectral profiles of the four species are reasonably well separated, it serves to illustrate the principles of the method. Figure 2.9 shows the data from the spectrophotometric titration. Figure 2.10a shows the expected distribution of species calculated using reported equilibrium constants [50]. Figure 2.10b shows the fit error traces in the forward direction. The traces are not smooth, but clearly indicate the presence of four components. The successive failure of the models approximately matches the expected distribution, although deviations for the two- and three- component models seem to lag the appearance

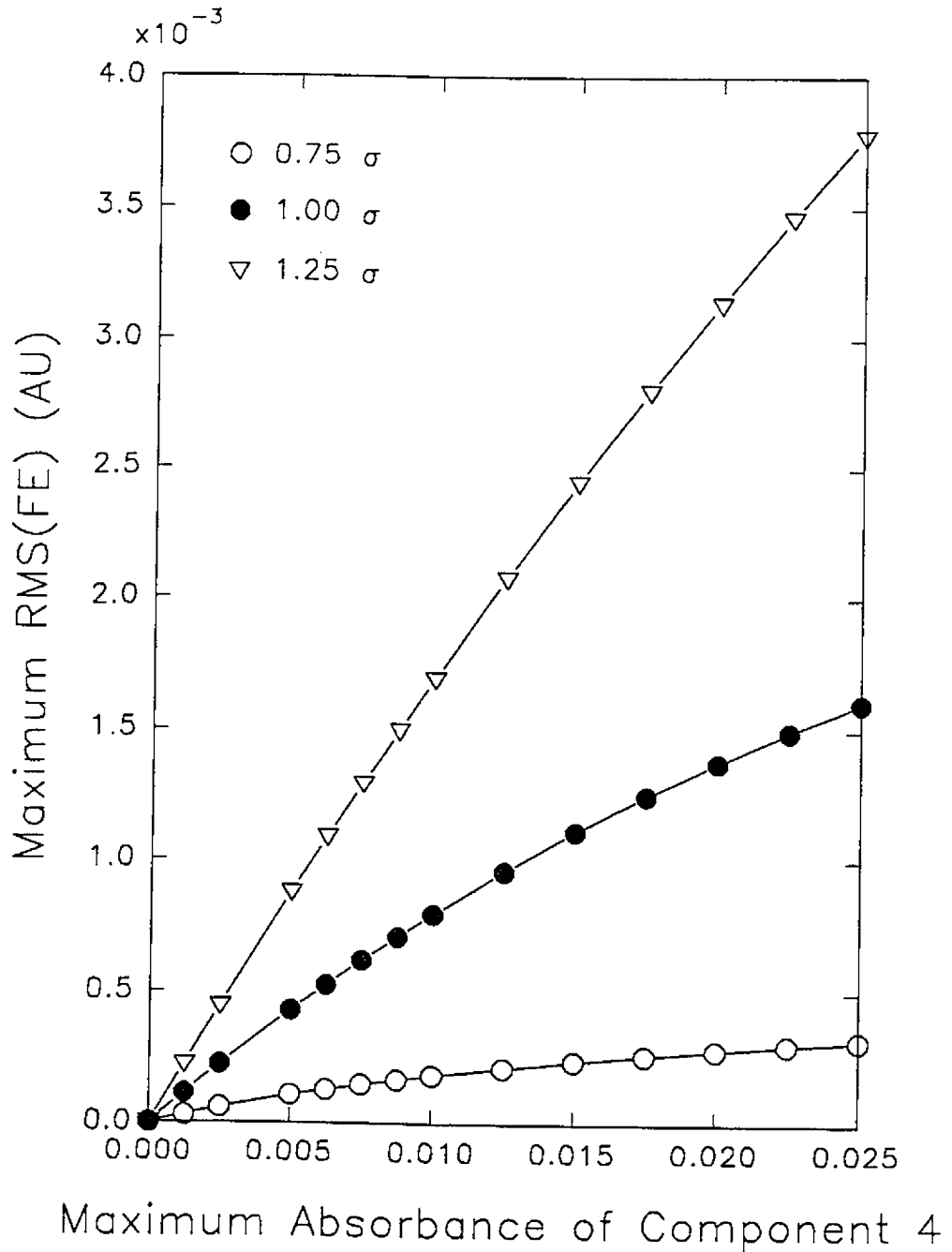


Figure 2.8 Maximum fit error obtained for the three-component model applied to a simulated four-component mixture as a function of the maximum absorbance of the fourth component. Three cases of chromatographic/spectral resolution are shown.

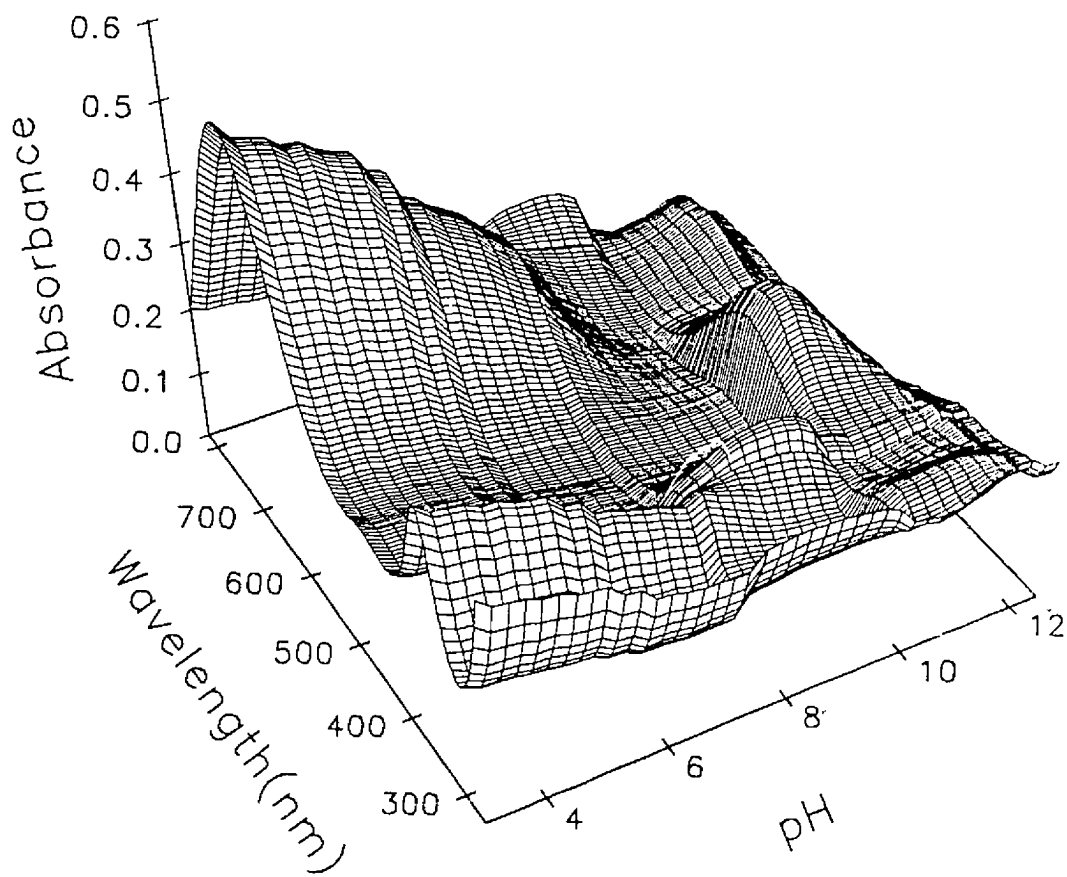


Figure 2.9 Data from the spectrophotometric titration of pyrocatechol violet.

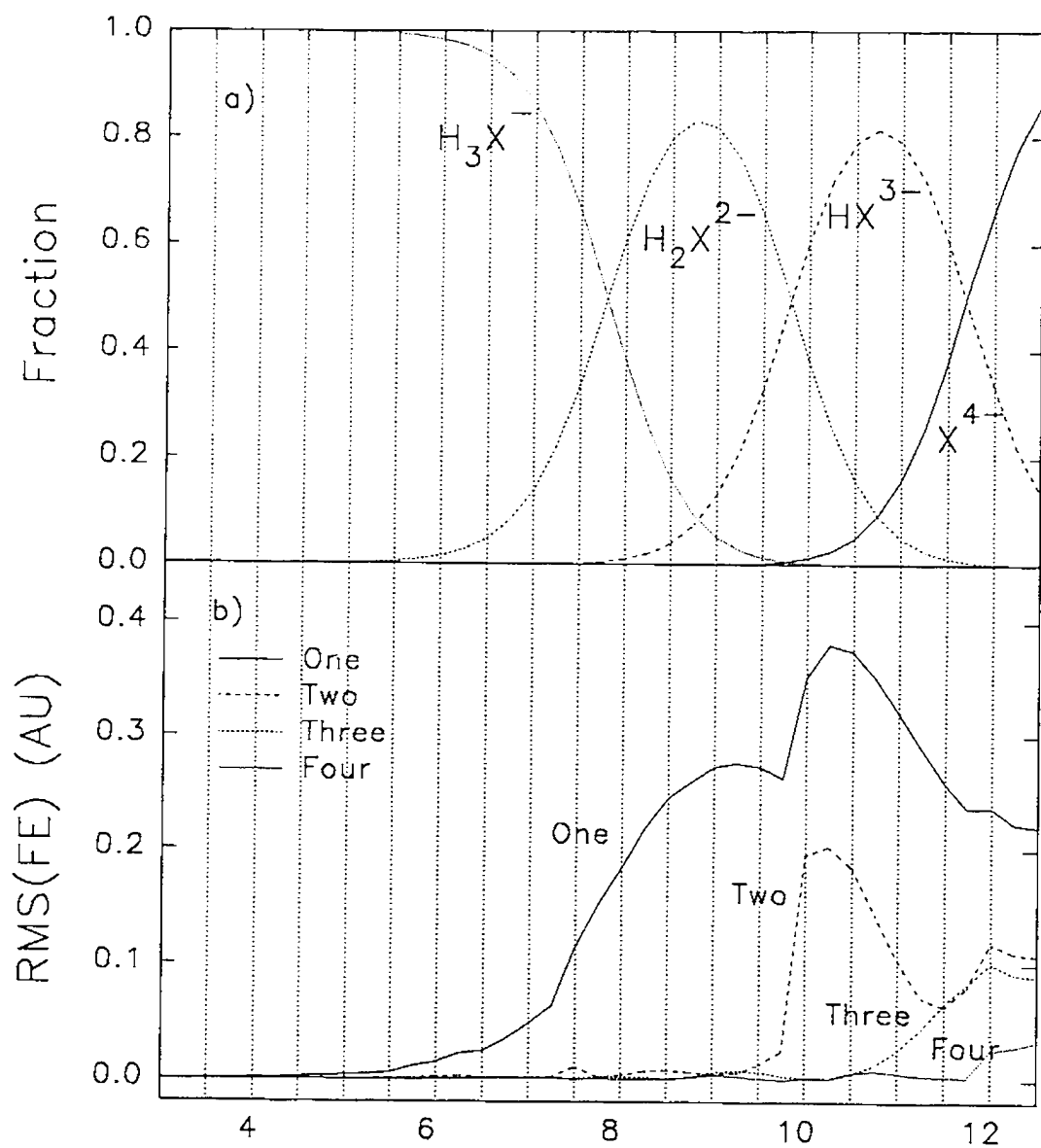


Figure 2.10 Evolving projection analysis of the data in Figure 2.9: **a)** distribution of four forms of pyrocatechol violet as calculated from equilibrium constants; **b)** evolving projection analysis for this data.

of the third and fourth components slightly. There is also an unexpected perturbation of the four-component model at the very end, but the number of points is insufficient to support the presence of a fifth species. In addition the pH measured could be outside the range of the pH electrode and the readings could be incorrect. The more challenging cases for EPA are the chromatographic systems.

The analysis of real chromatographic mixtures can be more problematic. A number of nonidealities such as baseline variations, scan time effects, and heteroscedastic noise are well-known to cause problems in the rank analysis of real chromatographic data [48,51]. These problems are described in greater detail in Chapter 3. Figure 2.11 shows the spectrochromatograms for the three- and four-components mixtures used in this study. Spectra for the four components are shown in Figure 2.12 and are presented in the order of elution. Figure 2.13a shows the fit error traces for the three-component system. As expected, the traces indicate the presence of the second and third components with the successive failure of the one- and two-component models, but the traces are somewhat different from the simulation results in several ways. First, the shape is different, but this is expected since the nonideal chromatographic conditions promote non-Gaussian peaks. Second, it is clear that the fit error traces for the three- and four-component models don't remain completely flat, making it difficult to detect the presence of additional minor components. It is expected that this is due to baseline variations and/or non-ideal detector behaviour. Finally, small

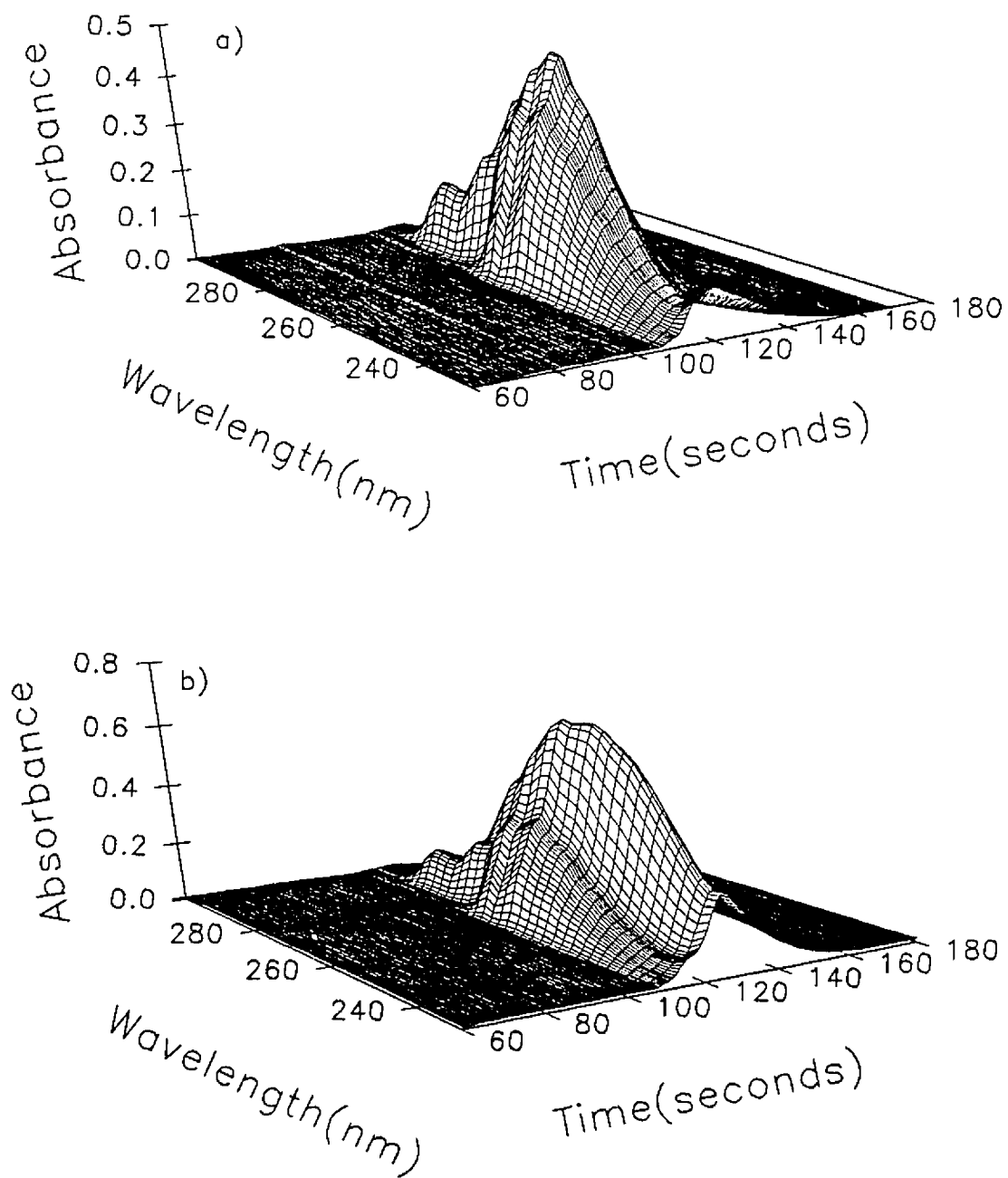


Figure 2.11 Experimental spectrochromatograms for the a) three-component and b) four-component mixtures.

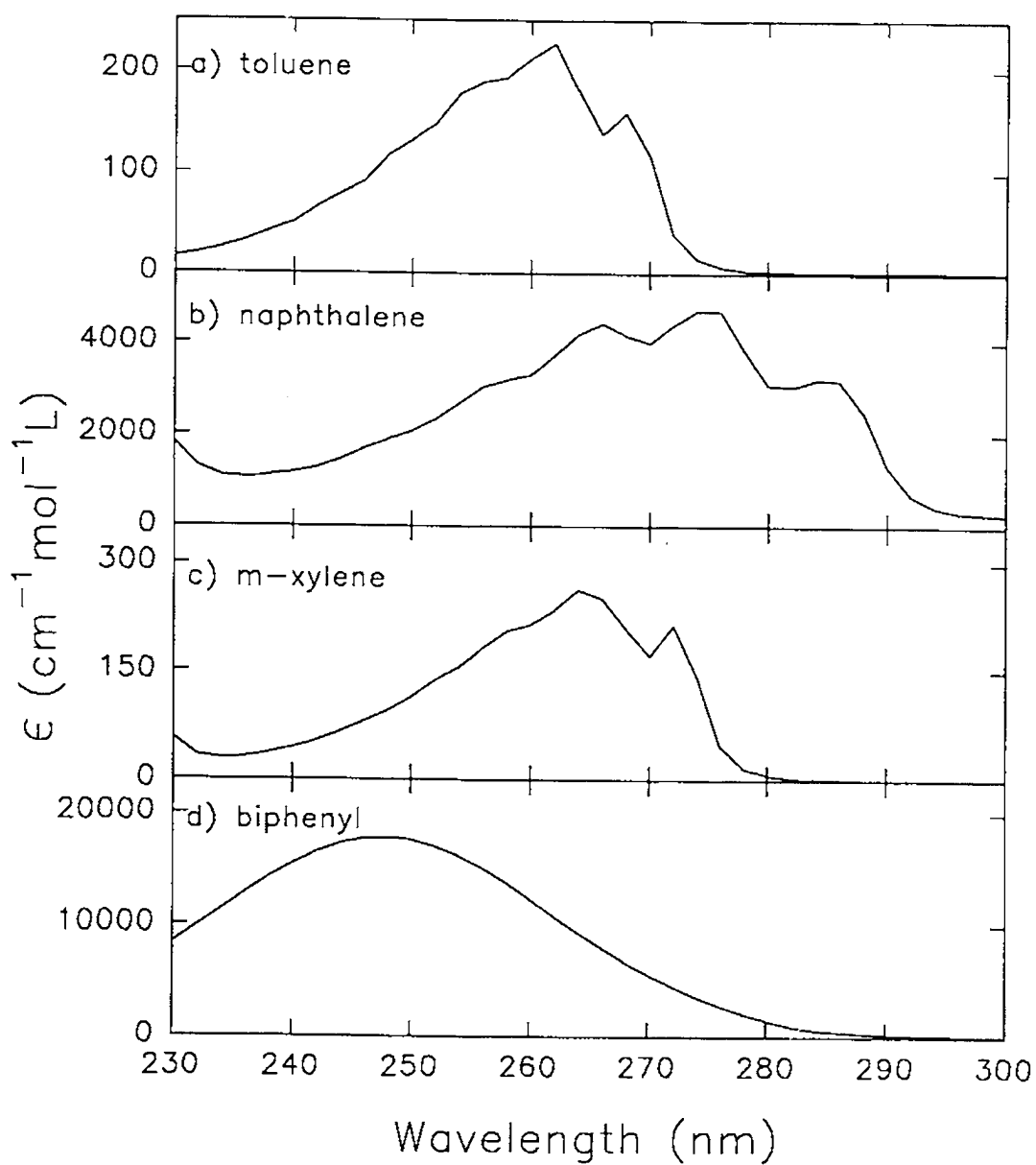


Figure 2.12 Spectra for the four components employed in the chromatography experiments.

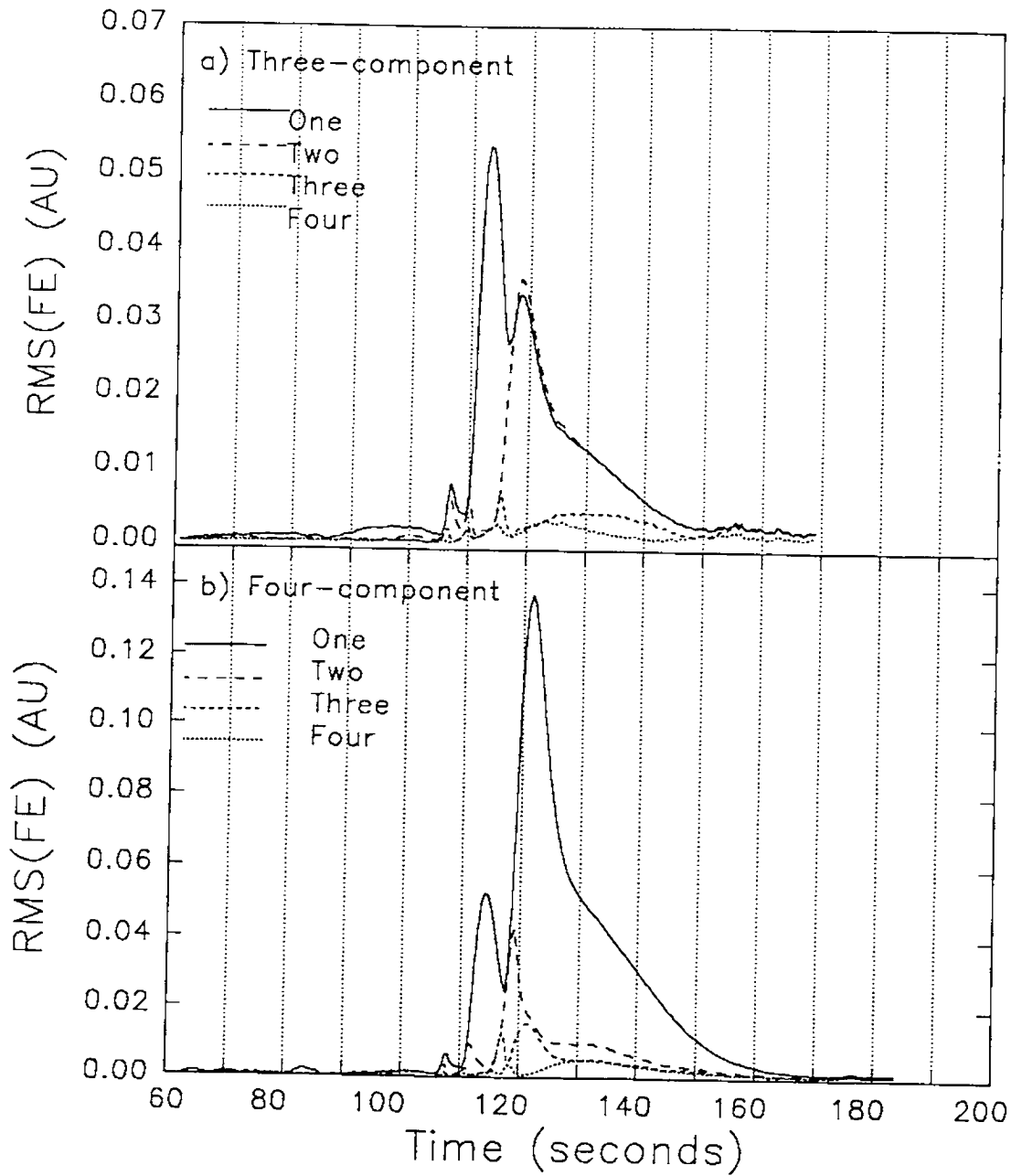


Figure 2.13 Results of evolving projection analysis for the data in Figure 2.12. *i.e.* **a)** the three-component system and **b)** the four-component system.

perturbations in the fit error traces are often observed coincident with the appearance of new components in the mixture. These perturbations, appearing as small "bumps", precede the true failure of the model when it occurs. This behaviour is routinely observed for real chromatographic data and arises from the fact that the baseline noise has some nonrandom structure. In the case of the one-component model, for example, this structure will be a fit to a model. When the first component does appear, a couple of iterations are necessary for the model to lock onto the true structure of the data, and it is during this transition that increased errors are observed. Similar effects are observed for the higher order models. Nevertheless, the fit error traces clearly indicate the presence of at least three components.

The fit error traces for the four-component mixture are shown in Figure 2.13b. The comments made for the three-component case are also valid here, except for the additional component. The deviations for the three-component model are smaller than for models of lower order, as expected from the simulation studies, and begin to approach levels that make them difficult to discern from the artifacts already noted. However the presence of a fourth component is still indicated in this case.

2.6 Conclusions

The results presented here demonstrate that EPA is a useful technique for the analysis of ordered data sets consisting of more than two components. Best

results were obtained when the original data were preprocessed with PCA and the scores were used in place of the original data. EPA has the advantage of providing relative measures of changes in rank and indicates where new components appear in the data set. No prior information needs to be presented to the algorithm. In the case of chromatographic data, simulations have shown that, in the ideal situation, minor components can be detected at relatively low concentrations.

Ordered Sets 2: Peak Purity Analysis: in the Presence of Non-ideal Detector Response

3.1 Peak Purity Analysis

It is often important in chemical analysis to establish the purity of a sample, such as in quality control applications in the pharmaceutical industry, where the purity of a product is a primary consideration. Usually purity can be established by subjecting the sample to an appropriate separation technique, such as chromatography. Sometimes the chromatographic separation shows two or more peaks, clearly indicating that the original sample was impure. However if only one peak is observed, one does not know whether this peak is due to one or a number of components. In the previous chapter the number of components present in unresolved chromatographic separations was determined for ideal systems, "ideal" referring to uniform (or homoscedastic) noise and linear detector response. This chapter will discuss extensions to EPA to include experimental non-idealities, specifically non-uniform (or heteroscedastic) noise and nonlinear detector response. These nonidealities are especially problematic in peak purity analysis where they can give a false indication of additional components in a pure sample.

3.2 EPA in Nonideal Systems

Ideally, a one-component system should fit a one-component model (*i.e.*

equation 2.1). However certain nonideal experimental conditions can lead to a false indication of additional components when the EPA algorithm is employed. Modifications to EPA are needed so that these nonideal conditions can be effectively incorporated into the one-component model. In order to illustrate extensions of the EPA algorithm to nonideal conditions, data obtained from an LC/DAD experiment will be considered. Four nonideal conditions are important for EPA when using absorbance spectra. These conditions are: 1) a nonzero or sloping baseline in the absorbance, 2) a scan-time effect in the measurement of absorbance, 3) a nonlinear relationship between absorbance and concentration and 4) heteroscedastic noise in the absorbance measurements. Each of these conditions will be discussed below.

Baseline effects in EPA for one and two component systems have been previously considered [41]. Baseline effects refer to nonzero or changing absorbance measurements that are not due to the species of interest. With EPA, the presence of a nonzero baseline will indicate that additional components are present in the mixture. For drifting or slowly varying baselines, the problem is normally reflected in the sequence of fit errors and can therefore be diagnosed. Transient perturbations in the baseline are more problematic, however. For liquid chromatographic-absorbance setups (*i.e.* high performance liquid chromatography-diode array detection (HPLC/DAD)) baseline effects are mainly caused by problems in the chromatographic analysis rather than from the detection system. Baseline effects can often be eliminated by carefully planned chromatographic

analysis. For the remaining discussion, a zero baseline will be assumed.

The scan-time effect arises from the fact that, although diode array detectors can acquire spectra very quickly, they do not acquire absorbance measurements at each wavelength simultaneously. For most diode array detectors, the absorbance measurements are obtained sequentially until the entire spectrum has been acquired. The total time to measure all the responses is known as the scan time (typically 10-100 ms). Non-simultaneous measurements can cause problems for data analysis when the sample composition in the cell is not fixed, as in chromatography (*i.e.* the leading and falling edges of a one-component peak will not produce the same spectra). Keller *et al* [52] suggest that the scan-time effect can be easily rectified by data pretreatment. Since scan-time corrections have already been dealt with in the literature they will not be treated here.

Nonlinearities that occur in absorbance detection are mainly due to apparent deviations from Beer's law. Nonlinearities in Beer's law found in chemistry are discussed in most undergraduate analytical chemistry texts [53,54] and a thorough discussion is given by Ingle and Crouch [55]. There are a number of causes of nonlinearities and these can be classed as chemical and instrumental effects. The chemical nonlinearities arise from sources such as chemical equilibria and a concentration dependence on molar absorptivity. Instrumental nonlinearities are mainly due to polychromatic radiation and stray light. Chemical nonlinearities are difficult to compensate for but can be minimized by careful consideration of the

system. Instrumental nonlinearities in absorbance spectra will be characterized in this work and they will be incorporated into EPA. Incorporation of nonlinearities is important because it allows one to model a one-component nonlinear system with a one-component model.

Heteroscedastic noise is noise whose magnitude depends in some fashion on the magnitude of the measured variables. For absorbances measured with the diode array detector in our laboratory, the measurement noise is generally found to increase with the measured absorbance and also varies with the wavelength used. Noise will therefore be largest where the absorbance is highest and the source intensity is lowest. This is a problem since the innovation sequence will estimate experimental noise, thereby producing a response which is not flat even when the correct model is used. Keller *et al* [34,56,57] and others [58] have suggested solutions to this problem based on normalization, but a more robust approach has been employed here.

The problem of nonideal response characteristics arises from the fact that we are no longer able to model a one-component system with a one-component model. For the discussion presented here, only extensions to the one-component models will be discussed, since the main problem being addressed is peak purity analysis, but in principle the methods could be extended to two or more components. To illustrate the problems caused by nonideal responses, Figure 3.1 a - c shows typical A^2 plots obtained for one-component systems under various conditions, whereas Figure 3.1d represents a two-component system.

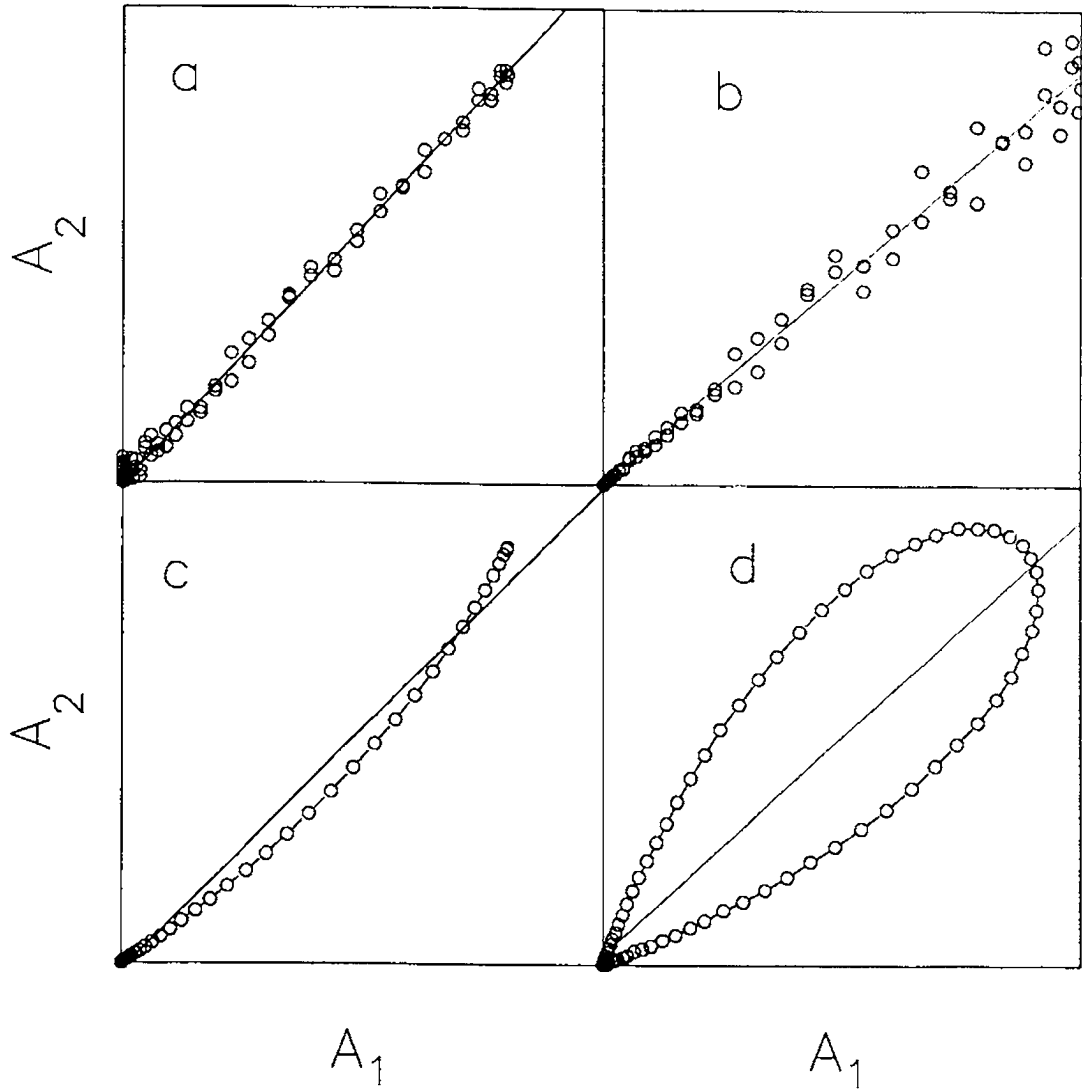


Figure 3.1 Effect of various experimental factors on A^2 plots. **a)** one-component, linear response, homoscedastic noise, **b)** one-component, linear response, heteroscedastic noise, **c)** one-component, nonlinear response, no noise, **d)** two-component, linear response, no noise.

Figures 3.1b and c show typical plots for cases of heteroscedastic noise and nonlinearities respectively. Parts a and d of the figure are for the one- and two-component systems with an ideal detector (linear response, homoscedastic noise). The presence of heteroscedastic noise (Figure 3.1b) will cause an increase in model errors as the absorbance increases, thereby suggesting failure of the one-component model. In chromatography, model errors will typically be greatest at the center of the eluting peak, since this is the point of greatest absorbance. In fact, the model is still valid, but the error structure suggests the presence of additional components. In the case of nonlinearities (Figure 3.1c), systematic deviations from the model will be observed as it tries to adjust to the curvature in the A^2 plot. Typically this results in model errors that first occur in one direction and then the other as the chromatographic signal rises and then falls. Since an RMS error is computed, this results in a bimodal trace in the error sequence which passes through a minimum near the peak maximum. Although both heteroscedastic noise and nonlinearities lead to an increase in model errors, it is clear by comparison with Figure 3.1d that the underlying causes are radically different from the model errors introduced by the two-component system. Ultimately one wants to develop models to distinguish between the one-component (a,b and c) and two-component systems (d).

3.3 Nonlinearities

The discussion here will focus on instrumental nonlinearities, although the

solution to this problem is general in nature and should also accommodate moderate nonlinearities introduced by chemical effects as well. For the instrument used in our laboratory, the dominant source of instrumental nonlinearities is polychromatic radiation. This results not from a failure of Beer's Law, but rather a violation of the assumptions for Beer's Law. Beer's Law assumes that radiation measured at a particular wavelength setting is monochromatic. DADs, like other spectrometers, measure a range of wavelengths for each setting. Nonlinearities are also observed due to stray light, but this is generally not a problem until absorbances become very high. Understanding when and to what extent nonlinearities occur will help to incorporate nonlinear effects into EPA.

Nonlinearities in Beer's Law due to polychromatic radiation in the DAD have been well characterized by Dose and Guichon [59]. The extent to which the nonlinearity is observed can be reasonably explained by considering the bandpass of the detector, the molar absorptivity of the analyte and its local spectral first derivative. Dose and Guichon derived approximate expressions for nonlinear behaviour in DADs and these agreed well with experimental results. The expression they derived is given below,

$$A_{obs} = A_{ideal} - \log \left(\frac{\sinh(Ka_1 \Delta J/2)}{Ka_1 \Delta J/2} \right) \quad (3.1)$$

where K is the natural logarithm of 10, Δ_d is the bandpass of the detector, A_{ideal} is the absorbance predicted by Beer's Law, A_{obs} is the observed absorbance and a_1

is the local spectral first derivative, given by,

$$a_1 = \frac{\partial A}{\partial \lambda} = \epsilon_1 bc \quad (3.2)$$

where ϵ_1 is,

$$\epsilon_1 = \frac{\partial \epsilon_0}{\partial \lambda} \quad (3.3)$$

and ϵ_0 is the molar absorptivity for the analyte at this wavelength. The first term on the right in Equation 3.1 is the absorbance that would be expected if Beer's Law were obeyed. The second term is the term responsible for the nonlinear behaviour. Since the second term will always give a negative value, the absorbance observed will always be less than that predicted by Beer's Law. The effect of various values of ϵ_0 and ϵ_1 is summarized in Figure 3.2. A bandpass (Δ_d) of 4.4 nm was chosen, since this bandpass is typical for the DAD used in this study, and the pathlength was taken to be 1 cm. Molar absorptivity (ϵ_0) values of 500 and 3000 $\text{cm}^{-1} \text{M}^{-1}$ were chosen to represent low and high absorbing wavelengths, respectively. The spectral first derivative (ϵ_1) values of 0.1, 200 and 300 $\text{cm}^{-1} \text{M}^{-1} \text{nm}^{-1}$ represent the range of derivatives commonly found in uv/visible spectra. Figure 3.2 illustrates the effect of nonlinearities as described by Dose and Guichon. As the derivative increases, the absolute contribution is the same for the two wavelengths, but the relative contribution of the nonlinearity to the observed absorbance is greater for the lower absorbing wavelength. In other words, for a given concentration, the nonlinearity contributes more for a smaller ϵ_0 . This nonlinearity will be incorporated into EPA.

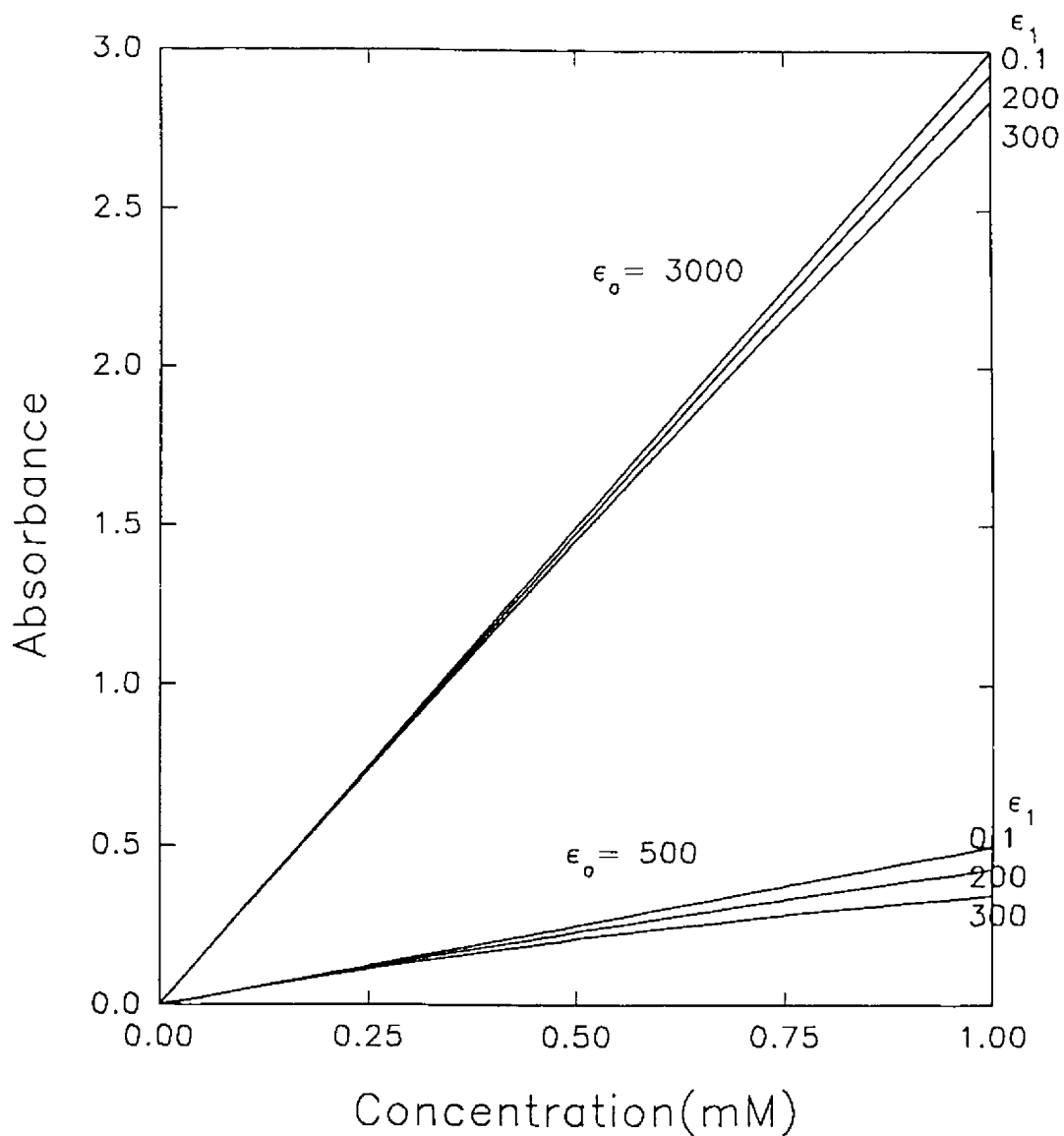


Figure 3.2 Calibration curves predicted by the equation of Dose and Guichon (eqn 3.1), at two wavelengths ($\epsilon_o = 500 \text{ M}^{-1} \text{ cm}^{-1}$ and $\epsilon_o = 3000 \text{ M}^{-1} \text{ cm}^{-1}$) for various values of ϵ_1 (0.1, 200 and 300 cm^{-1}).

The effect of stray light on the linearity of calibration curves is somewhat different from the effect of polychromatic radiation and is given by:

$$A_{stray} = -\log(T + r_s) + \log(1 + r_s) \approx -\log(T + r_s) \quad (3.4)$$

where T is the ideal transmittance, I_{stray} is the stray light, I_r is intensity of the reference, and r_s is the contribution of stray light (I_{stray}/I_r) [55]. The influence of stray light for conditions used in our laboratory ($r = 2.7 \times 10^{-3}$ [60]) for the cases used in Figure 3.2 are shown in Figure 3.3. The figure shows that the influence of stray light becomes more important at higher absorbance values and that the curvature eventually reaches a limiting value which corresponds to the stray light level. The model modifications proposed below have a much more difficult time dealing with the flat regions of the curve than with the relatively smooth variation introduced by polychromatic radiation. Nevertheless, it should still be able to handle relatively moderate stray light contributions.

Three approaches could be taken to model nonlinearities: 1) transform the nonlinear data into linear data, 2) break the nonlinear system into a concatenated series of linear systems and 3) propose a model incorporating the nonlinear characteristics of the responses. Of the three approaches the third approach was favoured as being the most flexible. In this work, the ability of a modified model to handle nonlinearities in one-component systems was investigated.

A representative A^2 plot for the calibration curves shown in Figure 3.2 is shown in Figure 3.1c. Curves of this type will be observed whenever the extent of the nonlinearity at wavelength 1 is greater than that at wavelength 2.

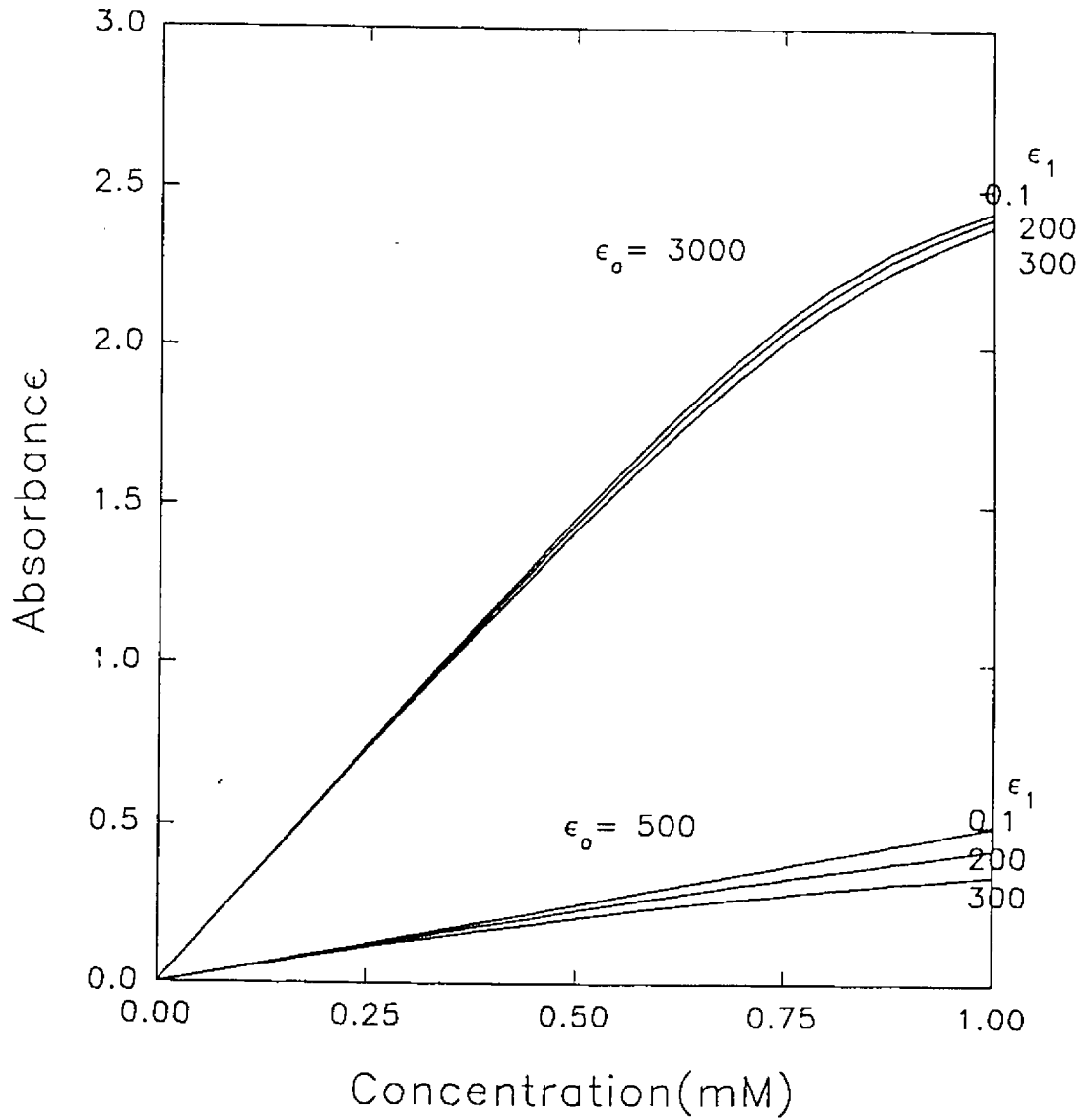


Figure 3.3 Figure showing the influence of stray light on calibration curves. This figure used equation 3.4 and the same conditions as Figure 3.2.

If the situation is reversed, the curve will be concave downward rather than upward. It is also possible (although less common) to observe S-shaped or reverse S-shaped curves if both polychromatic radiation and stray light play a role. It is clear that the model given by Equation 2.1 is inappropriate for nonlinear systems. Since the line in Figure 3.1c exhibits a relatively small degree of curvature and may be approximated by a low order polynomial, a reasonable model for a one-component system might be:

$$A_2^p = \alpha A_1 + \beta A_1^2 \quad (3.5)$$

where α and β are parameters to be estimated and A_1 and A_2^p were defined previously. The one-component model as given in Equation 3.5 will be called the one-component nonlinear model and the models given by Equations 2.1 and 2.2 will be referred to as the one- and two-component linear models, respectively. This nonlinear model, when used in conjunction with the one- and two-component linear models, will be able to distinguish between a one-component nonlinear system and a two-component linear system provided that nonlinearities are not severe.

In the implementation of the nonlinear EPA algorithm, the prediction error could no longer be reliably employed due to the poor extrapolation ability of polynomial models. On the rising portion of the chromatographic peak, where the absorbance is increasing, prediction of the next measurement meant extrapolation of the quadratic model. To circumvent this problem, the prediction error was replaced by the fit error (FE), defined simply as the residual for the most recent

point. This did not significantly alter the sensitivity of the algorithm to impurities and dramatically improved the performance of the quadratic model for the one-component systems. The FE and not the PE will be used throughout the rest of this work.

3.4 Heteroscedasticity

As in the case for nonlinearities, understanding when and to what extent heteroscedastic noise occurs will help to incorporate it into EPA. The HP 8452A DAD used in this work should be representative of DADs used for HPLC detection systems, and so was used for noise characterization. If the effect of heteroscedastic noise can be reduced or eliminated in this system, the approach should be applicable to other DADs. For the DAD used in this study the typical measurement noise in absorbance as a function of absorbance is shown in Figure 3.4. The plot was obtained by measuring the intensity of the sample (I_s), reference (I_r) and dark current (I_d) at each wavelength. The intensity at each wavelength could be varied by changing the concentration of an analyte, e.g. methyl orange but in this case cross-polarizers were used. In addition the standard deviation of the sample intensity (s_s) at each wavelength was obtained. The conversion of intensity readings to absorbance (A) is carried out in the instrument using:

$$A = -\log\left(\frac{I_s - I_d}{I_r - I_d}\right) \quad (3.6)$$

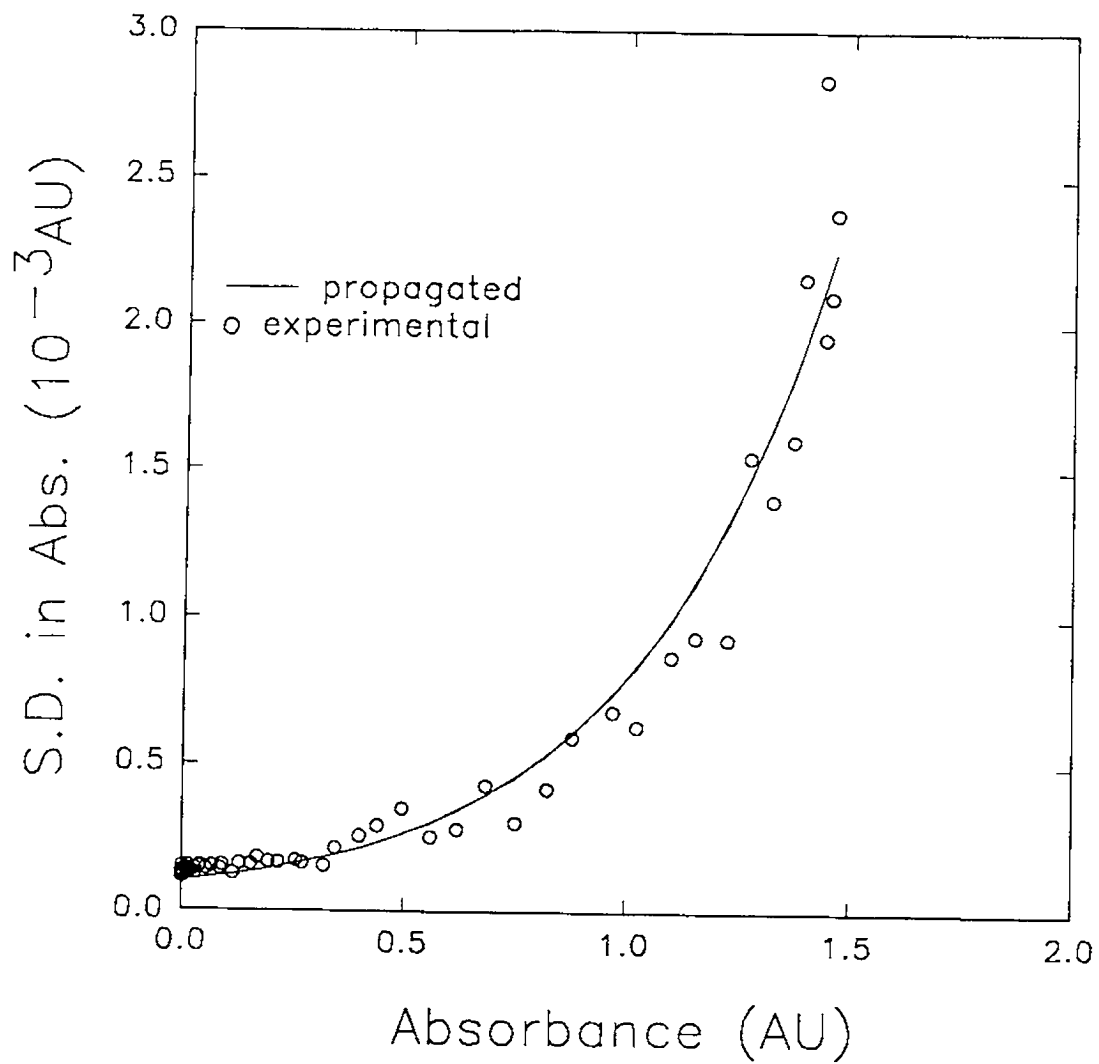


Figure 3.4 Measured errors in absorbance for the diode array detector.

The noise characteristics in the intensity were propagated to absorbance readings according to:

$$s_A = \frac{s_s}{2.303(I_s - I_d)} \quad (3.7)$$

where s_A designates the standard deviation in A. Note that the equation does not incorporate the variance of I_d or I_r , since these values are constant throughout a chromatographic run with a single-beam instrument. The trend of increasing standard deviation in absorbance with increasing absorbance was observed for all wavelengths in this study.

The one-component linear model will be used to develop a model incorporating noise correction. The one-component linear model was given in Equation 2.1 as

$$A_2^p = \alpha A_1$$

In dealing with heteroscedastic noise, the uncertainty in both A_1 and A_2 must be considered. The proper way to deal with heteroscedasticity and uncertainty in both variables (e.g. A_1 and A_2) is to perform weighted regression using a variance that takes uncertainty in both A_1 and A_2 into account. This variance will be referred to as effective variance and for the one-component linear model it is obtained by propagating the error in the x-axis onto the y-axis to give

$$s_{\text{eff}}^2 = \alpha^2 s_{A_1}^2 + s_{A_2}^2 \quad (3.8)$$

where α was defined previously, s_{eff}^2 is the effective variance, $s_{A_1}^2$ and $s_{A_2}^2$ are

variances in measuring A_1 and A_2 (obtained directly from the diode array or through an appropriate noise model). This effective variance is used for weighted regression as implemented with the Kalman filter. The problem with weighted regression of this form is that an estimate of α is needed. In practice first unweighted regression is performed to obtain an estimate of α , and then weighted regression is performed using this estimate to obtain an updated α . For this work satisfactory results were obtained by using only the unweighted α and the calculation of s_{eff}^2 as shown in Equation 3.8. Although this approach may provide the correct weighted regression parameters, it does not yet solve the problem of heteroscedastic noise. In order to obtain a flat innovation sequence it is necessary to weight the fit error by the effective variance (*i.e.* s_{eff}^2).

One way to compensate for increases in measurement errors is to scale the fit error according to the variance expected for the measurements at the wavelengths corresponding to the model in question. Thus for the one-component linear model the scaled fit error (SFE) is given by:

$$\text{SFE} = \frac{\text{FE}}{s_{\text{eff}}} = \frac{A_2(\text{measured}) - A_2(\text{fitted})}{\sqrt{s_2^2 + \alpha_2^2 s_1^2}} \quad (3.9)$$

However unlike the RMS(FE), the RMS(SFE) does not have units which allow it to be readily compared to the measurement error. Although such comparisons are not essential, they are sometimes useful, so in some cases the SFE is rescaled to the baseline noise level (s_b):

$$\text{SFE} = \frac{\text{FE} * s_b}{\sqrt{s_2^2 + \alpha_2^2 s_1^2}} \quad (3.10)$$

The above equation is for the one-component linear model. For the one-component nonlinear model it is necessary to modify the scaled fit error to reflect the propagation of error by the quadratic model. The equation for the quadratic model is,

$$\text{SFE} = \frac{\text{FE} * s_b}{\sqrt{s_2^2 + (\alpha_2 + 2\beta_2 A_1)^2 s_1^2}} \quad (3.11)$$

These modifications allow a direct comparison of modified and unmodified algorithms.

3.5 Experimental

All absorbance measurements were made on an HP 8452A diode array spectrophotometer (Hewlett-Packard, Palo Alto, CA) with a 30 μL flow cell (Hellma Cells, Jamaica, NY). In addition all solutions were transported through the flow cell using a peristaltic pump (Ismatec MS Reglo, Cole Palmer, Chicago, IL). Calculations were carried out on a 33 MHz 80486 based computer using Matlab 4.0 (Mathworks, Natwick, MA) for Windows (Microsoft Redmond, WA).

Several systems were studied to assess nonidealities. For some experiments, a photographic step tablet (#3, Eastman Kodak, Rochester, NY) was used to reduce the source intensity. This strip was linearly graduated with 21

neutral density filters ranging from approximately 0.05 to 3.05 AU. In other experiments a set of cross-polarizers (Melles Griot, Irvine, CA) was used to vary the intensity. Three sets of solutions were prepared using methyl orange in 0.1 M HCl, PrCl_3 in 0.1 M HCl and p-xylene in methanol.

In some experiments a chromatographic peak was emulated using a stopped-flow approach, this removed the effects of scan-time. Two computer controlled pumps were used to adjust the ratio of solvent to analyte which were mixed. The ratio was controlled in such a way to obtain the desired concentration profile, with the flow being stopped between measurements. This allowed close control over the shape of the peak and resulted in more reliable data.

3.6 Results and Discussion

One of the difficulties in studying the effects of nonideal detectors on peak purity analysis in chromatography is that the various instrumental effects are difficult to separate from one another. For example, the effects of heteroscedastic noise as well as nonlinearities due to polychromatic radiation and stray light will become more pronounced as absorbance increases. Scan time effects are a further complication present at all signal levels. In order to separate these effects from one another, it is convenient to incorporate them into a simulation program. To ensure a realistic simulation, it was modeled on the diode array spectrometer used in this work. This required that the instrument be well characterized. More exhaustive studies of photodiode array detectors have appeared in the literature

[61], but the results presented here permit a realistic replication of experimental results and exceed the capabilities of the simulation methods used in this work.

The 8452A diode array spectrometer was designed more as a general purpose spectrometer than as a chromatography detector. Although, it can be used for this purpose, design considerations place a lower limit of about 30 μL on the size of the flow cell. Nevertheless, it is expected that many of the features will be similar to those of a DAD designed specifically for chromatography. The 8452A DAD has a deuterium light source that is normally used for both uv and visible regions of the spectrum. The detection elements are 316 readable photodiodes read at 2 nm intervals over the range 190 to 820 nm. Each of these photodiodes has an independent gain setting that can be adjusted over a linear range of sixteen different values. The amplified intensity values recorded at each of the diodes are sampled sequentially by an A/D convertor. The amplified signals appear to have a fixed offset added to them prior to conversion. A cycle time of 100 ms is required to read all the diodes. For a typical LC/DAD, this value is reduced to 10 ms to minimize scan time effects.

The first step in characterizing the detector was to examine the linearity of the response in the absence of physical or chemical effects. To do this a calibrated step tablet was used. A reference scan was made using a step tablet index of 5 and absorbance measurements were made at other index numbers. One reason for making the reference scan at step index of five was that this index approximated the light levels for the flow cell used in this work. The plot of

absorbance vs the step tablet index exhibited excellent linearity up to at least 3 absorbance units, so this source of nonlinearity was not considered to be important and was ignored in developing a model for the diode array response.

Other sources of nonlinearity in making measurements on solutions are polychromatic radiation and stray light. Apparent deviations from Beer's Law due to polychromatic radiation depend on the absorbance, the bandpass of the spectrometer and the slope of the spectrum in the wavelength region of the measurement. This is illustrated in Figure 3.5, which plots measured absorbances vs those predicted on the basis of Beer's Law for various concentrations of *p*-xylene at several wavelengths [60]. Note that the greatest departure from linearity occurs for those wavelengths at which the spectrum is changing most rapidly. Stray light will also contribute to nonlinearity, but normally this only plays a role at higher absorbance values. This contribution is difficult to quantify and will vary with the solution being measured. In order to obtain an indication of the contributions from polychromatic radiation and stray light, data for *p*-xylene at six wavelengths (242, 250, 264, 270, 276 and 278 nm) were fit to a theoretical model based on a first-order Taylor series expansion described by Dose and Guichon [59] (see Equation 3.1). The fit of the experimental data at 278 is shown in Figure 3.6 (from [60]) and exhibits excellent agreement with the model. Fit parameters from the six wavelengths gave $\Delta = 4.4 \pm 0.2$ nm and $r = 2.7 \pm 1.1 \times 10^{-3}$. The spectral bandwidth determined was fairly repeatable at each of the wavelengths and is broader than the 2 nm spacing of the diodes, as might be expected. The

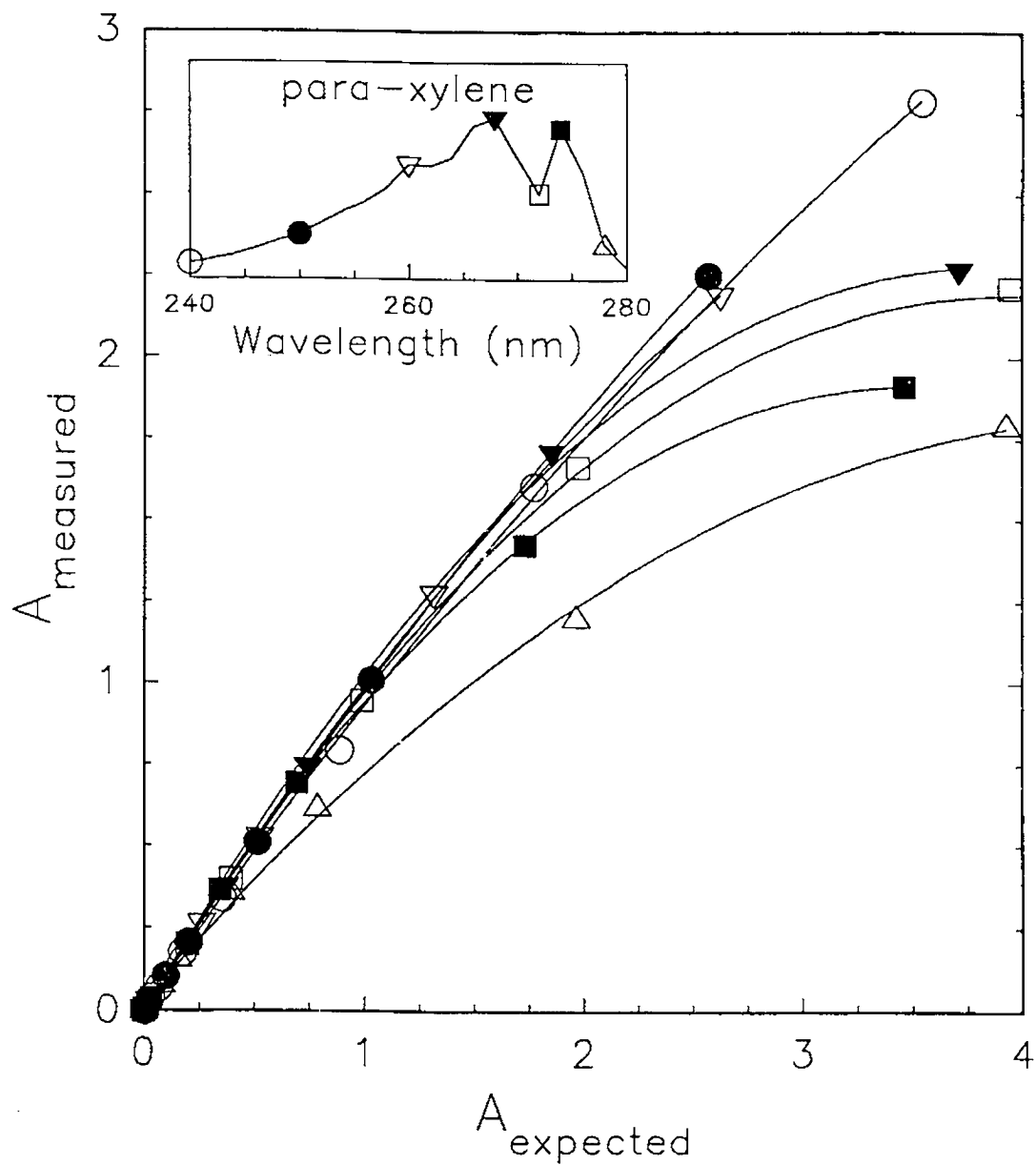


Figure 3.5 Detector response curves measured at various wavelengths for p-xylene showing nonlinearities. The spectrum of p-xylene indicating wavelengths used is inset (from Reference 60).

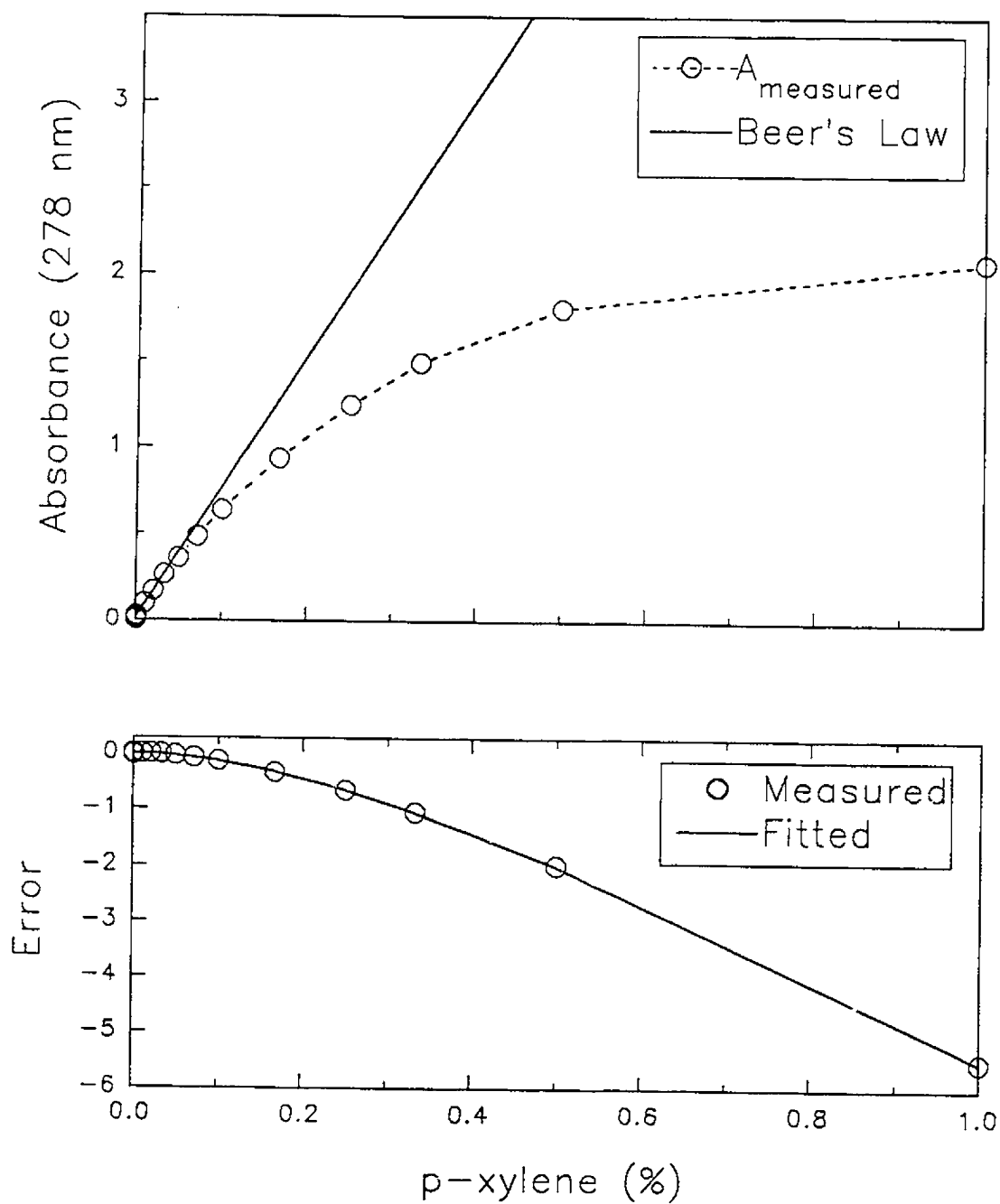


Figure 3.6 Upper plot shows nonlinear calibration curve obtained for *p*-xylene at 278 nm. Lower plot shows fit of deviations to a model incorporating polychromatic radiation (first-order correction) and stray light [60].

stray light exhibited a much greater degree of variability but was generally between 0.1 and 1 % in this region of the spectrum. It should be noted that measures of stray light are in general quite variable, depending on the methods used for their estimation. The numerical values obtained here are taken to be typical for the purposes of simulation.

The final step in the characterization of the DAD was an examination of the heteroscedasticity in the noise. This study was complicated by the fact that the gain for each diode is normally set automatically and independently. To characterize the noise, all diodes were forced to the lowest gain setting and the standard deviation in the intensity of the source was recorded at all wavelengths. This experiment was repeated for several levels of source intensity, which was varied through the use of cross-polarizers. The results are shown in Figure 3.7. It was found that the variation of noise with intensity was essentially independent of wavelength and moreover resulted in a smooth curve. The conversion of intensity readings to absorbance is carried out by the diode array using equation 3.6. The noise characteristics in the intensity were propagated to absorbance readings according to equation 3.7. This model worked well for describing the noise in the diode array, but it was usually necessary to scale the results to account for variations in the absolute level of the noise over time (*i.e.* the form of Equation 3.7 is correct, but the relative magnitude of s_e can vary somewhat over time). For this work, it is the form of the model rather than the actual magnitude of the errors that is important. Figure 3.4 showed a comparison of measured

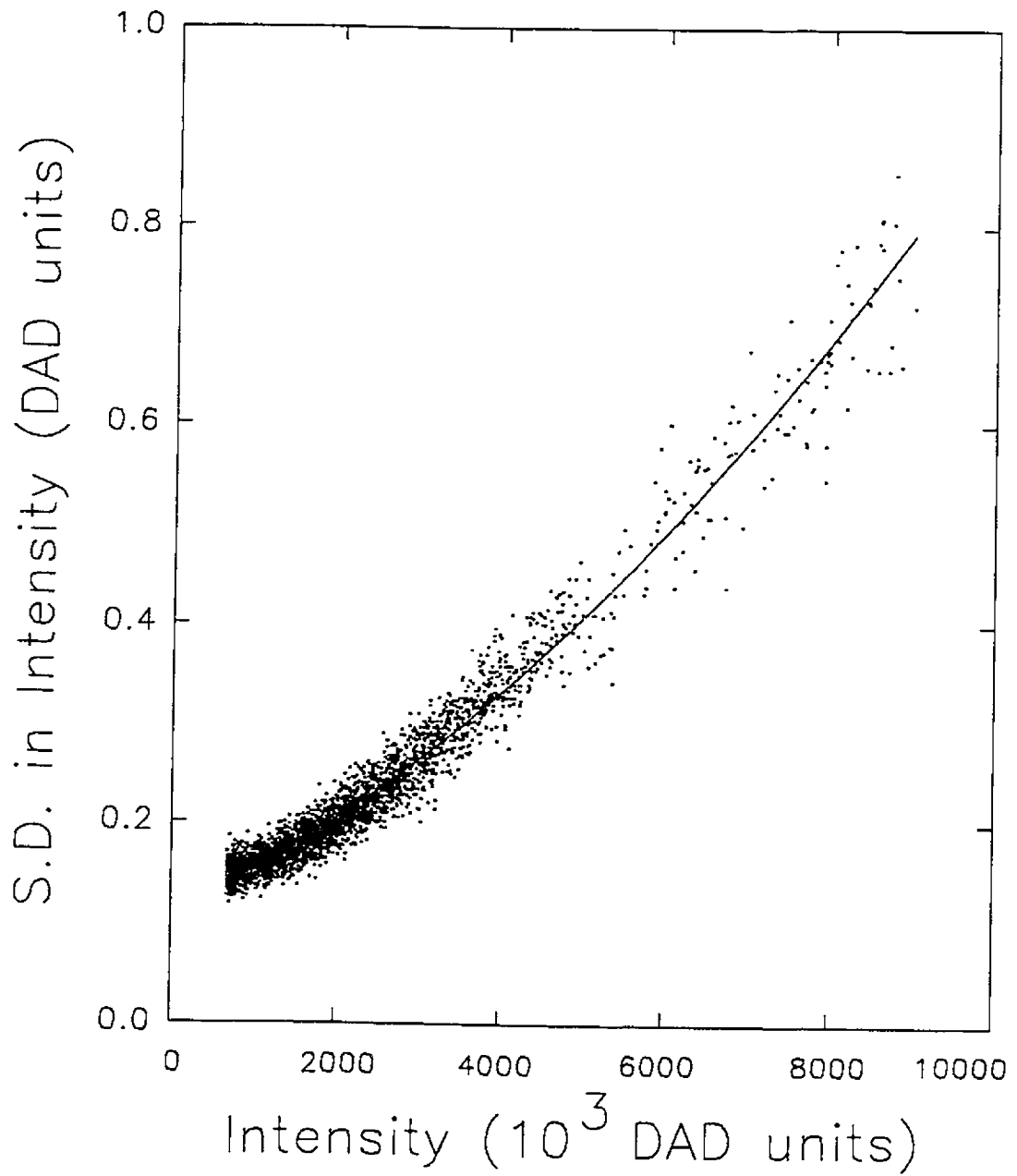


Figure 3.7 Noise (standard deviation) characteristics of the DAD. Points represent measurements made at all wavelengths in the visible region. The solid line is a second-order polynomial fit to the data.

uncertainties in absorbance and uncertainties predicted by a propagation of errors using the quadratic model (Figure 3.7) for heteroscedastic noise. For this plot the absorbance uncertainties were measured for varying concentrations of methyl orange and the errors predicted by Equation 3.7 were scaled by a factor of 1.2. Agreement between the two sets of results is generally very good and supports the form of the model.

To implement these results in a simulation, an intensity scan was recorded with the shutter closed for all wavelengths, and the results were stored in a file. These values, designated I_d , represent a measure of the dark current plus the constant offset employed by the spectrometer. Then, intensity scans were recorded for a variety of reference conditions (e.g. flow cell with distilled water). Both the dark current and reference scans were scaled to the lowest gain of the 16 possible settings. The actual gain for any diode depends on the source intensity at the corresponding wavelength and is important in the reduction of digitization errors, but is not important in the simulation, which is not limited in this regard. To determine I_s for the given absorbance, Equation 3.6 was used. Uncertainties for I_s were estimated using the fit to Figure 3.7, and errors were propagated using Equation 3.7. To adjust for nonlinearities, the noise-free spectra produced by the simulation were used along with the estimated values of spectral bandpass and stray light, and the appropriate equations. Spectral derivatives were calculated using a five-point quadratic (Savitzky-Golay [62]) filter. The noise is added after the nonlinearities have been introduced.

The EPA algorithm was first applied to data generated from the DAD simulator in order to independently evaluate the effectiveness of the heteroscedastic noise and the nonlinear corrections. As a model for this simulation, the spectrum of praseodymium chloride (PrCl_3) was used. This spectrum, shown in Figure 3.8, shows the sharp features characteristic of certain rare earth metal complexes. It was hoped that these features would exaggerate some of the nonlinearities in the response.

To examine the effect of the heteroscedastic noise corrections, nonlinear effects were excluded in the generation of simulated results, and Gaussian noise was added according to the model developed in the previous section. A typical reference scan for the flow cell was used as the reference spectrum. A Gaussian concentration profile ($\sigma = 10$ s, centered at 50 s, sampled at 1 s intervals for 100 s) was assumed. To produce a significant amount of heteroscedasticity, the concentration was adjusted to give a maximum absorbance of 2 AU. Figure 3.9a shows the RMS of the fit error as a function of time for the linear, one-component models with and without the noise correction. The model with noise correction used variance estimates calculated with the DAD noise model. The one-component model without noise correction shows a clear systematic variation in the fit error as the peak is eluting. This would normally signal the presence of additional components in the elution profile, but in this case is merely associated with the heteroscedastic noise. The noise-corrected one-component model, on the other hand, shows no deviation from the baseline noise level, correctly indicating

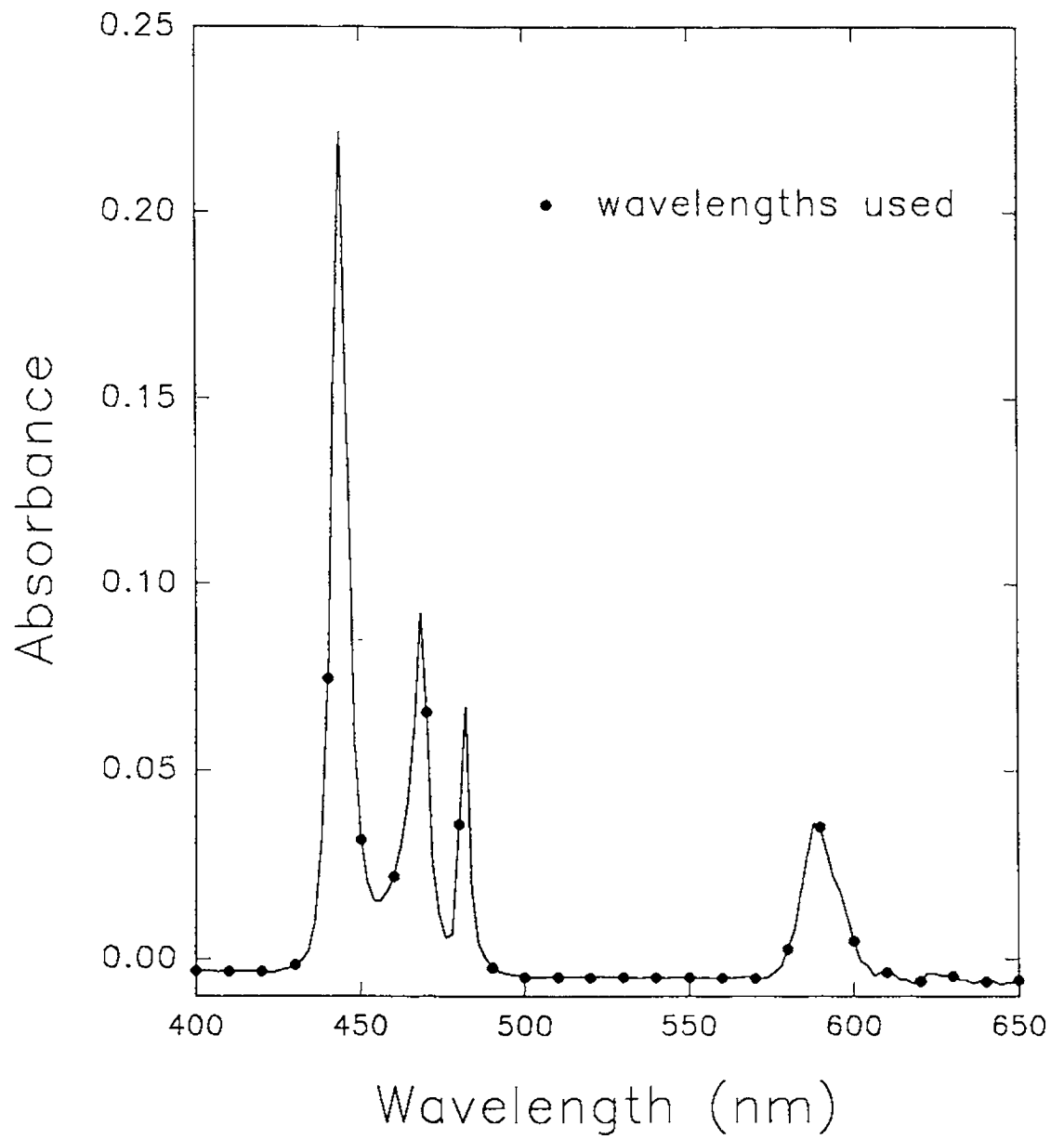


Figure 3.8 Spectrum of PrCl_3 , indicating the wavelengths used for the simulation.

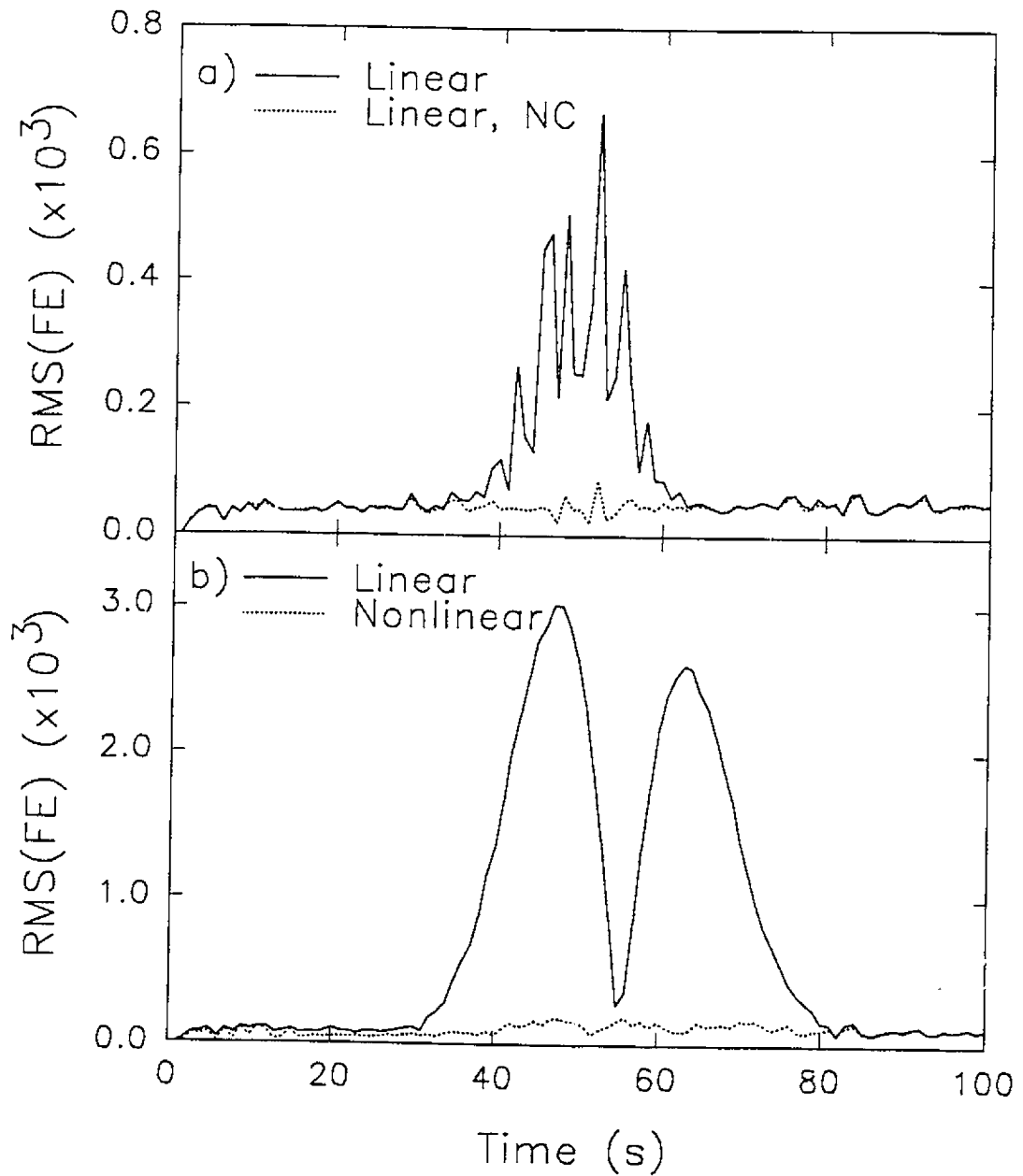


Figure 3.9 a) Effect of heteroscedastic noise on EPA models without (solid line) and with noise correction (dashed line) and b) the effect of nonlinear detector response on linear (solid line) and nonlinear (dashed line) EPA models for the simulated data.

that there is only one component present.

A similar study was used to investigate the effect of the correction for nonlinearities. In this case, homoscedastic noise at a level of 0.04% of the maximum absorbance (0.5 AU) was assumed. Nonlinearities due to the effect of polychromatic radiation were incorporated using a bandpass of 4.4 nm. The effect of the nonlinearity on the one-component fit error is shown in Figure 3.9b. Again, the nonideal behaviour produces systematic deviations from the baseline with the traditional linear model. One important difference from the heteroscedastic noise case, however, is that the trace of the fit errors is bimodal. This is common when nonlinearities are important and the minimum corresponds to the point where the data cross the model as they return to the baseline. It will be noted that the nonlinear one-component model (quadratic in this case) provides a flat trace for the fit error, correctly indicating the presence of only one component.

When linear two-component models are applied to the above situations, different effects are observed. When heteroscedastic noise is the dominant artifact, the linear two-component model shows little difference from the linear one-component model, as would be expected since the noise variations are random. In practice, this similarity can indicate that heteroscedastic noise may be a problem, but does not preclude the existence of additional components. In contrast, if nonlinearities are the dominant cause of variation, a two-component model will often dramatically flatten the fit error of its one-component counterpart, but this depends to some extent on the nature of the data. This would clearly give

a false indication of a second component in the absence of a nonlinear one-component model.

Although it has been shown that the one-component nonlinear model will decrease the likelihood of a false positive indication of a second component, it remains to be demonstrated that it does not increase the likelihood of a false negative in the case of a two-component mixture. Figure 3.10 is an illustration of this case. In this example, overlapped elution profiles (gaussian, $\sigma = 10$ s, $\Delta t = 10$ s, $R_s = 0.25$) with a 10:1 concentration ratio were simulated. The spectral profiles (gaussian, $\sigma = 10$ nm, $\Delta\lambda = 10$ nm) were set at equal heights and adjusted to give an overall absorbance maximum of 0.5 AU. The noise level was set at 0.04% of the maximum (homoscedastic) and a linear response was assumed. It is clear from Figure 3.10 that both the linear and nonlinear one-component models fail when there are two components present, whereas the two-component model gives a flat trace for the fit error, as expected. Although there are an infinite number of ways to vary the parameters for a two-component system, this example is representative. Thus, the use of one-component nonlinear models does not significantly diminish the ability of the algorithm to detect an impurity.

The corrections for heteroscedastic noise and nonlinearity can be combined into a single model as already noted. An experimental example is given below that deals with this case. However in most cases one of these two effects will dominate depending on the system under study.

In order to validate the results obtained from the simulations, several one

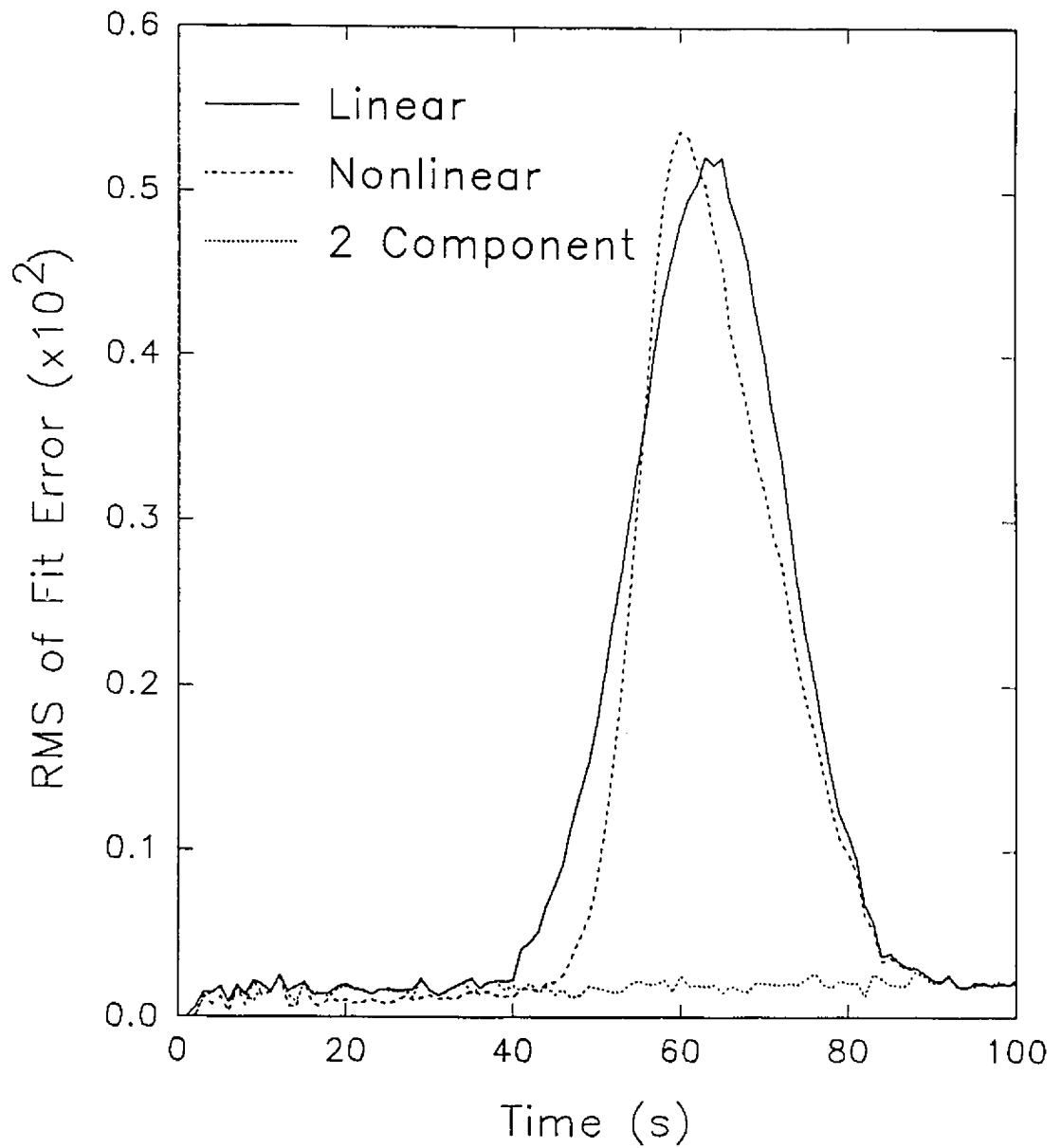


Figure 3.10 Effect of a two-component mixture on linear one-component, nonlinear one-component, and linear two-component EPA models.

component elution profiles were investigated using the stopped-flow-injection apparatus described in the section 3.5. One difficulty was isolating the effects of heteroscedastic noise and nonlinear response from one another so that each of the modified algorithms could be tested independently. Both effects occur at high absorbances, so it was necessary to modify measurement conditions somewhat to enhance individual contributions.

To isolate the effects of heteroscedastic noise, the dye methyl orange was used as the analyte. The chromophore of this dye has relatively broad spectral features and it was felt that this would reduce nonlinearities arising from polychromatic radiation. It was also necessary to place a neutral density filter (ca. 1.3 AU) in the light path to amplify the noise level. Figure 3.11a shows the effects of heteroscedastic noise on the modified and unmodified one-component models. For this experiment, wavelengths between 420 and 570 nm were used with steps of 6 nm. The stopped-flow apparatus was used to generate a Gaussian peak with a width (σ) of 10 s centered at 30 s, with an effective sampling interval of 1 s. The effective sampling period was 70 s and the maximum absorbance was 1.4 AU at 504 nm. The noise model described earlier for the diode array was employed to estimate errors in absorbance in this case. It is clear from the figure that the use of the unmodified EPA model would lead to the false conclusion that there is a second component present. The trace of the RMS of the fit error for the modified model is flat, however, indicating that such a conclusion is erroneous. These experimental results also bear a striking similarity to the simulated results

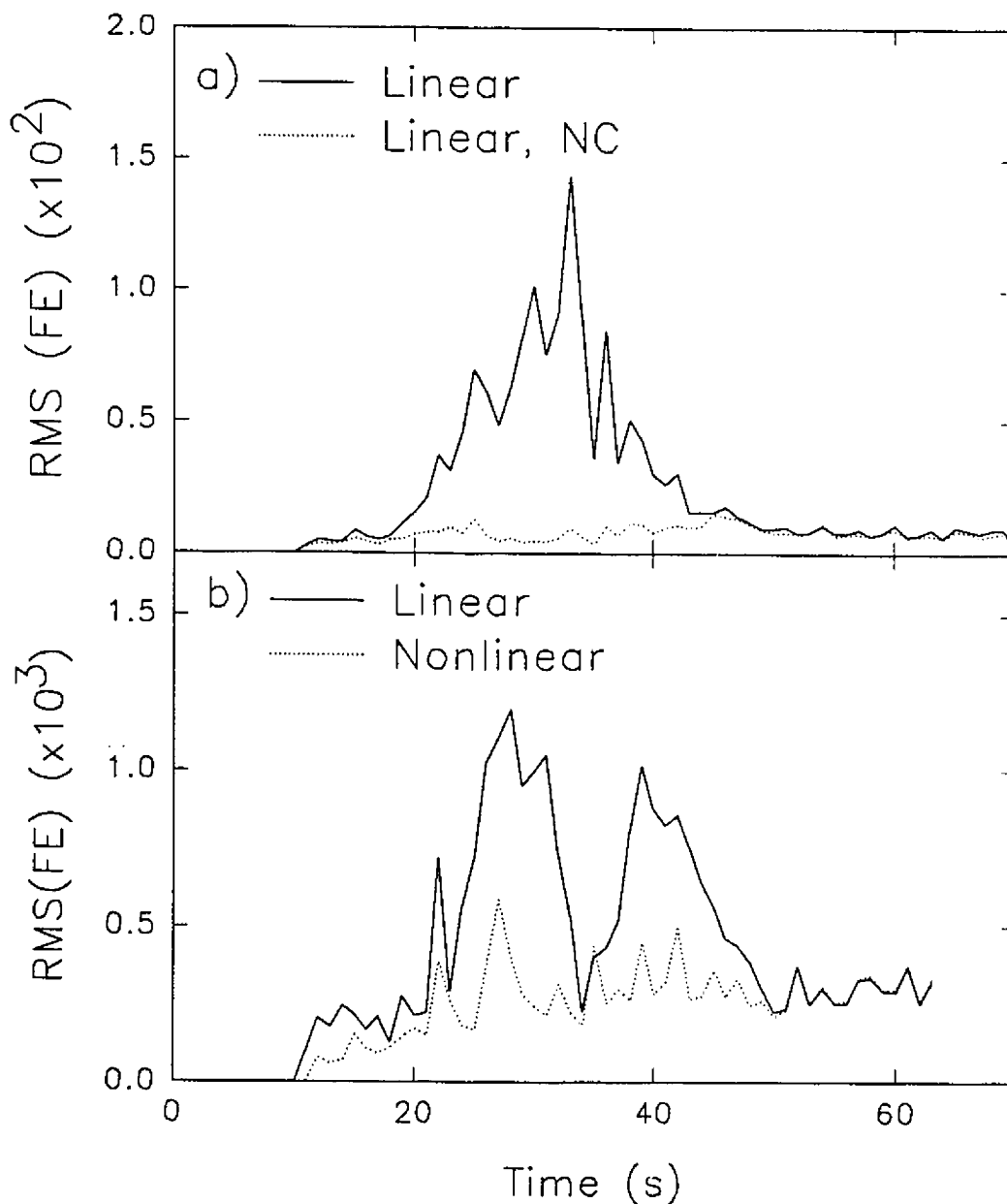


Figure 3.11 a) Effect of heteroscedastic noise on unmodified (solid line) and modified (dashed line) EPA models for an experimental system (10 ppm methyl orange). b) Effect of nonlinear detector response on linear (solid line) and nonlinear (dashed line) EPA models for an experimental system (3 wt% PrCl_3).

described above, validating the noise model for the DAD.

Experimental data in which the nonlinear response was accentuated were obtained using PrCl_3 as the chromophore. As noted previously, the sharp spectral features of these solutions exacerbate the effects of the polychromatic source, even at relatively low absorbances. Figure 3.11b shows the EPA traces obtained using the linear and nonlinear (second order) one-component models. For this study, wavelengths between 436 and 486 nm were used with steps of 2 nm. No neutral density filter was employed and the maximum absorbance of the solution (3 wt% PrCl_3) was 0.75 AU at 444 nm, which is below the level at which heteroscedastic noise starts to become significant. The typical bimodal deviation of the trace for the linear one-component model is evident, but missing for the modified model, indicating that at this level the correction for polychromatic radiation is successful.

To test the incorporation of both correction factors into the one-component model, a solution which was 8 wt% PrCl_3 was used (maximum absorbance = 1.77 AU). Although the trace of the fit error is much flatter for the modified model in Figure 3.12, it is still higher than one would expect for a fully compensating model. It is postulated that the higher absorbance levels necessary to enhance the heteroscedastic noise in this case also bring about larger contributions from stray light and reduce the effectiveness of the nonlinear correction. The high concentration of the chromophore also introduces some refractive index effects. Even the modified model will approach its limits under

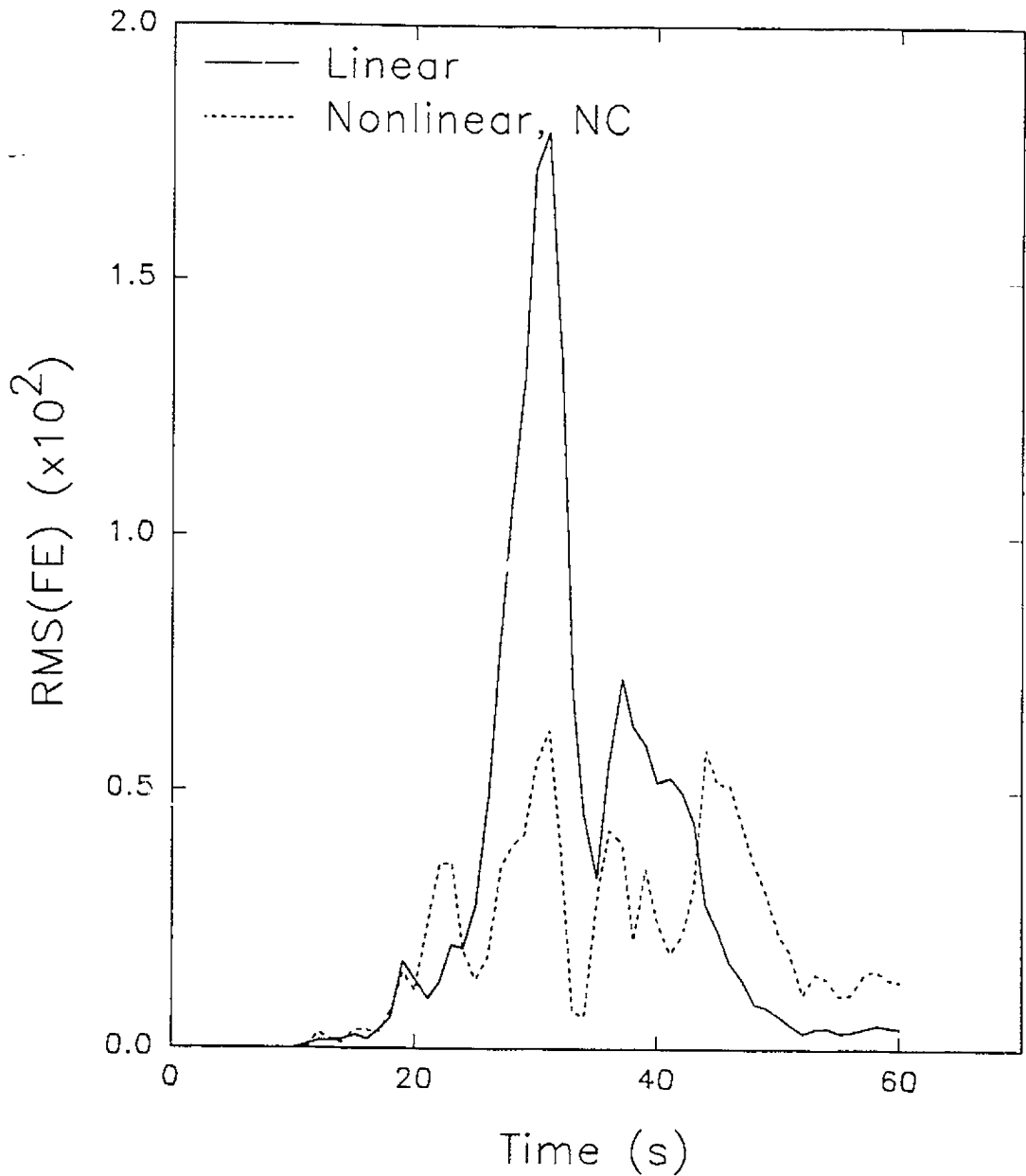


Figure 3.12 Comparison of unmodified EPA model (solid line) with model incorporating corrections for both heteroscedastic noise and nonlinearity (dashed line). Experimental data (8 wt% PrCl_3) were used.

such demanding spectroscopic conditions.

3.7 Conclusions

The analysis of multivariate data sets such as spectrochromatograms carries the risk of extracting incorrect information when the models used are inappropriate. As pointed out by Sánchez *et al.* [63] the most important barriers to reliable peak purity detection using multivariate data sets are the problems of nonlinear detector response and heteroscedastic noise. This study outlined when and why these effects occur and showed how to effectively correct for each. The problem of heteroscedastic noise has been treated to a limited degree in the literature. The algorithm presented here is the first treatment of correction for nonlinear detector response. The methodology could easily be adapted to other systems. The modified EPA algorithm is a straight-forward, robust solution to peak purity detection.

Disordered Sets 1: Conductivity Prediction

4.1 From Order to Disorder

The objective of chemometrics when applied to multivariate data sets is to find a model that adequately describes the systematic variations in the data. In Chapters 2 and 3, the model sought was one based on the spectra and elution profiles of the constituent components of the mixtures. It was demonstrated that this process can be aided by the fact that the data sets can be considered to be ordered. Because the rank of the data changes systematically with time or some other ordering variable, the estimation of overall rank and the identification of pure component regions were simplified. As a consequence, the underlying model can be estimated more reliably. In many analytical applications, however, the data are disordered, showing no systematic changes in rank. This may be an inherent property of the data set (*i.e.* all subsets have a rank equal to the overall rank) or may be due to the fact that the ordering variable is unknown or not measured. For the latter case, it will be shown in Chapter 6 that the underlying order may be established by a suitable algorithm. In the former case, a purely empirical approach to model development may be warranted. In this chapter, several empirical modelling methods are applied to a disordered data set of practical interest. In Chapter 5, a method for improving such empirical models through term

selection is investigated.

The problem under consideration is the estimation of conductivity in non-brine water samples from the concentrations of major ions and other analytical measurements. This problem is of interest for several reasons. First, conductivity prediction is one way of assuring quality control in the analysis of a large number of water samples. In routine analysis, errors can occur as the result of erroneous measurements or when the results are compiled. Since the conductivity depends on the concentration of the ionic species in the solution, gross errors in the determination of these ions can be detected if the measured conductivity is compared with that predicted by some model. The present standard method for quality control in this manner is based on a semi-empirical model developed by Rossum in 1975 [64]. This model has a dubious theoretical foundation and was only tested on one data set with relatively low conductivities (*i.e.* $<1400 \mu\text{S cm}^{-1}$). Thus, a second reason for carrying out this study was to evaluate the Rossum model. A third reason is that the estimation of conductivity in multi-electrolyte systems is a problem of general interest (*e.g.* in capillary electrophoresis) and is virtually intractable from a purely theoretical standpoint. Finally, as will be shown, this problem presents a formidable challenge to empirical modelling methods since semi-empirical approaches work remarkably well. In this chapter, modifications to the Rossum model and the use of a number of chemometric modelling techniques are used to help propose a better model for predicting conductivity. However, in order to place these techniques in a proper framework, a brief historical

perspective on conductivity prediction will be given.

4.2 Background

4.2.1 Theoretical and Semi-Empirical Modelling Techniques

In the late 1800's, Kohlrausch experimentally observed that for many dilute aqueous solutions of electrolytes their equivalent molar conductances (Λ) varied as the square root of their equivalent concentrations. (In this work, "conductance" will generally be used to refer to the molar quantity, whereas "conductivity" will be used for the extensive property.) He also proposed the Kohlrausch Law of the independent migration of ions for conductance at infinite dilution (Λ°) [65,66]. The law states that the conductance of an electrolyte at infinite dilution is the sum of contributions from each ion present. For a single electrolyte, the conductance at infinite dilution is the sum of the limiting equivalent conductances for the anion (λ_-°) and cation (λ_+°).

$$\Lambda^\circ = \lambda_+^\circ + \lambda_-^\circ \quad (4.1)$$

In 1923 Debye and Hückel described the theoretical basis for the concentration dependence of conductivity. Later, in 1926, Onsager modified their model and it can be summarized by the following formula.

$$\Lambda = \Lambda^\circ - (\alpha\lambda_+^\circ + \beta)\text{C}^{1/2} \quad (4.2)$$

In this equation, C is the equivalent concentration of the electrolyte and α and β are theoretically derived constants that depend on the temperature, dielectric

constant, viscosity and other physical constants. Equations for α and β are presented in slightly different forms by several authors [64-66]. A form similar to that used by Rossum will be employed here. The constants α and β are associated with relaxation and electrophoretic effects respectively. Both of these effects decrease the equivalent conductance of an electrolyte at high concentrations [66,67].

Although theoretical equations exist for predicting conductivity for multi-electrolyte cases [68] they are quite complex. Rossum suggested that the conductivity (G) of a water sample could be treated by considering the multi-electrolyte solution as a binary electrolyte in which the ionic concentrations and charges were represented as average or effective values. The Rossum Model (RM) is

$$G = G_+^o + G_-^o - \left[\frac{W\Lambda_o}{K_1(Z_+ + Z_-)} + K_2 \right] [(Z_+ + Z_-)C']^{3/2} \quad (4.3)$$

where

$$W = Z_+ Z_- \frac{2Q}{1 + \sqrt{Q}}$$

$$Q = \frac{Z_+ Z_- \Lambda_o^2}{(Z_+ + Z_-) (Z_+ \lambda_-^{\prime} + Z_- \lambda_+^{\prime})}$$

$$K_1 = \frac{24\pi(\epsilon_o DRT)^{3/2}}{\theta F^2}$$

$$K_2 = \frac{10^5 \theta F^2}{6\pi(\epsilon DRT)^{1/2} \eta}$$

Rossum used values of $K_1 = 115.2 \text{ mM}^{1/2}$ and $K_2 = 0.668 \text{ } \mu\text{S cm}^{-1} \text{ mM}^{-3/2}$ at 25 °C. The conductivity (G) calculated will be in units of $\mu\text{S cm}^{-1}$. Additional

parameters as defined by Rossum are as follows:

$$\begin{aligned}
 G_+^o &= \sum c_+ \lambda_+^o & G_-^o &= \sum c_- \lambda_-^o \\
 z_- &= \frac{\sum c_- z_-^2}{\sum c_- z_-} & z_+ &= \frac{\sum c_+ z_+^2}{\sum c_+ z_+} \\
 \lambda_+' &= \frac{G_+^o}{\sum c_+} & \lambda_- ' &= \frac{G_-^o}{\sum c_-} \\
 \Lambda_o' &= \lambda_+' + \lambda_- ' & C' &= \frac{(\sum c_+ + \sum c_-)}{2}
 \end{aligned} \tag{4.4}$$

The summations in the above expressions are over cations and anions that are included in the model. For a summary of symbols and their units see Table 4.1. The Rossum Model works reasonably well for solutions of relatively low conductivity ($<1400 \mu\text{S cm}^{-1}$) and has been recommended as a standard method since 1989 [69]. However, Rossum did not provide a theoretical basis for his definitions of effective concentration and charge, and only tested the model on a single set. Furthermore at higher conductivities, systematic deviations in the model become apparent.

Two modifications to the Rossum Model were considered in this work. The first dealt with the definition of effective charge and the second was the addition of an another term to the model. The alternative charge definition considered was based on a weighted average of individual charges. This new effective charge definition was

Table 4.1 List of symbols and units related to conductivity measurements.

| | |
|--|--|
| c_j, c_+, c_- | Equivalent concentration of an ion (mM). |
| C' | Equivalent concentration of a salt (defined by Rossum). (mM). |
| D | Dielectric constant of a solution. |
| e | Proton charge (C) |
| ϵ_0 | Vacuum permittivity ($J^{-1} C^2 m^{-1}$) |
| F | Faraday's constant ($C mol^{-1}$) |
| G | Solution conductivity ($\mu S cm^{-1}$) |
| G°_+, G°_- | Ionic conductivities in a many electrolyte solution (defined by Rossum). ($\mu S cm^{-1}$) |
| η | Solution viscosity (P) |
| $\lambda^{\circ}_+, \lambda^{\circ}_-$ | Limiting ionic equivalent conductance ($S cm^2 mol^{-1}$) |
| λ'_+, λ'_- | Limiting ionic equivalent conductance (defined by Rossum). ($S cm^2 mol^{-1}$) |
| Λ' | Equivalent conductance of a solution (defined by Rossum). ($S cm^2 mol^{-1}$) |
| R | Gas constant ($J mol^{-1} K^{-1}$) |
| z_+, z_- | Ionic charge (no units) |
| Z_+, Z_- | Effective charge in multiple electrolyte solution (defined by Rossum). |

$$Z_+ = \frac{\sum c_i z_i}{\sum c_i} \quad (4.5)$$

and likewise for Z_- . Rossum's only justification for using the previous definition (see Equation 4.4) was that it worked better. This assumption regarding the definition of charge was tested.

Extensions to Equation 4.2 were suggested by Shedlovsky in 1932 [70]. Shedlovsky observed, upon rearrangement of Equation 4.2 to,

$$\Lambda^o = \frac{(\Lambda + \beta C^{1/2})}{(1 - \alpha C^{1/2})} \quad (4.6)$$

that a plot of the right hand side of Equation 4.6 versus C was not constant but varied linearly. He suggested that a term in C be added to equation 4.2. Shedlovsky justified this modified equation on the basis that the equation developed by Debye, Hückel and Onsager was only valid as a limiting expression.

Thus, Equation 4.6 now becomes,

$$\Lambda^o = \frac{(\Lambda + \beta C^{1/2})}{(1 - \alpha C^{1/2})} - BC \quad (4.7)$$

where B is a fitted parameter and C , Λ^o , α and β were previously defined. In terms of Λ , rearrangement of Equation 4.7 gives,

$$\Lambda = \Lambda^o C - (\alpha \Lambda^o + \beta) C^{3/2} + BC(1 - \alpha C^{1/2}) \quad (4.8)$$

Using Rossum's approximation for many electrolytes,

$$G = G_{RM} + BC'(1-\gamma\sqrt{C}) \quad (4.9)$$

where G_{RM} is the conductivity accounted for by the Rossum Model and γ is given by

$$\gamma = \frac{W\Lambda^0(Z_+ + Z_-)^{1/2}}{K_1} \quad (4.10)$$

4.2.2.1 Empirical Modelling Techniques

Various chemometric techniques were used in this work an attempt to improve the prediction abilities of conductivity models. These techniques include multiple linear regression (MLR), principal component regression (PCR), partial least squares (PLS), continuum regression (CR) and artificial neural networks (ANN). Each of these techniques are discussed briefly below and references are given for more thorough treatments. However, a brief introduction to calibration and prediction will be presented.

In any calibration problem, one is normally trying to determine the model that best reflects the physical features of the system of interest. This involves both the selection of the form of the model and the estimation of parameters associated with it. To do this one usually builds a model based on one data set, called the calibration set. The model is generally built by minimizing the standard error of calibration (SEC), also known as the standard error of the estimate, and in this case is given by,

$$SEC = \sqrt{\sum_{i=1}^{m_{cal}} \left(\frac{(G_i - G_i^p)^2}{(m_{cal} - k)} \right)} \quad (4.11)$$

where G_i is the measured conductivity, G_i^p is the conductivity predicted by the trial model, m_{cal} is the number of calibration samples in the calibration set, and k is the number of parameters in the model. Unfortunately, a small SEC does not guarantee a good model since overfitting may be a problem. Because of this, validation of the model is usually carried out using an independent prediction data set, which measures the ability of the model to generalize. The measure of prediction ability chosen for this work is the standard error of prediction (SEP). The SEP is calculated as,

$$SEP = \sqrt{\sum_{i=1}^{m_{pred}} \left(\frac{(G_i - G_i^p)^2}{m_{pred}} \right)} \quad (4.12)$$

where G_i and G_i^p have the same definitions, and m_{pred} is the number of prediction samples. Normally the best calibration model is assumed to be the model that provides the best prediction ability, *i.e.* the lowest SEP.

In many instances data are pretreated before calibration is performed. The data pretreatment performed in this work is known as autoscaling, a two step process in which, for each variable, the data are adjusted so that they have a zero mean and a unit standard deviation. Mean-centering eliminates the need for calculating an intercept, while adjusting the data to a standard deviation of one

brings all variables within the same range, thus eliminating the excessive influence of variables with inherently large magnitudes. In this work autoscaling was used for MLR, PCR, PLS and CR.

4.2.2.2 Multiple Linear Regression

MLR [3] is probably the most familiar of the modelling techniques presently considered. In this work the equivalent concentrations of the ions were used to predict the conductivity of the sample. The MLR model applied can be expressed for the i th sample with n ions as

$$G_i = \sum_{j=1}^n (b_j c_{ij}) \quad (4.13)$$

The b 's are the regression coefficients that are to be determined by least squares. The above equation can be expressed more compactly in matrix form for m samples as,

$$\mathbf{G} = \mathbf{C}\mathbf{b} \quad (4.14)$$

where \mathbf{G} is a vector of dimensions $m \times 1$ of measured conductivities, \mathbf{C} is a matrix of dimensions $m \times n$ of equivalent concentrations and \mathbf{b} is a vector of dimensions $n \times 1$ of regression coefficients. Solution of the regression coefficients in Equation 4.14 is given as:

$$\mathbf{b} = (\mathbf{C}^T \mathbf{C})^{-1} \mathbf{C}^T \mathbf{G} \quad (4.15)$$

where Tr indicates the transpose of a matrix.

4.2.2.3 Principal Component Regression

While MLR is an effective modelling strategy for many situations, it suffers from two main disadvantages. First, for a reliable model it is generally required that the number of samples is much greater than the number of parameters to be estimated. Second, models are generally sensitive to collinearities within the modelling variables which can lead to unstable solutions. PCR [2] reduces these problems by using principal component analysis (PCA, discussed in Chapter 2) to compress the variability of the original variables in \mathbf{C} into a smaller set of new variables called latent variables. PCA decomposes the original data matrix into the product of a scores matrix, \mathbf{T} , and a loadings matrix, \mathbf{L} ,

$$\mathbf{C} = \mathbf{T}\mathbf{L} \quad (4.16)$$

The columns of the scores matrix represent the latent variables that are used for PCR. The total number of latent variables will be equal to the number of original variables, n , but by using only the first p latent variables (columns of \mathbf{T}), a data compression effect is achieved which reduces the number of variables used. These variables will also account for the greatest variability in the original data matrix and model any collinearity that exists. This data compression also has the effect of removing some of the random error from the data matrix (\mathbf{C}) and tends to lead to better predictive ability than MLR. The PCR model is developed by regressing the PCA scores according to the model,

$$G_i = \sum_{j=1}^p t_j b_j$$

$$\mathbf{G} = \mathbf{T}_p \mathbf{B} \quad (4.17)$$

where T_p is the reduced set of latent variables. The solution is analogous to eqn. 4.14. In PCR, the number of latent variables to be used in the development of the model, p , also needs to be considered.

4.2.2.4 Partial Least Squares

One difficulty with PCR is that, in determining the latent variables, only the variance in the concentration matrix, \mathbf{C} , is considered. The correlation of these variables with the desired property (\mathbf{G}) is not considered until the regression step, and at that point latent variables containing important information in this regard may already have been dropped. PLS [71] proceeds in a manner similar to PCR, but the latent variables are modified in such a way that they also consider correlation of the PCA eigenvectors with the property to be predicted. In this way, the latent variables are expected to have better predictive properties, and variance in \mathbf{C} which is not correlated with \mathbf{G} has a diminished influence. PLS is particularly effective when unrelated sources of variance (*e.g.* interferences) are present in the data. Except for the manner in which latent variables are calculated, the procedure for PLS regression is analogous to that described for PCA [72] and will not be included here. As with PCR, the number of latent variables used, p , is an

important parameter. In both cases, results are the same as MLR when $p=n$.

4.2.2.5 Continuum Regression

MLR, PCR, and PLS are all subsets of a more general approach to regression which has been called continuum regression (CR). This method, described by Lorber *et al.* [72], rationalizes the difference among the above-mentioned methods in terms of the extent to which correlations of the variables with the desired quantity are balanced against correlations among the variables themselves. At one end of the spectrum, MLR does not consider correlations among the predictor variables at all. At the other end, PCR considers only the relationship among the predictor variables generating the latent variables. PLS exists somewhere in between, considering both the *inner* and *outer* relationships. It has been noted that there are, in reality, a continuum of methods between MLR and PCR, and that these are characterized by a so-called *power term*, q , that describes the extent to which correlations with the desired property are considered in generating the latent variables. MLR corresponds to $q = 0$, PLS to $q = 1$, and PCR to $q = \infty$ (or in practice some large value). A continuum of solutions exists over the range of q , although in practice all solutions become equivalent after, say, $q = 10$.

CR can be regarded as a method which unifies the regression methods of this type. The *best* solution will depend on the problem and is reflected in the optimum choice of two variables: p , the number of latent variables, and q , the

power factor. The optimum model can be determined from a *continuum regression surface* in which the prediction error (SEP) is plotted as a function of p and q . The optimum model should correspond to the minimum in this surface.

4.2.2.6 Neural Networks

Neural network modelling is a technique based on how the brain is believed to process information. Biological neural networks are very complex and the computer simulations of these networks are known as artificial neural networks (ANNs). ANNs were first developed in the 1960's, but did not become widely used until the early 1980's through the efforts of Rumelhart and McClelland [73]. A recent tutorial article by Wythoff [74] provides a good discussion on neural networks in analytical chemistry. Although ANNs will be described briefly below, a thorough discussion on the neural networks applied here is available elsewhere [75-78].

Biological neural activities are based on a large number of neurons. These neurons process a number of inputs and produce an output that may combine with the outputs of other neurons and enter another neuron. A schematic diagram of an ANN similar to the one used in this work is shown in Figure 4.1. The ANN consists of a number of input neurons (equal to the number of independent variables), a number of output neurons (equal to the number of dependent variables), and a number of intermediate neurons (whose number depends on the complexity of the problem). These *layers* are known as the input, output, and

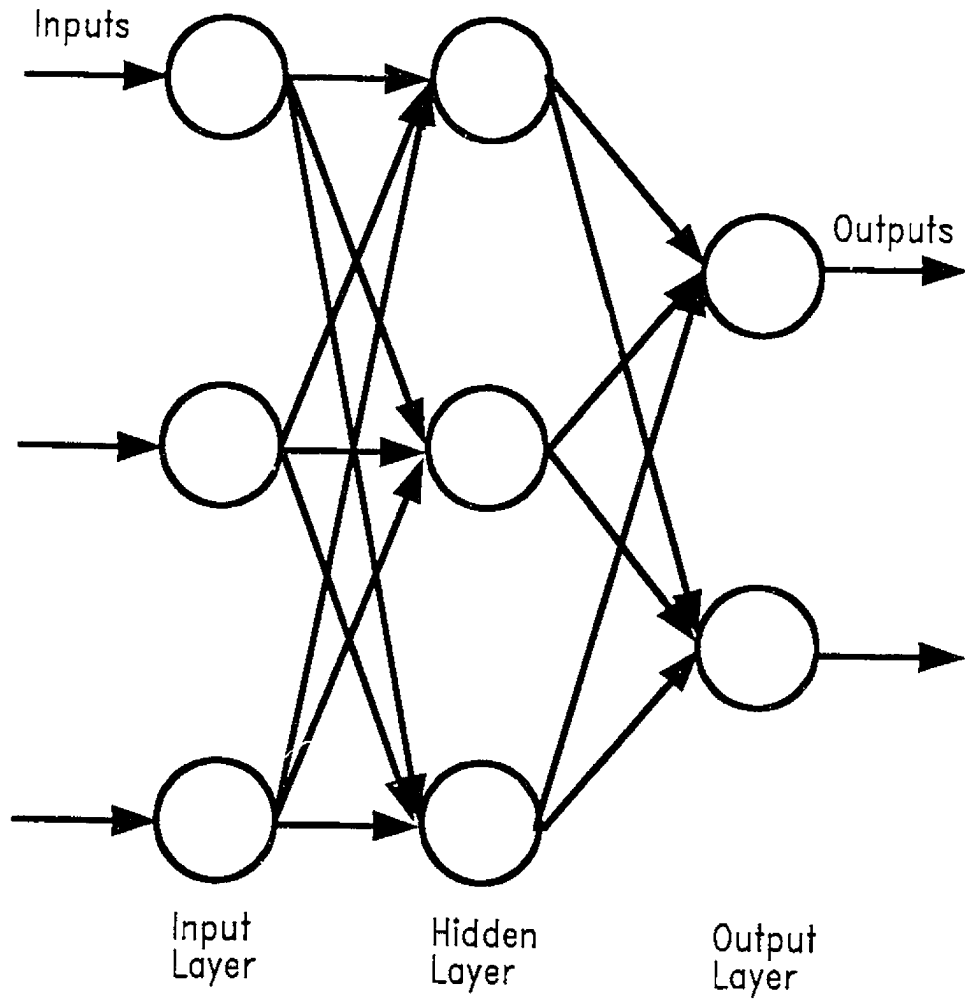


Figure 4.1 Neural Network topology like the one used in this work.

hidden layers respectively. The circles in Figure 4.1 represent the nodes, or processing elements of the ANN. It should be noted that the nodes on the input layer are simply convenient for most diagrams and do not do any processing (except perhaps a linear transformation for scaling purposes). A wide variety of geometries are possible based on this simple framework, the most common being the addition of more hidden layers.

The most common processing performed by nodes in the hidden and output layer involves a two-step calculation. First, a weighted sum, y_i , of the inputs to the node is calculated. For node i , this corresponds to,

$$y_i = \sum_{j=1}^n w_{ij} x_{ij} \quad (4.18)$$

where w_{ij} is the weight corresponding to input j of node i , x_{ij} is input j to node i , and the summation is over the number of input nodes. The output of the node, o_i , is given by a sigmoidal activation function,

$$o_i = \frac{1}{1 + e^{-y_i}} \quad (4.19)$$

which has a range of 0 to 1. Other activation functions can also be used, and a bias term is sometimes included in the exponent. In practice, it is usually easier to account for this bias by including an input fixed at unity for each layer.

The parameters associated with the scaling and transfer function are optimized for the outputs being predicted. The implementation of neural networks consists of a two step process involving training and validation. The training step

consists of presenting to the neural network a number of samples for which both the independent and dependent variables are known so that the operational parameters associated with the network can be optimized. These parameters are determined by a learning or fitting algorithm that makes adjustments until the error between the actual and predicted values is deemed acceptable. The so-called back-propagation algorithm is most widely used for training. The validation step involves presenting a different set of samples to the trained neural network and calculating the standard error of prediction (e.g. SEP) to evaluate performance.

4.3 Experimental

4.3.1 Chemical Analysis

Fenwick Laboratories of Halifax, Nova Scotia (now MDS. Environmental Services) analyzed all of the water samples reported in this work. The experimental procedures used in their analysis are based on standard methods of analysis [69] and will only be summarized here. Metallic ions (Na, K, Ca, Mg, Fe, Mn, Cu, Zn) were determined by flame atomic absorption on a Varian SpectrAA 40 spectrometer (Varian Canada, Geogretown, ON). Colorimetric methods were used for chloride, ammonia, nitrate + nitrite, orthophosphate, reactive silica, and alkalinity. These were carried out on an automated COBAS FARA centrifugal analyzer (Roche Diagnostics, Mississauga, ON). Chloride was determined using mercuric thiocyanate in the presence of Fe^{2+} . The method for ammonia utilized the formation of the blue indophenol by reaction with hypochlorite and alkaline phenol

(phenate method). Nitrate was reduced with hydrazine sulfate and total nitrate plus nitrite was then determined by reaction with sulphanilamide and N-(1-naphthyl)ethylenediamine. Orthophosphate and reactive silica were determined by reaction with ammonium molybdate and ascorbic acid (heteropoly blue method). Selectivity for phosphate was achieved kinetically, while, for silica, the phosphate interference was removed through the addition of oxalic acid. Alkalinity, which is the sum of all titratable bases [57], was measured colorimetrically using an automated methyl orange procedure. In a typical water sample, alkalinity depends predominately on the concentration of carbonate, bicarbonate, and hydroxide ions, so the first two can be estimated indirectly from the pH and alkalinity as was done here. Sulfate was determined by a turbidimetric method on a nephelometer (Turner Designs Model 40, Mountain View, CA) using BaCl_2 to precipitate BaSO_4 . Total organic carbon (TOC) was determined using the persulfate ultraviolet oxidation method performed on a Technicon autoanalyzer (Technicon, Tarrytown, NY) with a phenolphthalein indicator. A Radiometer GK2401C glass electrode (Radiometer America, Westlake, OH) was used to measure pH and a flow-through conductance cell (Radiometer Model CDC314) was used to measure conductivity, which was corrected to 25°C. A summary of the parameters measured is given in Table 4.2.

4.3.2 Computational Aspects

The computer analysis was conducted on a 486-based computer operating

Table 4.2 Statistical summary of water quality parameters.

| Measured Quantities | | | | | | | |
|-----------------------|------------------------|-------|------------|------------|-----------|--------|--|
| Parameter | Units | Mean | Min. Value | Max. Value | Std. Dev. | Median | 25 th & 75 th percentile |
| Sodium | mM | 2.09 | 0.01 | 71.7 | 6.17 | 0.66 | 0.24 1.71 |
| Potassium | mM | 0.36 | 0.00 | 22.5 | 1.55 | 0.06 | 0.03 0.18 |
| Calcium | mM | 2.17 | 0.00 | 17.5 | 2.42 | 1.66 | 0.61 2.74 |
| Magnesium | mM | 0.70 | 0.00 | 8.52 | 0.86 | 0.55 | 0.15 0.93 |
| Sulfate | mM | 0.82 | 0.01 | 16.2 | 1.97 | 0.23 | 0.09 0.52 |
| Chloride | mM | 1.94 | 0.00 | 90.3 | 7.42 | 0.62 | 0.20 1.43 |
| Nitrate+Nitrite | mM | 0.39 | 0.00 | 7.43 | 1.18 | 0.01 | 0.00 0.05 |
| Ammonia | mM | 0.20 | 0.00 | 14.6 | 1.03 | 0.00 | 0.00 0.01 |
| Orthophosphate | mM | 0.01 | 0.00 | 1.19 | 0.08 | 0.00 | 0.00 0.00 |
| Reactive Silica | mM | 0.37 | 0.00 | 1.80 | 0.25 | 0.36 | 0.18 0.49 |
| Iron | mM | 0.05 | 0.00 | 1.42 | 0.21 | 0.00 | 0.00 0.01 |
| Manganese | mM | 0.01 | 0.00 | 0.29 | 0.02 | 0.00 | 0.00 0.00 |
| Copper | mM | 0.00 | 0.00 | 0.02 | 0.00 | 0.00 | 0.00 0.00 |
| Zinc | mM | 0.01 | 0.00 | 2.49 | 0.12 | 0.00 | 0.00 0.00 |
| pH | | 7.54 | 4.3 | 12.0 | 0.7 | 7.6 | 7.2 8.0 |
| Alkalinity | mg/L CaCO ₃ | 214 | 0.00 | 2500 | 228 | 200 | 70 280 |
| TOC | mg/L Carbon | 11.6 | 0.5 | 850 | 64.6 | 1.7 | 0.8 4.9 |
| Conductivity | µS cm ⁻¹ | 801 | 4.1 | 10400 | 1070 | 528 | 288 956 |
| Calculated Quantities | | | | | | | |
| Carbonate | mM | 0.02 | 0.00 | 0.76 | 0.05 | 0.01 | 0.00 0.02 |
| Bicarbonate | mM | 4.24 | 0.00 | 50 | 4.56 | 3.94 | 1.34 5.46 |
| Hardness | mg/L CaCO ₃ | 287 | 0.00 | 2100 | 301 | 236 | 79.4 360 |
| Langlier Index | | -0.65 | -9.15 | 4.47 | 1.73 | 0.03 | -1.17 0.38 |
| Saturated pH | | 8.18 | 6.04 | 15.8 | 1.26 | 7.75 | 7.30 8.61 |

under DOS 6.0 and Windows 3.1 (Microsoft Corp., Redmond, WA). Excel 4.0 (Microsoft Corp.) and MATLAB 4.0 (MathWorks, Inc., Natick, MA.) were used for data treatment. In addition, some of the MATLAB programs used were based on code from the PLS Toolbox written by Barry Wise of Battelle Pacific Northwest Laboratories (Richland, WA) [79].

The Rossum Model applied here differs slightly from Rossum's original presentation [64]. Rossum did not include NH_4^+ , but it was found that its inclusion in the model gave improved performance, so it was included in our analysis. The limiting equivalent conductance for ammonium ion that was used was $73.5 \text{ S cm}^2 \text{ mol}^{-1}$ [80]. In addition, the λ° used for carbonate was $63.5 \text{ S cm}^2 \text{ mol}^{-1}$ [81], slightly different from Rossum's value.

For all of the models studied, the SEP was determined for two conductivity ranges: $<11000 \mu\text{S cm}^{-1}$ and $<1400 \mu\text{S cm}^{-1}$. The former range was selected because this included all samples presently available and will be referred to as the "all samples" range for the rest of the work. The $<1400 \mu\text{S cm}^{-1}$ range was the range of samples analyzed by Rossum. The number of samples in these ranges was 422 and 483 for $<1400 \mu\text{S cm}^{-1}$ and all samples, respectively. For calibration, 200 samples were randomly selected from each of these ranges and the remaining samples were used for the prediction. Unless otherwise stated, this was repeated 50 times to estimate the SEP.

4.4 Results and Discussion

4.4.1 Semi-Empirical Models

For the Rossum models, and unless otherwise stated, the set of ions used was Na^+ , K^+ , Ca^{2+} , Mg^{2+} , SO_4^{2-} , Cl^- , NO_3^- , NH_4^+ , CO_3^{2-} and HCO_3^- . The concentration of NO_3^- used in the computer analysis is actually the concentration of NO_3^- plus NO_2^- for the samples analyzed, but since it is unusual to have significant amounts of nitrite present, the assumption is regarded as valid. The NH_4^+ concentration was obtained from the total ammonia concentration and pH. These same 10 ions were included in all calibrations performed but additional parameters were also used for some of the neural network and MLR models.

Figure 4.2 shows how the Rossum Model performs over two conductivity ranges: a) $<1400 \mu\text{S cm}^{-1}$ and b) all samples. The solid line in Figure 4.2 is the measured conductivity versus the measured conductivity. It therefore has a slope of unity and represents the ideal model. The circles show the predicted conductivity based on the Rossum Model plotted against the measured conductivity. From Figure 4.2b the Rossum Model shows a systematic deviation above $2000 \mu\text{S cm}^{-1}$. Modifications are needed to this standard model to correct for this deviation.

As mentioned previously, two main modifications to the original Rossum Model were proposed for this work: the use of an alternate definition for the effective charge and the addition of another term to the model. To compare the influence of each of these modifications, the Rossum Model (RM) was compared

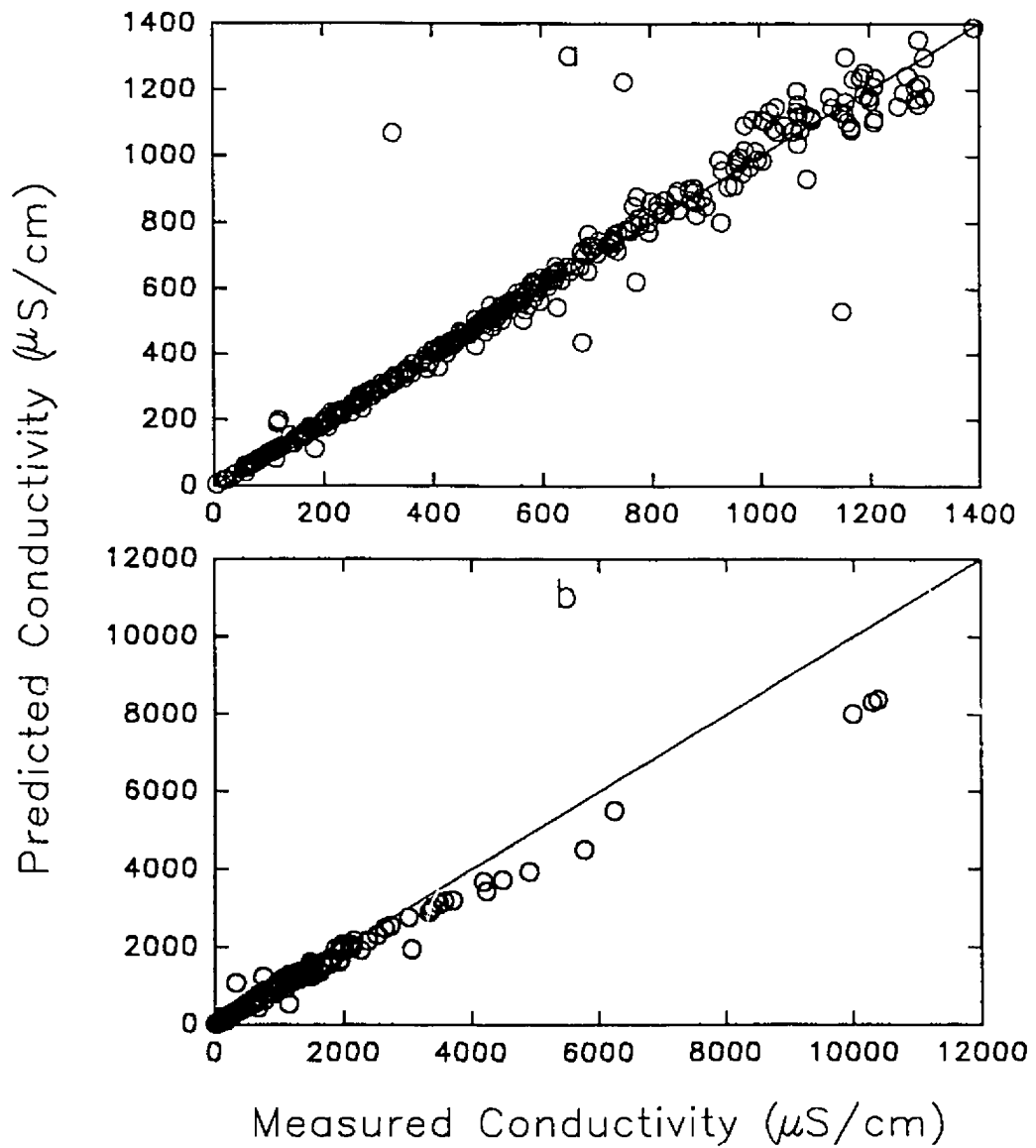


Figure 4.2 Comparison of measured and predicted conductivity for test data using the Rossum model over two ranges: a) $< 1400 \mu\text{S cm}^{-1}$, b) all samples.

to three other models. These other models are summarized in Table 4.3. The results for the Rossum and its variations are shown in Table 4.4.

Table 4.3 Summary of modified Rossum models.

| Model | Acronym | Comment |
|--------------------------------|---------|---|
| Modified Rossum Model | MRM | RM with alternative charge definition |
| Extended Rossum Model | ERM | RM with additional term |
| Extended Modified Rossum Model | EMRM | RM with alternative charge definition and additional term |

Table 4.4 Standard errors of prediction for models investigated.

| Model | Standard Error of Prediction ($\mu\text{S cm}^{-1}$) | |
|-------|--|-------------|
| | < 1400 $\mu\text{S cm}^{-1}$ | all samples |
| RM | 64 | 211 |
| MRM | 67 | 148 |
| ERM | 67 | 109 |
| EMRM | 67 | 95 |
| MLRa | 76 | 99 |
| MLRb | 68 | 89 |
| MLRc | 61 | 88 |
| PCR | 69 | 93 |
| PLS | 69 | 92 |
| CR | 69 | 92 |
| NN | 79 | 283 |

The modification of charge definition improved the prediction ability for the both

ranges. For the $<1400 \mu\text{S cm}^{-1}$ range, the Rossum Model provides marginally better results than the MRM, ERM and EMRM. However, for all samples, the EMRM outperformed any of the other Rossum-like models. The EMRM reduced the SEP by more than a factor of 2 over the RM when all samples were included. Figure 4.3 shows the improved prediction ability of the EMRM for a typical calibration/prediction set.

4.4.2 Empirical Models

MLR, PCR, PLS, CR and ANN were the empirical modelling techniques used to examine the applicability of this approach to the problem of conductivity prediction. With MLR, the conductivity is expressed as a linear combination of ionic concentrations (Equation 4.13). Three different MLR models are listed in Table 4.4. In addition to considering the MLR model including all 10 ions (MLRa), all possible combinations ($2^{10}-1 = 1023$) of these ions in the MLR model were also evaluated. MLRb is the best MLR model from this set of ions. Also, four additional variables were considered for MLR: total organic carbon (TOC), alkalinity, hardness, and silica. The results describe the best model from analyzing all 16383 ($2^{14}-1$) possible models from this set of 14 variables. However, in MLRc only 10 calibration/prediction sets were used rather than 50 to evaluate SEP due to time constraints.

As shown in Table 4.4, MLR and the EMRM showed similar prediction ability. The MLR model with all ten ions (MLRa) did not provide better prediction

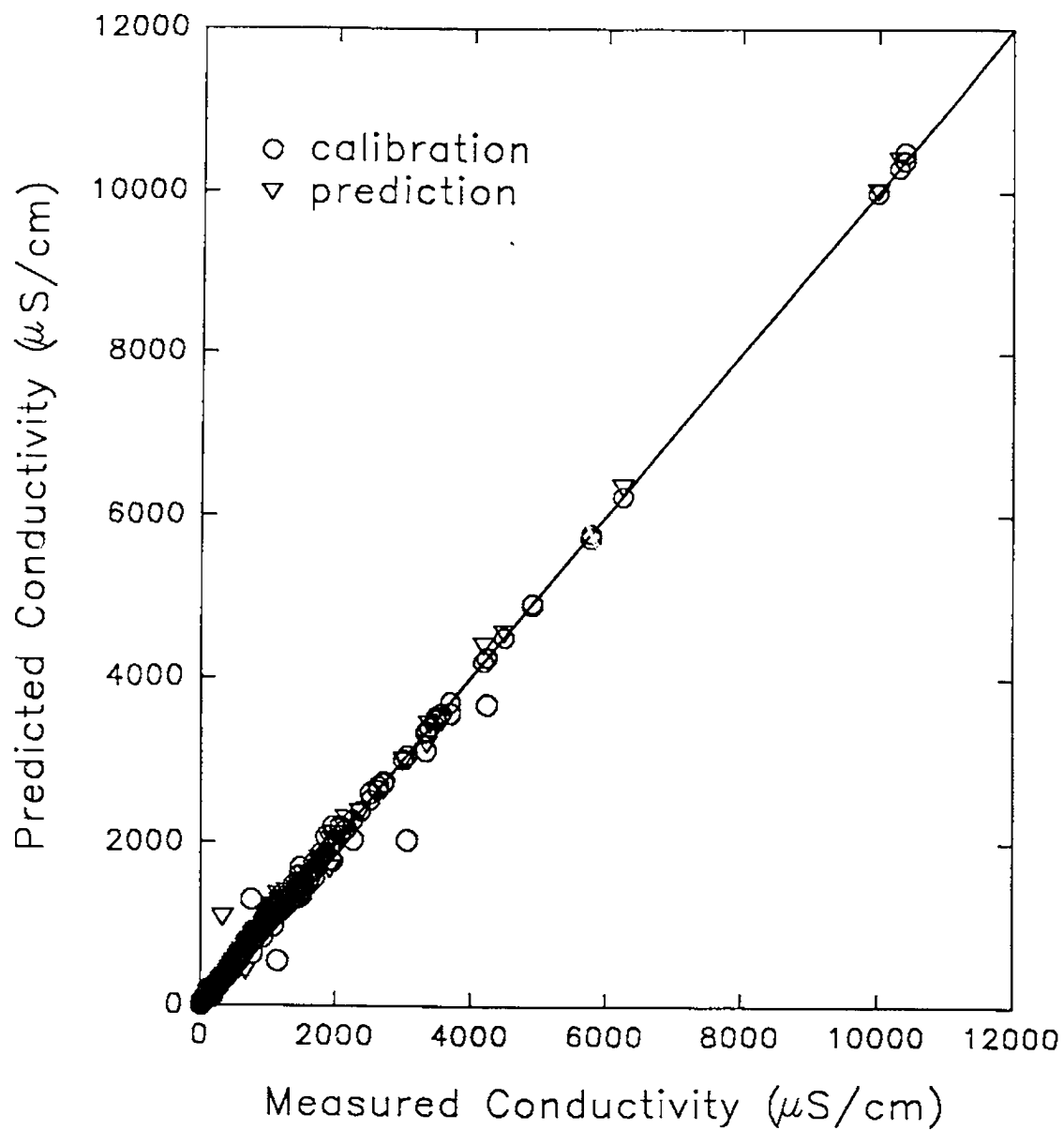


Figure 4.3 Comparison of measured and predicted conductivities using ERM.

than the EMRM in either range. For both ranges MLRb gave similar results to that of the EMRM. The best MLR model (MLRb) below $<1400 \mu\text{S cm}^{-1}$ was obtained when all five cations and no anions were used. In fact, the 30 best models of 1023 possible models all included the five cations with various combinations of the anions. When all samples were considered, the best model included Cl^- and NO_3^- in addition to all of the cations. Similarly, in this range the occurrence of all cations in the top 20 models was observed. Obviously, both anions and cations contribute to the conductivity of the sample, so it is interesting to note the predominance of the cations term in the optimum model. This may reflect the reliability of the respective measurements. In any case, this example illustrates what can be considered both a weakness and a strength of MLR models: they assume no underlying physical model and consider only the predictive abilities of the variables used. Figure 4.4 compares the measured and predicted conductivities for the MLRa and MLRb models for a typical calibration/prediction set in the all samples region. This figure shows no significant difference in prediction from that of EMRM, but are substantially better than the Rossum Model. MLRc shows only a small improvement in prediction over the EMRM and other MLR models (see Table 4.4). While there is no direct physical basis for the inclusion of the additional four terms in the MLRc model, the results suggest that they have some beneficial predictive value.

The PCR, PLS and CR results listed in Table 4.4 were similar to those of MLRa. The fact that PCR, PLS and CR provided results that were about the

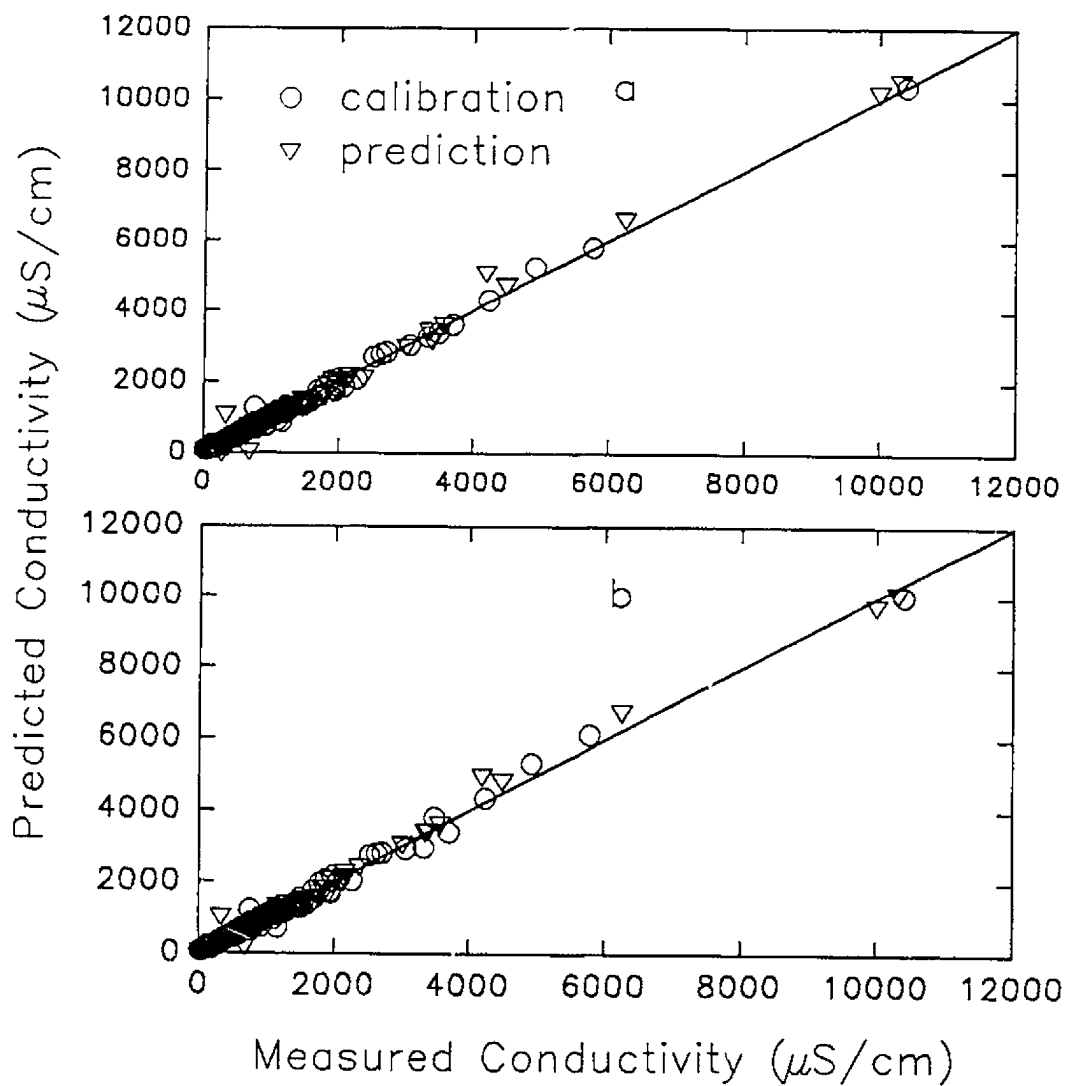


Figure 4.4 Comparison of measured and predicted conductivities using MLR models: a) model based on 10 ions (MLRa), b) optimized MLR model (MLRb).

same as the MLR results suggests that the ionic concentrations were not sufficiently correlated to reduce the number of effective variables. As mentioned previously, CR unifies MLR, PCR and PLS under one scheme through the use of a power factor, q , in the development of the models. The value of q influences the extent to which the dependent variables affect the determination of latent variables. It was shown by Lorber *et al* [72] that the MLR, PLS and PCR solutions are equivalent to $q = 0$, $q = 1$ and $q = \infty$ respectively. Figure 4.5 shows the SEP obtained by CR as a function of q and the number of latent variables included in the models or the all samples region. The best results for CR worked out to be the PCR and PLS solutions for $<1400 \mu\text{S cm}^{-1}$ and all samples ranges respectively. These techniques did not provide any significant improvement in prediction.

The neural networks used in this work consisted of one hidden layer. Due to the nature of the ANN's, the inputs and outputs were scaled for the calibration set. The inputs were adjusted to a mean of 0.5 and range of 2.0, and the output to mean of 0.5 and range of 0.5. Two sets of variables were used in the neural network analysis. One set contained the same 10 ions used previously and the other contained five other variables in addition to the ten ions. The additional variables were pH, alkalinity, TOC, hardness and silica.

Initially the optimum number of nodes in the hidden layer had to be determined. This can be viewed as analogous to finding the optimum number of latent variables for PCR, PLS and CR. To determine the optimum number of

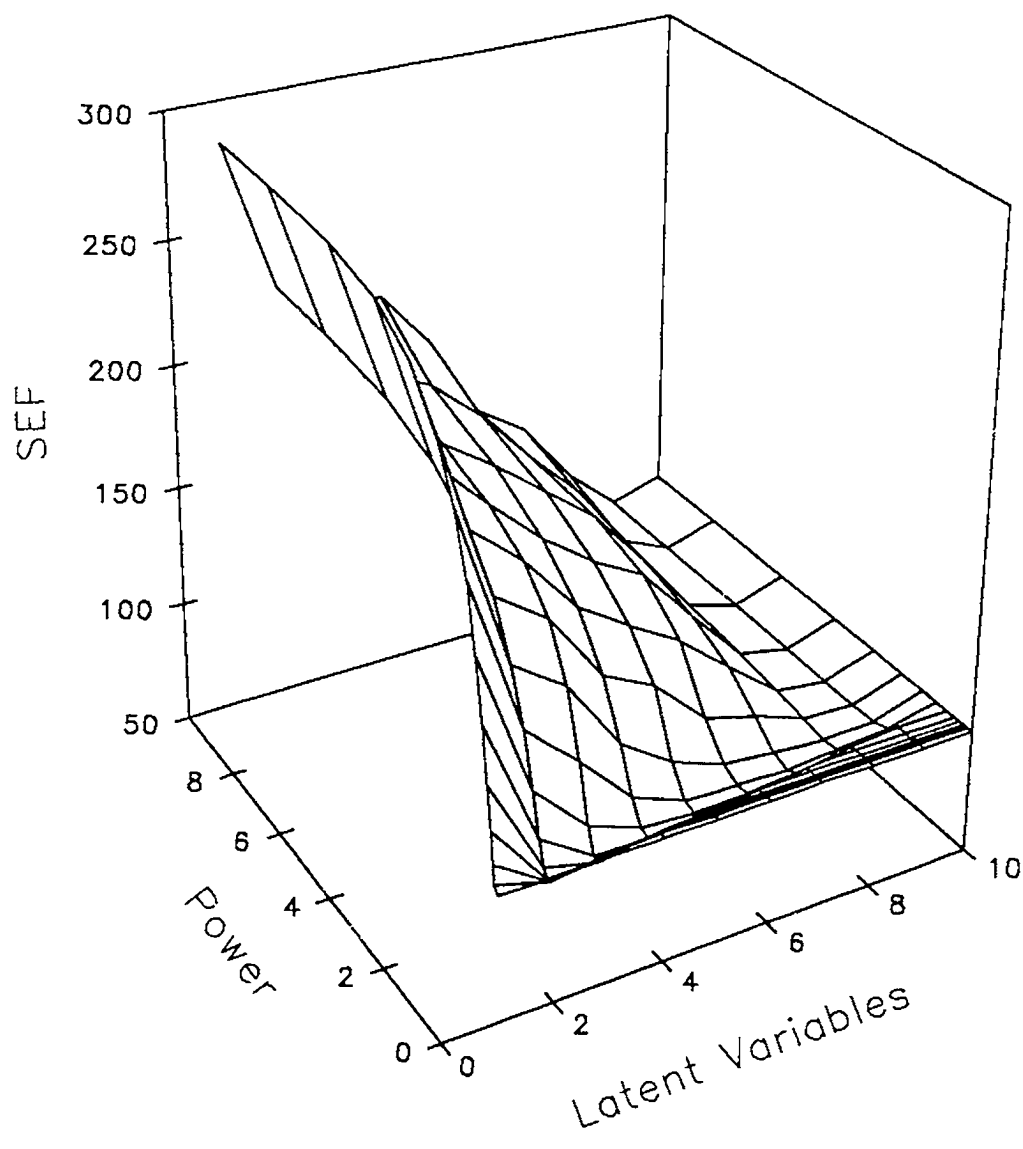


Figure 4.5 Prediction error surface for continuum regression for the all samples region.

hidden nodes the SEC and SEP were examined as a function of iteration number during the training process for various numbers of hidden nodes. Figure 4.6 shows a plot of these parameters for conductivities $<1400 \mu\text{S cm}^{-1}$. One iteration was considered to be when all 200 calibration samples are applied to the inputs of the neural network once. As shown in Figure 4.6a, the SEC initially drops off rapidly and then levels off where the neural network converges to a solution. The convergence level of SEC tends to decrease as the number of hidden nodes increases. However, as shown in Figure 4.6b the SEP passes through a minimum and levels off to the same magnitude for each number of hidden nodes. Two hidden nodes were chosen since this case was the smallest number of nodes giving good prediction abilities (one node gave poor results) and solutions after 5000 iterations were used. As with other estimates of SEP, 50 randomly selected calibration and prediction sets were used. The ANN using 10 ions (NN) gave similar results to the Rossum Model for conductivities below $1400 \mu\text{S cm}^{-1}$, but the prediction results when all samples were included gave poor results. The ANN using 15 variables also gave poor results.

The use of ANNs to predict conductivities in water has been further investigated by Gates [82]. Although the same type of neural network was used, modifications to the architecture and training algorithms were thoroughly investigated. Variations to the basic back-propagation learning algorithm included the use of a gradient search method with an entropic learning algorithm and a random gradient search method. These modifications result in more efficient

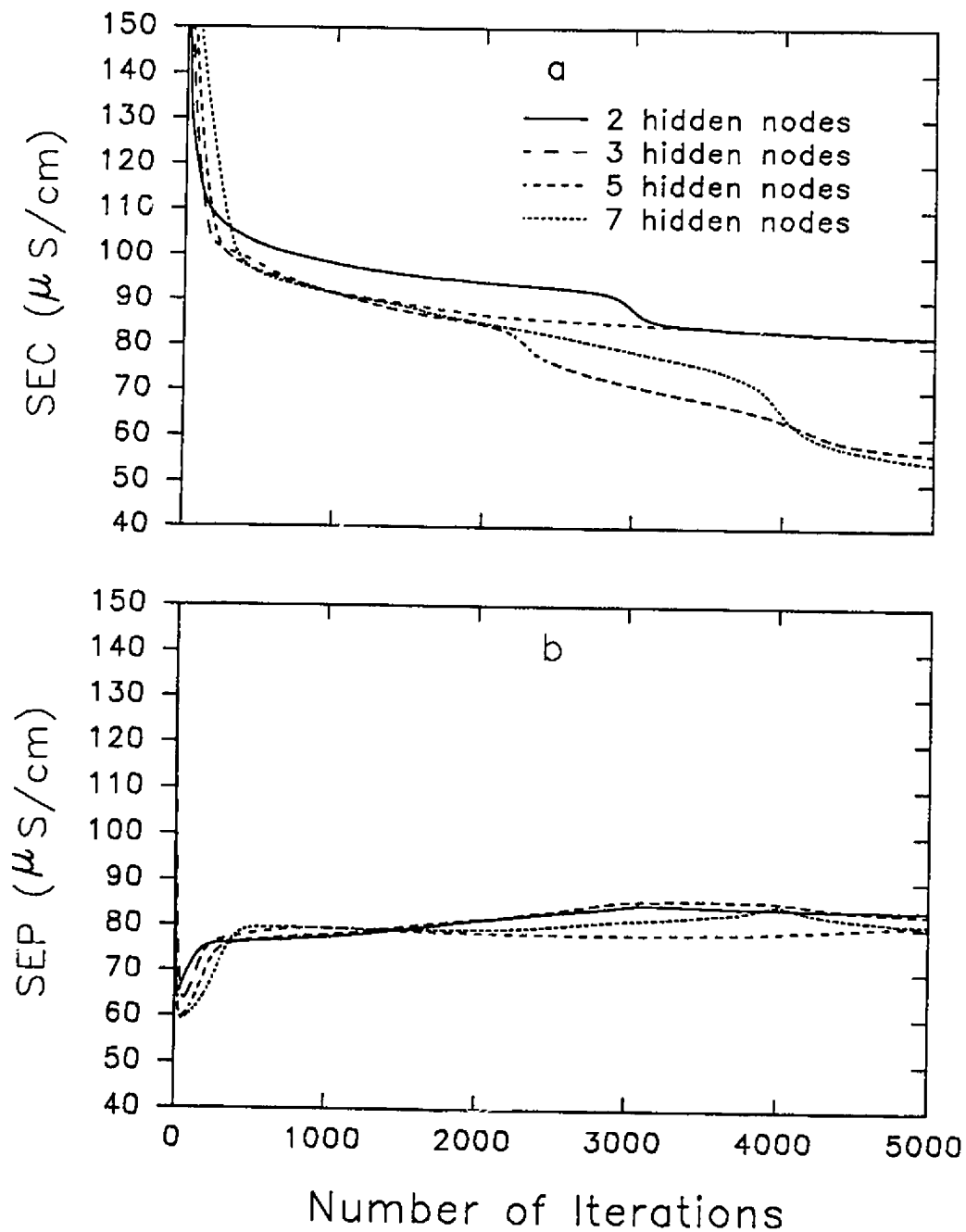


Figure 4.6 Variation of calibration and prediction errors during typical training of neural networks with number of hidden nodes. ($< 1400 \mu\text{S cm}^{-1}$).

training and more reliable location of the global optimum. The effect of the number of hidden nodes was also more thoroughly investigated. Although this study was able to improve on the ANN reported earlier, it was not able to produce better results than the EMRM.

The poor performance of the ANN, especially for the all samples case, was somewhat surprising. ANNs are a nonlinear modelling strategy and should be able to perform at least as well as the linear models in this study. One of the problems inherent in ANNs is the optimization of architecture and training methods, and that is reflected in this study. One possible way to improve the results would be to use an alternative convergence criterion. It will be noted in Figure 4.6b that the SEP curve passes through a minimum. This is a commonly observed phenomenon and results from overtraining of the network on subsequent iterations. While the minimum represents the best predictive ability, it is not considered legitimate to stop training at this point because the prediction set is presumably *unknown* in the training. An alternative strategy is to divide the data set into three groups: a calibration set, a prediction set used to determine the stopping or convergence conditions, and a prediction set used to evaluate the model. In this way, a better convergence point can be obtained without compromising the model. This approach was not used here, however.

4.5 Conclusions

In this work, the Rossum Model was found to work remarkably well for

samples with relative low conductivity ($<1400 \mu\text{S cm}^{-1}$), but alternative models were found necessary for extending the range of conductivity prediction. Modifications to the Rossum Model were incorporated into the EMRM, which exhibited the best performance of the Rossum-based models and greatly improved the prediction ability in the extended range.

It is surprising and somewhat disappointing that the empirical models (MLR, PCR, PLS, CR and ANNs), with a greater number of adjustable parameters, did not substantially improve on the EMRM, which has only one adjustable parameter. This does not mean that these multivariate models are not generally useful, but in this case the integrity of a model based on physical principles proved to be superior. Although some of the empirical methods gave equivalent or improved results, the EMRM is favoured by the principle of parsimony, which holds that, when two models give comparable performance, the simpler of the two (*i.e.* the one with the fewer adjustable parameters) should be used.

Another problem often inherent in empirical methods is the time required to find optimum models (*e.g.* the terms in MLR, architecture for ANNs). For MLR, the selection of optimum terms (variable selection) can be especially problematic because the number of combinations can quickly become astronomical. Solutions to this problem are considered in the next chapter.

Disordered Sets 2: Term Selection

5.1 Introduction

5.1.1 Term Selection

The problem of conductivity prediction presented is an interesting example of a disordered data set because of its practical relevance and because of the failure of empirical methods loosely based on physical models. Because of this, further efforts were made in an attempt to improve the empirical models. These attempts, made through the use of term selection, are described in this chapter. One of the empirical modelling methods used was multiple linear regression (MLR). MLR assumes that a quantity y can be modelled by an equation of the following form:

$$y = b_1 f_1(x) + b_2 f_2(x) + \dots + b_n f_n(x) \quad (5.1)$$

where b_i represents the coefficient for the corresponding basis function, $f_i(x)$. In this formulation, each $f(x)$ is a linear function of the independent variables. In Chapter 4, the basis functions used were simply the individual ionic concentrations. Results suggested, however, that improved performance could be achieved if only selected terms were used. Furthermore, the choice of basis functions was somewhat arbitrary, leading to speculation that an alternate choice might yield improved models, especially if those functions were based on physical principles.

In Chapter 4, it was noted that Shedlovsky [70] had shown that the conductance of a salt could be estimated by,

$$\Lambda = \Lambda^{\circ} - (A + B\Lambda^{\circ})C^{1/2} + mC - BmC^{3/2} \quad (5.2)$$

where Λ , Λ° , A, B, C and m were defined in the preceding chapter. This equation was shown to be valid for a number of binary electrolytes. However, the water samples examined in this work contain significant amounts of several cations and anions, not just a single salt. An analogous expression for a single ion would be,

$$\lambda = \lambda^{\circ} - (A + B\lambda^{\circ})C^{1/2} + mC - BmC^{3/2} \quad (5.3)$$

where A, B and m are modified appropriately. The parameters λ , λ° and C are the ionic conductance, limiting ionic conductance and concentration of a particular ion, respectively. The general form for Equation 5.3 is,

$$\lambda = a + bC^{1/2} + cC + dC^{3/2} \quad (5.4)$$

where λ and C are the same as before and a, b, c, and d are coefficients to be determined. Assuming that this equation is valid and the conductivities are additive, a reliable estimate of conductivity might be obtained from 10 most abundant ions; that is,

$$\begin{aligned} G &= \sum_{i=1}^{10} \lambda_i C_i = \sum_{i=1}^{10} (a_i + b_i C_i^{1/2} + c_i C_i + d_i C_i^{3/2}) C_i \\ &= \sum_{i=1}^{10} (a_i C_i + b_i C_i^{3/2} + c_i C_i^2 + d_i C_i^{5/2}) \end{aligned} \quad (5.5)$$

This equation suggests that an appropriate MLR model should contain not

only terms that are first order in C , but also terms with order $3/2$, 2 and $5/2$. This immediately poses two problems, however. First, this results in a total of 40 terms. Such a large number of terms in an MLR model is not desirable since it is likely to lead to unstable solutions with poor predictive ability. The principle of parsimony suggests that one should attempt to find a suitable model with the smallest number of terms. Thus, we should like to include only those terms which have good predictive ability. This leads to the second problem. If we are to seek the best combination of terms, there are $(2^{40}-1)$ possible combinations to evaluate. In Chapter 4, there were only 1023 models to evaluate for 10 terms, making an exhaustive evaluation of all possible models feasible. In this case, however, a similar study would require ca 4000 years! Clearly, then, an alternative approach must be sought.

Although the problem under consideration here is that of term selection for conductivity prediction, variable selection is a general problem in chemometrics. Other examples are the selection of wavelengths for calibration [83] and feature selection in pattern recognition problems [84]. The goal in all of these cases is to find the optimum set of variables to solve the problem under consideration.

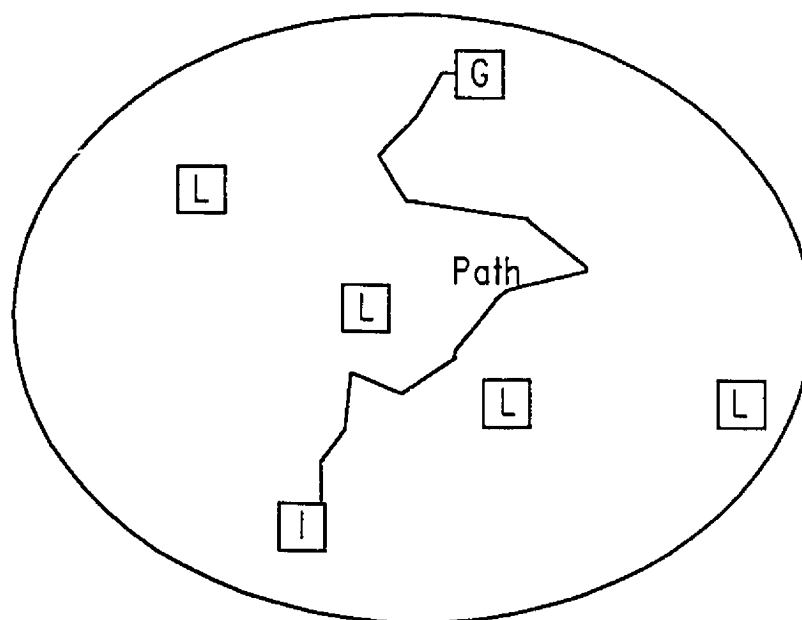
Optimization problems are very common in chemometrics [1-3]. In addition to variable selection problems, experimental optimization problems are also widely encountered. This might mean finding the set of conditions for which a response is maximized (e.g. yield of a reaction, signal-to-noise ratio of an instrument) or minimized (e.g. yield of a side reaction, deviation of an observed quantity from the

ideal). The general goal in an optimization strategy is to proceed along the shortest path from an initial guess to the global optimum. This is represented pictorially in Figure 5.1. A major difficulty in optimization problems is the existence of local optima, which often obscure the search for the true or global optimum. Various optimization methods can be employed and some of these are discussed in the next section.

5.1.2 Optimization Methods

In an optimization problem, perhaps the most straightforward way to find the global optimum is to evaluate all the possible solutions and then pick the best result. Although this exhaustive, or enumerative, approach is guaranteed to find the global optimum, it is only practical when the search space is small. As previously noted, many problems are so complex that evaluation of all possible solutions is not feasible. Thus a search method is needed that is able to reliably find the global optimum while only evaluating a small number of cases. For illustration purposes consider the case of a multiple parameter optimization, such as might be encountered in an experimental optimization or nonlinear curve fitting. One of the simplest approaches to parameter optimization is find the optimum value for each parameter independently (univariate optimization) [1]; that is, varying one parameter at a time while holding the others constant. Once each parameter is optimized, the process is repeated for all of the remaining parameters. Although this method may work in some cases, it is not able to deal

Set of all possible
solutions



I Initial Guess

L Local Optima

G Global Optima

Figure 5.1 Graphical representation of optimization.

with interdependence among the parameters. Therefore an optimization method is needed that allows all of the parameters to change simultaneously. One class of methods with this feature is gradient methods [70].

Gradient methods, such as steepest descent and Levenberg-Marquardt algorithms, have been used quite successfully in optimization problems. Such methods, as the name implies, search along the steepest gradient in search space. To do so, the derivative of the response surface for each of the search variables must be known. The success of the gradient methods depends on the validity of these derivatives and on the ability to calculate them. In many instances, such as experimental optimization, these derivatives are not directly available or would be cumbersome to calculate.

An alternative method known as simplex optimization [86] can be used if little information is known about the shape of the surface being searched. Simplex optimization does not require the estimation of derivatives, but rather uses simple geometric rules to navigate the search space. Simplex optimization is often applied to problems that are not suitable to solution by gradient methods. However, like gradient methods simplex optimization requires the search surface to be well-behaved, is usually only effective for a limited number of parameters, and is prone to finding local optima. Furthermore, these methods are mainly suited to numerical optimization problems and are not well adapted to the term selection problem currently under consideration. Two methods which are well-suited to large scale optimization problems of this type are simulated annealing

[87] and genetic algorithms [88-93].

Simulated annealing is an optimization procedure that mimics the process of annealing a solid. The physical process of annealing involves heating a solid to just below its melting temperature, at which time all the particles of the solid are randomly arranged. The liquid phase is then slowly cooled by gradually reducing the temperature. During this cooling the particles reach equilibrium at each temperature, tending to arrange themselves in low-energy ground states. The cooling is continued and the solid reaches its lowest energy state. The optimization method of simulated annealing begins with a high energy state (the initial guess) and explores large areas of the solution space by assuming a Boltzmann distributions of solutions. When a lower energy solution is found, smaller steps are taken until the global optimum is reached. Simulated annealing is particularly well-suited to large scale optimization problems which are prone to local minima since there is an element of randomness in the search algorithm. In this work, however, another technique with similar characteristics, genetic algorithms, was employed. This approach is described in the next section.

5.1.3 Genetic Algorithms

A genetic algorithm (GA) can be described as a method for solving large scale optimization problems that is loosely based on Darwin's principles of evolution: *the struggle for life* and *the survival of the fittest*. Principles of evolution hold that the development of any species did not occur by accident, but by a long

process of slow modification that encouraged passing on desirable traits from one generation to the next. For example, if two individuals of the same species differed by one trait, and if this trait was essential to the survival of the individual, the individual without the trait would not likely live to reproduce, whereas the other would reproduce and the trait would be passed on. For each species, numerous competitions have occurred to produce the diversity in nature. In a similar fashion, the GA starts with a *population* of trial solutions and through competition and modification, the desirable traits in the solutions are passed on from one generation to the next. The GA moves progressively toward the optimum in a systematic manner, but occasionally takes random steps (or *mutations*) that help to search many regions of the search space.

For any search algorithm, there needs to be a balance between two principles: exploration and exploitation. Exploration is the ability to effectively search the whole set of possible solutions and exploitation is the ability to utilize the characteristics of the search space or solution space to efficiently proceed toward the optimum. Since approaches such as gradient methods and simplex optimization are susceptible to finding local minima, one could view them as having a greater emphasis on exploitation than exploration. On the other hand, simulated annealing and genetic algorithms seem to strike a better balance between exploration and exploitation for finding the global optimum on complex surfaces with many parameters. In this work, only GAs were considered for the term selection problem.

The types of problems that can be treated with GAs can be divided into three areas: 1) numerical parameter estimation, 2) sequencing, and 3) subset selection problems. No matter what type of problem a GA is used for, a numerical sequence of *genes* is used to find the optimum. This sequence is analogous to a chromosome in a living organism and is referred to as a *string* in GA terminology. Figure 5.2a shows some simple string representations for each of the three problem types. For numerical parameter estimation problems (such as curve fitting), each of the parameters is coded in a binary sequence. In the example shown, three parameters are coded in the string, each as a five-bit binary number. Numerical parameter estimation is not used in this work and will not be considered further. Sequencing problems are concerned with finding the optimum order of, say, samples [94] or events [95-97]. In this case, each *gene* is represented by a number which corresponds to the position in the original sequence, and it is the order of these genes in the string that is important. A sequencing problem is considered in Chapter 6 and this category will not be discussed further in this chapter. The last type of problem, subset selection, can be represented in two ways, depending on whether a variable or fixed size subset is desired. For a variable size subset, there are n genes, where n is the total number of variables or terms. In this case, each gene is either 0 or 1, indicating whether or not that variable is included in the subset. For a fixed size subset containing m variables, there are m genes, each containing a number corresponding to the variable to be included in the subset. In this case, there will be $m!$ redundant solutions forms for

a) Types of problems

1) Numerical parameter estimation

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

2) Sequencing

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 6 | 3 | 5 | 1 | 8 | 7 | 4 | 2 | 9 |
|---|---|---|---|---|---|---|---|---|

3) Subset selection

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

or

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 6 | 3 | 5 | 1 | 8 | 7 | 4 | 2 | 9 |
|---|---|---|---|---|---|---|---|---|

b) Population

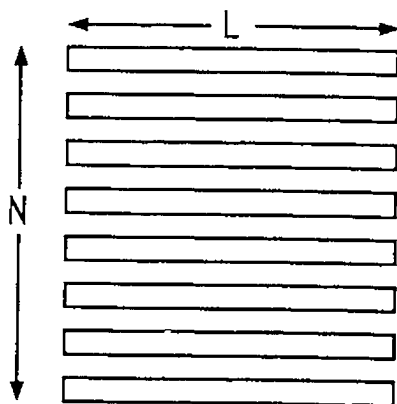


Figure 5.2 a) String representation of the types of problems a GA can handle. b) A graphical representation of a population.

each solution. The problem of subset selection the one of interest in this work.

Once a string is defined for a problem, its length, L , is usually held constant for the running of the GA. The GA uses a collection of different strings called a *population*, shown pictorially in Figure 5.2b. The size of the population or the number of strings, N , can be held constant or vary as the GA runs. This population not only allows the GA to explore different areas of the search space, but also permits information to be exchanged among members of the population. This exchange helps to locate the global optimum by exploiting complementary genetic characteristics in different strings.

In order to show how a GA works, the term selection problem will be used. Two important questions need to be considered before a GA is run: (1) how to represent the problem being solved, and (2) how to discriminate among the various test solutions?. In this case, the 40 terms were represented as a 40 bit binary code, (*i.e.* variable size subset problem) where each bit represents a variable or term in the model. As shown in Equation 5.5, all the terms here are functions of ionic concentrations. The bit variable assignment is given in Table 5.1.

Table 5.1 The bit-term assignments for the term selection problem given by their bit location and chemical symbol.

| Bit | Term | Bit | Term | Bit | Term | Bit | Term |
|-----|------------------------|-----|----------------------------|-----|------------------------|-----|----------------------------|
| 1 | $(\text{Na}^+)^1$ | 11 | $(\text{Na}^+)^{3/2}$ | 21 | $(\text{Na}^+)^2$ | 31 | $(\text{Na}^+)^{5/2}$ |
| 2 | $(\text{K}^+)^1$ | 12 | $(\text{K}^+)^{3/2}$ | 22 | $(\text{K}^+)^2$ | 32 | $(\text{K}^+)^{5/2}$ |
| 3 | $(\text{Ca}^{2+})^1$ | 13 | $(\text{Ca}^{2+})^{3/2}$ | 23 | $(\text{Ca}^{2+})^2$ | 33 | $(\text{Ca}^{2+})^{5/2}$ |
| 4 | $(\text{Mg}^{2+})^1$ | 14 | $(\text{Mg}^{2+})^{3/2}$ | 24 | $(\text{Mg}^{2+})^2$ | 34 | $(\text{Mg}^{2+})^{5/2}$ |
| 5 | $(\text{SO}_4^{2-})^1$ | 15 | $(\text{SO}_4^{2-})^{3/2}$ | 25 | $(\text{SO}_4^{2-})^2$ | 35 | $(\text{SO}_4^{2-})^{5/2}$ |
| 6 | $(\text{Cl}^-)^1$ | 16 | $(\text{Cl}^-)^{3/2}$ | 26 | $(\text{Cl}^-)^2$ | 36 | $(\text{Cl}^-)^{5/2}$ |
| 7 | $(\text{NO}_3^-)^1$ | 17 | $(\text{NO}_3^-)^{3/2}$ | 27 | $(\text{NO}_3^-)^2$ | 37 | $(\text{NO}_3^-)^{5/2}$ |
| 8 | $(\text{NH}_4^+)^1$ | 18 | $(\text{NH}_4^+)^{3/2}$ | 28 | $(\text{NH}_4^+)^2$ | 38 | $(\text{NH}_4^+)^{5/2}$ |
| 9 | $(\text{CO}_3^{2-})^1$ | 19 | $(\text{CO}_3^{2-})^{3/2}$ | 29 | $(\text{CO}_3^{2-})^2$ | 39 | $(\text{CO}_3^{2-})^{5/2}$ |
| 10 | $(\text{HCO}_3^-)^1$ | 20 | $(\text{HCO}_3^-)^{3/2}$ | 30 | $(\text{HCO}_3^-)^2$ | 40 | $(\text{HCO}_3^-)^{5/2}$ |

For illustration purposes, consider the first 6 bits. An example of a trial solution might be,

$$1 \ 0 \ 1 \ 0 \ 0 \ 1 \quad (5.6)$$

where the 1's indicate inclusion of this term in this model and 0's indicate

exclusion. Therefore the model presently under consideration is,

$$G = a_1(Na^+)^1 + a_3(Ca^{2+})^1 + a_6(Cl^-)^1 \quad (5.7)$$

The best string will normally be the one for which the corresponding model gives the lowest prediction error. The prediction error is therefore the obvious choice for discriminating among strings. The standard error of prediction (SEP) was used as the objective function for this problem; that is,

$$SEP = \sqrt{\frac{\sum_{i=1}^N (G_i - G_i^p)^2}{N}} \quad (5.8)$$

where G_i^p is the predicted conductivity for prediction sample i , G_i is the measured conductivity, and N is the number of prediction samples. In GA terminology, a string is usually evaluated in terms of its *fitness*, which is assumed to be maximized for an optimum solution. Therefore, since the SEP is minimized for the optimum solution, we can define the fitness function, f , as,

$$f = \frac{1}{SEP} \quad (5.9)$$

In some implementations of GA's, the absolute magnitude of f plays a role, but in this work only relative magnitudes are considered, so the actual definition is not of critical importance.

GAs are usually referred to in the plural because there are numerous variations in the specific algorithms used for a particular problem [98]. Nevertheless, the basic steps remain the same and these are represented as a

flowchart in Figure 5.3. Each of these steps is discussed in more detail below, with specific attention given to the implementation used in this work.

1. Initiation

The initial population or the set of test strings is usually randomly generated. Specifically, the strings are generated using a random number generator with a uniform distribution and converting the output to its corresponding binary code. The number of strings in the initial population, N , is an adjustable parameter for the GA. In this work, the value of N was typically 100.

2. Evaluation

For each string in the population, the objective function is evaluated for its fitness. First the appropriate terms are selected, then the model is built using multiple linear regression, and finally the SEP (and the fitness) is evaluated. In this work a calibration set of 200 samples was used to build the model (*i.e.* the least-squares solution is used to generate the coefficients). The remaining samples are used in the prediction set to evaluate the fitness.

3. Exploitation

Once the strings have been evaluated, the fittest strings are selected

for reproduction. A number of strategies are commonly used in this step. In some cases, a *roulette wheel* approach is used in which the probability that a string will be reproduced is proportional to its fitness. In this work, a simpler approach was employed where copies of the m fittest strings are made to carry on to the next step (exploration). The best $(N-m)$ strings in the original population are also retained, but are not subjected to the next step. The worst m strings are eliminated and are eventually replaced by the m copies after the exploration step. In this way, the size of the population was maintained constant. The proportion m/N will be referred to as the reproduction rate and in this work a rate of 20% was typically used.

4. Exploration

The m selected strings copied in the previous step are modified in the exploration step by two procedures: recombination and mutation. These two procedures are meant to be analogous to their biological counterparts. Recombination, as shown in Figure 5.3, involves swapping *genes* (bits in this case) between pairs of strings and is meant to mimic sexual reproduction. The m strings copied in the previous step are paired up randomly before the bits are swapped in this step. Several strategies can be employed for the actual exchange of bits. In one approach, a crossover point is selected and all of the bits before (or after) that point are

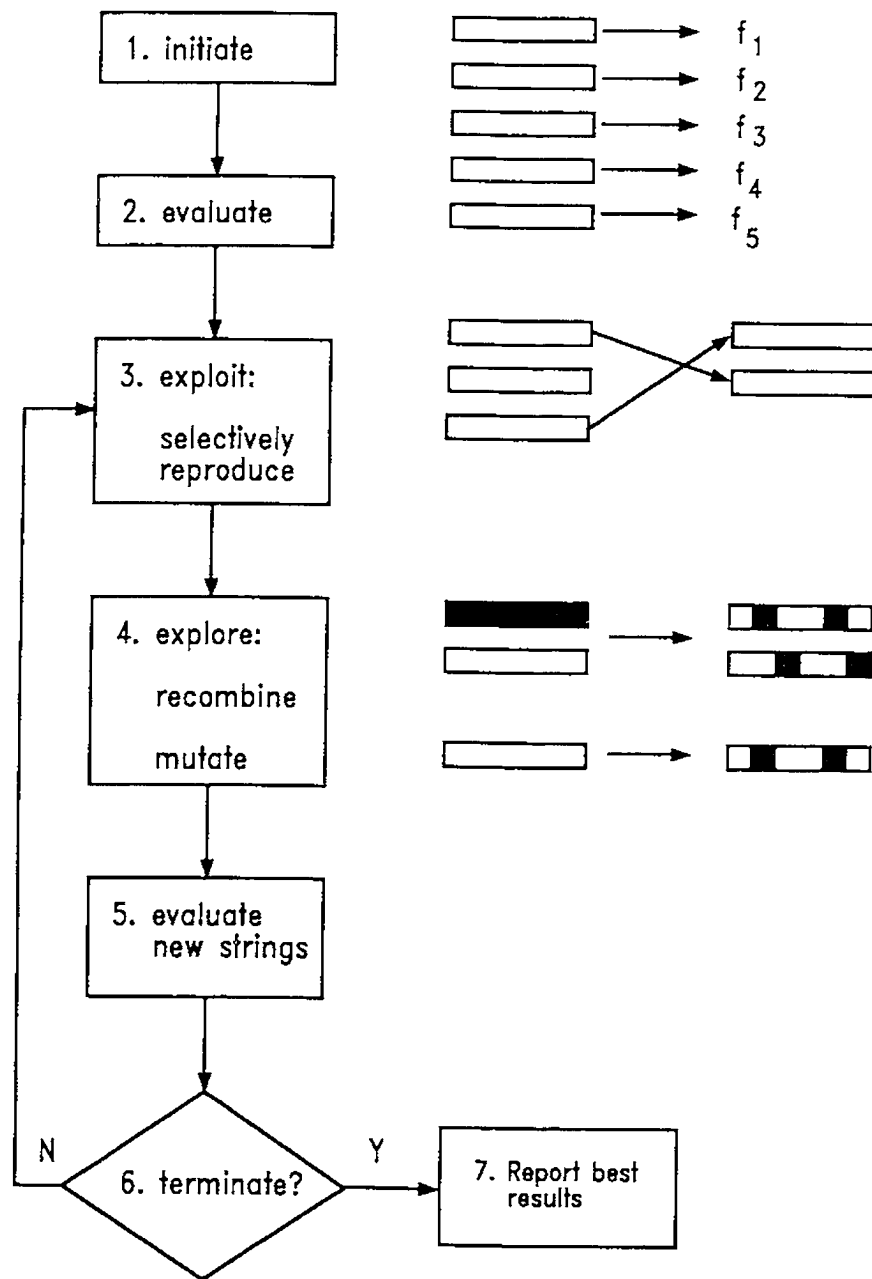


Figure 5.3 A flow chart of a typical GA.

exchanged. In this work bits were exchanged according to a random selection procedure. For each pair of bits, the probability of exchange is given by the recombination rate, typically 30% in this work. This means, that on average, each string would have 30% of its bits swapped with its partner. Note that this does not mean that 30% will necessarily be changed, however, since the bits swapped may be identical. The purpose of the recombination step is to combine the complementary favourable characteristics of the *partners* in the *children*. Of course, there is an equal chance that undesirable characteristics will be passed on.

The problem with recombination alone as an operator is that it limits exploration of the search space. This is particularly true if the population becomes fairly homogenous so that most of the bits exchanged are identical. For this reason, the mutation operator is employed following recombination. This operator is intended to simulate the process in nature by which *genes* are randomly altered. In the mutation stage, each bit has a certain probability of being *flipped* to its complementary value. This probability, termed the mutation rate, was typically set to 3% in this work. The mutation operator incorporates a certain randomness to the exploration and is instrumental for ensuring that the global optimum is found. As an illustration of these operators recombination of two parent strings 110001 and 101101 at the fourth bit would result in two child strings 110101 and 101001, and mutation at the third bit of 110101 would result in 111101.

The new strings generated in the exploration step replace the m worst strings from the previous generation.

5 & 6. Evaluation and Termination

In the evaluation step, the fitness values of the new strings generated at the exploration step are evaluated according to the objective function. A decision is made at this stage to continue the GA or not. As with any optimization method, the decision to terminate can be based on a number of criteria, none of which are foolproof. Most often with GAs, convergence to a consistent solution is evaluated graphically. This is done by plotting the objective function (or the fitness function) for each generation as a function of the generation number. Typically, this plot approaches some constant value in an exponential fashion and is considered to have converged when the curve is essentially flat. If this has not occurred, the process is continued with the new population at step 2.

The steps outlined above give the essential aspects of at least one of the GAs used in this work for term selection. Depending on the type of the problem and its representation, the nature of the recombination and mutation operators may be different, but the basic steps remain the same. As well, the choice of parameters (population size, reproduction rate, recombination rate, mutation rate) is subject to the type of problem and is usually arrived at by trial and error and/or

common sense. The choice of parameters used here is not necessarily optimal, but seemed to work reasonably well.

5.2 Experimental

In the present study only ten species were considered: sodium, potassium, calcium, magnesium, sulfate, chloride, nitrate, ammonium, carbonate and bicarbonate. The details on the measurement of conductivity and ionic concentrations were reported in Chapter 4.

Most calculations were performed on 486-based computers (486-DX, 486-DX2 and 486-DX4) under DOS 6.0 and Windows 3.1 (Microsoft, Redmond WA) with programs written in MATLAB 4.0 for Windows (The Math Works Inc., Natick, MA). The 20 variable exhaustive search calculation was run on a Sun Sparc 10 under the Solaris operating system.

5.3 Results and Discussion

5.3.1 Variable Number of Terms

Using the methods described in section 5.1.3, preliminary studies were conducted to select the best subset of terms from 10 term and 20 term models (*i.e.* using only first order terms and then first and 3/2 terms.) This was done so that the results of the GA could be compared to an exhaustive search of solutions. The GA converged rather quickly for the 10 variable case as expected, since there were only 1023 solutions and there were 100 strings in the population for each

generation. The 20 term case was more difficult, since there are over a million possible solutions. Figure 5.4 shows convergence for a typical 20 term case. The prediction error (SEP) for the best string in each generation is plotted against the generation number. As expected, the SEP declines in an exponential fashion with generation. In both the 10 term and 20 terms cases, the GA generally converged on the correct subset of terms. Note, however, that these trials used the same calibration and prediction sets throughout the GA run. This means that the search space was essentially noise free, giving a relatively smooth trace in Figure 5.4 and making it easier for the optimum to be found. Unfortunately, a solution found in this manner is not necessarily robust. In other words, although the solution may be optimal for the particular calibration and prediction set used, it may not be optimal for another combination of calibration and prediction samples.

In order to produce a solution which is more robust, a better method for detecting a globally optimum solution (*i.e.* one that gives the best average performance over all combinations of calibration and prediction sets) is needed. One way to do this would be to determine an average prediction error based on multiple calibration and prediction sets for each trial solution. This would greatly increase computation time however, and instead a method for incorporating this into the GA itself was sought. Two alternatives were investigated with this objective in mind: (1) using a new calibration and prediction set for each generation, and (2) using a new calibration and prediction set for each string. Although the second method introduces more variations, results showed that the

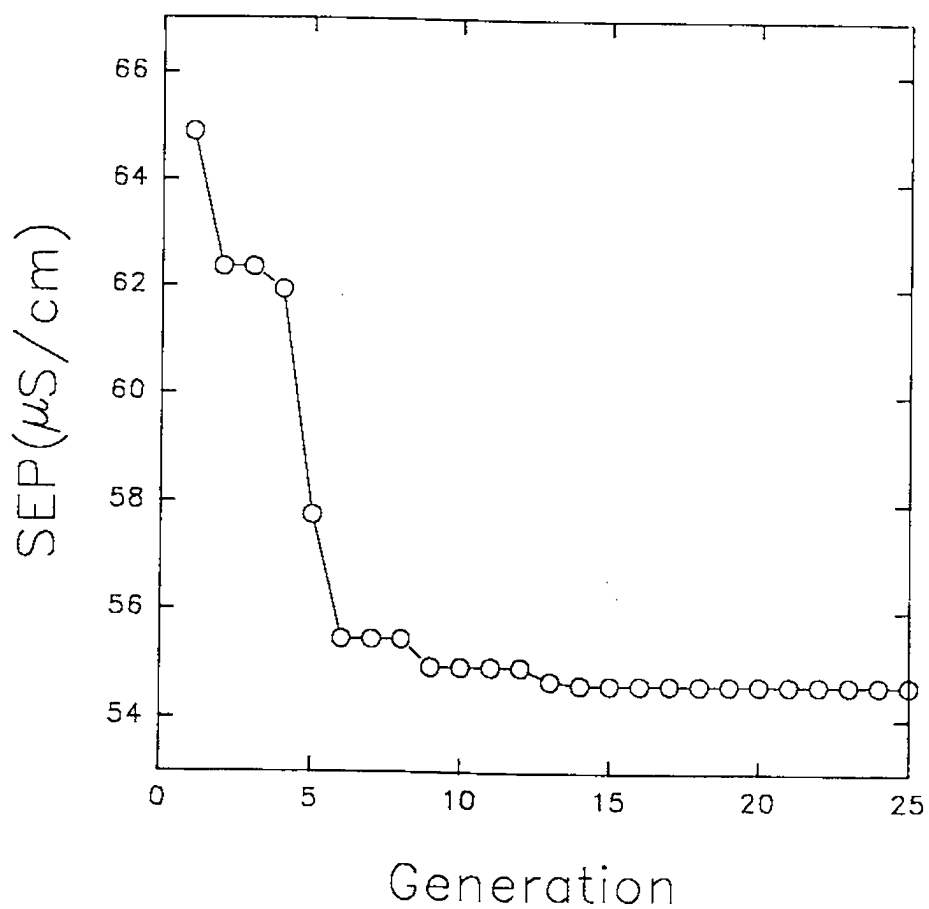


Figure 5.4 Convergence of a typical GA run for the 20 term case using the all samples region.

first method gave equivalent results and was computationally more efficient. The idea behind this modification was that a solution which performed well for one generation (*i.e.* a particular calibration and prediction set) would not necessarily perform well in the next generation, but those solutions that consistently performed well (although not necessarily the best) would be retained over many generations. This modification necessitated the recalculation of the SEP for all strings with each new generation, but also presented an opportunity for further improvement. For strings that were retained for more than one generation, an average SEP could be calculated. This allows a more reliable estimate of the SEP for retained strings that would be less prone to anomalous results from a particular calibration and prediction set. For example, if two strings were retained by the GA for 5 and 7 generations, their performance would be evaluated over 5 and 7 prediction sets. This would smooth the search landscape and therefore should make the optimum model easier to find.

The modifications described above (new calibration and prediction set for each generation and averaged SEPs) were implemented in the search for the best subset for the 40 term model. The GA results for the two samples ranges used are shown in Figure 5.5 and exhibit several features worth noting. First, although the prediction errors attained with the GA looks lower for the two conductivity regions when compared with the values obtained with the MLRb model described in Chapter 4 their average SEP's (when calculated with 50 calibration/prediction sets) were actually higher (*i.e.* $69 \mu\text{S cm}^{-1}$ for the $< 1400 \mu\text{S cm}^{-1}$ and $125 \mu\text{S cm}^{-1}$

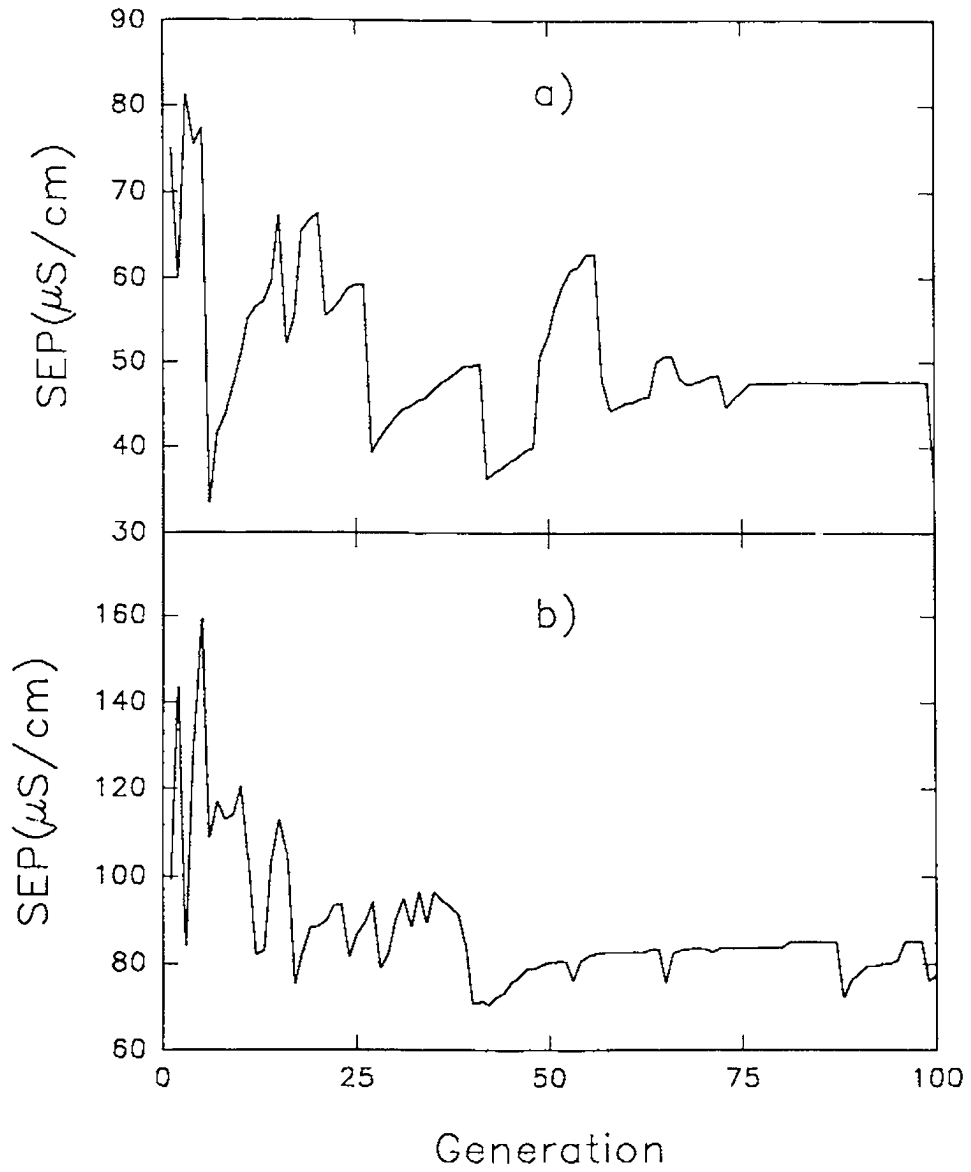


Figure 5.5 SEP versus generation using the 40 term model for a) samples below $1400 \mu\text{S}/\text{cm}$ and b) all samples.

for the all samples region). This is unfortunate since it was hoped that including more terms would improve the predictive ability. A second feature worth noting is that the results shown in Figure 5.5 do not exhibit a large change from the initial population results. This is somewhat disconcerting, since a larger improvement was anticipated. Finally, the traces in Figure 5.5 are noisier than those in Figure 5.4, but this was expected because of the new calibration and prediction sets used for each generation. Unfortunately, this characteristic is indicative of another problem with the GA results: the solutions obtained on subsequent GA runs were not entirely consistent. In other words, different terms were represented in the final solutions obtained from different runs.

To summarize, the results for the 40 term case with a variable size subset did not show significant improvement over the rigorously optimized 10 term model, exhibited unexpected convergence characteristics and solutions that were not particularly consistent. The reasons for these last two problems are addressed in the next section.

5.3.2 Analysis of GA Results for the 40-term Model

In this section, a closer look at the solution landscape is undertaken in an attempt to diagnose the problems noted in the preceding section. One of these problems was the poor convergence characteristics of the GA. This can be understood by considering the characteristics of the solution space. To do this, consider that all of the possible solutions are exhaustively evaluated, ranked in

terms of the objective function from best to worst and then plotted. The shape of this plot will depend on the problem under consideration and three possible representative cases are shown in Figure 5.6. Curve a represents the case where most of the solutions give poor results, curve b shows the situation where there is a gradual change in the quality of solutions, and curve c represents the extreme where many solutions produce reasonably good results and a minority produce very poor results. In this last case, a random sampling of solutions for the initial population would be expected to yield a substantial number of solutions with reasonably good results. It is this type of surface that is anticipated here, so the net improvement seen is not large. Although all of the solutions for the 40-term case ($ca\ 10^{12}$) cannot be evaluated, a view of the surface can be inferred from a random sampling of 1000 solutions. This result is shown in Figure 5.7 and confirms the anticipated shape. For a surface such as this, the solutions are not highly discriminating in the quality of results produced, so the application of the GA does not lead to substantial improvement over a trial-error approach.

Even though the surface observed in this work may contain large regions that are fairly flat, the GA is based on relative comparisons, so one would expect that it would be able to arrive at a stable solution. To understand why this is not the case here, further analysis of the search landscape is useful. One way a landscape can be characterized is by using an autocorrelation function. The autocorrelation function gives a direct indication of the surface complexity.

Autocorrelation functions are very useful for revealing correlations among

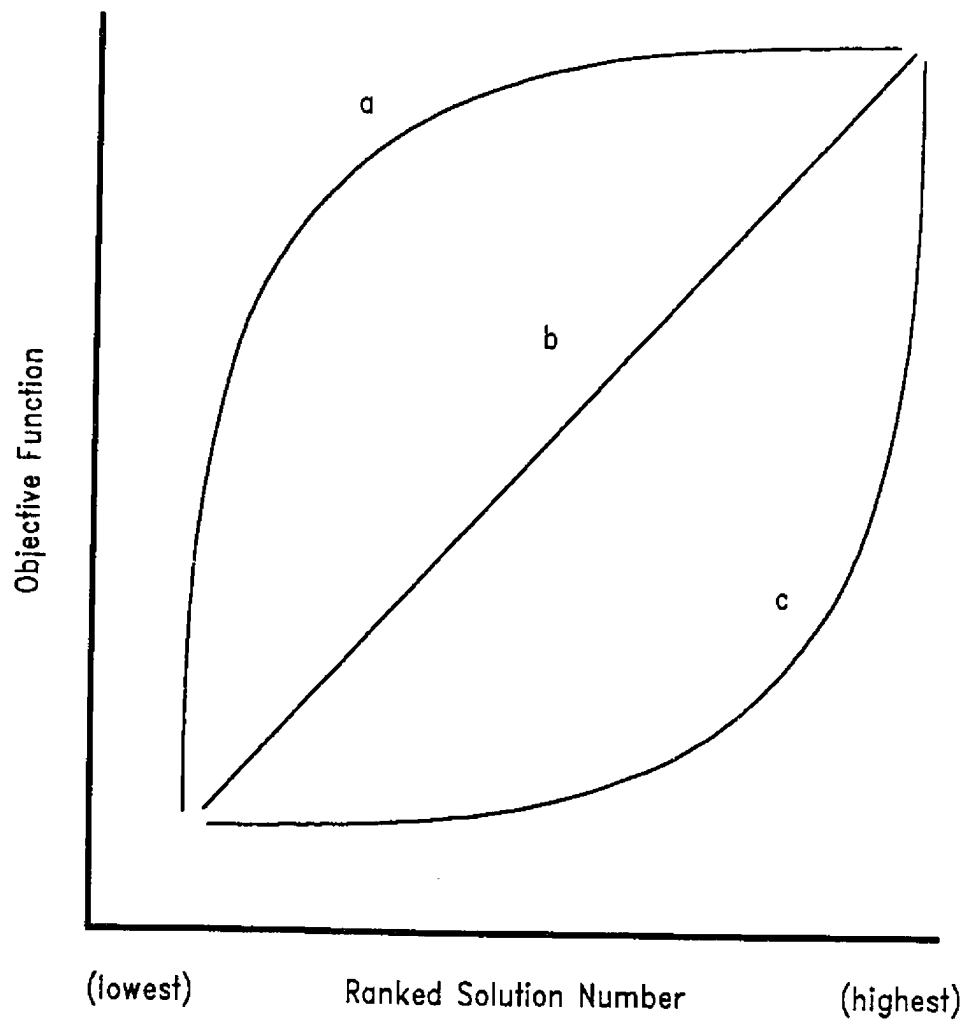


Figure 5.6 Characteristics of representative solution spaces.

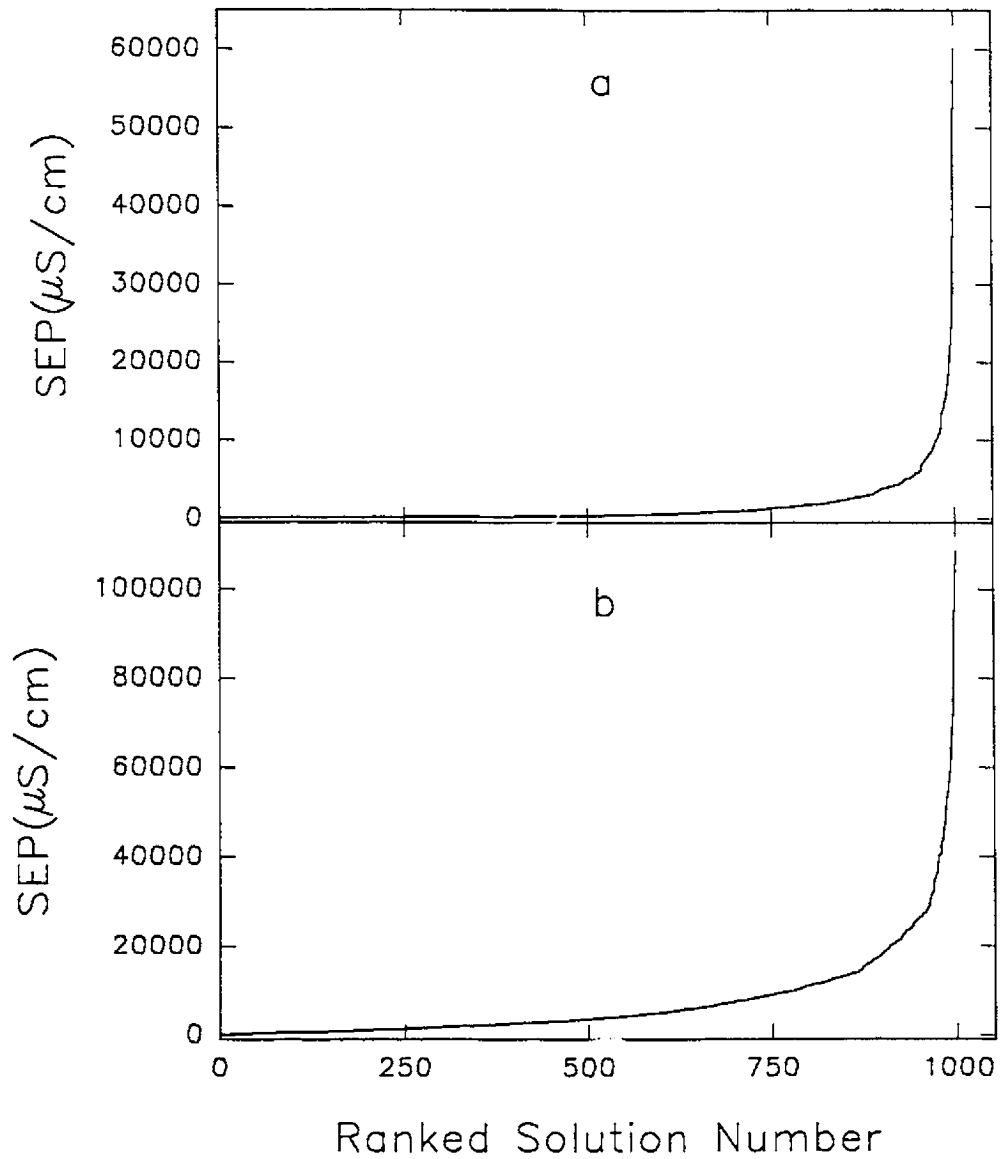


Figure 5.7 Characteristics of 40 term model for samples a) below $1400 \mu\text{S cm}^{-1}$ and b) all samples.

measurements in a sequence, *i.e.* if there is a relationship between a measurement and ones close to it. Autocorrelation functions can be most easily illustrated graphically. Consider a variable x that varies over time, such as the time series shown in Figure 5.8a. For a function with mean (x_m) and standard deviation (s_x) the autocorrelation is calculated as,

$$r(\tau) = \frac{\sum_{t=1}^{n-\tau} [x(t) - x_m][x(t+\tau) - x_m]}{(n-1-\tau)s_x^2} \quad (5.10)$$

where $x(t)$ is the value of the series at time t , and $x(t+\tau)$ is the value of x at a later time, n is the number of points and τ is the delay time. The above equation gives the correlation coefficient at each τ . Usually r is calculated from $\tau = 0$ to large values of τ . In Figure 5.8 the autocorrelation function for the series in part a is shown in part c. The appearance of the autocorrelation function depends on the series evaluated. A more correlated series is shown in part b along with its corresponding autocorrelation function in part d. Note the behaviour of the autocorrelation function for each series. For the random series, *i.e.* part a, the autocorrelation function equals 1 at $\tau = 0$ but quickly drops off to near zero. For the more correlated series, the autocorrelation also equals 1 at $\tau = 0$, but decreases more gradually and only approaches zero at $\tau = 15$. This gradual decay is indicative of a correlated series of measurements and roughly gives the period of the correlation function.

In the context of GAs, correlation means that solutions that are close to one

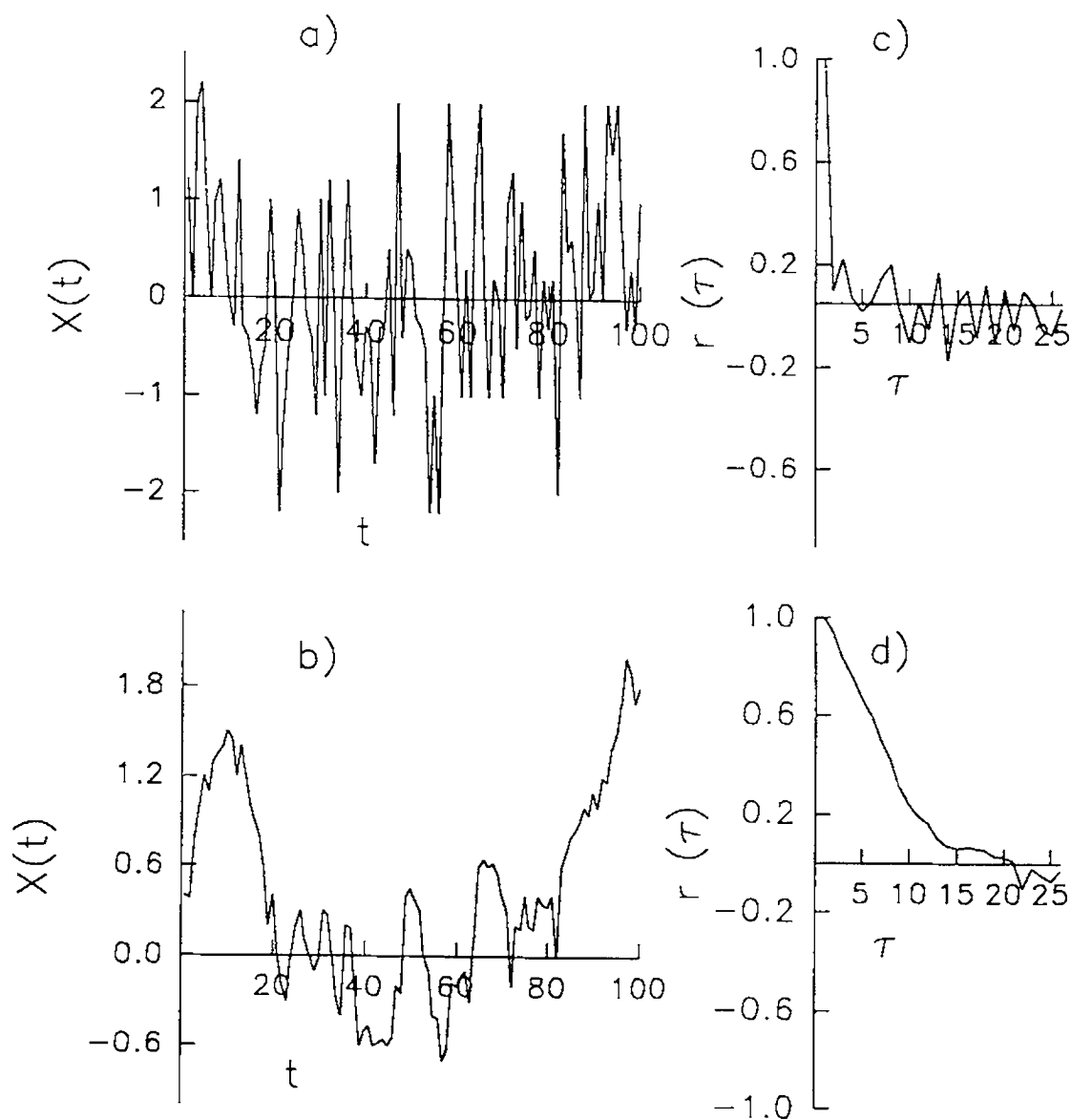


Figure 5.8 Two examples illustrating autocorrelation functions. **a)** A time series with low correlation among observations and **b)** a time series with high correlations among observations. Parts **c)** and **d)** are the corresponding autocorrelation functions for part **a)** and **b)** respectively.

another in the search space should produce similar results. It is this assumption that allows the GA to exploit the surface topology in the recombination step. Note that the way a problem is encoded in the development of the GA is critical in influencing this correlation. In order to determine if such a correlation exists, the following approach can be used [98]. First a solution is chosen at random, and the objective function (or fitness) is evaluated. Then, a step on the search landscape is made by randomly changing one bit in the solution, and again the objective function is evaluated. This process is repeated multiple times and constitutes a *random walk* over the solution surface. The series of objective functions generated is the sequence analyzed by the autocorrelation function.

For the current study this method was used to generate autocorrelation functions for the 40 term model in both the low and high conductivity regions. The results, shown in Figure 5.9, were calculated using a series of 25,000 values (SEPs) in each case. As shown, both autocorrelation functions drop off very quickly after just one step, suggesting a low correlation among solutions in the same region of the search space. Under these conditions, the GA becomes less effective, since the exploitation features of the algorithm are not as useful.

A low correlation among solutions in the search space can be caused by two factors. First, the problem may be defined or improperly encoded so that solutions in the same region of the space have unrelated fitness values. By way of a physical analogy, consider a solid surface which has a matrix of holes drilled to random depths. If the objective is to find the deepest hole, there is no

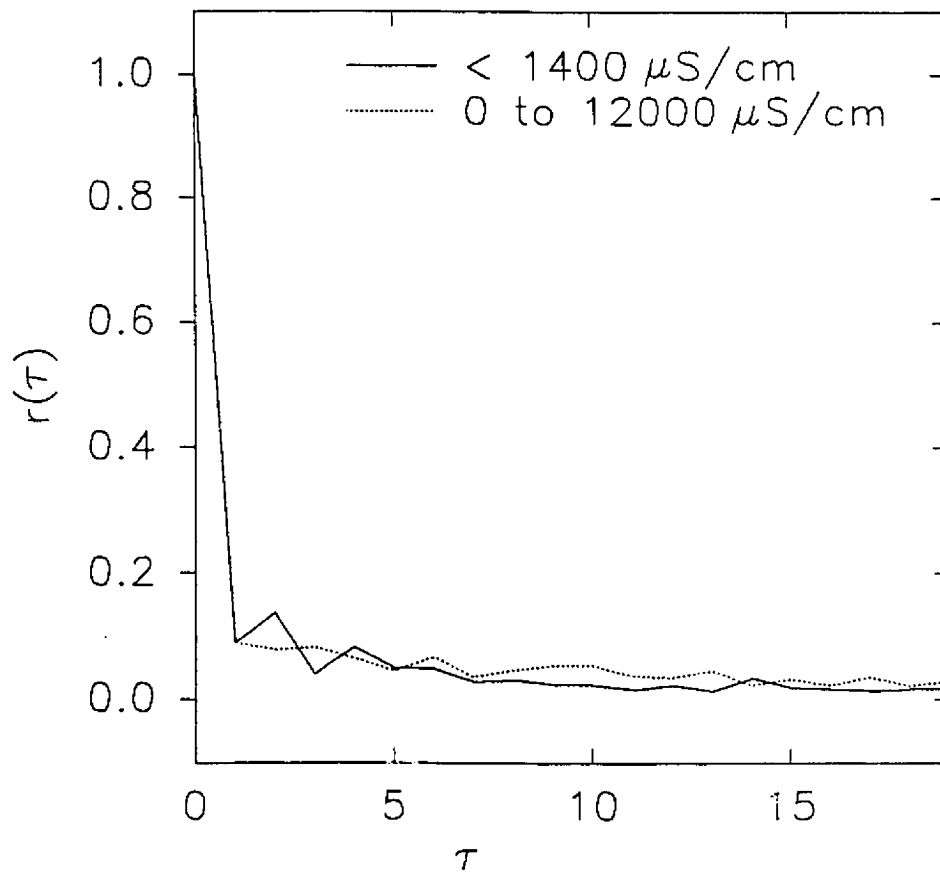


Figure 5.9 Autocorrelation function for the search space in the two conductivity regions employed.

systematic way to exploit the features of the surface (since there are none) and an exhaustive search is the only option. This is in contrast to a surface that is smoothly contoured, guiding an algorithm to the lowest point. The second factor that can lead to a low correlation is noise on the surface. If subsequent evaluations of the objective function for the same solution give significantly different results, then there will be noise on the surface. This will be the case, for example, when a new calibration and prediction set is used for each generation. Again, by physical analogy, consider the problem of finding the lowest point in a wheat field. Although the *surface* itself is smooth varying, small gulleys will be obscured by the varying heights of the wheat shafts, which are analogous to noise.

It is impossible to tell for certain from the autocorrelation function which of the two problems (or both) is the source of the low correlation. However, the nature of problem encoding would suggest that the solutions should be correlated and that noise on the search landscape is likely to be the bigger problem. This is also suggested by small, but nonzero, values of the autocorrelation function for the first several steps. The problem of surface noise is discussed in greater detail in the next section.

In order to show that the stability/noise problem has to do with the landscape and not the GA or its implementation, a simulation experiment was conducted. In this study the model was assumed to be:

$$y = 100x_1 + 100x_2 + 100x_3 \quad (5.11)$$

where y and x are the dependent and independent variables respectively. As with

the experimental data set, 10 variables and 40 terms were used. A set of 483 samples was generated using a random number generator with a uniform distribution. The new x replaced the ionic concentrations and the y was calculated as given in Equation 5.11 without the addition of measurement noise. With this set of data, GA quickly converged on the optimum (see Figure 5.10), although an interesting complication arose. Even though no measurement noise was included in the simulation, solutions determined by the GA always had several terms in addition to the three that were expected. These additional terms varied from run to run, whereas the three expected terms did not change. This is one of the big problems with models of this magnitude. Because of the enormous number of possible solutions (ca 10^{12}), correlations with noise are virtually guaranteed, even with the large number of samples here, and this will result in a number of models that are (apparently) superior to the true model. The solution in this case was simply to use a logical AND to combine the results of several runs. This left only the bits (terms) that corresponded to the true model. Although this simulation demonstrated that the GA was indeed working correctly (in fact, perhaps too well) it also showed some the difficulties in searching such large spaces.

5.3.3 Fixed Number of Terms

Perhaps the most important element in the successful implementation of any GA is the way the problem is encoded. When the GA does not perform as anticipated, an alternative strategy for representing the problem may be warranted.

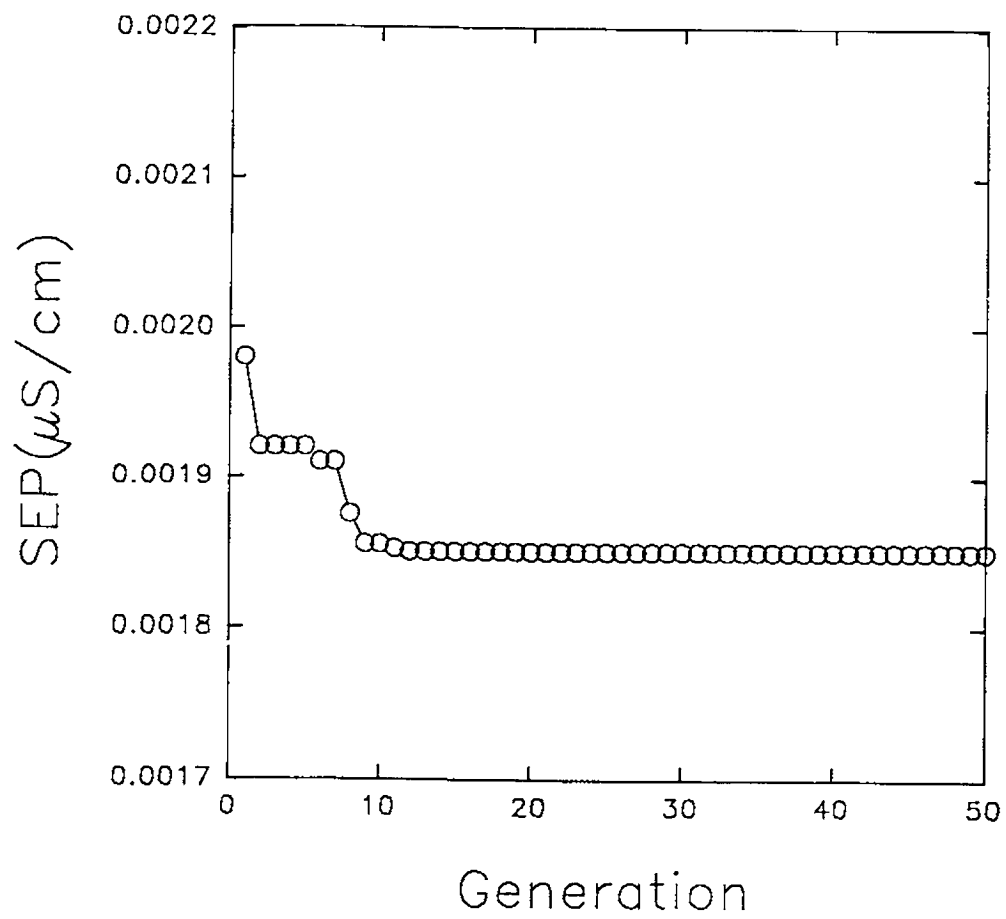


Figure 5.10 GA convergence for the simulated set.

In this case, this was accomplished by revising the question being asked. Instead of asking *what is the best model?*, the new question was *what is the best model with n terms?*. The latter question embodies the second form of subset selection represented in Figure 5.2 in which a fixed size subset is used, and for this problem a new GA approach had to be developed. In this case the strings were not encoded using 1's and 0's, but rather as integers from 1 to 40. Instead of including or excluding a term based on its value, models were built by including the first n terms. For example if we want to build a 4 term model from a 12 integer string, we would use the first 4 terms. If the string was 3 7 11 5 12 6 2 1 10 8 4 a model would be built using terms 3, 5, 7 and 11. In this term selection problem the model would be:

$$G = b_3(Ca^{2+})^1 + b_5(SO_4^{2-})^1 + b_7(NO_3^-)^1 + b_{11}(Na^+)^{3/2} \quad (5.12)$$

Another aspect in which this GA differed from the previous GA is in the exploration operators. Both the mutation and recombination operators were different than the ones used previously. Figure 5.11 is a pictorial representation of the mutation operator used in this new method for the case of selecting 4 terms from a total of 9.

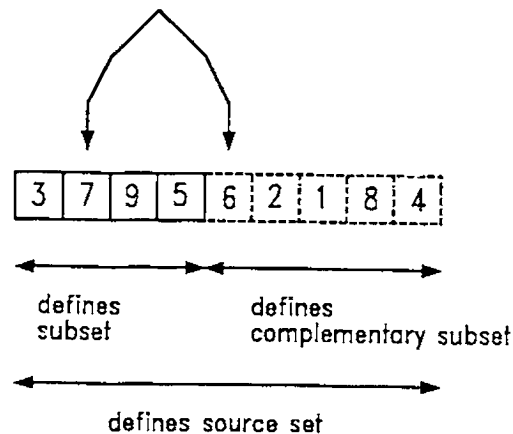


Figure 5.11 New mutation operator.

The model in Figure 5.11 was built with 4 terms. New strings were created by using the trade or transfer operator. Once a mutation is to occur, a gene from the solution subset is switched with a gene from the complementary subset. A simple recombination operator could be as follows, using the case of selecting 4 terms from a total of 9 for illustration. Consider two strings 123654789 and 124567893, where recombination is to occur between the third bits of the strings (i.e. 3 and 4). Find the location of the third bit from the first string (i.e. 3) in the second string (i.e. position 9) and find the location of the third bit from the second string (i.e. 4) in the first string (i.e. position 6). Switch the two bits in each string, if this gives a new solution, otherwise do nothing. The new strings after recombination would be 124653789 and 123567894. Although other operators could be developed, results seemed to be satisfactory using these two operators.

Figure 5.12 shows the GA convergence for subset of 40 terms in the two conductivity regions. Only the results from the first 5 terms are shown, *i.e.* what is the best 1 term model?, what is the best 2 term model and so on, up to 5 terms. The figure shows that all of the models converge on an improved solution. although this is not always apparent because the noisy surface often leads to individual cases with lower SEPs. Also there is not a significant change in some instances for the one and two terms models. This is expected, since there are only 40 combinations of solutions for the one term case, and 780 for the two term case where there are 100 strings in the initial population. The number of combinations increases rapidly after this, however, reaching 658,008 for the five term model. There is also a substantial improvement in the quality of the final model (as measured by the SEP) as the number of terms is increased, but this improvement becomes less significant after five terms.

These results can be more easily interpreted if the SEP at a particular generation is plotted against the number of terms in the model. Figure 5.13 shows the SEP as a function of the number of terms in the model at the 100th generation. The figure shows that for a small number of terms, the prediction error decreases as the number of terms increases, but then begins to degrade as the number of terms becomes large. This is consistent with overfitting, which results from modelling noise in the calibration data. Furthermore, the principle of parsimony dictates that the best model of those with equivalent performance is the one with fewest terms [99]. This would suggest that in both regions, the best

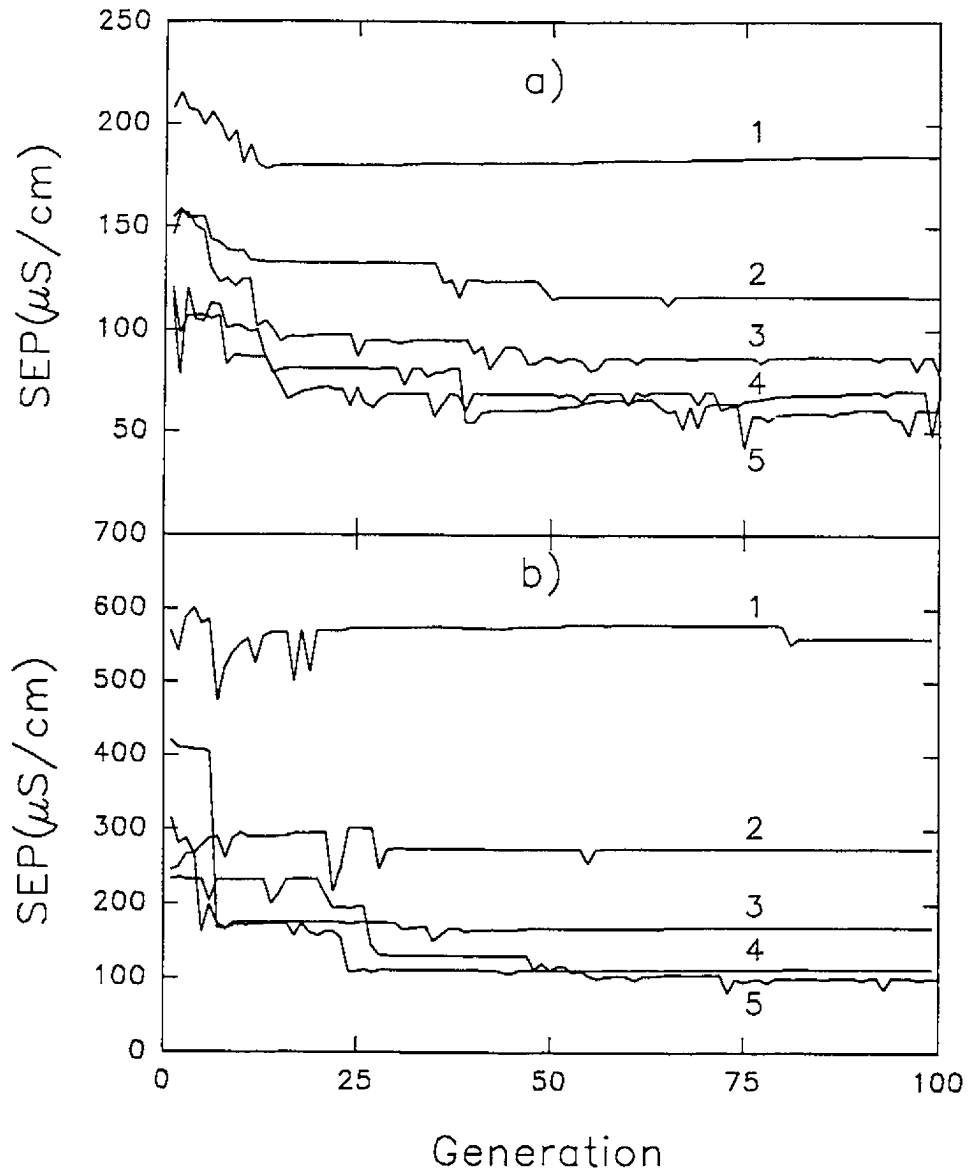


Figure 5.12 GA convergence for models of fixed size n in the two conductivity regions: **a)** $< 1400 \mu\text{S}/\text{cm}$ and **b)** all samples. The number of terms in the model is given above the corresponding curve.

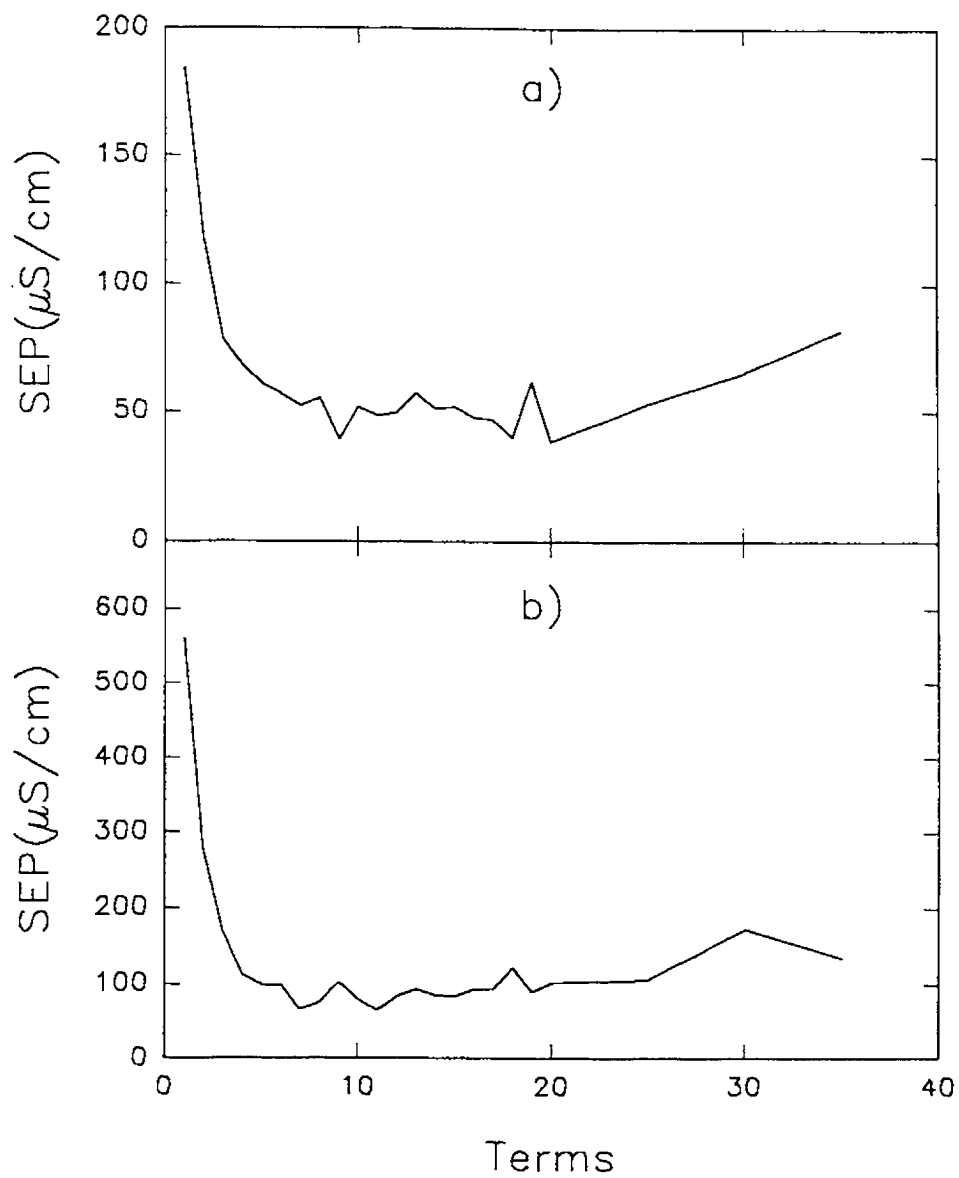


Figure 5.13 Plot of SEP versus number of terms for the two conductivity regions: **a)** $< 1400 \mu\text{S}/\text{cm}$ and **b)** all samples.

model would be one with fewer than 10 terms. Clearly, large values of n do not help improve the predictive ability.

An advantage of this formulation of the problem is that it led to more consistent solutions from the GA, something which was a problem in the original model with a variable number of terms. This raises the question *how has the new approach made the problem more manageable?* This can be answered by considering the complexity of the solution space for each approach. In the last section, it was noted that the autocorrelation function for the variable number of terms suggested a complex solution space, and it was postulated that this was due to noise on the surface. This noise makes it difficult to distinguish among solutions with similar SEPs. The role this noise plays in the success of the GA will be investigated here.

Since it is impractical to look at all of the models for the 40 term case, the 10 term case was investigated first. Although the following treatment has been developed for the 10 term case, the conclusions can be easily applied to the 40 term case. Table 5.2 lists the number of possible fixed size models for 1 to 10 terms for the 10 term case. The first GA approach tried to find the optimum solution from the set of all models, in this case 1023 models. Whereas the second approach looked only at subsets of these models, the number of which is given in Table 5.2. The ability of the GA to locate an optimum solution on a noisy surface will be determined by the shape of the surface (*i.e.* how flat it is in the region of the optimum) and the level of noise. Figure 5.14 shows the

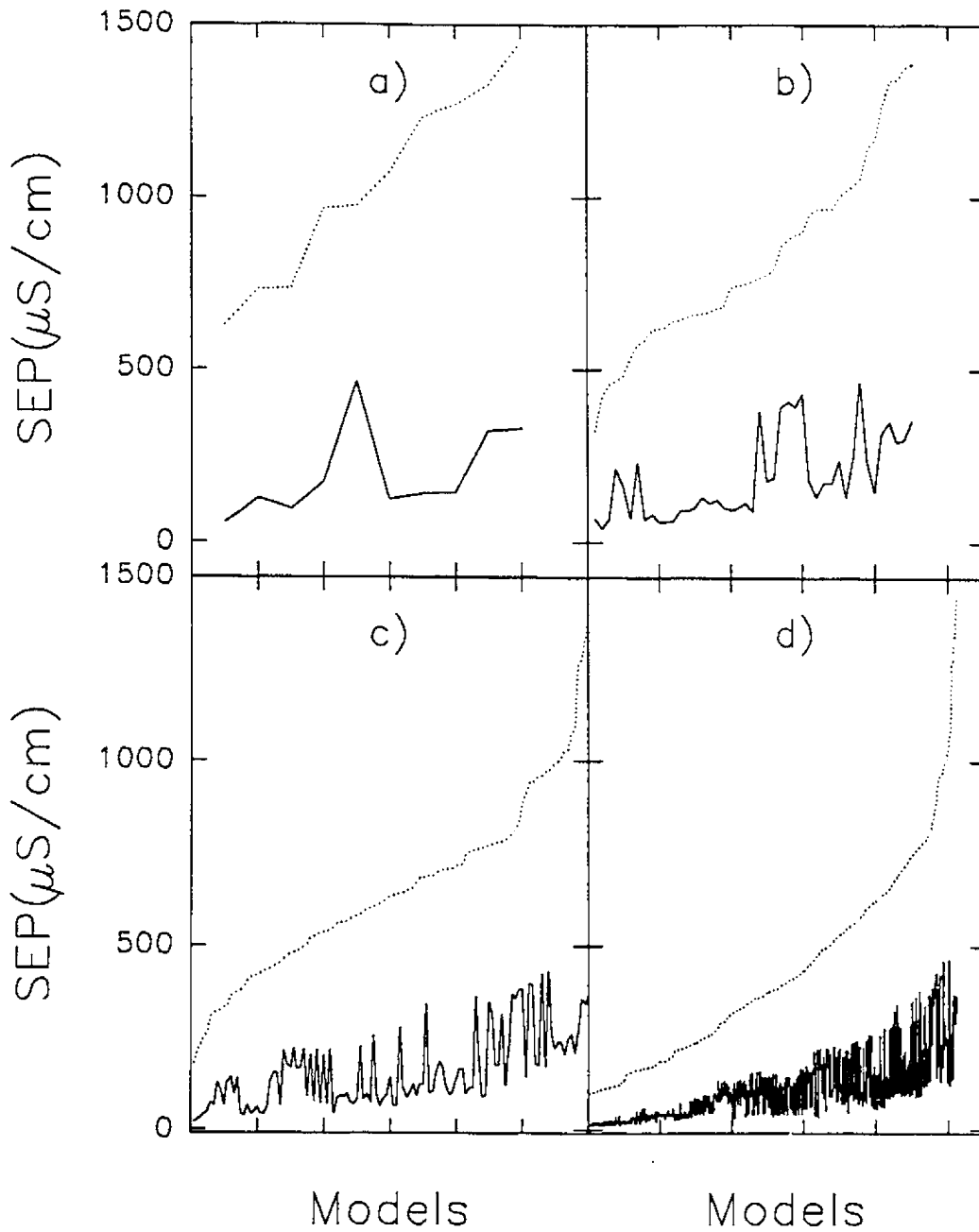


Figure 5.14 Comparison of search landscapes for the 10 term case with a) 1 term, b) 2 term, c) 3 term and d) variable size models. The dotted and solid lines are the average prediction error for the ranked model and its standard deviation, respectively.

Table 5.2 A simple example of the number of possible combinations

| k | Combinations | k | Combinations |
|---|--------------|----|--------------|
| 1 | 10 | 6 | 210 |
| 2 | 45 | 7 | 120 |
| 3 | 120 | 8 | 45 |
| 4 | 210 | 9 | 10 |
| 5 | 252 | 10 | 1 |

characteristics of the search landscape (dashed lines) for the 1 term, 2 term and 3 term subsets of the 10 term case, as well as the case of all models. The plots were generated by first exhaustively evaluating the average prediction error for all combinations (50 calibration and prediction sets evaluated for each combination) and then ranking these according to the average SEP. It will be noted that, as the number of terms increases, models become increasingly flat in the region of the optimum. The *degree of flatness* can only be assessed in relation to the noise on the surface however. For this reason, Figure 5.14 also shows the standard deviation in the SEP (solid line) for the associated model. The old GA would look at surface d whereas the new GA would look at a, b, and c. Although Figure 5.14 is helpful, one needs to clearly and concisely analyze the GA's ability to distinguish among models. A crude way of assessing surface complexity is to examine the change in SEP between adjacent solutions in the ranked sequence relative to the

surface noise (*i.e.* the standard deviation in the SEP). In general, the bigger the ratio $(\Delta\text{SEP})/\sigma_{\text{SEP}}$ the easier the region of the surface will be to search. In order to get a rough measure of model complexity, a parameter called the Model Distinction Indicator (MDI) was developed. MDI is calculated as,

$$\text{MDI} = \frac{\sum_{i=1}^{N-1} \left(\frac{\text{SEP}_{i+1} - \text{SEP}_i}{\sqrt{(\sigma_{i+1}^2 + \sigma_i^2)}} \right)}{(N - 1)} \quad (5.13)$$

In this equation, the term in the numerator of the summation is the difference in the SEP for adjacent models in the ranked sequence and the denominator is the pooled standard deviation for the two values. The summation is over all pairs of adjacent models, and is divided by $(N-1)$, where N is the number of solutions, to get an average value. In general one expects the MDI to decrease as the complexity of the noisy surface increases. The MDI is plotted in Figure 5.15 for the 1 to 9 term models, (the 10 term case has only one model so no MDI could be calculated). It will be noted that the MDI is directly related to the number of solutions possible, as might be expected since large surfaces tend to be more complex. This is not the only factor, however, since the shape of the surface and noise levels are also changing. Also shown in the figure (dashed line) is the MDI for the original variable size GA. Note that this is somewhat lower, indicating that the search space is more complex. Thus, the MDI seems to be a useful indicator of surface complexity and suggests why the fixed size approach seems to give better convergence and more stable solutions.

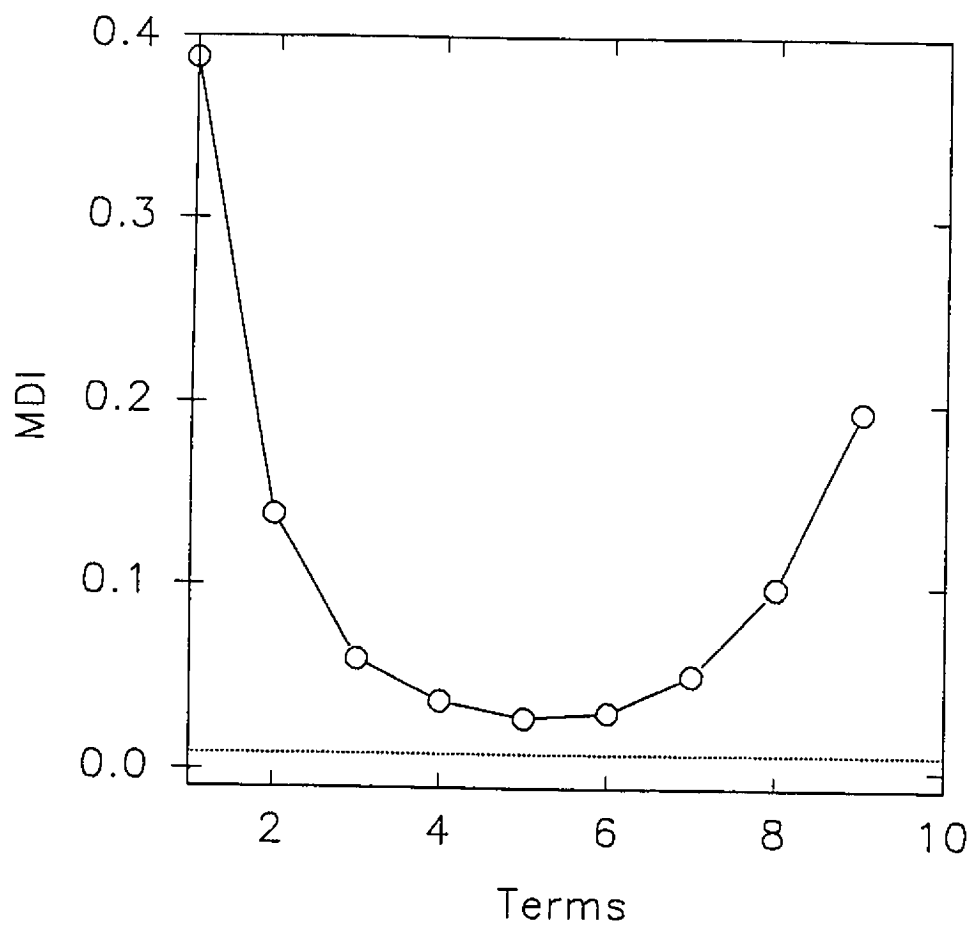


Figure 5.15 MDI as function number of terms for the fixed size subset selection for the 10 term case. The dashed line shows the magnitude of the MDI for the variable size subset selection.

For the 40 term case, analysis of model complexity in this way is more difficult. For a small number of terms the same approach can be used, and Figure 5.16 shows the characteristics of the search landscape in a manner analogous to Figure 5.14 for the 10 term case. It will be noted that the surfaces in this case are flatter, already suggesting a more complex problem. It would be useful to evaluate the MDI for this case as well, but this calculation becomes impractical very quickly because of the enormous number of solutions as the number of terms is increased. To circumvent this problem, a Monte Carlo approach was used to calculate the MDI for models with a large number of terms. To do this, random solutions were chosen from the search space and the average SEP and its standard deviation were evaluated as before (*i.e.* 50 calibration and prediction sets each). Also, as before, the solutions were ranked and the MDI was evaluated, except that m solutions were used rather than the total number, N , to give $MDI(m)$. The true MDI was then estimated by,

$$MDI = \frac{m}{N} MDI(m) \quad (5.14)$$

This approach rationalizes that the complementary set of $(N-m)$ solutions are likely to fall randomly between the selected solutions, so the ΔSEP is likely to decrease by a factor of m/N , whereas the noise level should remain roughly the same. The validity of this approach was checked with the three term model. The exhaustive calculation gave $MDI = 7 \times 10^{-4}$, whereas the Monte Carlo method gave $MDI = 4 \times 10^{-4}$, approximately the same.

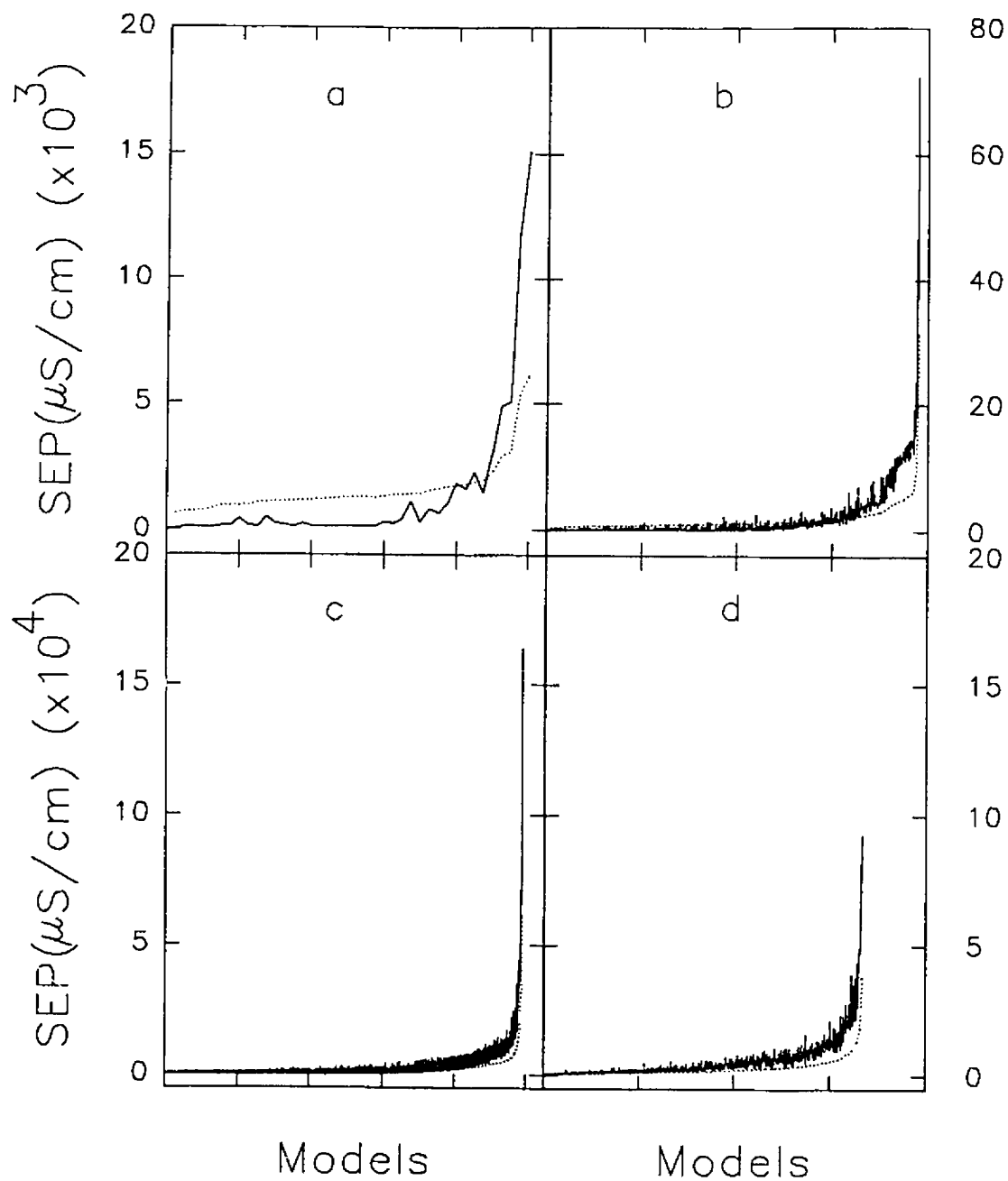


Figure 5.16 Comparison of search landscapes for the 40 term case with a) 1 term, b) 2 term, c) 3 term and d) variable size models. The dotted and solid lines are the average prediction error for the ranked model and its standard deviation, respectively.

A figure analogous to Figure 5.15 for the 40 term case could not be developed since this would require evaluation of about 50×10^{12} models. However the MDI was able to be calculated for the 1 to 3 term cases, and they are 0.117, 0.008, and 0.0007 respectively. For the all term case the MDI was 1.8×10^{-12} . This value was calculated using 1000 strings and Equation 5.14. As with the 10 term case, for the points calculated, the MDI for the variable size model is lower than for the fixed size case, reflecting the greater difficulty in searching the solution space for the former.

5.3.4 Comparison with Iterative Searching

Although the results in the last section demonstrate that the GA apparently converges to an optimum solution when searching for the best n-term model from a set of 40 terms, a question that remains is *could the results have been obtained more easily by another method?*. One method that is commonly used for this purpose is called the iterative method. The iterative method is analogous to univariate optimization and searches the space by first finding the best one term model and then using this to search for the best second term. This procedure continues until the *best* n terms have been found. The number of possible combinations using the iterative method and the GA method differs considerably, as shown in Table 5.3. It is clear from the table that the search space for the iterative approach is only a fraction of that for the GA. However, the iterative solution assumes that the best n-term model will contain the terms from the best

(n-1) term model. While this may often be valid, it restricts the search space and may lead to suboptimal solutions.

Table 5.3 Comparison of the number of possible evaluations for iterative and GA approaches.

| Terms | Iterative | GA |
|-------|-----------|---------|
| 1 | 40 | 40 |
| 2 | 79 | 780 |
| 3 | 117 | 9,880 |
| 4 | 154 | 91,390 |
| 5 | 190 | 658,008 |

To compare the performance of the GA and the iterative approach for the 40 term case, the *all samples* range was used and the average SEP was calculated for the best n term model found by each method, where n ranged from 1 to 10. The results are plotted in Figure 5.17. The GA and iterative methods find identical 1 term solutions, but the GA gives significantly better results for the 2 and 3 term models, illustrating the weakness in the assumptions made by the iterative method. The results are again identical for the 4 and 5 term models, even though the search space for the GA was more than 3000 times as large for the 5 term case. For models with more than 5 terms, the iterative method actually gives

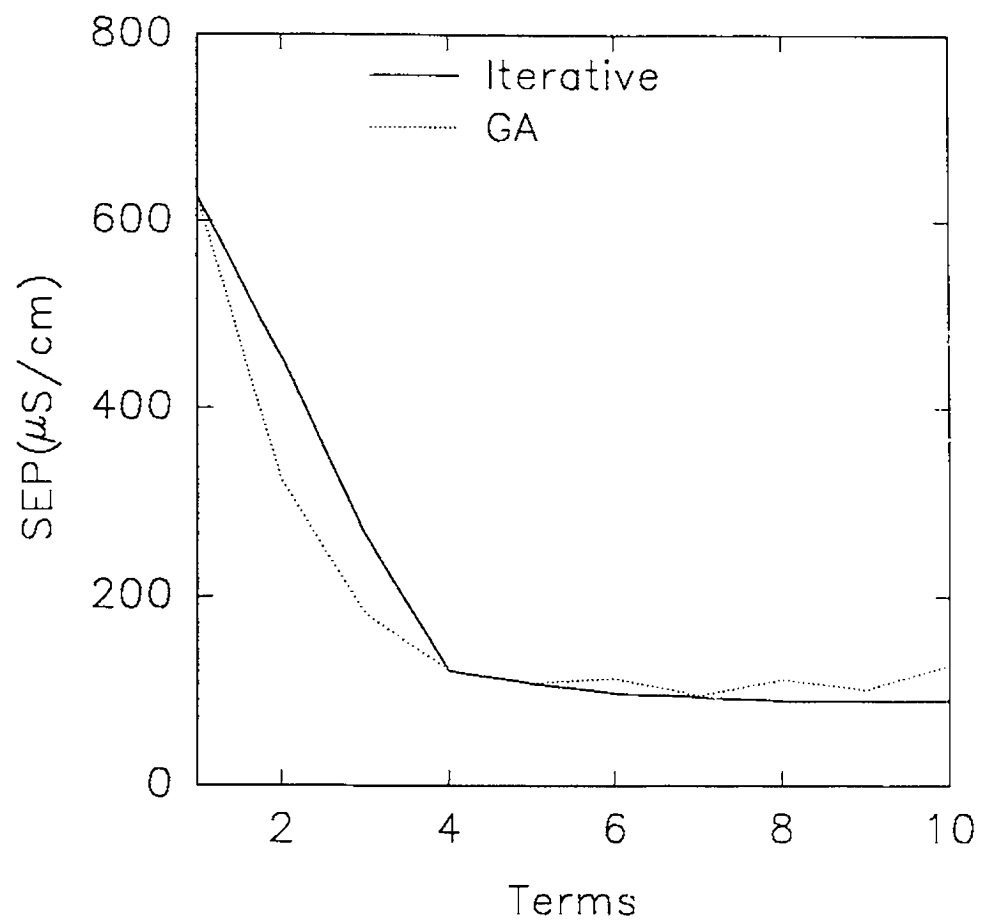


Figure 5.17 Comparison of the iterative and GA term selection solutions.

slightly better results in this case, probably due to the increasing complexity of the noisy surface being explored by the GA. In any given problem, it is probably wise to use both approaches, balancing the more extensive GA search with the more restrictive but more efficient iterative method. The terms for the best models found by each method are given in Table 5.4. for the first five models. The results are

Table 5.4 Best models found by the iterative method and the GA for the all samples region.

| Terms | Iterative | GA |
|-------|--|---|
| 1 | $(\text{Mg}^{2+})^1$ | $(\text{Mg}^{2+})^1$ |
| 2 | $(\text{Mg}^{2+})^1, (\text{Na}^+)^1$ | $(\text{Na}^+)^1, (\text{Ca}^{2+})^1$ |
| 3 | $(\text{Mg}^{2+})^1, (\text{Na}^+)^1, (\text{K}^+)^1$ | $(\text{Na}^+)^1, (\text{Ca}^{2+})^1, (\text{K}^+)^1$ |
| 4 | $(\text{Mg}^{2+})^1, (\text{Na}^+)^1, (\text{K}^+)^1, (\text{Ca}^{2+})^1$ | $(\text{Mg}^{2+})^1, (\text{Na}^+)^1, (\text{K}^+)^1, (\text{Ca}^{2+})^1$ |
| 5 | $(\text{Mg}^{2+})^1, (\text{Na}^+)^1, (\text{K}^+)^1, (\text{Ca}^{2+})^1,$ & $(\text{SO}_4^{2-})^1$ | $(\text{Mg}^{2+})^1, (\text{Na}^+)^1, (\text{K}^+)^1, (\text{Ca}^{2+})^1,$ $(\text{K}^+)^1, \& (\text{SO}_4^{2-})^1$ |

consistent with those found in Chapter 4, especially with respect to the dominance of the cation terms. It will also be noted that only first order terms are observed for the first five models. This perhaps explains why the use of the higher order terms in the 40 term model did not give results significantly superior to those in Chapter 4 with the 10 term model, since the first order terms have the best predictive ability.

5.4 Conclusions

The GA method did not provide any better results than, the 10 terms considered in Chapter 4. In Chapter 4, only the first order terms were considered. The GA showed that the best models contained only the first order terms. Although the GA did not suggest any new terms it was able to test other terms that would not have been rigorously test by other methods.

In addition to the problems outlined here for term selection Leardi [100,101] also outlines term selection problems.

Ordering of Disordered Data Sets

6.1 Introduction

Thus far, the work presented has dealt with two kinds of data sets: ordered and disordered. In Chapters 2 and 3 it was shown that pre-existing knowledge of order in a data set can be of great utility in determining its rank and assist in identifying components present (since the regions of lower rank are located). In Chapter 4 it was noted that, although a data set may be disordered, it may possess an inherent order and it is only because we don't know the ordinal variable that the data *appear* disordered. In such a case, it may be possible to organize the samples in a logical manner according to similarities in the measured features, such as the spectra. This assumes that changes in the ordinal variable will be reflected by a relatively smooth transition in the feature space. The ordering of samples according to the features has at least three useful functions: (1) it allows relationships among the samples to be more clearly elucidated, (2) it may permit the effective use of evolving rank analysis methods, such as EPA and evolving factor analysis, and (3) it may aid in the discovery of an ordinal variable, which could provide a better understanding of the system dynamics. Because of these potential benefits it was the objective of the work presented in this chapter to seek methods for ordering disordered data sets.

6.1.1 The Ordering Problem

The concept of reordering is illustrated pictorially in Figure 6.1, where the objects have been reordered by the reordering operator, R , using the features of shape and shading. The ordering process reveals a systematic transition among the objects (samples) that otherwise would not have been known. For a typical chemical data set, there will be more features and more samples, but the principles remain the same.

Clearly there are two problems in this reordering process: (1) how to determine when the samples are correctly ordered, and (2) how to proceed with the reordering process to obtain that objective efficiently. The former problem has no single solution, but the approach used in this work is outlined in section 6.1.3. The latter problem, finding the optimum order, is non-trivial and also has a number of possible solutions. In trying to order samples there are a great number of possible ways to arrange the data set. For example, 3 samples can be arranged in six ways: (1 2 3), (1 3 2), (2 1 3), (2 3 1), (3 1 2) and (3 2 1). In general, for n samples the number of possible arrangements, or the number of permutations is $n!$, where $!$ denotes factorial. As n increases the number of permutations increases rapidly. For $n = 2, 4, 6, 8$ and 10 , the number of permutations are 2, 24, 720, 40 320, and 3 628 800, respectively. That is, for each incremental increase in n the number of samples, the number of permutations increases by a factor of n . For a typical chemical set, we might have $n = 30$, for which there are 2.65×10^{32} permutations. One can easily appreciate that evaluating all

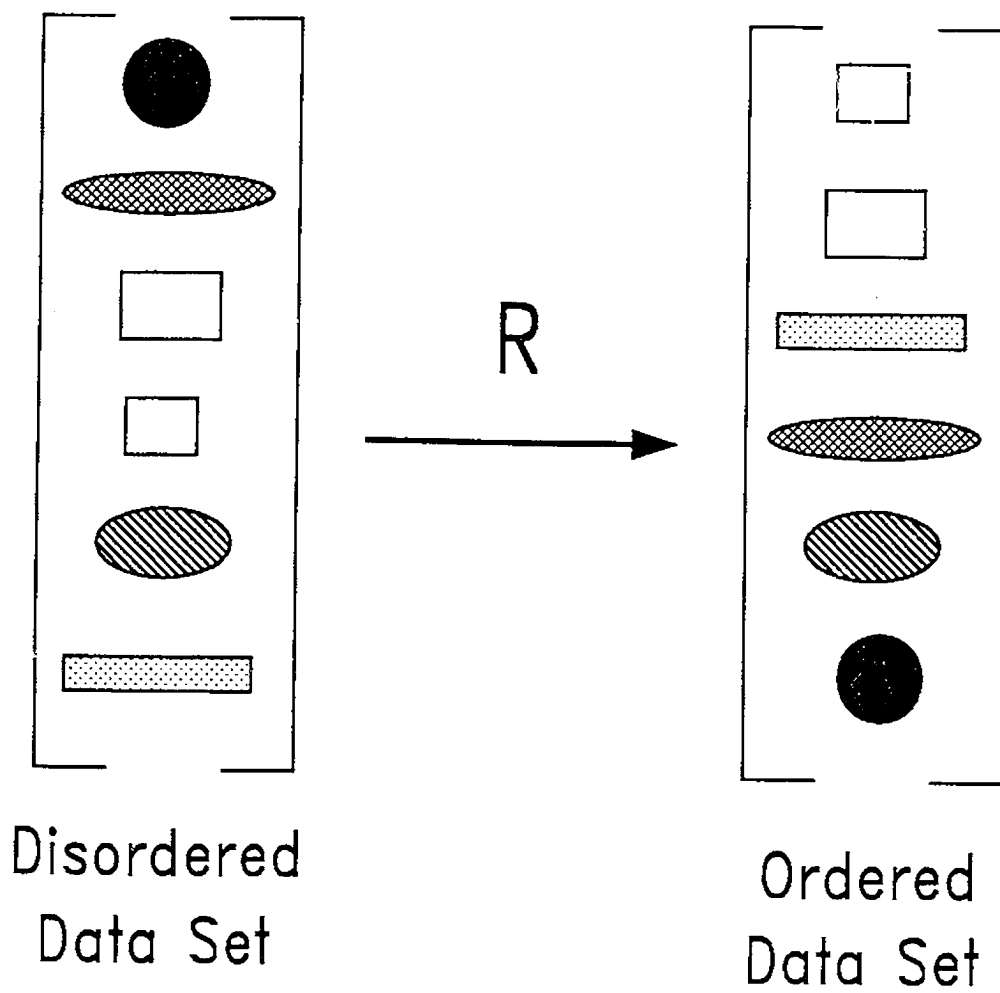


Figure 6.1 A pictorial representation of the ordering of samples.

arrangements would be almost impossible, so an efficient optimization procedure is essential. In this work, the same general approach to optimization as was used in Chapter 5, genetic algorithms, was employed for the ordering problems. Details are discussed in section 6.1.4.

6.1.2 Problem Types

The reordering process is applicable to two general problem types: (1) cluster analysis and (2) evolving data sets. The distinction is one of degree rather than absolute categorization. In cluster analysis, there are distinct boundaries between groups of samples and the ordinal variable can be considered to be a class designation (*i.e.* which group does the sample belong to?). For evolving data sets there is usually a much smoother transition between samples, and the ordinal variable can be considered to be continuous (*e.g.* time, pH). These two cases are considered in more detail below.

Cluster analysis is defined as the study of algorithms and methods for grouping, or classifying objects [102]. Cluster analysis techniques are often classified in one of two ways: partitioning or hierarchical [103]. Partitioning techniques construct groups, or classes, by defining regions of the measurement space that determine a sample's membership in a class. An example of partition clustering for a simple case of two measurements (features) is shown in Figure 6.2a. The lines divide the space into three regions and the points are measurements from the 15 samples.

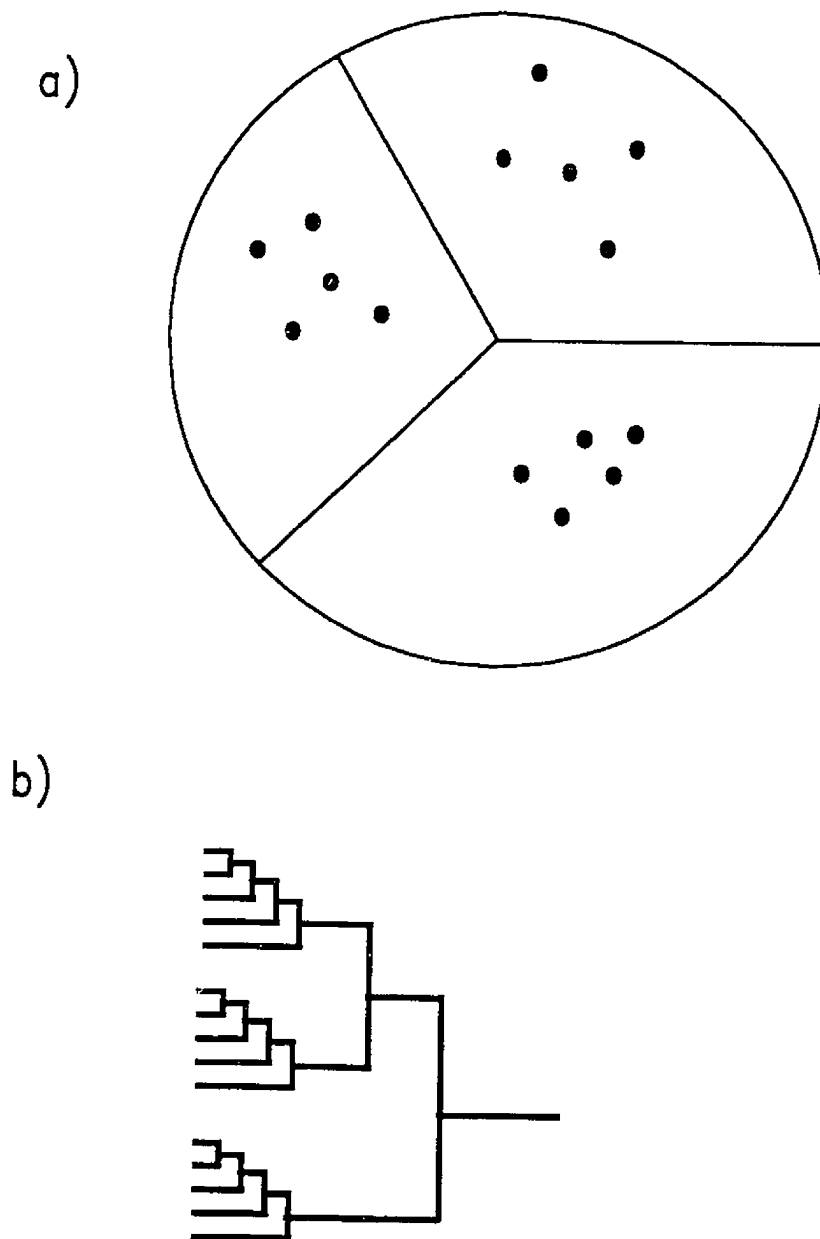


Figure 6.2 Diagrams illustrating two common cluster analysis techniques: a) partitioning clustering analysis and b) hierarchical clustering analysis.

Generally, the effective partitioning classification (or *pattern recognition*) is only effective when there are samples whose classification is known so that they can be used to generate the partitions (*i.e.* supervised pattern recognition). A more difficult problem is finding natural clusters within the data with no *a priori* knowledge of class structure (*i.e.* unsupervised pattern recognition). For this purpose, hierarchical clustering analysis (HCA) is normally used.

Hierarchical techniques classify samples by grouping them according to their similarity to one another in feature space. The clustering is usually represented in the form of a *dendrogram* [104]. This can be constructed in a manner similar to the following procedure. First, a distance matrix is calculated which contains the distance from every sample to every other sample in feature space (normally Euclidean distances are used, but other variants are possible). This will be a $n \times n$ matrix (where n is the number of samples, or objects) as shown below,

$$\begin{bmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \vdots & \vdots & \dots & \vdots \\ d_{n1} & d_{n2} & \dots & d_{nn} \end{bmatrix} \quad (6.1)$$

where d_{ij} represents the distance from sample i to sample j . Note that the diagonal of this matrix will contain zeros and it will be symmetric across the diagonal. Next, the smallest distance is found and those two samples are combined into one cluster. The distance matrix is then recalculated, treating the two objects as one,

thereby reducing the size of the distance matrix by 1 row and column. This process is continued until there is only one cluster remaining. (Note that in calculating the distance to the cluster, the median of the cluster is normally used, but other variations are also possible). Finally, a dendrogram, such as that represented pictorially in Figure 6.2b is generated. In this picture, the horizontal length of the lines indicates the distance between the corresponding objects or cluster. In this example (Figure 6.2b), three clusters are evident (*i.e.* distance between is greater than distance within).

Hierarchical clustering can be very effective for exploratory data analysis, but one of its weakness is that it is hierarchical. Because it tries to compress the information about several samples into a single cluster, important relationships among the individual samples may be obscured. The ordering process, on the other hand, seeks to connect each sample with the one closest to it in the feature space. In this way it is non-hierarchical and is analogous to the classical "travelling salesman problem" in that it seeks the order that will minimize the total distance connecting all of the samples. Furthermore, this approach may be more effective in cases where clustering is not distinct.

The second type of problem involves ordering evolving data sets. An evolving data set refers to one in which the component contributions for each sample are related to some continuous ordering variable. A common chemical example is a simple reaction of the type $A \rightarrow B$. Since the reaction leads to a continuous decrease in the amount of A and an increase in the amount of B, the

measurements in this case are ordered by time. This type of data, like the chromatographic and titration data sets in Chapter 2 and 3, is said to be naturally ordered. Of course, there is no point in reordering samples that are naturally ordered, but it would be useful to develop a methodology to order a disordered data set. It is hoped that ordering data sets so that their evolutionary nature is discovered will reveal new information about the system.

6.1.3 The Objective Function

In order to determine when the best sequence of samples has been found, some quantitative measure of the relative quality of a particular order is necessary. To illustrate how this might be accomplished, consider the problem of ordering the integers between 1 and 5. There are 120 arrangements of these five integers, but only three will be considered for illustration. These three possible arrangements are (1 3 2 5 4), (5 1 3 4 2) and (1 2 3 4 5). One can distinguish among these sequences if they are summed as shown in Figure 6.3. Simply taking the sum (or the integral) of each arrangement is not enough since this is identical for all sequences. It is necessary to take what can be called the cumulative sum of the cumulative sum in order to find the preferred arrangement. This *double sum* distinguishes among the different arrangements. It is clear from Figure 6.3 that the ordered set, *i.e.* case (c), has the lowest objective function of the three, and, in fact of all possible arrangements. Thus, this can be used to determine the correct order for this problem. Although an identical approach will not work for chemical

$$\begin{array}{cccccc}
 \text{a)} & 1 & + & 3 & + & 2 & + & 5 & + & 4 \\
 & & & \parallel & & \parallel & & \parallel & & \parallel \\
 & & & 4 & + & 6 & + & 11 & + & 15 \\
 & & & & & & & & & \parallel \\
 & & & & & & & & & \boxed{36}
 \end{array}$$

$$\begin{array}{cccccc}
 \text{b)} & 5 & + & 1 & + & 3 & + & 4 & + & 2 \\
 & & & \parallel & & \parallel & & \parallel & & \parallel \\
 & & & 6 & + & 9 & + & 13 & + & 15 \\
 & & & & & & & & & \parallel \\
 & & & & & & & & & \boxed{43}
 \end{array}$$

$$\begin{array}{cccccc}
 \text{c)} & 1 & + & 2 & + & 3 & + & 4 & + & 5 \\
 & & & \parallel & & \parallel & & \parallel & & \parallel \\
 & & & 3 & + & 6 & + & 10 & + & 15 \\
 & & & & & & & & & \parallel \\
 & & & & & & & & & \boxed{34}
 \end{array}$$

Figure 6.3 Illustration of the *double sum* discrimination ability for the two disordered arrangements (parts a and b) and ordered arrangement (part c).

problems, this numerical example is used for illustration later.

For chemical problems, a somewhat different approach to the objective function is taken. In this case, the goal is to maximize the similarity of adjacent samples. This can be done by minimizing the variance of adjacent samples in such a way that optimum continuity is maintained. The objective function must also be flexible enough to accommodate a range of different data sets. To illustrate the objective function used, a simple cluster analysis example will be employed. This example consists of 30 samples, each with two simulated measurements (x_1 and x_2). The samples are organized into three classes and they are shown projected into measurement space in Figure 6.4. The calculation of the objective function used involves several steps. The process, illustrated in Figure 6.5, is outlined below:

1. First the samples are grouped by using a sliding window of length p . The first such window ($p = 9$) is shown in Figure 6.5 and uses samples in positions 1 to 9. The second window moves one position to the right and uses samples in positions 2 to 10. The window continues to move right until the last sample is encountered, *i.e.* the last window would include samples in positions 22 to 30. The window size can be adjusted for the problem under consideration.
2. For each of these windows a variance is calculated for each variable. Using the example presented in Figure 6.4 if the first window contained samples 3, 6, 8, 13, 14, 18, 21, 23, and 25, the variance for the x_1 will be

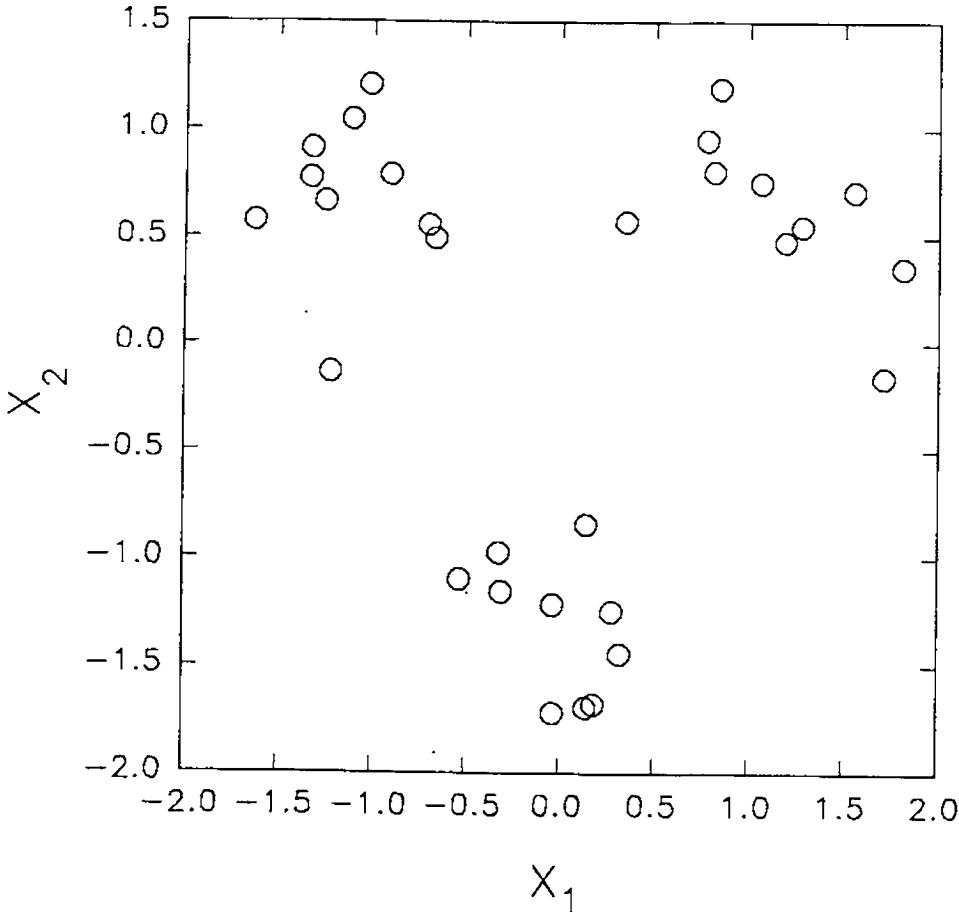


Figure 6.4 The clustering example to illustrate the new objective function.

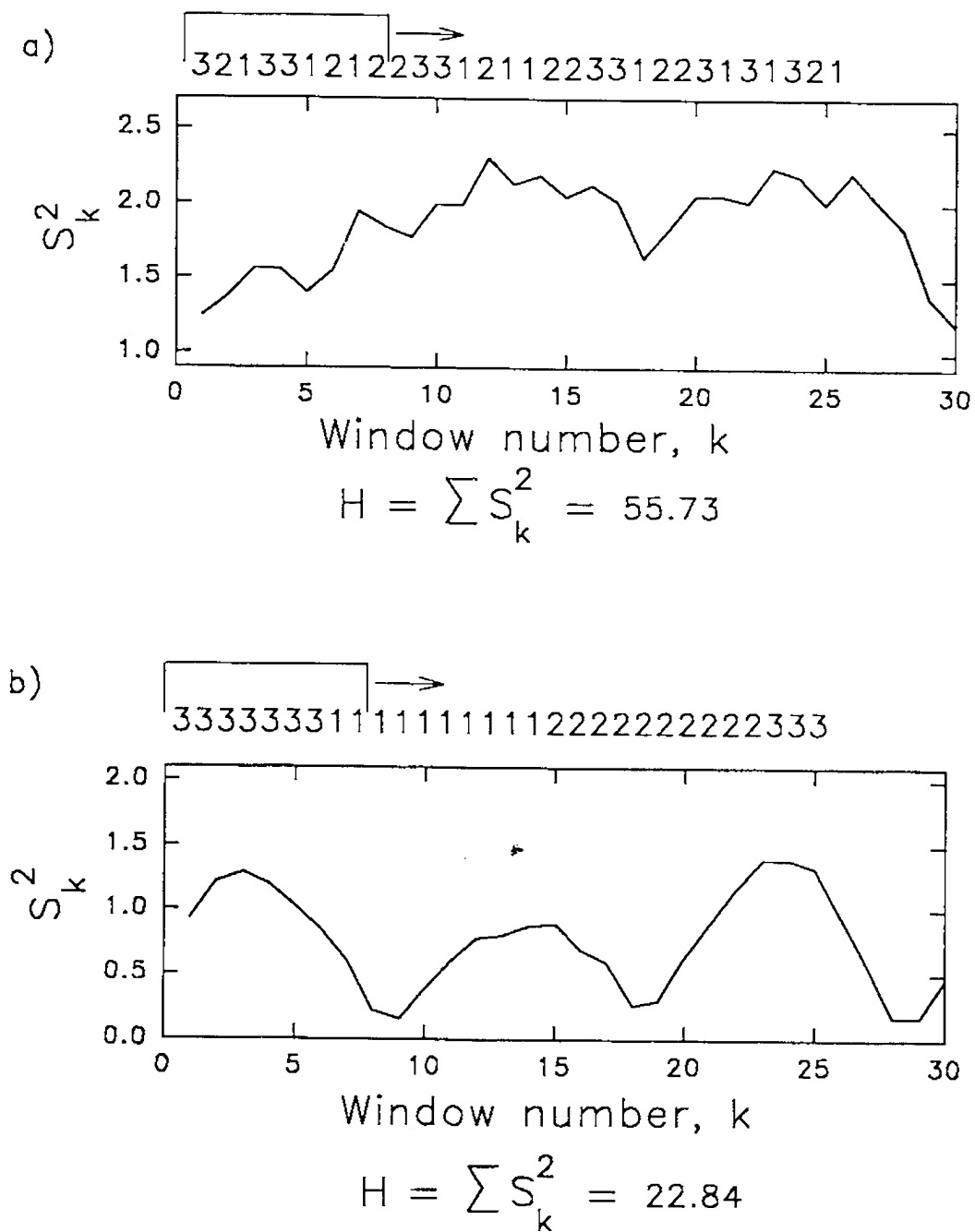


Figure 6.5 Illustration of calculations for the new objective function with a) a disordered data set and b) a ordered data set.

calculated using -1.33, -0.70, -0.67, 1.81, 0.34, 0.81, -0.03, 0.32 and -0.04.

Likewise a variance would be calculated for x_2 .

3. The variance for each window, s_k^2 , is calculated as the mean of the variances for each variable in each window. Thus, the overall calculation is,

$$s_k^2 = \frac{1}{n(p-1)} \sum_{j=1}^n \sum_{i=k}^{k+p-1} (x_{ij} - x_{mj})^2 \quad (6.2)$$

where n is the number of variables ($n = 2$ in this case), x_{ij} is the value for variable j for sample i , and x_{mj} is the mean for variable j in window k . This function is shown in Figure 6.5a for the disordered data set and Figure 6.5b for the ordered set.

4. The objective function is calculated as the integral of s_k^2 ; that is,

$$H = \sum_{k=1}^{m-p+1} s_j^2 \quad (6.3)$$

where m is the number of samples. In some cases, it was found advantageous to integrate the standard deviation rather than the variance for the objective function; that is,

$$H' = \sum_{k=1}^{m-p+1} s_j \quad (6.4)$$

The idea in using these objective functions is to minimize the variability within each of the overlapping windows. This should be accomplished when the samples are correctly ordered so that there is a maximum similarity between

adjacent samples. As shown in Figure 6.5a, the disordered data set leads to what is essentially a random trace for the mean variance, since samples from different classes will be included in each window. For the ordered data set, the variance trace is much lower, except at the boundaries between classes. Consequently this leads to a smaller value for the objective function.

In defining the objective function, two important aspects have not been mentioned. The first has to do with scaling of the variables. If the measurements represent significantly different ranges, (e.g. different absolute concentrations or different types of measurements) or are on a different scale some kind of scaling is necessary to ensure a consistent measure of similarity. This problem is addressed later for particular cases. The second important aspect concerns the inclusion of samples at the beginning and end of the sequence. This problem is treated in the next section.

6.1.4 Ordering Modes

As noted in the last section, a problem arises with the objective function as presented in equations 6.3 and 6.4. This problem relates to the fact that samples near the ends of the sequence are included in the variance calculation in fewer than p windows, unlike the samples in the middle of the sequence which will appear in p windows. This means that very dissimilar samples may be pushed to the ends, where their influence on the objective function will be reduced. The result is that their order in the sequence may not be correctly specified. There are

two possible remedies to this problem:

1. At the end of the sequence (*i.e.* $k=m-p+1$), the following $(p-1)$ windows can include the samples at the beginning of the sequence. In other words, the sliding window cycles the sample sequence in a circular fashion. For instance, for the example in Figure 6.4, window number 29 would include samples 29, 30, 1, 2, 3, 4, 5, 6 and 7.
2. At the beginning and end of the sequence, a variable size sliding window can be used. In the example presented, the first window would include samples 1 and 2, the second would use (1, 2, 3), the third would include (1, 2, 3, 4) and so on, until the desired window size is reached. At that point, the window size remains constant and it simply shifts position. A similar contraction of the window would occur at the right-hand side of the sequence.

Both of these approaches have their advantages, but which one is used depends on the nature of the data set. For a *closed* data set, there is a cyclical relationship in the sequence, as shown in Figure 6.6a. In this case there is a relationship between the last sample in the sequence and the first one, and the cyclical window approach is an elegant solution to the boundary problem. In chemistry, this case does not arise often, but can be approximated by certain problems in cluster analysis, such as the example given in Figure 6.4. In these cases, there are a number of redundant solutions due to the nature of the data sets. (*i.e.* there is no definitive beginning and end). The number of redundant

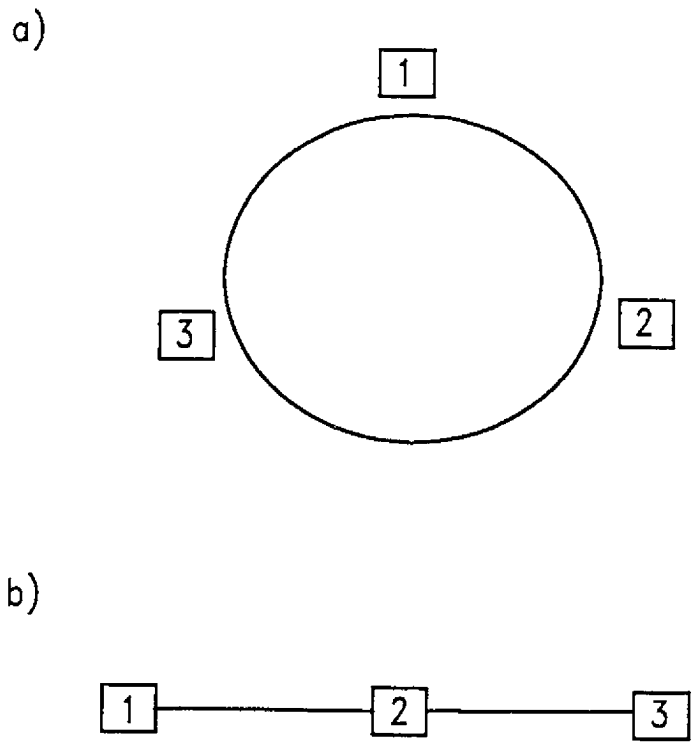


Figure 6.6 Ordering modes: **a)** closed data set, **b)** open data set.

solutions in this case is $2m$, where m is the number of samples.

A much more common case in chemistry is an *open* data set, as represented in Figure 6.6b. In this case, there is no relationship between the first and last samples. If a cyclical sliding window is used in this case, problems arise because the reordering algorithm tries to bridge the first and last sample with some of the intermediate samples. For these cases, there will only be 2 redundant solutions those corresponding to the forward and reverse sequences.

6.1.5 Genetic Algorithms

The two preceding sections dealt with finding a suitable objective function for the reordering problem. Having addressed this problem, the task remaining is to find a suitable way to search for the optimum order. The approach used here is the general method outlined in Chapter 5, genetic algorithms.

As mentioned in Chapter 5, the problems a GA can handle can be divided into three classes: 1) numerical parameter estimation, 2) subset selection and 3) sequencing. Much of the work on GAs has focussed on numerical parameter estimation [89-93], and this will not be discussed in this work. The term selection problem, the subject of Chapter 5, is classed as a subset selection problem, and ordering, the focus of this chapter, can be considered to be a sequencing problem. By ordering, or sequencing, is meant that samples are arranged according to their features or responses. The goal is to go from a disordered arrangement of samples to an ordered arrangement. This can be achieved by using the ordering

operator (R) as shown in Figure 6.1. A method using GAs to order samples will be outlined here.

As previously noted, the outline of the GA is same for any problem being studied. The aspects that differ from problem to problem are the problem definition and the exact search methodology used. For this present problem the strings consist of a sequence of numbers, where the number represents the position of the sample in the original sequence and its location in the string represents its position in the test sequence. The objective function is calculated as described in the preceding two sections. The search methodology refers to the exploitation and exploration operators (*i.e.* reproduction recombination and mutation) and the searching parameters (*i.e.* number of strings, selective reproduction rate, recombination rate and mutation rate) used.

The recombination operator used in this work is described as follows. Let P_1 , and P_2 be the two parent strings (or source strings), and C_s will denote a child string, *i.e.* the result of the recombination of two parent strings. The recombination will be described using the two disordered strings from Figure 6.3. Let $P_1 = 1\ 3\ 2\ 4\ 5$ and $P_2 = 5\ 1\ 3\ 4\ 2$. The production of the child strings is outlined below:

1. Create a binary template (T) that is equal to the length of the parent strings.

A possible template could be:

$$T = 0\ 1\ 0\ 1\ 1$$

This template was generated simply by employing a uniform random number generator to produce an integer between 0 and 2^N , where N is the

number of samples to be ordered.

2. Let C_s initially be a copy of P_1 :

$$C_s = 1\ 3\ 2\ 4\ 5$$

3. Align C_s , T and P_2 as shown below.

$$C_s = 1\ 3\ 2\ 4\ 5$$

$$T = 0\ 1\ 0\ 1\ 1$$

$$P_2 = 5\ 1\ 3\ 4\ 2$$

4. Select the elements in C_s that correspond to the positions of the 1 bits in T .

$$C_s = 1\ \underline{3}\ 2\ \underline{4}\ \underline{5}$$

5. Select the elements in P_2 that correspond to the elements selected in C_s .

$$P_2 = \underline{5}\ 1\ \underline{3}\ \underline{4}\ 2$$

6. Rearrange the elements in C_s according to their order in P_2 . The resulting child string is:

$$C_s = 1\ 5\ 2\ 3\ 4$$

To create the other child string, the procedure starts from Step 2 with the role of parents switched. In this case the other child string is 5 1 3 2 4.

One of the parameters in the recombination step that is adjustable is the number of bits that are activated in the template. In this work, this parameter was set at 50 % (*i.e.* the number of zeros equalled the number of 1's in the binary template) and this will be referred to as the exchange rate. One approach to selecting the template might be to always activate the same number of bits, but

the actual percentage of bits activated was around 50 %. Because of the method used to generate the template, the mean number of activated bits can be calculated from the binomial distribution as half the total number of samples. Furthermore, the standard deviation in the exchange rate can be also calculated from the binomial distribution, giving,

$$\text{Exchange Rate} = 50\% \pm \frac{50\%}{\sqrt{N}} \quad (6.5)$$

where the latter represents the absolute standard deviation in the exchange rate. The exchange rate is defined as percentage of bits in the total population that undergo recombination. For example, for $N = 30$, the standard deviation in the exchange rate is about 9 %. It was felt that this approach was more consistent with the natural process, but alternatives were not rigorously investigated.

The mutation operator that was used involved switching locations of the integers in the string. For example consider the string:

1 3 2 4 5

The first step was to select the locations where the switch is to occur. If the switch was to occur at the second and fourth bit locations,

1 3 2 4 5

Then the mutated string was,

1 4 2 3 5

The mutation rate, an adjustable parameter in this work, refers to the percentage of child strings to which a single mutation of this type was applied.

The adjustable parameters for the GA used in the ordering problem were the number of strings in the population, the selective reproduction rate, the recombination rate, and the mutation rate. In this work acceptable results were obtained by keeping the searching parameters constant throughout the run, with the exception of the mutation rate which was allowed to vary with the number of generations. The parameters that were used are summarized in Table 6.1.

The population size of 100 strings seemed to work well for all of the problems employed, although a thorough optimization was not carried out. The selective reproduction rate of 50% was used to maximize the turnover of the population. Unlike the problem in Chapter 5, the search space in this case was noise free in that the objective function for a given string never changed. Therefore, there was no advantage in retaining any strings other than the ones used for reproduction, since all of those ranked lower would always remain lower. Thus, each new generation consisted of the best 50 in the previous generation plus the new the children. As already noted, the recombination rate was 50 % (*i.e.* 50 % of the *genes* or bits were affected), with some level of variability built into this value. Finally, it was observed that the GA seemed to perform better where the mutation (percentage of child strings undergoing one mutation) was increased as the algorithm proceeded, as indicated in the table. This was particularly true for larger numbers of samples. It is thought that recombination,

Table 6.1 Summary of the GA configuration used for the integer example.

| Parameter | Value (%) |
|---------------------------------------|-----------|
| Number of Strings (Population) | 100 |
| Selective Reproduction Rate | 50 |
| Recombination Rate (Exchange Rate) | 100 50 |
| Mutation Rate (Generation < 20) | 5 |
| Mutation Rate (Generation 21-30) | 30 |
| Mutation Rate (Generation 31-40) | 50 |
| Mutation Rate (Generation 41-50) | 70 |
| Mutation Rate (Generation > 50) | 100 |

at least under the conditions employed here, is not as effective for *fine tuning* of the order in the later stages and that population homogeneity may be a problem.

6.2 Experimental

In this section, the seven data sets used to test the ordering algorithm are discussed. These consisted of both simulated data sets (to control the conditions of the ordering) and experimental data sets (to evaluate performance on real systems).

6.2.1 Integer Sorting

The simplest data set used in this work consisted of a sequence of integers that was randomly arranged and then sorted with the GA using the double sum procedure described in section 6.1.3. This permitted an initial evaluation of the GA

without using the windowing approach for calculating the objective function described for chemical data sets. Complications arising from the definition of the objective function and boundaries were thus removed.

6.2.2 Cluster Analysis - Simulated Data

To evaluate the ordering algorithm for clustering applications, a data set was generated consisting of 30 samples in three groups of 10. The three groups were centered at the corners of an equilateral triangle in two-dimensional space corresponding to the two simulated measurements made on each sample were generated by adding normally distributed random values ($\mu = 0$, $\sigma = 0.25$) to the coordinates of the center of the cluster. The clusters were separated by 10σ .

6.2.3 Cluster Analysis - Experimental Data

The data set used in this study was previously described by White *et al* [105]. Briefly, a chromatographic procedure was used to measure the concentration of primary amino acids in the digestion of 60 mushroom samples representing six species. For the work presented here, a subset of this data set consisting of 30 samples and representing three species was employed. These three species were those most easily separated by traditional cluster analysis techniques. The feature space consisted of the responses for six selected amino acids (integrated signals) normalized by the total integrated signal for each chromatogram.

6.2.4 Simulated Chromatographic Data

As a simple example of an evolving data set, a three-dimensional chromatogram was simulated (absorbance vs. wavelength vs. time). The specifications for this data set were as follows. The three concentration profiles were Gaussian with $\sigma = 7.5$ s and a peak to peak separation of 10 s between components 1 and 2 and 12 s between components 2 and 3. The middle profile was centered at 35 seconds and all profiles were of equal height. The spectra were also generated as Gaussians, with $\sigma = 20$ nm and a peak to peak separation of 20 nm. Here the middle spectrum was centered at 250 nm and corresponded to the middle concentration profile. Again, all three spectra were of equal amplitude. Data for the spectrochromatogram were generated at intervals of 2 s between 0 and 70 s, and at intervals of 8 nm between 204 nm and 300 nm. Gaussian noise at a level of 0.01 % of the maximum signal was added to the final spectrochromatogram.

Of course, the spectrochromatogram generated in this manner is already ordered, so to test the ordering algorithm, the spectral slices were randomly ordered along the time axis. The ability of the GA to obtain the original order was then examined as a test of its performance for ordering evolving data sets.

6.2.5 The Clock Reaction

An oscillating chemical reaction that will be referred to as the clock reaction was used as an example of a time variant process that was only partially ordered.

The procedure used was adapted from Summerlin and Ealy [106] and was carried out in a spectrophotometric cell with a pathlength of 1 cm. Five drops of ferrion (phenanthroline ferrous sulfate) was added to the cell containing a solution of NaBr, NaBrO₃ and malonic acid (full details are given in [107]). Spectra were obtained at 2 nm intervals between 320 nm and 650 nm every 5 s for 480 s. Approximately five oscillations were observed in this period.

6.2.6 Pyrocatechol Violet

The acid-base equilibrium characteristics of pyrocatechol violet were employed in Chapter 2 to illustrate a data set where the ordinal variable is pH. The same equilibrium, which consists of four absorbing species, was used in the current study to simulate an evolving data set which has a totally disordered time sequence. In order to generate this disordered time sequence, the flow system shown in Figure 6.7 was used. Pump 1 was used to provide a variable flow of pyrocatechol violet (0.01 M) at a flow rate of ca 1 ml/min. Pumps 2 and 3 were used to control the pH and the combined flow rate was set at ca 2 ml/min. The composition of the buffer used was similar to the system described by Perrin and Dempsey [108] and consisted of Na₂HPO₄ (7.10 g), citric acid monohydrate (7 g), C3-cyclohexylamino-1-propane sulfonic acid (CAPS) (11.2 g), and (250 ml) 1 M NaOH diluted to 1 L with CO₂ free distilled water. A solution of 0.1 M HCl was added to this at a mixing tee to adjust the pH, and this mixture was then combined with the pyrocatechol violet stream. The combined flow rate of the buffer and HCl

streams was held constant. The ratio of flow rate of the buffer to acid and the flow rate of the pyrocatechol violet stream were determined by a computer generated random number sequence. The sequence values were drawn from a uniform distribution. Although the pH of the buffer-base mixture is not exactly linearly related to the ratio, the change is fairly gradual and monotonic. The pyrocatechol violet mixture was directed to the diode array spectrophotometer a 30 μ L flow cell (Hellma Cells, Jamaica, NY) and spectra were obtained on the flowing stream. The pH of the stream exiting the flow cell was measured with a flow through pH electrode (Fisher Model # 3D) with a cell constructed in the departmental machine shop.

The conditions used here were intended to roughly simulate a process stream in which there are four observable components in the process stream whose contributions change in accordance with some unknown process variable(s).

6.2.7 St. Louis Data.

The data used in this study were provided by Prof. P. K. Hopke of Clarkson University and were obtained as a part of an air pollution study in the St. Louis, MO area carried out during the months of July and August, 1976. The data consisted of concentrations of 27 elements in particulate matter gathered at one monitoring site (site 112), and were divided into four subgroups by particle size (< 2.4 μ m and 2.4 μ m - 20 μ m) and time of collection (a.m. and p.m.). For this

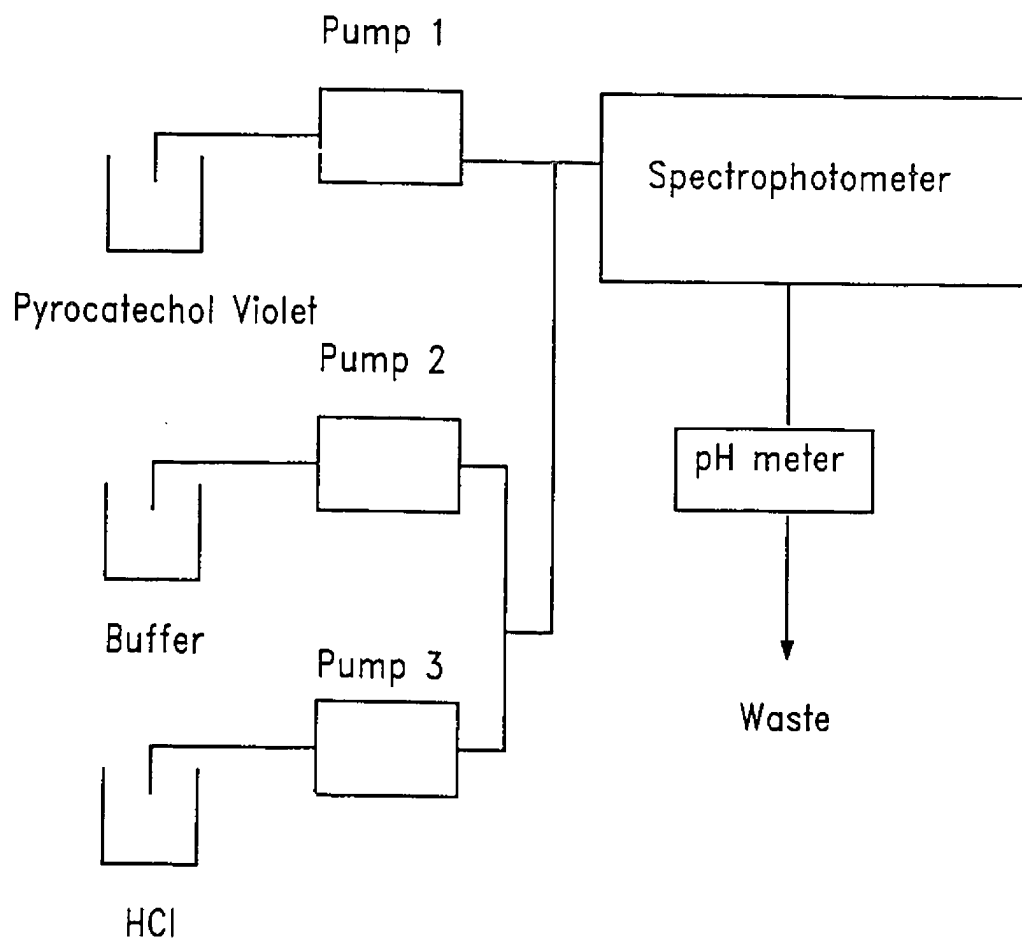


Figure 6.7 Flow system for pyrocatechol violet study.

study, the coarse fraction collected during the a.m. period was used. Elemental concentrations that were below the detection limits were excluded, leaving 17 elements (Al, Si, S, Cl, K, Ca, Ti, Mn, Fe, Ni, Cu, Zn, Se, Br, Sr, Ba, and Pb) and 56 samples. In order to account for the different concentration ranges, the data were autoscaled prior to analysis. Further details on this data set can be found in a paper by Hopke *et al* [109].

6.2.8 Computational Aspects

Calculations were performed on 486-based computers (486-DX, 486-DX2 and 486-DX4) under DOS 6.0 and Windows 3.1 (Microsoft, Redmond WA) with programs written in MATLAB 4.0 for Windows (The Math Works Inc., Natick, MA).

6.3 Results and Discussion

6.3.1 Integer Sorting

The integer sorting problem was selected as the initial test of the GA ordering algorithm for several reasons. First, complications arising from an inappropriate choice of objective function were avoided since the simple double-sum approach was known to work for this example. Also, the double-sum does not require any adjustment at the boundaries. Because of this, the problem was suitable as a test case for adjusting the GA parameters to near optimal values. Finally, the nature of the ordering problem is such that the number of generations required for convergence (assuming a starting point that is totally random) should

be independent of the objective function used. This is because the search space can be regarded as essentially *noise free*. Thus, the integer ordering problem serves as a means to evaluate the convergence ability as a function of the number of samples (of course, the significance of the final solution will depend on the objective function, but that is not the issue in this case).

Figure 6.8 shows a typical result for the integer ordering problem. In this case ordering of integers from 1 to 30 (*i.e.* $n = 30$) is shown. The objective function, *i.e.* double sum, decreased dramatically for about the first 30 generations and marginally improved after 50 generations. For this particular run, the GA converged in 64 generations (*i.e.* the integers were ordered from 1 to 30). The mean number of generations required for convergence as a function of n is shown in Figure 6.9 a. The results are the averages of 3 runs and the error bars shown represent standard deviations. These results show that the number of generations required for convergence is nonlinear in n . Figure 6.9 tries to show that although the search space increases quickly (*e.g.* logarithmic as shown in Figure 6.9 b) the number of generations does not. For the present problem each generation corresponds to approximately 7 seconds on a 100 MHz 486 computer. The actual times change somewhat with the objective function and the dimensionality of the problem.

6.3.2 Cluster Analysis - Simulated Data

The simulated cluster analysis data set was used as a test case for the

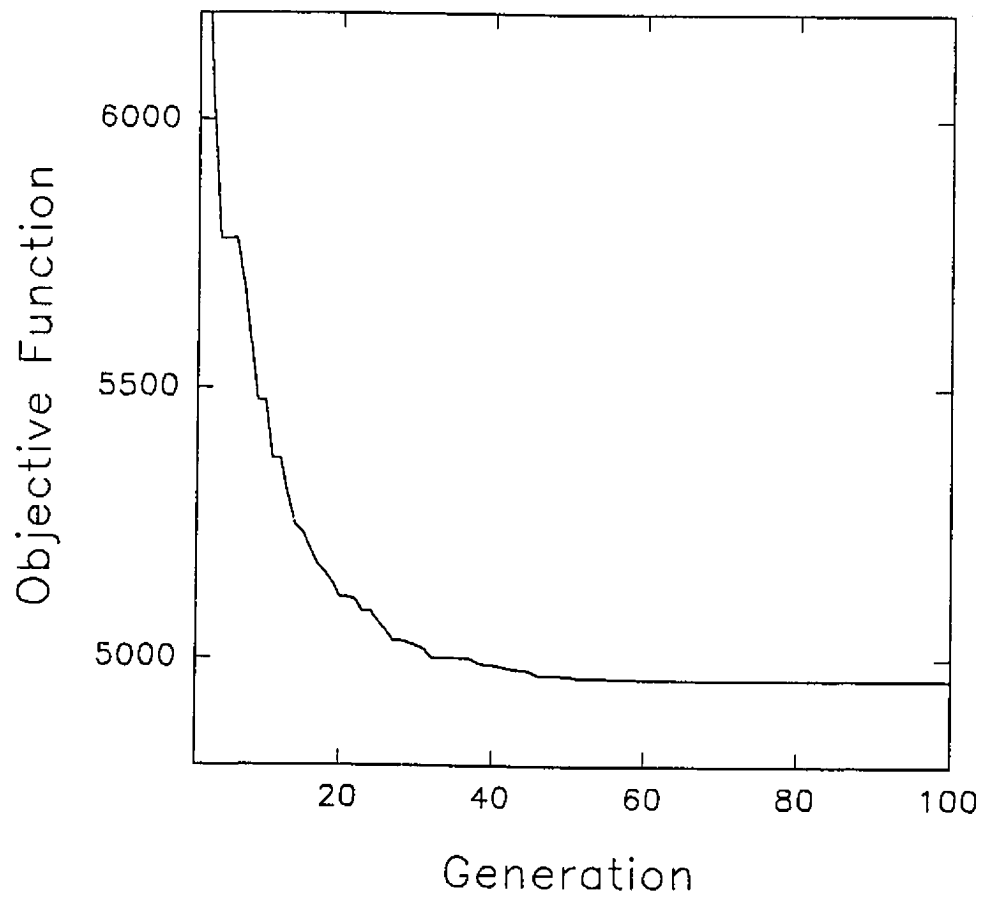


Figure 6.8 A typical run for the integer example.

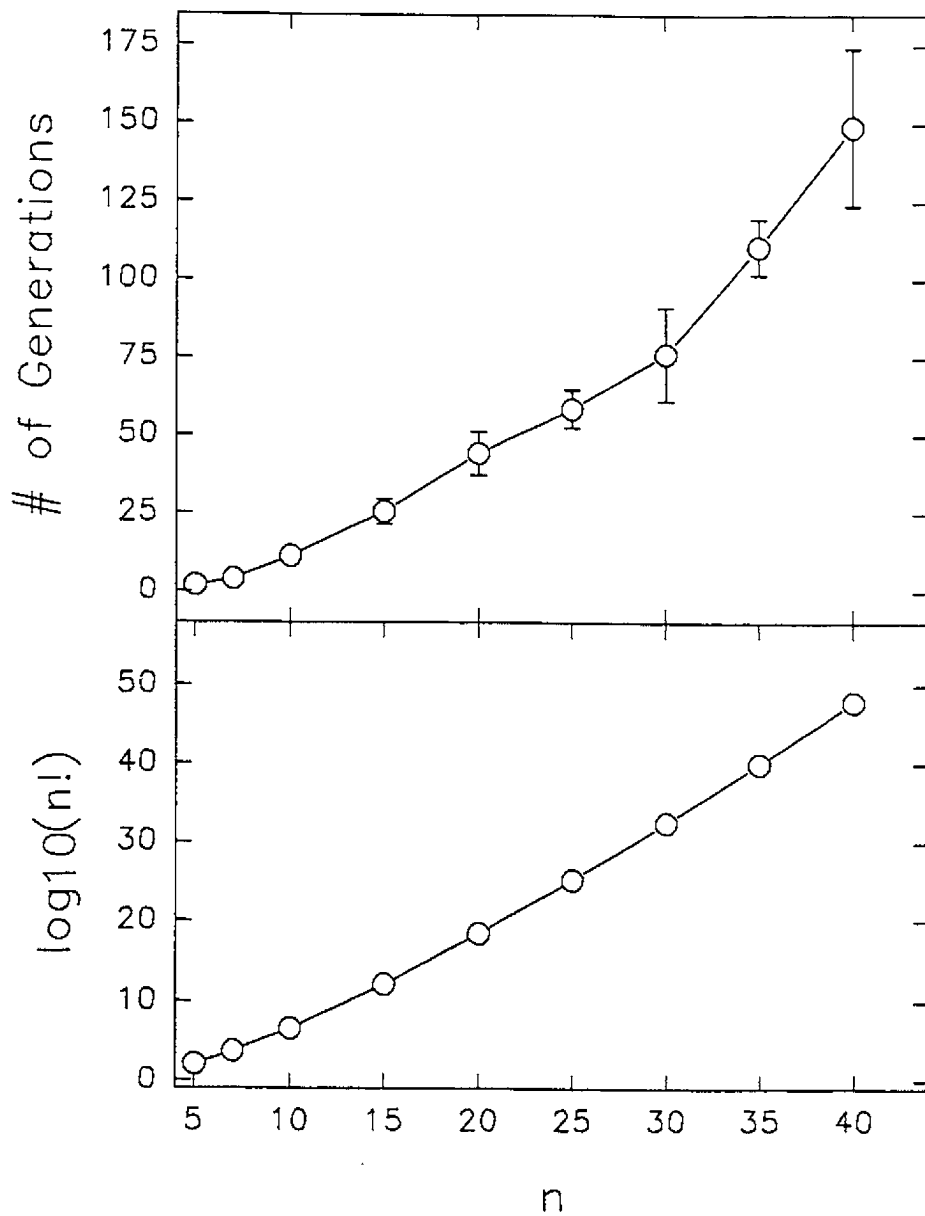


Figure 6.9 a) Mean number of generations required for convergence of the GA as a function of the integer number. b) The common logarithm of the number of permutations for different n.

sliding window objective function. Since the data in this case were symmetrically arranged around a central point, the data set was considered to be closed and the sliding window was fixed at a size of 9 samples. Although such closure is not required in the general case, in this instance it permitted the more elegant approach of a circular boundary to be used.

Figure 6.10 shows the convergence of the GA for the simulated set. It will be noted that correct clustering of the sample (*i.e.* into groups 1, 2 and 3) is obtained after about 37 generations, but more generations are needed to order the samples within each group. The clustering shown in Figure 6.10, *i.e.* 33333331111111112222222222333 is equivalent to 111111111122222222223333333333 because of the nature of the circular approach to accommodate boundaries.

The GA will group samples that are most similar to each other even if they are not in the same class. This can be illustrated through the use of a connective diagram in which lines are drawn between the ordered samples in feature space, indicating the sequence determined by the GA. Such a diagram is shown in Figure 6.11 for the simulated set used here. Note that the sequence chosen minimizes the total distance connecting all of the samples. Of special interest is how the GA organizes the samples that bridge the classes. For example in this case three pairs of samples bridge the three classes: 8 & 14, 20 & 29 and 4 & 30. While the GA correctly ordered the samples in the data set, the resulting sequence does not provide a direct indication of the number of clusters present or where

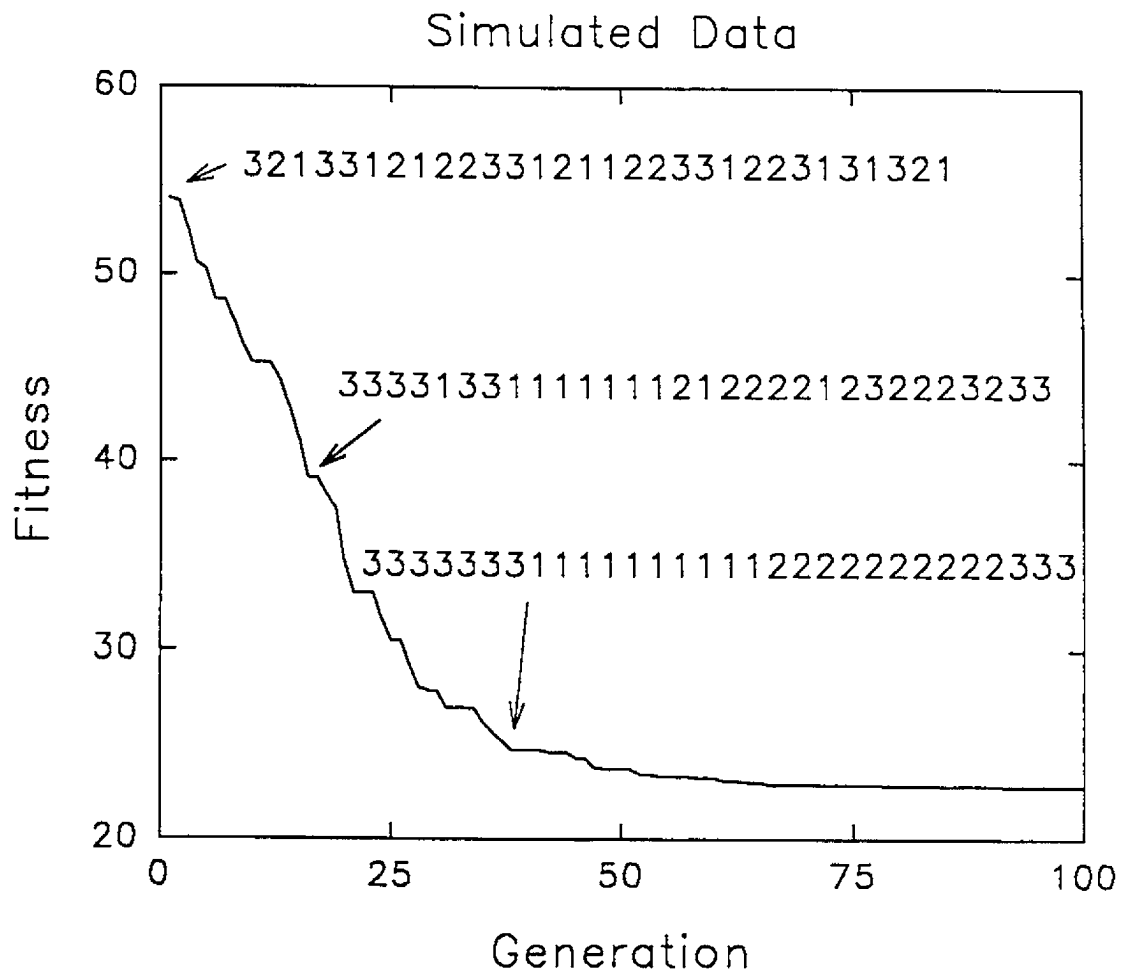


Figure 6.10 GA results for the simulated cluster analysis data set.

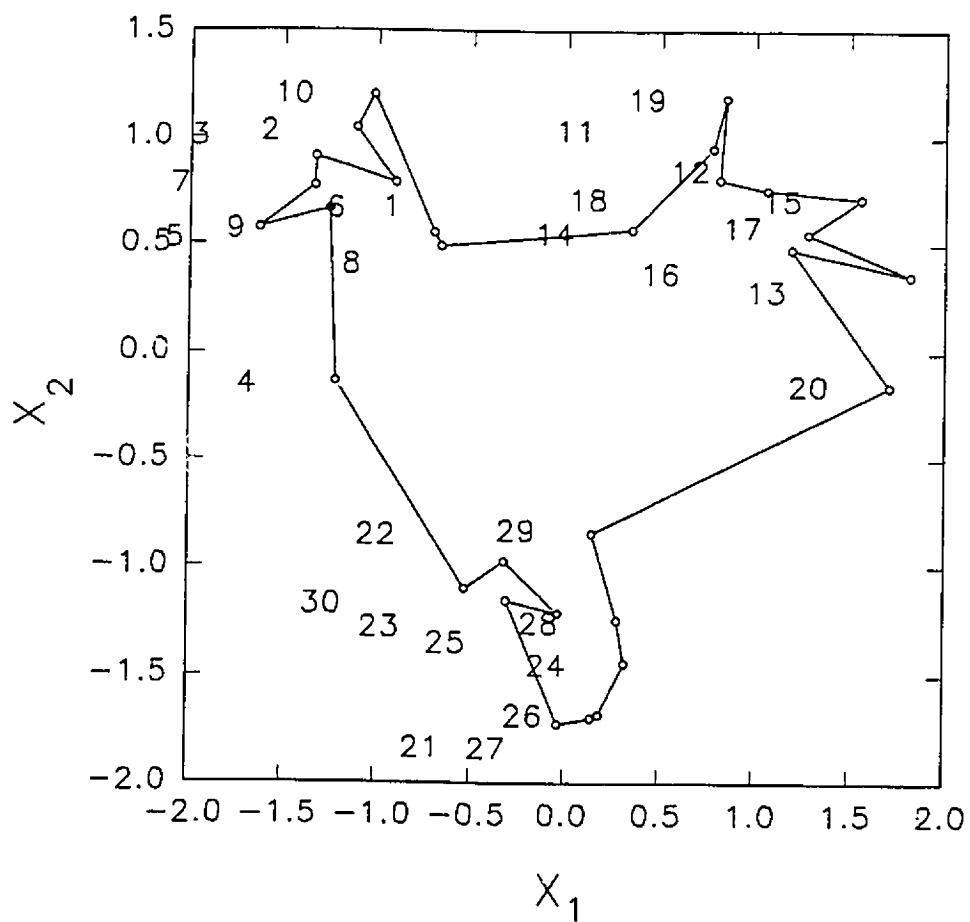


Figure 6.11 Connection diagram for the simulated cluster analysis data set.

their boundaries lay. This information can be obtained by a number of methods, however. One of these is a plot of the window variance as a function of the location of the center of the window within the ordered sequence. Such a plot was shown in Figure 6.5b. Note that the variance function increases significantly at the boundaries of the clusters and that there are 3 such maxima indicating the presence of three clusters.

6.3.3 Cluster Analysis - Experimental Data

To illustrate the application of GA ordering to the cluster analysis of experimental data, the mushroom data set described in section 6.2.3 was used. The classification of these mushroom samples was previously investigated by White *et al* [105]. In their study three species were able to be spatially discriminated using only two principal component vectors extracted from a six dimensional space. In this work the same six experimentally measured features (*i.e.* normalized amino acid responses) were used as the sequencing variables. As for the simulation, the data set was considered to be closed. The samples are plotted in the space of the first two principal components in Figure 6.12. Samples 1 to 10 belong to the first species (*Amanita flavoconia*), samples 11 to 20 belong to the second species (*Amanita gemmata*), and samples 21 to 30 belong to the third species (*Amanita virosa*). The sequence as determined by the GA is shown in Figure 6.12b in the form of a connection diagram in the principal component space. Note that all of the samples are grouped in the appropriate cluster.

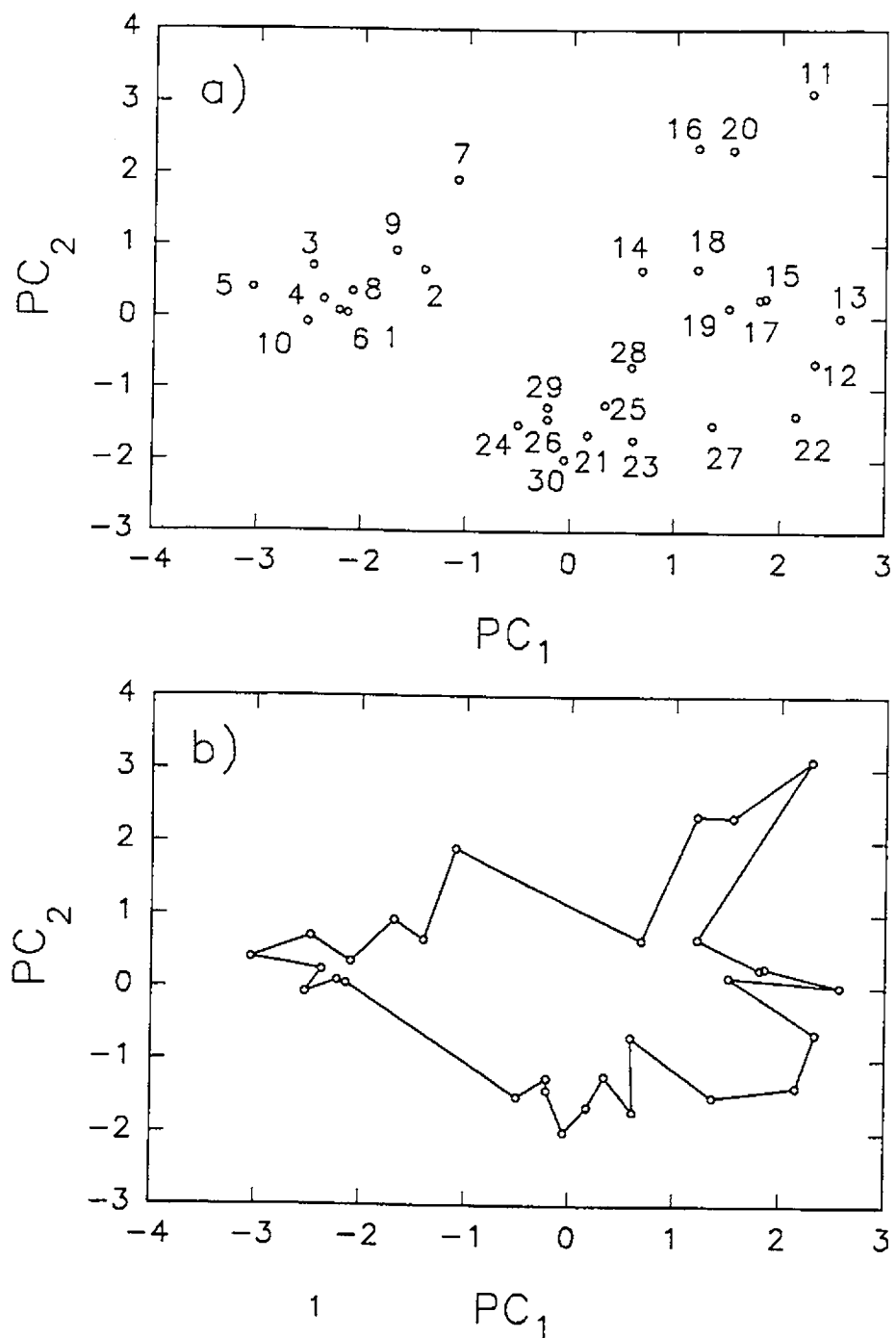


Figure 6.12 a) Classification of mushroom data using two principal components. b) Connection diagram for the mushroom data.

In this example, the clustering provided by the GA proved to be better than that obtained with other exploratory data analysis techniques, such as PCA or hierarchical clustering (dendrograms). PCA is restricted to low dimensional mapping, which may obscure spatial relationships, and hierarchical clustering (because of its hierarchical nature) may suggest incorrect associations. This latter point is illustrated for the mushroom data set by the dendrogram in Figure 6.13 obtained using the complete link method. It can be seen that the dendrogram misclassified some samples, while the GA was able to provide the correct classification for all samples. This suggests that the GA may be a better classifier than the HCA method in some methods.

6.3.4 Simulated Chromatographic Data

The preceding two examples show how GA sequencing can be applied to clustering problems. The remainder of this chapter will focus on the second type of problem, evolving data sets. In this section the simulated chromatographic data set described in section 6.2.4 was treated with the sequencing algorithm. Of course, chromatographic data are naturally ordered in time, so the spectra obtained at each time interval were randomly ordered prior to initiation of the algorithm. By using simulated data, factors such as the noise level and degree of spectral and chromatographic resolution could be controlled, and the success of the ordering algorithm could be readily accessed. In this case, a closed data set was once again assumed and the circular boundary was used. This was possible

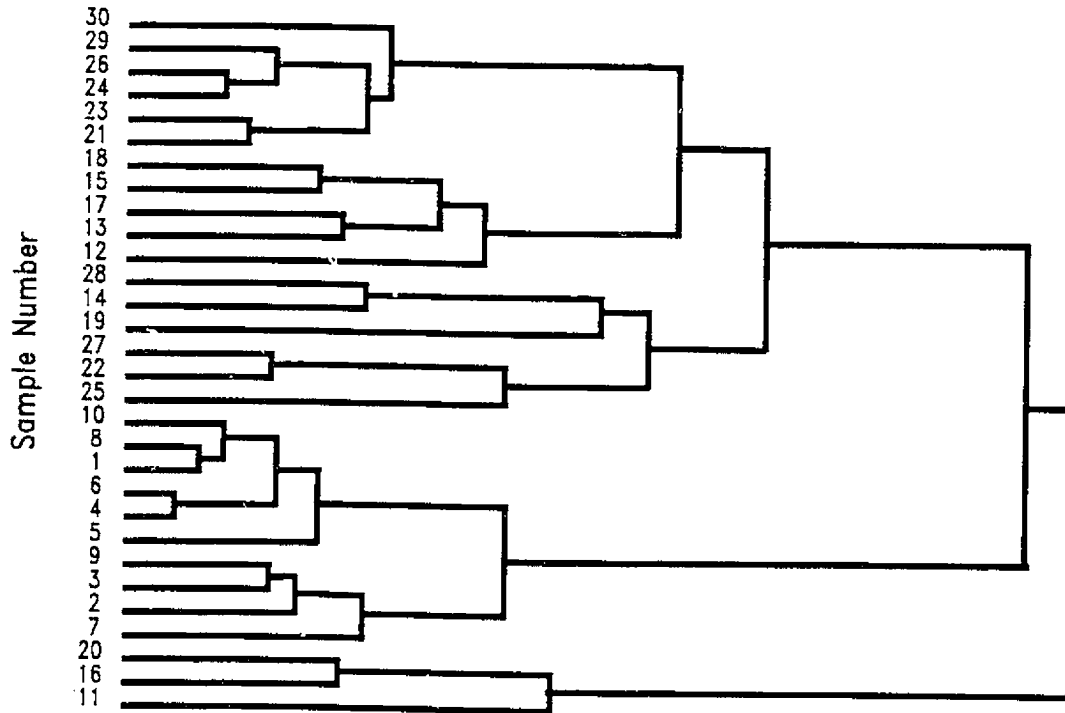


Figure 6.13 Dendrogram of the mushroom data. The sample numbers are the correspond to the numbers in Figure 6.12.

here since the baseline regions permitted continuity to be maintained between the first and last components. A window size of 12 was used in treating these data.

Figure 6.14a shows the concentrations profiles for the three components in the simulated mixture and 6.14b shows the ordering of component concentrations determined by the GA. It is clear that the GA was able to correctly order the data set, although the beginning and end of the sequence were arbitrarily defined.

6.3.5 The Clock Reaction

A further test of the GA method employed an experimental data set where the samples were not entirely ordered by time. This set consisted of spectra obtained over several cycles of the iodine clock reaction described in section 6.2.5. This data set was a good choice for initial tests because it consisted of only two observable components that were spectrally different. Since there were only two components, it could be treated as a open data set. As shown in Figure 6.15, the set is not entirely ordered by time and there appear to be almost five complete cycles. The only difference in the treatment of this set from the chromatographic data set is that a window of length 7 was used. The results of ordering for the clock reaction are shown in Figure 6.16. Figure 6.16a shows the absorbance measurements in the original order at one wavelength (444 nm). The bottom panel shows the absorbance at the same wavelength obtained after GA sequencing. For this example, the smooth transition of absorbance values demonstrates that the GA has arranged the data in a consistent fashion, with

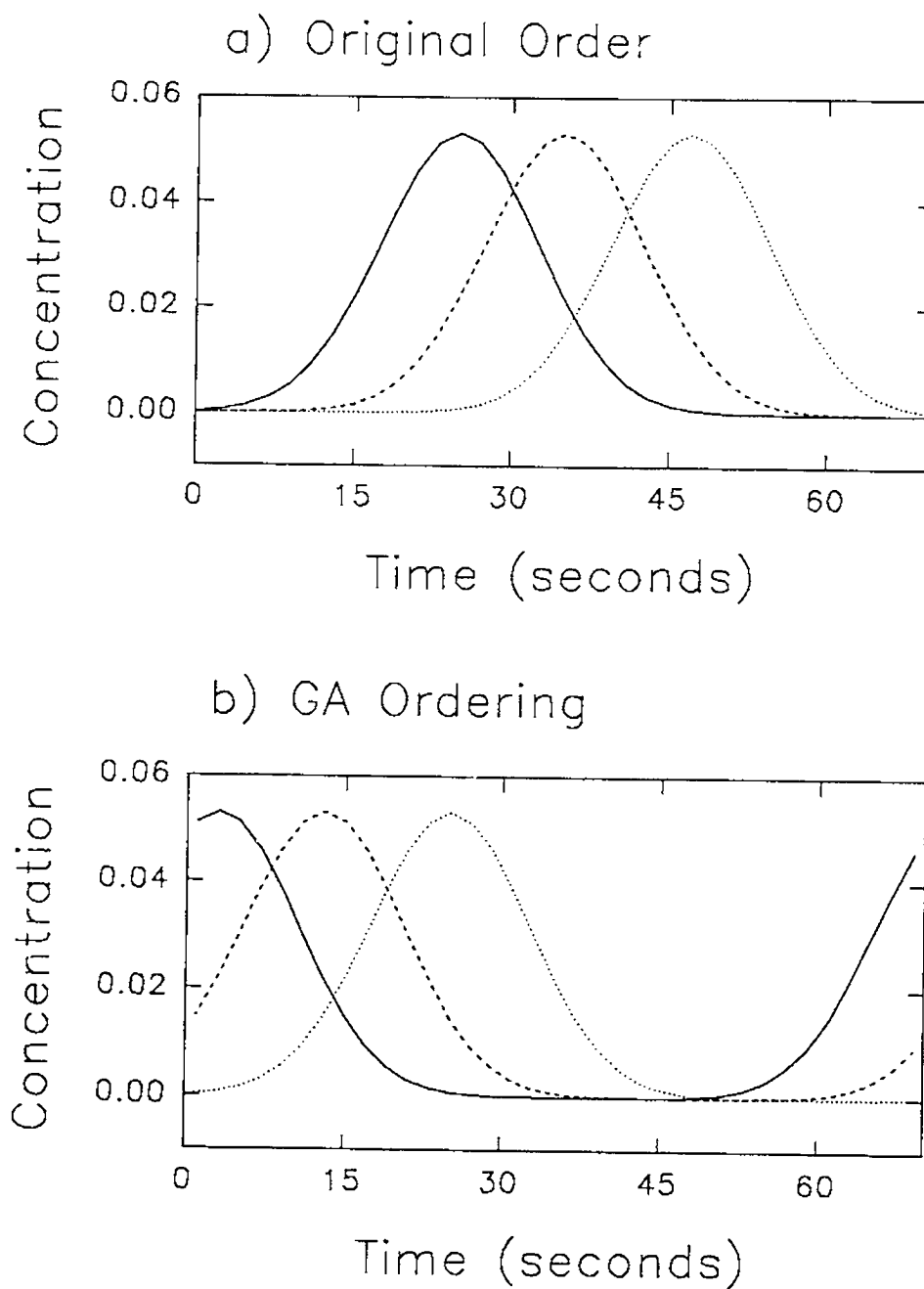


Figure 6.14 Results of GA ordering of the simulated chromatographic data: **a)** original concentration profiles (before randomization), **b)** concentration profiles of randomized chromatograms determined by the sequencing algorithm.

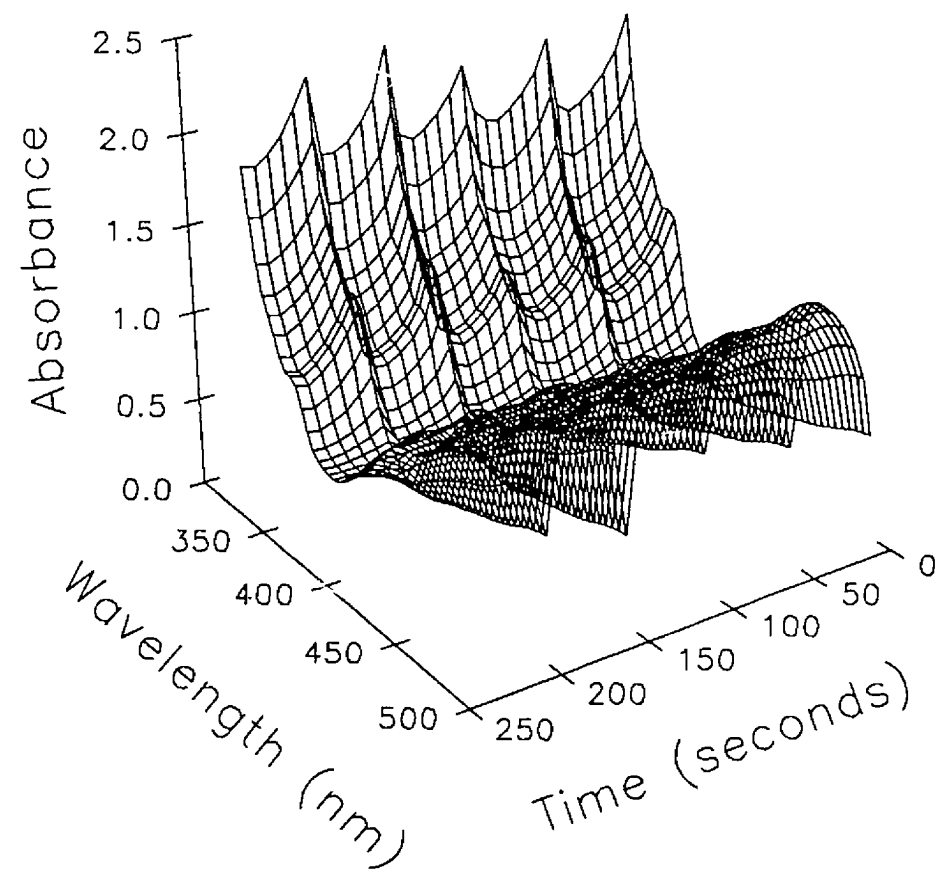


Figure 6.15 Spectra obtained from the clock reaction as a function of time.

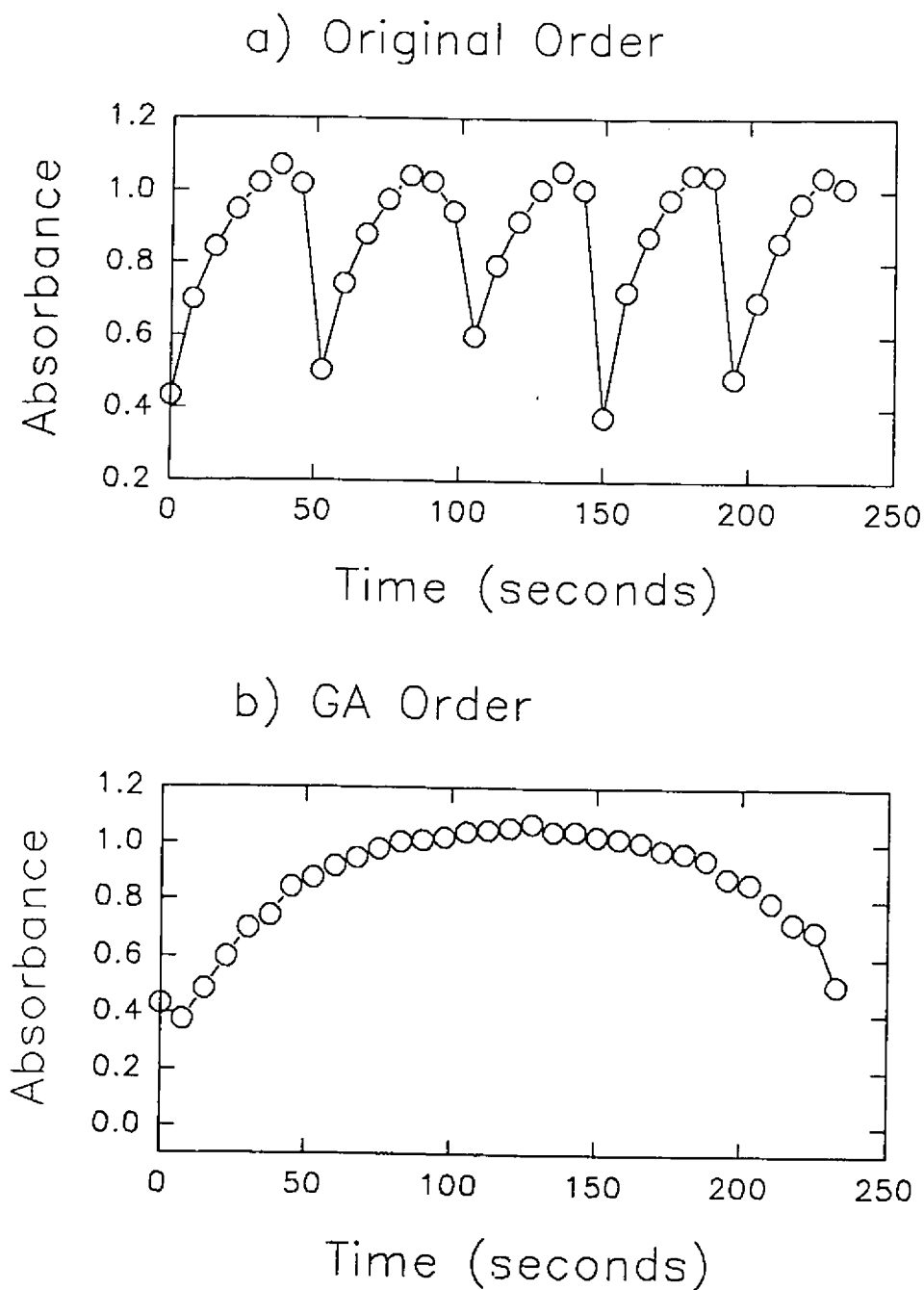


Figure 6.16 Results of GA sequencing of clock reaction as illustrated by the absorbance at 444 nm: **a)** original sequence, **b)** ordered sequence.

composition of one sample merging smoothly into the next. However, a truly correct ordering would result in a monotonic change in absorbance values. The results obtained here are a consequence of the circular boundary, and indicates weakness of this approach for a data set which is not truly closed.

6.3.6 Pyrocatechol Violet

The clock reaction described in the last section consisted of only two components and was already partly ordered. The pyrocatechol data set served as a more rigorous test of the GA applied to experimental data since it consisted of four components and was randomly sequenced by the nature of the experiment. The nature of this data set necessitated some changes in the treatment. Unlike the other cases discussed so far, this case was treated as an open data set with a variable sized window at the boundaries. The full window size was 12 samples. In order to account for variations in the magnitude of the spectra due to changes in the dye concentration, spectra within each window were normalized to unit area. Finally, it was shown that better results were obtained when the objective function was calculated using the standard deviation rather than the variance (i.e. equation 6.4).

The original sequence of spectra obtained with the randomized flow system is shown in Figure 6.17. In this case, the ordering ability of the GA can be accessed by examining the correlation of the sequence number with the pH, since that is the underlying ordinal variable in this case. In Figure 6.18, pH is plotted as

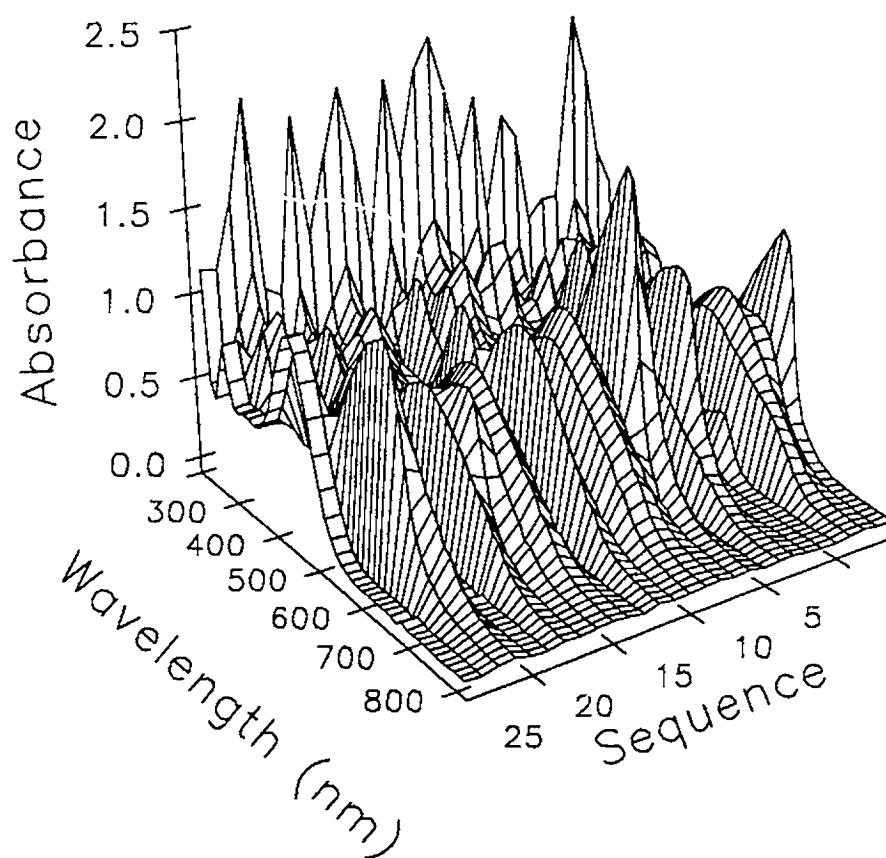


Figure 6.17 Spectra obtained from the randomized pyrocatechol violet experimental data.

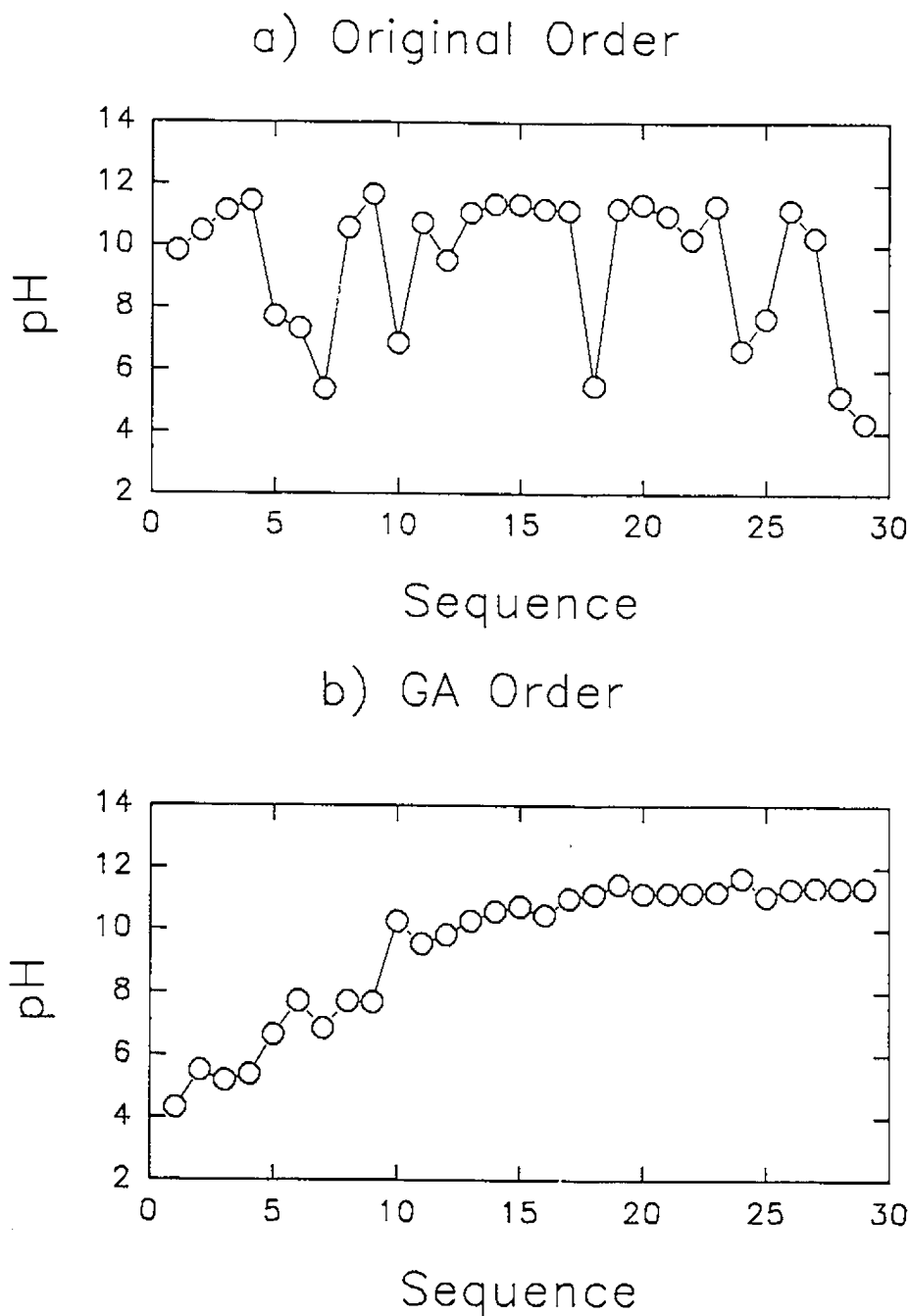


Figure 6.18 Results of GA ordering of the pyrocatechol violet data set illustrated via the corresponding pH values: **a)** original sequence, **b)** ordered sequence.

a function of sequence for the original data set (6.18a) and the GA ordered data set (6.18b). Although the series of pH values is not perfectly monotonic, the progression showed a definite positive correlation. Improperly ordered samples are likely the result of variations in the dye concentration. Low concentrations, upon scaling, will have inflated errors relative to the higher concentrations and thus may lead to an inflated estimate of the variance within the window. An alternative objective function which accounts for this problem may yield better results.

6.3.7 St. Louis Data Set

For all of the evolving data sets presented so far, an underlying ordering variable was already known. It remains to be shown that the ordering process can provide information that was not already available. In order to demonstrate this, a suitable data set must meet two requirements. First, it must be truly disordered and therefore represent a real problem and, second, there must be a way of verifying that the order determined by the GA is somehow meaningful. To this end, the St. Louis data set was chosen. This data is an example of data that is used in receptor modelling [110]. The original elemental profiles for this data set, ordered by day, are shown in Figure 6.19. As can be seen from the figure, there is no obvious order to these data. It is also apparent that the ranges of elemental concentrations differ significantly. For this reason, each column of elemental concentration was variance scaled prior to ordering to reduce the influence of values with a large magnitude. Since errors were generally of the proportional

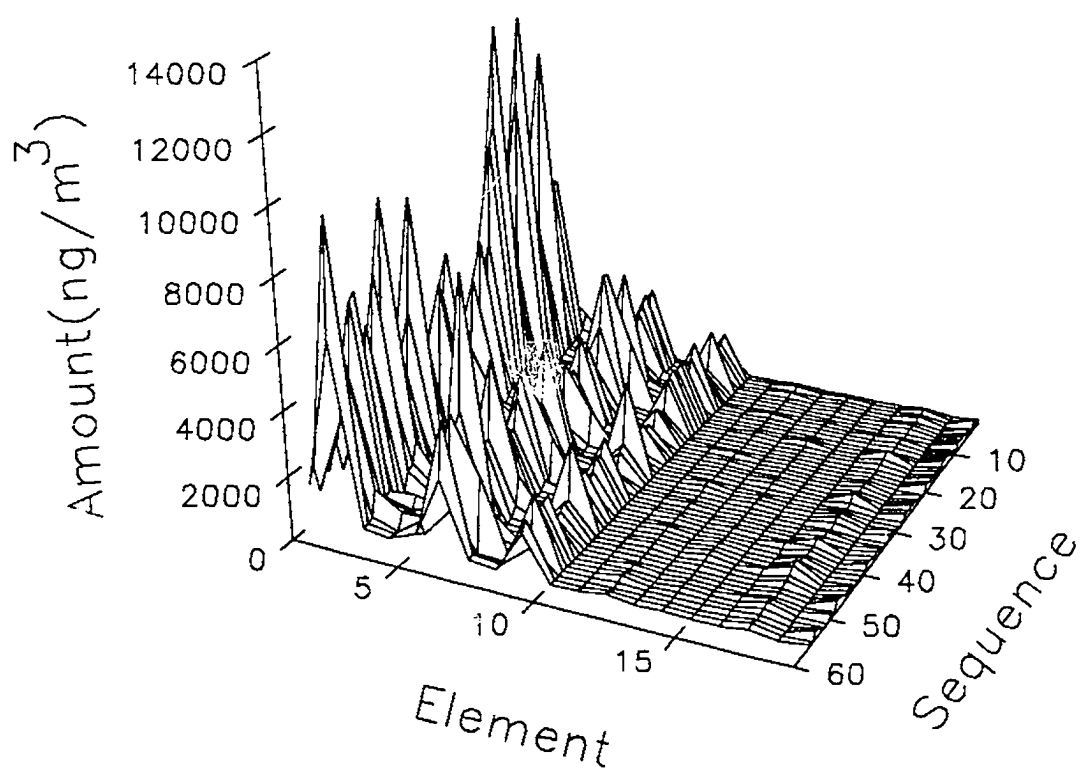


Figure 6.19 Elemental profiles at the receptor site as a function of time. Elements (in order) are Al, Si, S, Cl, K, Ca, Ti, Mn, Fe, Ni, Cu, Zn, Se, Br, Sr, Ba, Pb.

type, this type of scaling was not regarded as particularly detrimental. Also, as in the previous case, the profiles were scaled to unit area within each window. The data set was treated as closed with a window size of 12.

Following ordering by the GA, a corresponding ordinal variable was sought by examining correlations with a number of auxiliary variables recorded with the data set. When the ordering was tested against variables such as day of the week or wind speed, no systematic relationship was observed. With wind direction, however, a significant correlation was observed with the sequence number. This is illustrated in Figure 6.20. Although the relationship is not perfect (as expected, since wind direction can vary over a twelve hour period), these results serve as independent confirmation of a meaningful sequencing. Furthermore, the correlation could have been anticipated, since the contribution of particular sources to the overall receptor profile should be related to the direction of the wind carrying the pollutants. Such information could prove very valuable in identifying the sources of particular emissions and characterizing them in the absence of known source profiles.

6.4 Conclusions

In this chapter, the ability of the GA to order disordered data sets from a variety of sources has been demonstrated. These data sets differed in terms of their origin, the nature of variables used, the number of variables, the number of samples and the type of ordinal variable. The sets are summarized in Table 6.2.

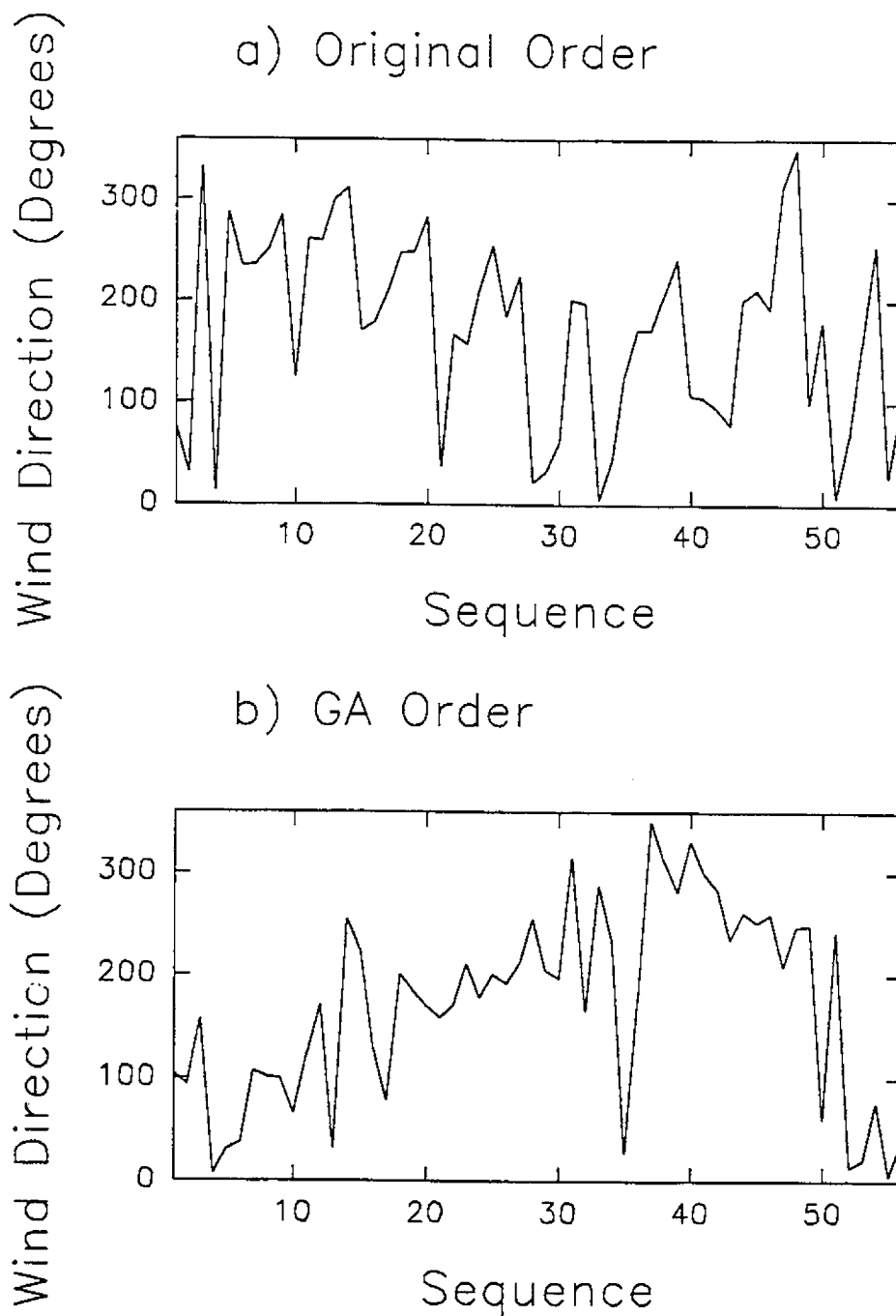


Figure 6.20 Results of GA ordering of the St. Louis data set illustrated via the corresponding wind direction values: **a)** original sequence, **b)** ordered sequence.

Table 6.2 The six sets analyzed by the GA sequencing method.

| | Data Set | Number of Samples | Number of Variables |
|----|---------------------------------------|-------------------|---------------------|
| 1. | Cluster Analysis Simulated Data | 30 | 2 |
| 2. | Cluster Analysis Experimental Data | 30 | 6 |
| 3. | Simulated Chromatographic Data | 36 | 13 |
| 4. | The Clock Reaction | 33 | 43 |
| 5. | Pyrocatechol Violet | 29 | 50 |
| 6. | St. Louis Data | 56 | 17 |

The algorithm described in the introduction was found to be generally applicable to all of the problems discussed here with minor changes. For convenience, some of the parameters associated with the GA were summarized in Table 6.1. Additional details for each problem are given in Table 6.3. Although the performance of the GA was not optimized (*i.e.* window size, window mode, scaling and the type of objective function) in every case, acceptable convergence was obtained for all problems. Further work needs to be done to refine the objective function so that it can better accommodate variables of different scale and compensate for measurement errors.

Table 6.3 Summary of windowing GA parameters for the six sets considered in Table 6.2.

| Data Set | Window Size | Scaling | Obj. Function | Window Mode | # of gen. to converge |
|----------|-------------|--------------------|---------------|-------------|-----------------------|
| 1 | 9 | Autoscaled | Eq. 6.3 | Closed | 40 |
| 2 | 9 | Autoscaled | Eq. 6.3 | Closed | 125 |
| 3 | 12 | Scale To Unit Area | Eq. 6.3 | Closed | 150 |
| 4 | 7 | Scale To Unit Area | Eq. 6.3 | Closed | 40 |
| 5 | 12 | Scale To Unit Area | Eq. 6.4 | Open | 50 |
| 6 | 12 | Scale To Unit Area | Eq. 6.3 | Closed | 250 |

For the clustering problems presented here, the results of GA ordering appear to be quite useful in providing more information than traditional exploratory data analysis methods. Further work needs to be done to examine the utility of this method for dealing with more difficult clustering problems. For most of the evolving data sets examined, the underlying ordinal variable was already known, but for the last data set, it was shown that uniquely useful information could be provided by the sequencing process. Traditionally, this method of treating disordered data sets has not been used, so it remains to be seen whether or not wider application will bring more useful results. For example, the use of evolving factor analysis methods in conjunction with GA ordering could prove to be especially fruitful.

7 Conclusions

Data can be acquired from many different types of instruments. Even though this data might be collected in a specific format one is not limited to analyzing data in that format. This work demonstrated that many types of data formats can be analyzed in the form of a matrix (See Eq. 1.1).

In Chapter 2, a method for determining the number of components in an unresolved chromatographic peak was developed. A technique called Evolving Projection Analysis (EPA) was originally developed for systems of one- and two-components. This present work extended the EPA method to systems of three- and four-components. The algorithm was successfully tested on a simulated system (a four-component system) and three experimental system (a three-component chromatographic system, a four-component chromatographic system, and a four-component spectrophotometric titration). This method is easily able to discern how many components are present in a sample. The main advantages of EPA are that it is able to locate where components appear and disappear and little or no prior information is needed about the system. This analysis focussed on measurements from a diode array spectrometer, which was also used in Chapter 3.

In Chapter 3, determining the number of components in a system can be complicated if the detector is not ideal. Although the analysis focussed on a absorbance measurements from a diode array detector this treatment can be

extended to other detectors. The two problems investigated were nonlinear instrument response and heteroscedastic noise in instrument response. The present discussion focussed on a one-component system but the treatment can be extended to higher order systems if needed. The analysis showed that with modifications to the EPA method, one can correctly identify the purity of a sample. However, during the analysis it was found that for high absorbance measurements ($ca > 2$), where stray light becomes important, the method fails.

Analysis in Chapters 2 and 3 used data from a diode array spectrometer, which commonly gives a series of measurements per sample. In contrast Chapter 4 looks at measurements from instruments that give one measurement per sample. In this chapter, the main analysis was the conductivity prediction. The problem is that the present recommended model, the Rossum model, is not able to deal with all samples. The validity of the Rossum model is questionable and a variety of methods that can be classed as empirical and semi-empirical were thoroughly investigated. The chapter concluded by recommending a method that modifies the Rossum model.

Using the same data in Chapter 4, Chapter 5 also looked at trying to find the best conductivity prediction model. Only the ten most abundant ions were considered. Equation 5.3 showed that 40 terms were considered. It is not possible to evaluate all combinations since this would require evaluation of $ca 10^{12}$ models, which would take around 4000 thousand years. To find the best model, one needs an intelligent search strategy. Genetic algorithms were used to find this

model. GA's were helpful in the data analysis in Chapter 6.

In this chapter the following question was considered: How can disordered samples be ordered? The motivation here is that placing samples in order may give new information about the system. The main problem was finding a function that ordered samples for different systems. The most successful of the methods tried was called the variance method. Two areas of this work could be more carefully investigated. The first area is that other ordering functions could be considered and secondly that GA conditions could be more fully optimized. However this chapter clearly showed how ordering can be accomplished.

This work has looked at a number of types of analysis. Although the analysis from chapter to chapter may seem unrelated, a number of scenarios are possible where one may want to analyze data using techniques from several chapters. For example, consider the collection of samples from a process stream. Sometimes these samples contain an analyte that is not ordered in time, but by some other variable. One could use the technique outlined in Chapter 6 to order the samples, and monitor the analyte's presence/absence using techniques from Chapter 2 and 3. This is just one example how these techniques are related.

REFERENCES

1. Massart, D. L.; Vandeginste, B. G. M.; Deming, S. N.; Michotte, Y.; Kautman, L. *Chemometrics: a textbook*, Elsevier: New York, 1988; pp 5-9.
2. Sharaf, M. A.; Illman, D. L.; Kowalski, B. R. *Chemometrics; Chemical Analysis Series 82*; Wiley-Interscience: New York, 1986.
3. *Practical Guide to Chemometrics*; Haswell, S. J., Ed.; Dekker: New York, 1992.
4. Brown, S. D.; Bear, R. S. Jr.; Blank, T. B. *Anal. Chem.* **1992**, *64*, 22R-49R.
5. Heise, H. M.; Marbach, R.; Janatsch, G.; Kruse-Jarres, J. D. *Anal. Chem.* **1989**, *61*, 2009-2015.
6. Ward, K. J.; Haaland, D. M.; Robinson, M. R.; Eaton, R. P. *Appl. Spectrosc.* **1992**, *46*, 959-965.
7. Bhandre, P.; Mendelson, Y.; Peura, R. A.; Janatsch, G.; Kruse-Jarres, J. D.; Marbach, R.; Heise, H. M. *Appl. Spectrosc.* **1992**, *47*, 1214-1221.
8. Robinson, M. R.; Eaton, R. P.; Haaland, D. M.; Koepp, G. W.; Thomas, E. V.; Stalland, B. R.; Robinson, P. L. *Clin. Chem.* **1992**, *38*, 1618-1622.
9. Heise, H. M.; Marbach, R.; Koschinsky, T. H.; Gries, F. A. *Appl. Spectrosc.* **1994**, *48*, 85-95.
10. Haaland, D. M.; Robsinson, M. R.; Koepp, G. W.; Thomas, E. V.; Eaton, R. P. *Appl. Spectrosc.* **1992**, *46*, 1575-1578.
11. Arnold, M. A.; Small, G. W. *Anal. Chem.* **1990**, *62*, 1457-1464.
12. Marquardt, L. A.; Arnold, M. A.; Small, G. W. *Anal. Chem.* **1993**, *65*, 3271-3278.
13. Small, G. W.; Arnold, M. A.; Marquardt, L. A.; *Anal. Chem.* **1993**, *65*, 3279-3289.
14. Bhandre, P.; Mendelson, Y.; Stohr, E.; Peura, R. A.; *Appl. Spectrosc.* **1994**, *48*, 271-273.

15. Rosenthal, D. *Anal. Chem.* **1982**, *54*, 63-66.
16. Davis, J. M.; Giddings, J. C. *Anal. Chem.* **1983**, *55*, 418-424.
17. Lawton, W.H.; Sylvestre, E.A. *Technometrics* **1971**, *13*, 617-633.
18. Sharaf, M. A.; Kowalski, B.R. *Anal. Chem.* **1982**, *54*, 1291-1296.
19. Osten, D. W.; Kowalski, B.R. *Anal. Chem.* **1984**, *56*, 991-995.
20. Ohta, N *Anal. Chem.* **1973**, *45*, 553-557.
21. Chen, J.; Hwang, L. *Anal. Chim. Acta.* **1981**, *133*, 271-281.
22. Borgen, O. S.; Kowalski, B. R. *Anal. Chim. Acta.* **1985**, *174*, 1-26.
23. Gemperline, P.J. *Anal. Chem.* **1986**, *58*, 2656-2663.
24. Gerritsen, M. J. P.; Tanis, H.; Vandeginste, B. G. M.; Kateman, G. *Anal. Chem.* **1992**, *64*, 2042-2056.
25. Vandeginste, B G. M.; Derks, W.; Kateman, G. *Anal. Chim. Acta*, **1985**, *173*, 253-26.
26. Gampp, H.; Maeder, M.; Meyer, C. J.; Zuberbühler, A. D. *Talanta*, **1985**, *32*, 1133-1139.
27. Gampp, H.; Maeder, M.; Meyer, C. J.; Zuberbühler, A. D. *Talanta*, **1986**, *33*, 943-951.
28. Maeder, M. *Anal. Chem.* **1987**, *59*, 527-530.
29. Gampp, H.; Maeder, M.; Meyer, C. J.; Zuberbühler, A. D. *Anal. Chim. Acta*, **1987**, *193*, 287-293.
30. Maeder, M.; Zillian, A. *Chemom. Intell. Lab. Syst.* **1988**, *3*, 205-213.
31. Gemperline, P. J.; Hamilton, J. C. *J. Chemom.* **1989**, *3*, 455-461.
32. Malinowski, E. R. In *Computer-Enhanced Analytical Spectroscopy*, Vol. 1; Mleuzelaar, H. L. C.; Isenhour, T. L. Eds.; Plenum Press: New York, 1987; pp. 55-102.
33. Keller, H R.; Massart, D. L. *Anal. Chim. Acta.* **1991**, *246*, 379-390.

34. Keller, H.R.; Massart, D.L.; De Beer J.O. *Anal. Chem.* **1993**, *65*, 471-475.
35. Schostack, K. J.; Malinowski, E. R.; *Chemom. Intell. Lab. Syst.*, **1993**, *20*, 173-182.
36. Den, W.; Malinowski, E. R.; *J. Chemom.* **1993**, *7*, 89-98.
37. Kvalheim, O. M.; Liang, Y. Z. *Anal. Chem.* **1992**, *64*, 936-946.
38. Liang, Y. Z.; Kvalheim, O.M.; Rahmani, A; Brereton, R.G. *J. Chemom.* **1993**, *7*, 15-43.
39. Liang, Y. Z.; Hämäläinen, M. D.; Kvalheim, O. M.; Anderson, R. *J. Chromatogr.*, **1994**, *662*, 113-122.
40. Keller, H. R.; Massart, D. L.; Liang, Y. Z.; Kvalhiem, O. M. *Anal. Chim. Acta.* **1992**, *267*, 63-71.
41. Vanslyke, S.J.; Wentzell, P.D. *Anal. Chem.* **1991**, *63*, 2512-2519.
42. Kalman, R.E. *Journal of Basic Engineering* **1960**, *82D*, 35-45.
43. Brown, S.D. *Anal. Chim. Acta* **1986**, *181*, 1-26.
44. Rutan, S.C. *J. Chemom.* **1987**, *1*, 7-18.
45. Cooper, W. S. *Rev. Sci. Instrum.* **1986**, *57*, 2862-2869.
46. Tranter, R. L.; *Anal. Proc.* **1990**, *27*, 134-136.
47. Wentzell, P.D.; Wade, A.P.; Crouch, S.R. *Anal. Chem.* **1988**, *60*, 905-911.
48. Vanslyke, S.J.; Wentzell, P.D. *Chemon. Intell. Lab. Syst.* **1993**, *20*, 183-195.
49. Wentzell, P.D.; Hughes, S.G.; Vanslyke, S. J. *Anal. Chim. Acta* **1995**, *307*, 459-470.
50. Kennedy, J. H. *Analytical Chemistry: Principles 2nd ed.*; Saunders, New York 1990; p. 309.
51. Keller, H.R.; Massart, D.L. *Anal. Chim. Acta.*, **1992**, *263*, 21-28.
52. Keller, H.R.; Massart, D.L.; Kiechle, P.; Erni, F. *Anal. Chim. Acta.*, **1992**, *256*, 125-131.

53. Skoog, D.A.; West, D.M. *Analytical Chemistry*, third edition, Saunders College Publishing, Philadelphia, 1980.
54. Harris, D.C. *Quantitative Chemical Analysis*, Second Edition, W.H. Freeman, New York, 1987.
55. Ingle, J.D.; Crouch, S.R. *Spectrochemical Analysis*, Prentice-Hall, New Jersey, 1988.
56. Keller, H. R.; Massart. D. L.; Liang, Y. Z.; Kvalhiem, O. M. *Anal. Chim. Acta* **1992**, 263, 29-36.
57. Keller, H. R.; Massart. D. L.; Liang, Y. Z.; Kvalhiem, O. M. *Anal. Chim. Acta* **1992**, 267, 63-71.
58. Kvalhiem, O. M.; Brakstad, F.; Liang, Y. Z. *Anal. Chem.* **1994**, 66, 43-51.
59. Dose, E.V.; Guichon, G. *Anal. Chem.*, **1989**, 61, 2571-2579.
60. Vanslyke, S. J. Ph. D. Thesis, Department of Chemistry, Dalhousie University, Halifax, Nova Scotia, Canada, 1994.
61. Voigtman, E. *Analytical Instrumentation*, **1993**, 21, 43-62.
62. Savitzky, A.; Golay, M. J. E. *Anal. Chem.* **1964**, 36, 1627-1639.
63. Sánchez, F.C.; Khots, M.S.; Massart, D.L.; De Beer, J.O. *Anal. Chim. Acta* **1991**, 285, 181-192.
64. Rossum, J.R. *J. Amer. Water Works Assoc.* **1975**, 67, 204-205.
65. Barrow, G. M. *Physical Chemistry*, 4th ed.; M^oGraw-Hill: New York, 1979.
66. Atkins, P. W. *Physical Chemistry*, 4th ed.; W.H. Freeman: New York, 1990.
67. Fuoss R. M.; Accascina, F. *Electrolytic Conductance*; Interscience Publishers: New York, 1959.
68. Harned, H. S.; Owen, B. B. *The Physical Chemistry of Electrolytic Solutions*; 3rd ed.; Reinhold: New York, 1958.
69. Clesceri, L. S.; Greenberg, A. E.; Trussell, R. R., Eds.; *Standard Methods For The Examination of Water and Wastewater*, 17th ed.; American Public Health Association: Washington DC, 1989.

70. Shedlovsky, T. *J. Amer. Chem. Soc.* **1932**, *54*, 1405-1411.
71. Martens, H.; Naes, T.; *Multivariate Calibration*; Wiley; New York, 1987.
72. Lorber, A.; Wangen, L.E.; Kowalski, B. R. *J. Chemom.* **1987**, *1*, 19-31.
73. Rumelhart, D. E.; McClelland, J. L. *Parallel Distributed Processing*, Vols 1 and 2; MIT Press: Bradford, 1986.
74. Wythoff B. *J. Chemom. Intell. Lab. Syst.* **1993**, *18*, 115-155.
75. Bos M.; Bos A.; Van Der Linden, W. E. *Anal. Chim. Acta* **1990**, *233*, 31-39.
76. Hecht-Nielson, R. *Neurocomputing*, Addison-Wesley; New York, 1991.
77. Liu, Y.; Upadhyaya, B. R.; Naghedolfeizi, M. *Appl. Spec.* **47**, 12-23.
78. Blank, T. B.; Brown, S. D.; *Anal. Chim. Acta*, **1993**, *277*, 273-287.
79. Wise, B. M. *PLS_Toolbox*, version 1.2, 1992.
80. Kaye, G. W. C.; Laby, T. H. *Tables of Physical and Chemical Constants*; Longman: London, 1973.
81. Robinson, R. A.; Stokes, R. H. *Electrolyte Solutions*; Butterworth: London, 1959.
82. Gates, C. *The Application of Neural Networks to Predicting The Conductivity of Water*, Masters Thesis, Department of Mathematics Statistics and Computer Science, Dalhousie University, 1995.
83. Sasaki, K.; Kawata, S.; Minami, S. *Appl. Spectrosc.* **1986**, *40*, 185-190.
84. Carey, W.P.; Beebe, K.R.; Kowalski, B.R.; Illman, D.L.; Hirschfeld, T. *Anal.*
85. Press, W.H.; Flannery, B.P.; Teukolsky, S.A.; Vetterling, W.T. *Numerical Recipes: The Art of Scientific Computing*; Cambridge University Press: New York, 1986.
86. Hecht, H.G. *Mathematics in Chemistry: An Introduction To Modern Methods*; Prentice-Hall Inc.: New Jersey, 1990.
87. Kalivas, J.H.; Roberts, N.; Sutter, J.M. *Anal. Chem.* **1989**, *61*, 2024-2030.

88. Holland, J. H. *Scientific American*, **1992**, July, 66-72.
89. Lucasius, C. B.; Kateman, G. *Chemom. Intell. Lab Syst.* **1993**, 9, 1-33.
90. Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*; Springer-Verlag: New York, 1992.
91. Rawlins, G. J. E. *Foundations of Genetic Algorithms* Morgan Kaufmann Publishers Inc. San Mateo, California, 1991.
92. Davis, L. Ed.; *Handbook of Genetic Algorithms*; Van Nostrand Reinhold: New York, 1991.
93. Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning* Addison-Wesley Publishing Company Inc. Don Mills, Ontario, 1989.
94. Li, T. H.; Lucasius, C. B.; Buydens, L.; Kateman, G. *Anal. Chim. Acta*, **1992**, 268, 123-134.
95. Bos, M.; Weber, H. T. *Anal. Chim. Acta* **1991**, 247, 97-105.
96. Fontain, E. *Anal. Chim. Acta* **1992**, 265, 227-232.
97. Wehrens, R.; Lucasius, C. B.; Buydens, L.; Kateman, G. *Anal. Chim. Acta* **1993**, 277, 313-324.
98. Lucasius, C. B. Ph. D. Thesis, Laboratory for Analytical Chemistry, Faculty of Science, Katholieke Universiteit, Nijmegen, Netherlands, 1993.
99. Seasholtz, M. B.; Kowalski, B. R. *Anal. Chim. Acta* **1993**, 277, 165-177.
100. Leardi, R.; Boggia, R.; Terrile, M. J. *Chemom.* **1992**, 6, 267-281.
101. Leardi, R.; *J. Chemom.* **1994**, 8, 65-79.
102. Jain, A, K; Dubes, R, C, *Algorithms for Clustering Data*; Prentice-Hall: New Jersey, 1988.
103. Kaufman, L; Rousseeuw, P.J. *Finding Groups In Data: An Introduction To Cluster Analysis*, John-Wiley & Sons, Inc., New York, 1990.
104. Sibbald, D. B.; Wentzell, P.D.; Wade, A.P *Trends in Anal. Chem.* **1989**, 8, 289-291.

105. White, R. L.; Wentzell, P.D.; Beasey, M. A.; Clark, D.S.; Grund, D.W. *Anal. Chim. Acta* **1993**, *277*, 333-346.
106. Summerlin, L. R.; Ealy, J. L. *Chemical Demonstrations: A Sourcebook for Teachers Volume 1*, Second Edition, American Chemical Society, 1988.
107. Walker, J. *Scientific American*, **1978**, *July*, 152-160.
108. Perrin, D. D.; Dempsey, B. *Buffers for pH and Metal Ion Control*, Chapman and Hall, London, 1974.
109. Albert, D. J.; Hopke, P. K. *Atmospheric Environment* **1981**, *15*, 675-687.
110. Hopke, P. K. Editor, *Receptor Modelling For Air Quality Management*, Elsevier, Science Publishers, Amsterdam, 1991.

Appendices

Several programs were written to analyze the data in this thesis. Some of the program listings follow. All of all the programs were written in MATLAB. A summary of the programs listed here is given below..

| <u>Program</u> | <u>Description</u> |
|------------------|--|
| Chapter 2 | |
| A multkf.m | Program that perfoms EPA on a multicomponent system. |
| Chapter 3 | |
| B henosim.m | Program to simulate the response from the HP 5242A diode array detector. |
| C kfcorr.m | Program that performs EPA on nonideal systems. |
| Chapter 4 | |
| D fenfinal.m | Program that calculates SEP for RM, MRM, ERM and EMRM. |
| E mlrfinal.m | Program that calculates SEP for various MLR models. |
| F crfinal.m | Program that calculates SEP for continuum regression. |
| Chapter 5 | |
| G gavar.m | Program to select terms using the variable size method. |
| H gafixed.m | Program to select terms using the fixed size method. |
| Chapter 6 | |
| I gaorder.m | Program to order samples. |

Appendix A

Program listing for multkf.m

```
% multkf.m
% Written by Stephen Hughes
% Dalhousie University
% Last modified July 1993
clear
clf
tic
%Options
% a) orthogonal or vertical fit
% if want vertical set vert=1;
% if want orthogonal set vert=0;
vert=0;
% b) forward and/or reverse filtering
% if want forward only set forrev=1;
% if want forward and reverse set forrev=2;
forrev=2;
% c) select wavelength
% if want to specify independent wavelengths set wave_select=0;
% if want to search independent wavelengths set wave_select=1;
wave_select=0;
% d) Scan time correction
% yes set scantime==1
% no set scantime==0
scantime=1;

% Load in data

!timread4
load absorb.dat

D = absorb;
nmd=4;

t=61:0.5:183.5;
w=230:2:300;

nw = length(w);
k = length(t);
```

```

[m,n]=size(D);
worig=w;
rmsfitfr=zeros(2*nmd,k);
rmsinnfr=zeros(2*nmd,k);
Dcorr=D;

if scantime==1;
    tscan=0.1;
    ndiodes=328;
    for i=2:m
        for j=1:n
            lamda=((w(j)-190)/2)+9;
            Dcorr(i,j)=D(i,j)-(D(i,j)-D(i-1,j))*((lamda-1)/(ndiodes-1))*(tscan/0.5);
        end
    end
end
clear D

D=Dcorr(2:m,:);
Dorig=D;
meanD=mean(Dorig);
stdD=std(Dorig);
for i=1:length(meanD)
    asD(:,i)=(Dorig(:,i)-meanD(i))/stdD(i);
end
[V,eigD]=eig((asD'*asD)/(length(meanD)-1));
V=flip1r(V);
T=asD*V;
V(1:10,1:3)
T(1:10,1:3)
%pause
nw=36;
Dorig=T(:,1:nw);
D=Dorig;

pass=1;
fr=1;

while fr<=forrev
    fr
    sent=zeros(nmd,1);

    if fr==2;
        D=flipud(Dorig);
    end
end

```

```
end

if wave_select==0;
    if fr==1;
        id(1)=1;
        id(2)=2;
        id(3)=3;
        id(4)=4;
        sent=ones(nmd,1);
    end
end

if pass==2;
    temp=id;
    id(1)=temp(1);
    id(2)=temp(2);
    id(3)=temp(3);
    id(4)=temp(4);
    sent=ones(nmd,1);
end

e=zeros(k,nw*nmd);
f=zeros(k,nw*nmd);
maxf=zeros(k,nmd);
e2sum=zeros(k,nw*nmd);
f2sum=zeros(k,nw*nmd);
rmsinn=zeros(nmd,k);
rmsfit=zeros(nmd,k);

bg = 75;
if fr ==2
    bg=5;
end
Back = D(bg,:);
sd = std(Back);
clear Back
R = sd.^2;
thres = 6*sd

ct=2

while ct<=k
    ct
```

```

if ct==2
    xprev=zeros(nmd^2,nw);
    Pinit1=diag(linspace(10e10,10e10,nmd));

    for i=1:nmd
        if i==1
            Pinit2=[Pinit1];
        else
            Pinit2=[Pinit2 Pinit1];
        end
    end

    for i=1:nw
        if i==1
            Pinit=[Pinit2];
        else
            Pinit=[Pinit2; Pinit];
        end
    end

end

if max(D(ct,:))>= thres | sent(1,1)>0

    for md=1:nmd

        if sent(md,1)>0

            iw=md+1;

            if sent(md,1)==1

                Did = D(:,id(md)); % Locate the position of the ind. variable in D
                Dbe = D(:,md:id(md)-1);
                Daf = D(:,id(md)+1:nw);% Reconstruct D so that id in the 1st col. of D

                wid = w(:,id(md)); % Locate the position of the ind. variable in D
                wbe = w(:,md:id(md)-1);
                waf = w(:,id(md)+1:nw);

                if md==1
                    D = [Did,Dbe,Daf];
                    w = [wid,wbe,waf];
                    wid
                end
            end
        end
    end
end

```



```

end %md==1

if md==2
    D = [D(:,1),Did,Dbe,Daf];
    w = [w(:,1),wid,wbe,waf];
    wid
end %md==2

if md==3
    D = [D(:,1:2) Did,Dbe,Daf];
    w = [w(:,1:2) wid,wbe,waf];
    wid
end %md==3

if md==4
    D = [D(:,1:3) Did,Dbe,Daf];
    w = [w(:,1:3) wid,wbe,waf];
    wid
end %md==4

Dall(:,(md-1)*nw+1:md*nw)=D;

clear wid wbe waf Did Dbe Daf

end % sent(md,1)==1

Trans = diag(linspace(1,1,md));
Q = diag(linspace(0,0,md));

for cw = iw:nw

    xprevsb = xprev(nmd*(md-1)+1:nmd*(md-1)+(md-1)+1,cw);
    Pinitsub = Pinit(nmd*(cw-1)+1:nmd*(cw-1)+md,nmd*(md-1)+1:nmd*
(md-1) +(md-1)+1);

    if md==1 H=[Dall(ct,nw*(md-1)+1)]; end
    if md==2 H=[Dall(ct,nw*(md-1)+1) Dall(ct,nw*(md-1)+2)]; end
    if md==3 H=[Dall(ct,nw*(md-1)+1) Dall(ct,nw*(md-1)+2)
Dall(ct,nw*(md-1)+3)]; end
    if md==4 H=[Dall(ct,nw*(md-1)+1) Dall(ct,nw*(md-1)+2)
Dall(ct,nw*(md-1)+3) Dall(ct,nw*(md-1)+4)]; end

    HT = H';
    K = (Pinitsub*HT)/(H*Pinitsub*HT + R);

```

```

e(ct,cw+nw*(md-1)) = Dall(ct,cw+nw*(md-1))-H*xprevsub;
xest = xprevsub+K*e(ct,cw+nw*(md-1));
f(ct,cw+nw*(md-1)) = Dall(ct,cw+nw*(md-1))-H*xest;

% Calculate the innovations eij

if vert==0;
    if md == 1, pdt=xest(1)*xest(1); end
    if md == 2, pdt=xest(1)*xest(1)+xest(2)*xest(2); end
    if md == 3, pdt=xest(1)*xest(1)+xest(2)*xest(2)+xest(3)*xest(3); end
    if md == 4,
pdt=xest(1)*xest(1)+xest(2)*xest(2)+xest(3)*xest(3)+xest(4)*xest(4); end
        e(ct,cw+nw*(md-1))=e(ct,cw+nw*(md-1))/sqrt(pdt+1);
    end

% Calculate the innovations fij

if vert==0;
    if md == 1, pdt=xest(1)*xest(1); end
    if md == 2, pdt=xest(1)*xest(1)+xest(2)*xest(2); end
    if md == 3, pdt=xest(1)*xest(1)+xest(2)*xest(2)+xest(3)*xest(3); end
    if md == 4,
pdt=xest(1)*xest(1)+xest(2)*xest(2)+xest(3)*xest(3)+xest(4)*xest(4); end
        f(ct,cw+nw*(md-1))=f(ct,cw+nw*(md-1))/sqrt(pdt+1);
    end

Tpose = (Trans - K*H)';
P = (Trans - K*H)*Pinitsub*(Tpose)+K*R*K';

TransT = Trans';
xprevsub = Trans*xest;
Pinitsub = Trans*P*TransT + Q;

xprev(nmd*(md-1)+1:nmd*(md-1)+(md-1)+1,cw)=xprevsub;

Pinit(nmd*(cw-1)+1:nmd*(cw-1)+md,nmd*(md-1)+1:nmd*(md-1)+(md-1)+1)=Pinit
sub;

end % for wavelength w

clear HT K xest Tpose P TransT

sent(md,1) = sent(md,1)+1;

```

```

    e2sum = sum((e(ct,iw+nw*(md-1):nw+nw*(md-1))).^2);
    rmsinn(md,ct) = sqrt(e2sum/(nw-md));
    f2sum = sum((f(ct,iw+nw*(md-1):nw+nw*(md-1))).^2);
    rmsfit(md,ct) = sqrt(f2sum/(nw-md));

    end %if sent(md,1)>0

    end % if model md

end % if max(D(ct,:))>= 6*sd | on(1) > 0

if ct==1
    clear Pinit Pinit1 Pinit2
end

ct=ct+1;

end % while time t

if fr==2
    rmsfit = fliplr(rmsfit);
end

rmsfitfr(nmd*(fr-1)+1:fr*nmd,:)=rmsfit;
rmsinnfr(nmd*(fr-1)+1:fr*nmd,:)=rmsinn;
if fr==1, wfr=w; end
if fr==2, wba=w; end
fr = fr+1;
%if pass ~=2, fr = fr+1; end
if pass <3, pass=pass+1; end

if pass==2;
    plot(t,rmsfitfr(1,:),'-',t,rmsfitfr(2,:),'-.',t,rmsfitfr(3,:),':',t,rmsfitfr(4,:),'--')
    title('1com - 2com -. 3com : 4com --')
    xlabel('Time')
    ylabel('Rms Fit')
    pause(0.1)
end % fr==1

if pass==3
    plot(t,rmsfitfr(1,:),'-',t,rmsfitfr(2,:),'-.',t,rmsfitfr(3,:),':',t,rmsfitfr(4,:),'--')
    hold

```

```

plot(t,rmsfitfr(nmd+1,:),'-',t,rmsfitfr(nmd+2,:),'-',t,rmsfitfr(nmd+3,:),'-',t,rmsfitfr(nmd
+4,:),'--')

plot(t,rmsfitfr(nmd+1,:),'o',t,rmsfitfr(nmd+2,:),'o',t,rmsfitfr(nmd+3,:),'o',t,rmsfitfr(nm
d+4,:),'o')
    hold
    title('1com - 2com -. 3com : 4com --')
    xlabel('Time')
    ylabel('Rms Fit')
    pause(0.1)
end % fr==2

end %for fr

clg

if forrev==1;
    plot(t,rmsfitfr(1,:),'-',t,rmsfitfr(2,:),'-',t,rmsfitfr(3,:),'-',t,rmsfitfr(4,:),'--')
    title('1com - 2com -. 3com : 4com --')
    xlabel('Time')
    ylabel('Rms Fit')
    pause(0.1)
end % forrev==1

if forrev==2;
    plot(t,rmsfitfr(1,:),'-',t,rmsfitfr(2,:),'-',t,rmsfitfr(3,:),'-',t,rmsfitfr(4,:),'--')
    hold

plot(t,rmsfitfr(nmd+1,:),'-',t,rmsfitfr(nmd+2,:),'-',t,rmsfitfr(nmd+3,:),'-',t,rmsfitfr(nmd
+4,:),'--')

plot(t,rmsfitfr(nmd+1,:),'o',t,rmsfitfr(nmd+2,:),'o',t,rmsfitfr(nmd+3,:),'o',t,rmsfitfr(nm
d+4,:),'o')
    hold
    title('1com - 2com -. 3com : 4com --')
    xlabel('Time')
    ylabel('Rms Fit')
    pause(0.1)
end % forrev==2

rmsfitfr = rmsfitfr';

```

Appendix B

Program listing for henosim.m

```
% henosim.m
% Written by Stephen Hughes
% Dalhousie University
% Last modified January 1994

% Program to simulate nonidealities in "LC/DAD"
% 1) Scantime effect
% 2) Nonlinearities
% 3) Heteroscedatic Noise

% Need the following files in directory
% henosim.m
% prcl3lo2.txt
% blankflo.txt
% blank1cm.txt
% dark.dat
% gauss.m

% Input section
% 1) Nonidealities
% a) Include Scantime effect?
%   yes, then set scantime = 1
%   no, then set scantime = 0
%   scantime = 0;

% b) Include nonlinearity?
%   yes, then set nonlin = 1
%   no, then set nonlin = 0
%   nonlin = 1;

% c) Include heteroscedastic noise?
%   yes, then set hetnoise = 1
%   no, then set hetnoise = 0
%   hetnoise = 0;
%   also if hetnoise = 0 must set amount of homoscedastic noise
%   fraction(f) of maximum absorbance
%   fr = 0.0004;

% 2) Blank (assumes water as blank)
```

```

% Do you want the flow cell or 1cm cell?
% if want flow cell set flow = 1 and onecm = 0;
% if want 1 cm cell set flow = 0 and onecm = 1;
flow = 1; % Need to modify
onecm = 0;

% 3) Load in spectrum and concentration profile of analyte(s)

analyte =1;

% a) Load in spectrum of analyte
% Need spectrum where Beer's Law holds.
if analyte == 1
    load prcl3lo2.txt; % load spectrum
    abs = prcl3lo2;
    c = 0.025; % concentration in moles/liter
end
if analyte == 2
    load methor.txt; % load spectrum
    abs = methor;
    c = 2/(1000*327.347); % concentration in moles/liter
end

b = 1; % pathlength in cm
w = 190:2:820; % wavelength of spectrum in nm

% b) Load in spectrochromatogram or generate
% (done later within simulation program)
t = 1:1:100; % t = Time
tr = [50]; % tr = retention times (#m = #components)

% 4) Kalman filter parameters
delwkf = 10; % for kalman filtering want to select wavelengths
% delwkf(in nm) apart
% wavelength of interest wi:dwkf:wf

idw(1) = 440; % independent wavelength
idw(2) = 480;

niter = 1; % number of passes through kalman filter
% 1 pass unweighed regression
% 2 or more weighted regression

```

```

nmd = 3;    % if nmd = 3 want to analyze
            % 1L,1NL and 2L

% 5) Other parameters
maxabs = 0.5;    % maximum absorbance
wi = 400;    % lowest wavelength to analyze
wf = 650;    % highest wavelength to analyze

scantime = 0;

% 6) Include Stray Light effect?
% yes, then set stray = 1
% no, then set stray = 0
stray = 0;
r=0.0020;    % Fraction of Stray Light

% Assumptions
% 1) length(blank) = length(dark) = 316
% 2) bandpass = 4.5; %
bpass = 4.4;

% Linear One Component Model md=1
% Nonlinear One Component Model md=2
% Linear Two Component Model md=3
% Nonlinear Two Component Model md=4

% load in blank and dark current
randn('seed',0)
wr = 190:2:820;
% load in blank
if flow == 1 & onecm == 0
    load blankpo
    bi = blankpol;
end
if flow == 0 & onecm == 1
    load blank1cm.txt
    bi = blank1cm;
end

% load in dark current
load dark.dat
dark = dark';

delw = w(2)-w(1);

```

```

int = delwkf/delw; % assumes dw is a multiple of dwkf

% Estimate spectral first derivative with a
% 5 Point Quadratic polynomial Savitzky-Golay type filter

[m,n] = size(abs);
e1 = zeros(m,1);

eo = abs/(b*c);
for i=3:m-2
    e1(i) = ((-2)*eo(i-2)+(-1)*eo(i-1)+(1)*eo(i+1)+(2)*eo(i+2))/10/delw;
end

% Want to take subset of data set to do EPCIA
% wid is the location of wi in vector w
% wfd is the location of wf in vector w
wid = ((wi-w(1))/delw)+1;
wfd = ((wf-w(1))/delw)+1;
wp = w(wid:int:wfd);
eop = eo(wid:int:wfd);
e1p = e1(wid:int:wfd);
absp = abs(wid:int:wfd);
% widr is the location of wi in vector wr
% wfdr is the location of wf in vector wr
widr = ((wi-wr(1))/delw)+1;
wfdr = ((wf-wr(1))/delw)+1;
darkp = dark(widr:int:wfdr);
blp = bl(widr:int:wfdr);

idwloc(1) = ((idw(1)-wp(1))/delwkf)+1;
idwloc(2) = ((idw(2)-wp(1))/delwkf)+1;

% Generating spectrochromatogram

delt = t(2)-t(1);
C = gauss(t,tr);          % C = concentration matrix
D = C*eop;                % Data matrix
Scale = max(max(D));      % Maximum in D
D = (maxabs/Scale)*D;     % Scale D to maxabs
Csc = D*eop*(inv(eop*eop)); % Scale conc profile

clf
plot(t,Csc);
%pause

```



```

plot(wp,absp)
%pause

[mD,nD] = size(D);
Dcorr = zeros(mD,nD);
strayD = zeros(mD,nD);

% Now calculate contribution due to stray light

if stray == 1;
  for i=1:mD
    for j=1:nD
      T = (10.^(-1*D(i,j)));
      strayD(i,j) = -log10(T+r);
    end
  end
  D = strayD;
end

nonl = zeros(mD,nD);
a1 = zeros(mD,nD);

% Now calculate contribution due to polychromatic radiation

if nonlin == 1;
  K = log(10);
  for i=1:mD
    for j=1:nD
      a1(i,j) = e1p(j)*Csc(i)*b;
      pdt = K*a1(i,j)*bpass/2;
      nonl(i,j) = -log10((sinh(pdt))/pdt);
      Dcorr(i,j) = D(i,j)+nonl(i,j);
    end
  end
  D = Dcorr;
end

mesh(D)
%pause

% Now Incorporate the Scan Time Effect
% The scan time effect is discussed in Anal. Chim. Acta. 256, 125-131,1992

```

```

if scantime == 1;
    tscan = 0.1;
    ndiodes = 316;

    for i=2:mD
        for j=1:nD
            lamda = (delwkf/delw)*j-(delwkf/delw)+wid;
            Dcorr(i,j) = D(i,j) + (D(i,j)-D(i-1,j))*((lamda-1)/(ndiodes-1))*(tscan/delt);
        end
    end
    Dcorr(1,:) = D(1,:);
    D = Dcorr;
end % if Scan ==1

fit = polyfit(Csc(1:10),D(1:10,idwloc(1)),1);
Dfit = polyval(fit,Csc);

plot(Csc,D(:,idwloc(1)),'o')
str=sprintf('Calibration Curve at %g nm',idw(1));
title(str)
ylabel('absorbance')
xlabel('concentration')
hold
plot(Csc,Dfit,'-')
hold
%pause

% Now incorporate heteroscedastic noise

% Now want to calculate variance in measurement
% Must calculate measurement standard deviation in intensity
% with ( std_dev = aI.^2+bI+c)

[mD,nD] = size(D);
NR = randn(size(D));
N = zeros(mD,nD);

if hetnoise == 1;
    a = 4.3056e-9;
    b = 3.6897e-5;
    c = 0.1125;

    R = zeros(mD,nD);
    Is = zeros(mD,nD);

```

```

for i=1:mD
    for j=1:nD
        ls(i,j) = (10.^(-1*D(i,j)))*(blp(j)-darkp(j))+darkp(j);
        stdls = a*(ls(i,j).^2)+b*ls(i,j)+c;
        stdbl = a*(blp(j).^2)+b*blp(j)+c;
        stddk = a*(darkp(j).^2)+b*darkp(j)+c;
        stdpart1 = (stdls.^2)/((ls(i,j)-darkp(j)).^2); %+stddk.^2
        stdpart2 = 0; %(stdbl.^2+stddk.^2)/((blp(j)-darkp(j)).^2);
        var_abs = (1/(2.303).^2)*(stdpart1+stdpart2);
        R(i,j) = var_abs;
    end
end

rootR = sqrt(R);

for i=1:mD
    for j=1:nD
        N(i,j) = NR(i,j)*rootR(i,j);
    end
end
end % hetnoise = 1

if hetnoise == 0
    fr = fr/(maxabs/0.25);
    sigma=fr*maxabs;
    N = NR*sigma;
    R = ones(mD,nD)*(sigma.^2);
end

D=D+N;

Rorig = R;
R1L = zeros(mD,nD);
R1NL = zeros(mD,nD);
R2L = zeros(mD,nD);
R2NL = zeros(mD,nD);

nt = length(t);
nw = length(wp);
e = zeros(nt,nw);

xestt = zeros(1,nw);
xesttiter = zeros(niter,nw);

```

```

rmsinn = zeros(nmd,nt);
rmsfit = zeros(nmd,nt);
rmsinnhe = zeros(nmd,nt);
rmsfithe = zeros(nmd,nt);

bg = mD; % variance estimate
% as estimates of background variance

Back = D(bg,:);
sd = std(Back);
clear Back
varhoabs = mean(mean(Rorig(bg-5:1:bg,:)));
for md=1:nmd

    if md == 1
        n = 1;
        dw = 2;
    end
    if md == 2
        n = 2;
        dw = 2;
    end
    if md == 3
        n = 2;
        dw = 3;
    end
    if md == 4
        n = 4;
        dw = 3;
    end

    xestt = zeros(n,nw);
    e2he = zeros(nt,nw);
    f2he = zeros(nt,nw);

    for l = 1:niter

        if md == 1 | md == 3

            if l==1

                Didw = D(:,idwloc(n)); % Locate the position of the independent
                variable in D
                Dbe = D(:,n:idwloc(n)-1);

```

```

    Daf = D(:,idwloc(n)+1:nw);    % Reconstruct D so that id in the 1st
column of D

    Ridw = R(:,idwloc(n));        % H Locate the position of the independent
variable in R
    Rbe = R(:,n:idwloc(n)-1);
    Raf = R(:,idwloc(n)+1:nw);    % Reconstruct R so that id in the 1st
column of R

    if md == 1
        D = [Didw,Dbe,Daf];        % Reconstruction or swap assignment
        R = [Ridw,Rbe,Raf];
    end % md == 1

    if md == 3
        D = [D(:,1),Didw,Dbe,Daf]; % Reconstruction or swap assignment
        R = [R(:,1),Ridw,Rbe,Raf];
    end % md == 3

    clear Didw Dbe Daf Ridw Rbe Raf

end % if l==1

end % if md == 1 | md == 3

Reff = R;

for cw = dw:nw

    % Initialize x and Pinit
    xprev = zeros(n,1);

    Pinit = diag(linspace(10e10,10e10,n));

    % Define the Transition matrix and Q matrix

    Trans = diag(linspace(1,1,n));

    Q = diag(linspace(0,0,n));

    for ct=1:nt

        % Use D in the Kalman Filter
        if md == 1

```

```

    H=[D(ct,1)];
    end

    if md == 2
        H=[D(ct,1) D(ct,1)*D(ct,1)];
    end

    if md == 3
        H=[D(ct,1) D(ct,2)];
    end

    if md == 4
        H=[D(ct,1) D(ct,1)*D(ct,1) D(ct,2) D(ct,2)*D(ct,2) ];
    end

    HT = H';

    % Kalman Gain

    K = (Pinit*HT)/(H*Pinit*HT + Reff(ct,cw));

    % Update the estimate (xest) with measurement Dn

    e(ct,cw) = D(ct,cw) - H*xprev;
    xest = xprev + K*e(ct,cw);
    f(ct,cw) = D(ct,cw) - H*xest;

    % Compute Error Covariance for updated estimate

    Tpose = (Trans - K*H)';
    P = (Trans - K*H)*Pinit*(Tpose)+K*Reff(ct,cw)*K';

    % Project Ahead values of xestnext and Pestnext

    TransT = Trans';
    xprev = Trans*xest;
    Pinit = Trans*P*TransT + Q;

end % for time ct

xestt(:,cw) = xest;

end % for wavelength cw

```

```

if md ==1
  for i=1:nt
    for j=dw:nw
      R1L(i,j) = (xestt(1,j).^2)*(R(i,1)) + R(i,j);
    end
  end
  Reff = R1L;
end % if md ==1

if md ==2
  for i=1:nt
    for j=dw:nw
      R1NL(i,j) = ((xestt(1,j) + 2*xestt(2,j)*D(i,1)).^2)*(R(i,1)) + R(i,j);
    end
  end
  Reff = R1NL;
end % if md ==2

if md ==3
  for i=1:nt
    for j=dw:nw
      R2L(i,j) = (xestt(1,j).^2)*(R(i,1)) + (xestt(2,j).^2)*(R(i,2)) + R(i,j);
    end
  end
  Reff = R2L;
end % if md ==3

if md ==4
  for i=1:nt
    for j=dw:nw
      R2NL(i,j) = ((xestt(1,j) + 2*xestt(2,j)*D(i,1)).^2)*(R(i,1)) + ((xestt(3,j) +
2*xestt(4,j)*D(i,2)).^2)*(R(i,2)) + R(i,j);
    end
  end
  Reff = R2NL;
end % if md ==4

end % for niter l

e2 = e.^2;
f2 = f.^2;

e2sum = sum(e2(:,dw:nw)');
rmsinn(md,:) = sqrt(e2sum/(nw-n));

```

```

f2sum = sum(f2(:,dw:nw)');
rmsfit(md,:) = sqrt(f2sum/(nw-n));

scale=1;
if scale==0;
    Rprop(md,:) = mean(Reff');
    for i=1:mD
        rmsinnhe(md,i) = sqrt((rmsinn(md,i)).^2 - Rprop(md,i));
        rmsfithe(md,i) = sqrt((rmsfit(md,i)).^2 - Rprop(md,i));
    end
end

if scale==1;
    for i=1:mD
        for j=dw:nD
            e2he(i,j) = e2(i,j)*(varhoabs/Reff(i,j));
            f2he(i,j) = f2(i,j)*(varhoabs/Reff(i,j));
        end
    end

    e2hesum = sum(e2he');
    rmsinnhe(md,:) = sqrt(e2hesum/(nw-n));
    f2hesum = sum(f2he');
    rmsfithe(md,:) = sqrt(f2hesum/(nw-n));
end

clear xestt e2sum f2sum

end % nmd

if nmd == 2

    subplot(211)
    plot(t,rmsinn(1,:),'-',t,rmsinn(2,:),'-')
    title('1L -, 1NL -.')
    xlabel('Time')
    ylabel('Rms Inn')

    subplot(212)
    plot(t,rmsinnhe(1,:),'-',t,rmsinnhe(2,:),'-')
    title('Corrected for heter 1L -, 1NL -.')
    xlabel('Time')
    ylabel('Rms Inn')

```



```

subplot(211)
plot(t,rmsfit(1,:),'-',t,rmsfit(2,:),'-')
title('1L -, 1NL -')
xlabel('Time')
ylabel('Rms Fit')

subplot(212)
plot(t,rmsfithe(1,:),'-',t,rmsfithe(2,:),'-')
title('Corrected for heter 1L -, 1NL -')
xlabel('Time')
ylabel('Rms Fit')
pause

end % if nmd == 2

if nmd == 3

subplot(211)
plot(t,rmsinn(1,:),'-',t,rmsinn(2,:),'-',t,rmsinn(3,:),':')
title('1L -, 1NL -. 2L:')
xlabel('Time')
ylabel('Rms Inn')

subplot(212)
plot(t,rmsinnhe(1,:),'-',t,rmsinnhe(2,:),'-',t,rmsinnhe(3,:),':')
title('Corrected for heter 1L -, 1NL -. 2L:')
xlabel('Time')
ylabel('Rms Inn')

subplot(211)
plot(t,rmsfit(1,:),'-',t,rmsfit(2,:),'-',t,rmsfit(3,:),':')
title('1L -, 1NL -. 2L:')
xlabel('Time')
ylabel('Rms Fit')

subplot(212)
plot(t,rmsfithe(1,:),'-',t,rmsfithe(2,:),'-',t,rmsfithe(3,:),':')
title('Corrected for heter 1L -, 1NL -. 2L:')
xlabel('Time')
ylabel('Rms Fit')
end % if nmd == 3

```

Appendix C

Program listing for kfcorr.m

```
% kfcorr.m
% Written by Stephen Hughes
% Dalhousie University
% Last modified January 1994

% Evolving Principal Component Innovation Analysis 11:34AM 1/12/94
% Program to compensate for
% 1) Uncertainty in both x and y -- Doubly Weighted Regression
% 2) Scan Time Effect -- Pretreat data by Using Massart Equation
% 3) Heteroscedastic Errors -- Weigh innovations by their variances
% 4) Nonlinearities due to polychromatic radiation -- Quadratic Model
% 5) Multicomponent Nonlinear Models

% Linear One Component Model md=1
% Nonlinear One Component Model md=2
% Linear Two Component Model md=3
% Nonlinear Two Component Model md=4

clear
!timread2
load abs.dat
load var.dat

Dall = abs;
Rall = var;
D = Dall(:,1:3:76);
R = Rall(:,1:3:76);
w = 420:6:570;
t = 1:1:70;
save art15abs.dat D /ascii
save art15var.dat R /ascii
save art15w.dat w /ascii
bg = 70;
id(1) = 15;
id(2) = 10+1;
niter=1;
nmd=3
pause
```

```

delwkf = 2; % for kalman filtering want to select wavelengths
% delwkf(in nm) apart
% wavelength of interest wi:dwkf:wf

subplot(211)
plot(w,D)
xlabel('wavelength')
ylabel('Absorbance')
subplot(212)
plot(w,R)
xlabel('wavelength')
ylabel('Variance in Abs')
pause

clg
plot(t,D')
xlabel('time')
ylabel('Absorbance')
pause

[mD,nD] = size(D);

Rorig = R;
R1L = zeros(mD,nD);
R1NL = zeros(mD,nD);
R2L = zeros(mD,nD);
R2NL = zeros(mD,nD);

% Now Incorporate the Scan Time Effect
% The scan time effect is discussed in Anal. Chim. Acta. 256, 125-131,1992

Scan = 0; %1 scan time correction applied 0 not applied

% Scan time effect will have to be considered later
if Scan ==1
    Dcorr = zeros(mD,nD);
    for i=1:mD
        for j=1:nD
            lamda = (delwkf/delw)*j-(delwkf/delw)+wid+9;
            ndiodes = 328; % ???
            Dcorr(i,j) = D(i,j) - (D(i,j)-D(i-1,j))*((lamda-1)/(ndiodes -1))*(tscan/delt);
        end
    end
    D = Dcorr;

```

```

end % if Scan ==1

% A program to do Kalman Filtering in Matlab
%
% Matrices and vectors defined
%   H = observation matrix
%   x = state vector
%   K = Kalman gain
%   P = Covariance Matrix

% As mentioned earlier this program incorporates uncertainty in both the
% independent and dependent axis.
% Reference Data Analysis For Scientists and Engineers by S.L.Meyer p.75

k = length(t);
nw = length(w);
e = zeros(k,nw);
f = zeros(k,nw);
rmsinn = zeros(nmd,k);
rmsfit = zeros(nmd,k);
rmsinnhe = zeros(nmd,k);
rmsfithe = zeros(nmd,k);
varho = zeros(nmd,k);

Back = D(bg,:);
sd = std(Back);
clear Back
varhoabs = mean(mean(Rorig(65:1:70,:)));

for md=1:nmd
md
  if md == 1
    n = 1;
    iw = 2;
  end
  if md == 2
    n = 2;
    iw = 2;
  end
  if md == 3
    n = 2;
    iw = 3;
  end
  if md == 4

```

```

n = 4;
iw = 3;
end

xestt = zeros(n,nw);

for l =1:niter

    if md ==1 | md ==3
        if l==1
            in D
                Did = D(:,id(n));          % Locate the position of the independent variable
            D
                Dbe = D(:,n:id(n)-1);
                Daf = D(:,id(n)+1:nw);    % Reconstruct D so that id in the 1st column of
            D

            in D
                Rid = R(:,id(n));          % Locate the position of the independent variable
            D
                Rbe = R(:,n:id(n)-1);
                Raf = R(:,id(n)+1:nw);    % Reconstruct D so that id in the 1st column of
            D

            if md ==1
                D = [Did,Dbe,Daf];          % Reconstruction or swap assignment
                R = [Rid,Rbe,Raf];
            end %md==1

            if md ==3
                D = [D(:,1) Did,Dbe,Daf];    % Reconstruction or swap assignment
                R = [R(:,1) Rid,Rbe,Raf];
            end %md==3

            clear Did Dbe Daf Rid Rbe Raf

            Reff=R;

        end % if l==1

    end % if md ==1 | md ==3

    for cw = iw:nw

        on = 0; % 0

```

```

% Initialize x and Pinit
xprev = zeros(n,1);

Pinit = diag(linspace(10e10,10e10,n));

% Define the Transition matrix and Q matrix

Trans = diag(linspace(1,1,n));

Q = diag(linspace(0,0,n));

for ct=1:k

if max(D(ct,:))>= 6*sd | on == 1 % | is or
    on =1;

    % Use D in the Kalman Filter

    if md == 1
        H=[D(ct,1)];
    end

    if md == 2
        H=[D(ct,1) D(ct,1)*D(ct,1)];
    end

    if md == 3
        H=[D(ct,1) D(ct,2)];
    end

    if md == 4
        H=[D(ct,1) D(ct,1)*D(ct,1) D(ct,2) D(ct,2)*D(ct,2) ];
    end

    HT = H';

% Kalman Gain

    K = (Pinit*HT)/(H*Pinit*HT + Reff(ct,cw));

% Update the estimate (xest) with measurement Dn

    e(ct,cw) = D(ct,cw) - H*xprev;
    xest = xprev + K*e(ct,cw);

```

```

    f(ct,cw) = D(ct,cw) - H*xest;

    % Compute Error Covariance for updated estimate

    Tpose = (Trans - K*H)';

    P = (Trans - K*H)*Pinit*(Tpose)+K*Reff(ct,cw)*K';

    % Project Ahead values of xestnext and Pestnext

    TransT = Trans';
    xprev = Trans*xest;
    Pinit = Trans*P*TransT + Q;

    end % if D > 6*sd

end % for time ct

xest(:,cw) = xest;

end % for wavelength w

if md ==1
    for i=1:k
        for j=iw:nw
            R1L(i,j) = (xestt(1,j).^2)*(R(i,1)) + R(i,j);
        end
    end
    Reff = R1L;
end % if md ==1

if md ==2
    for j=1:k
        for j=iw:nw
            R1NL(i,j) = ((xestt(1,j) + 2*xestt(2,j)*D(i,1)).^2)*(R(i,1)) + R(i,j);
        end
    end
    Reff = R1NL;
end % if md ==2

if md ==3
    for i=1:k
        for j=iw:nw
            R2L(i,j) = (xestt(1,j).^2)*(R(i,1)) + (xestt(2,j).^2)*(R(i,2)) + R(i,j);

```

```

    end
  end
  Reff = R2L;
end % if md ==3

if md ==4
  for i=1:k
    for j=iw:nw
      R2NL(i,j) = ((xestt(1,j) + 2*xestt(2,j)*D(i,1)).^2)*(R(i,1)) + ((xestt(3,j) +
2*xestt(4,j)*D(i,2)).^2)*(R(i,2)) + R(i,j);
    end
  end
  Reff = R2NL;
end % if md ==4

end % for niter I

e2 = e.^2;
f2 = f.^2;

e2sum = sum(e2(:,iw:nw));
rmsinn(md,:) = sqrt(e2sum/(nw-n));
f2sum = sum(f2(:,iw:nw));
rmsfit(md,:) = sqrt(f2sum/(nw-n));

scale=1;
if scale==0;
  Rprop(md,:) = mean(Reff);
  for i=1:mD
    rmsinnhe(md,i) = sqrt((rmsinn(md,i)).^2 - Rprop(md,i));
    rmsfithe(md,i) = sqrt((rmsfit(md,i)).^2 - Rprop(md,i));
  end
end

if scale==1;
  for i=1:mD
    for j=iw:nD
      e2he(i,j) = e2(i,j)*(varhoabs/Reff(i,j));
      f2he(i,j) = f2(i,j)*(varhoabs/Reff(i,j));
    end
  end

  e2hesum = sum(e2he');
  rmsinnhe(md,:) = sqrt(e2hesum/(nw-n));

```



```

    f2hesum = sum(f2he');
    rmsfithe(md,:) = sqrt(f2hesum/(nw-n));
end

clear xestt e2sum f2sum
end % nmd

clg

if nmd == 1
    subplot(211)
    plot(t,rmsfit(1,:),'-')
    title('x 1com -')
    xlabel('Time')
    ylabel('Rms Fit')

    subplot(212)
    plot(t,rmsfithe(1,:),'-')
    title('Hetcor x 1com -')
    [a,maxfit] = max(rmsfithe(1,:));
    str = sprintf('Std dev in abs %4.2e',sqrt(varhoabs));
    text(t(2),rmsfithe(1,maxfit),str);
    xlabel('Time')
    ylabel('Rms Fit')
end % nmd ==1

if nmd == 2
    subplot(211)
    plot(t,rmsfit(1,:),'-',t,rmsfit(2,:),'-')
    title('x 1com - x x^2 1com -')
    xlabel('Time')
    ylabel('Rms Fit')

    subplot(212)
    plot(t,rmsfithe(1,:),'-',t,rmsfithe(2,:),'-')
    title('Hetcor x 1com - x x^2 1com -')
    [a,maxfit] = max(rmsfithe(1,:));
    str = sprintf('Std dev in abs %4.2e',sqrt(varhoabs));
    text(t(2),rmsfithe(1,maxfit),str);
    xlabel('Time')
    ylabel('Rms Fit')
end % if nmd == 2

```

```
if nmd == 3
```

```
    subplot(211)
    plot(t,rmsfit(1,:),'-',t,rmsfit(2,:),'-',t,rmsfit(3,:),':')
    title('x 1com - x x^2 1com - x1 x2 2com :')
    xlabel('Time')
    ylabel('Rms Fit')
```

```
    subplot(212)
    plot(t,rmsfithe(1,:),'-',t,rmsfithe(2,:),'-',t,rmsfithe(3,:),':')
    title('Hetcor x 1com - x x^2 1com - x1 x2 2com :')
    [a,maxfit] = max(rmsfithe(1,:));
    str = sprintf('Std dev in abs %4.2e',sqrt(varhoabs));
    text(t(2),rmsfithe(1,maxfit),str);
    xlabel('Time')
    ylabel('Rms Fit')
```

```
end % if nmd == 3
```

```
if nmd == 4
```

```
    subplot(211)
    plot(t,rmsfit(1,:),'-',t,rmsfit(2,:),'-',t,rmsfit(3,:),':',t,rmsfit(4,:),'-')
    title('1 com x1 - x1 x1^2 -- 2 com - x1 x2 : x1 x1^2 x2 x2^2 -.')
    xlabel('Time')
    ylabel('Rms Fit')
```

```
    subplot(212)
    plot(t,rmsfithe(1,:),'-',t,rmsfithe(2,:),'-',t,rmsfithe(3,:),':',t,rmsfithe(4,:),'-')
    title('1 com x1 - x1 x1^2 -- 2 com - x1 x2 : x1 x1^2 x2 x2^2 -.')
    [a,maxfit] = max(rmsfithe(1,:));
    str = sprintf('Std dev in abs %4.2e',sqrt(varhoabs));
    text(t(2),rmsfithe(1,maxfit),str);
    xlabel('Time')
    ylabel('Rms Fit')
```

```
end % if nmd == 4
```

Appendix D

Program listing for fenfinal.m

```
% fenemp.m
% Written by Stephen Hughes
% Dalhousie University
% Last modified July 1993

% A Program to analysis conductance data
% cond.dat contains the conductance values ns x 4
% Column      Comment
% 1           Rossum
% 2           Diluted Sample
% 3           Infinite Dilution
% 4           Measured Conductance

% nine.dat contains the ionic concentrations expressed in mg/L ns x 9
% Column      Comment
% 1           Sodium
% 2           Potassium
% 3           Calcium
% 4           Magnesium
% 5           Sulfate
% 6           Chloride
% 7           Nitrate + Nitrite
% 8           Carbonate
% 9           Bicarbonate

% other.dat contains the other measured parameters ns x 5
% Column      Comment
% 1           Alkalinity
% 2           Silica
% 3           Orthophosphate
% 4           Ammonia
% 5           pH

clear
load cond.dat
load other.dat
load nine.dat

[mc,nc] = size(cond);
```

% Eliminate #448 Incomplete Data Sample

```
nine = [nine(1:447,:); nine(449:mc,:)];
cond = [cond(1:447,:); cond(449:mc,:)];
other = [other(1:447,:); other(449:mc,:)];
```

```
Cation = [ nine(:,1:4) other(:,4)];
Anion = [ nine(:,5:9) other(:,3)];
pH = other(:,5);
```

% Cation contains the ionic concentrations expressed in mg/L ns x 5

| % Column | Comment |
|----------|-----------|
| % 1 | Sodium |
| % 2 | Potassium |
| % 3 | Calcium |
| % 4 | Magnesium |
| % 5 | Ammonia |

% Anion contains the ionic concentrations expressed in mg/L ns x 6

| % Column | Comment |
|----------|-------------------|
| % 1 | Sulfate |
| % 2 | Chloride |
| % 3 | Nitrate + Nitrite |
| % 4 | Carbonate |
| % 5 | Bicarbonat |
| % 6 | Orthophosphate |

```
[mc,nc] = size(cond);
[mc,ncat] = size(Cation);
[mc,nani] = size(Anion);
```

```
LamdaCa = [ 50.11 73.5 59.5 53.06 73.5];
LamdaAn = [ 80.0 76.35 71.4 69.3 44.5];
```

```
ValenceCa = [ 1 1 2 2 1];
ValenceAn = [ 2 1 2 2 1];
```

```
IonsCa = [ 1 1 1 1 1 ];
IonsAn = [ 1 1 1 1 1 ];
```

```
Phos = 0; % 0 off 1 on
quart = 0; % 0 off 1 on
```

% Convert to equivalents

```

Cationeq = [Cation(:,1)*(1/23) Cation(:,2)*(1/39.1) Cation(:,3)*(2/40.08)
Cation(:,4)*(2/24.31) Cation(:,5)*(1/14)];
Anioneq = [Anion(:,1)*(2/96.06) Anion(:,2)*(1/35.45) Anion(:,3)*(1/14)
Anion(:,4)*(2/100) Anion(:,5)*(1/50) Anion(:,6)*(1/30.97)]; %Anion(:,5)*(1/100)

```

```

% Need to modify Ammonia Concentrations reported

```

```

OH = zeros(mc,1);
NH4 = zeros(mc,1);
Kb = 1.774e-5; % From CRC Handbook 58th edit D-151
for i=1:mc
    OH(i) = 10.^(-1*(14-pH(i)));
    NH4(i) = (Kb*Cationeq(i,5))/(OH(i)+Kb);
end % for i

```

```

%plot(pH,NH4, '*')
%xlabel('pH')
%ylabel('NH4')
%pause

```

```

%Cationeq(:,5) = NH4;

```

```

if Phos == 1

```

```

    % Need to modify Phosphate Concentrations reported
    % Columns 6 of Anioneq is divided into three other columns
    % Column Ion    Lamda
    % 6    H2PO4- 33
    % 7    HPO42- 57
    % 8    PO43- 69

```

```

LamdaAn = [ LamdaAn 33 57 69];
ValenceAn = [ ValenceAn 1 2 3];
IonsAn = [ IonsAn 1 1 1];

```

```

H2PO4 = zeros(mc,1);
HPO4 = zeros(mc,1);
PO4 = zeros(mc,1);

```

```

Ka1 = 7.11e-3;
Ka2 = 6.34e-8;
Ka3 = 4.2e-13;

```

```

for i=1:mc

```

```

    H = 10.^(-1*pH(i));

```

```

Denom = H*H*!i + Ka1*H*H + Ka1*Ka2*H + Ka1*Ka2*Ka3;
H2PO4(i) = (Anioneq(i,6))*Ka1*H*H/(Denom);
HPO4(i) = 2*(Anioneq(i,6))*Ka1*Ka2*H/(Denom);
PO4(i) = 3*(Anioneq(i,6))*Ka1*Ka2*Ka3/(Denom);
end

Anioneq(:,6) = H2PO4;
Anioneq(:,7) = HPO4;
Anioneq(:,8) = PO4;

plot(pH,H2PO4,'*')
xlabel('pH')
ylabel('H2PO4')
pause

plot(pH,HPO4,'*')
xlabel('pH')
ylabel('HPO4')
pause

plot(pH,PO4,'*')
xlabel('pH')
ylabel('PO4')
pause

end % if Phos == 1

if Phos == 0
  Anioneqnew = [Anioneq(:,1:5)];
  Anioneq = Anioneqnew;
  Anionnew = Anion(:,1:5);
  Anion = Anionnew;
end

[mc,ncat] = size(Cationeq);
[mc,nani] = size(Anioneq);

% Now calculate conductance using Rossum model

G = zeros(mc,1);
K1 = zeros(mc,1);
Gnewcon = zeros(mc,1);
K1newcon = zeros(mc,1);
C = zeros(mc,1);

```

```

for i=1:mc

    Gop = 0;
    CpZpsq = 0;
    CpZp = 0;
    SumCa = 0;

    for j=1:length(IonsCa)
        Gop = Gop + (IonsCa(j))*(LamdaCa(j))*(Cationeq(i,j));
        CpZpsq = CpZpsq + (IonsCa(j))*(ValenceCa(j).^2)*(Cationeq(i,j));
        CpZp = CpZp + (IonsCa(j))*(ValenceCa(j))*(Cationeq(i,j));
        SumCa = SumCa + (IonsCa(j))*(Cationeq(i,j));
    end

    Gon = 0;
    CnZnsq = 0;
    CnZn = 0;
    SumAn = 0;
    for j=1:length(IonsAn)
        Gon = Gon + (IonsAn(j))*(LamdaAn(j))*(Anioneq(i,j));
        CnZnsq = CnZnsq + (IonsAn(j))*(ValenceAn(j).^2)*(Anioneq(i,j));
        CnZn = CnZn + (IonsAn(j))*(ValenceAn(j))*(Anioneq(i,j));
        SumAn = SumAn + (IonsAn(j))*(Anioneq(i,j));
    end

% Zn = CnZnsq/CnZn;
% Zp = CpZpsq/CpZp;

% Definition of Effective Charge
Zp = CpZp/SumCa;
Zn = CnZn/SumAn;

Yp = Gop/SumCa;
Yn = Gon/SumAn;
Yo = Yp+Yn;

C(i) = (SumCa + SumAn)/2;

Q = (Zn*Zp*Yo)/((Zp+Zn)*(Zp*Yn+Zn*Yp));

K1(i) = (Zp*Zn/115.2)*(2*Q/(1+sqrt(Q)))*((Zp+Zn).^0.5);
K1newcon(i) = (Zp*Zn/114.4)*(2*Q/(1+sqrt(Q)))*((Zp+Zn).^0.5);
Gi = Gop + Gon - (Yo*K1(i)+0.668*((Zp+Zn).^1.5))*(C(i)^1.5);
G(i) = Gi;

```

```

    Gnewcon(i) = Gop + Gon -
    (Yo*K1newcon(i)+0.676*((Zp+Zn).^(1.5)))*(C(i)^1.5);

end

% Sort Data according to measured conductance values

Cationeqsort = zeros(mc,ncat);
Anioneqsort = zeros(mc,nani);
Condsort = zeros(mc,nc);
Gsort = zeros(mc,1);
Gnewconsort = zeros(mc,1);
Csort = zeros(mc,1);
K1sort = zeros(mc,1);
pHsort = zeros(mc,1);
othersort = zeros(mc,4);

[conds,lc] = sort(cond(:,4));

for i=1:mc
    Cationeqsort(i,:) = Cationeq(lc(i),:);
    Anioneqsort(i,:) = Anioneq(lc(i),:);
    Condsort(i,:) = cond(lc(i),:);
    Gsort(i,1) = G(lc(i),1);
    Gnewconsort(i,1) = Gnewcon(lc(i),1);
    Csort(i,1) = C(lc(i),1);
    K1sort(i,1) = K1(lc(i),1);
    pHsort(i,1) = pH(lc(i),1);
    othersort(i,1:3) = other(lc(i),1:3);
    othersort(i,4) = other(lc(i),5);
end

condsub = [cond(1:436,4); cond(438:mc,4)];

[conds,lcsub] = sort(condsub);

% Sample # 422 Conductivity 1390
% Sample # 423 Conductivity 1430
cutoff = 422;

plot(Condsort(:,4),Condsort(:,4),'-r',Condsort(:,4),Gsort,'+g');
title('Rossum')
text(2000,10000, '- Actual')
text(2000,28000,'+ Rossum')

```



```
xlabel('Actual Conductance')
ylabel('Predicted Conductance')
pause
```

```
plot(Condsort(1:mc-1,4),Condsort(1:mc-1,4),'-r',Condsort(1:mc-1,4),Gsort(1:mc-1),'+g')
title('Rossum')
text(2000,10000,'- Actual')
text(2000,9000,'+ Rossum')
xlabel('Actual Conductance')
ylabel('Predicted Conductance')
pause
```

```
plot(Condsort(1:cutoff,4),Condsort(1:cutoff,4),'-r',Condsort(1:cutoff,4),Gsort(1:cutoff),'+g')
title('Rossum')
text(200,1300,'- Actual')
text(200,1200,'+ Rossum')
xlabel('Actual Conductance')
ylabel('Predicted Conductance')
pause
```

```
% Calculate Errors
```

```
errRossum = zeros(mc-1,1);
errRossumNC = zeros(mc-1,1);
errRossumfen = zeros(mc-1,1);
```

```
for i=1:mc-1
    errRossum(i) = Condsort(i,4) - Gsort(i);
    errRossumNC(i) = Condsort(i,4) - Gnewconsort(i);
    errRossumfen(i) = Condsort(i,4) - Condsort(i,1);
end
```

```
RMSRossum1400 = sqrt(sum(errRossum(1:cutoff).^2)/cutoff);
RMSRossumall = sqrt(sum(errRossum(1:mc-1).^2)/(mc-1));
```

```
RMSRossumNC1400 = sqrt(sum(errRossumNC(1:cutoff).^2)/cutoff);
RMSRossumNCall = sqrt(sum(errRossumNC(1:mc-1).^2)/(mc-1));
```

```
RMSRossumfen1400 = sqrt(sum(errRossumfen(1:cutoff).^2)/cutoff);
RMSRossumfenall = sqrt(sum(errRossumfen(1:mc-1).^2)/(mc-1));
```

```
RMSSummary = [ RMSRossum1400 RMSRossumall RMSRossumNC1400
```

```
RMSRossumNCall ]
pause
```

```
RMSfensum = [RMSRossumfen1400 RMSRossumfenall]
pause
```

```
plot(Condsort(1:cutoff,4),Condsort(1:cutoff,4),'-r',Condsort(1:cutoff,4),Condsort(1:
cutoff,2),'+g')
title('Diluted Sample')
text(200,1400,'- Actual')
text(200,1300,'x Diluted Sam')
xlabel('Actual Conductance')
ylabel('Predicted Conductance')
pause
```

```
plot(Condsort(1:cutoff,4),Condsort(1:cutoff,4),'-r',Condsort(1:cutoff,4),Condsort(1:
cutoff,3),'+g')
title('Infinite Dilution')
text(200,1400,'- Actual')
text(200,1300,'* Inf Dilution')
xlabel('Actual Conductance')
ylabel('Predicted Conductance')
pause
```

```
errDil = zeros(mc-1,1);
errInf = zeros(mc-1,1);
```

```
for i=1:mc-1
    errDil(i) = Condsort(i,4) - Condsort(i,2);
    errInf(i) = Condsort(i,4) - Condsort(i,3);
end
```

```
RMSDil1400 = sqrt(sum(errDil(1:cutoff).^2)/cutoff);
RMSDilall = sqrt(sum(errDil(1:mc-1).^2)/(mc-1));
```

```
RMSInf1400 = sqrt(sum(errInf(1:cutoff).^2)/cutoff);
RMSInfall = sqrt(sum(errInf(1:mc-1).^2)/(mc-1));
```

```
RMSDilInf = [ RMSDil1400 RMSDilall RMSInf1400 RMSInfall ]
pause
```

```
% Sample # 422 Conductivity 1390
% Sample # 423 Conductivity 1430
cutoff2 = 483
```

```

Cationeqsort = Cationeqsort(1:cutoff2,:);
Anioneqsort = Anioneqsort(1:cutoff2,:);
Gsort = Gsort(1:cutoff2,1);
Condsort = Condsort(1:cutoff2,:);
Csort = Csort(1:cutoff2,1);
K1sort = K1sort(1:cutoff2,1);
pHsort = pHsort(1:cutoff2,1);
othersort = othersort(1:cutoff2,1:4);

% First And Third Quartiles Stuff
data = [ Cationeqsort Anioneqsort pHsort ];

[md,nd] = size(data);

first = (md/4);
third = (md/4)*3;

if quart == 1
    for i =1:nd
        [sorted,ldata] = sort(data(:,i));
        for j=1:md
            Gsortquant(j,:) = Gsort(ldata(j),:);
            Condsortquant(j,:) = Condsort(ldata(j),:);
        end
        plot(Condsortquant(1:first),Gsortquant(1:first),'*r')
        xlabel('Actual Conductance')
        ylabel('Predicted Conductance')
        title('r first g middle b third')
        hold
        plot(Condsortquant(first+1:third),Gsortquant(first+1:third),'*g')
        plot(Condsortquant(third+1:md),Gsortquant(third+1:md),'*b')
        hold
        pause
    end
end % if quart == 1

ncolumn = 0;

for i=1:ncat
    if lonsCa(i) == 1
        if ncolumn == 0
            X1Cat = [Cationeqsort(:,i)];
        end
        if ncolumn == 1

```

```

        X1Cat = [X1Cat Cationeqsort(:,i)];
    end
    ncolumn = 1;
end
end

ncolumn = 0;

for i=1:nani
    if lonsAn(i) == 1
        if ncolumn == 0
            X1Ani = [Anioneqsort(:,i)];
        end
        if ncolumn == 1
            X1Ani = [X1Ani Anioneqsort(:,i)];
        end
        ncolumn = 1;
    end
end

X1 = [X1Cat X1Ani];

[mX1,nX1] = size(X1);

% Modelling Section

nset = 1;
ncalsam = 200; %cutoff2; 200;
npredsam = (cutoff2)-ncalsam; %cutoff2; (cutoff2)-ncalsam;
predsam1 = 201;
Bi = zeros(nset,1);

exfit = zeros(npredsam,1);
funC = zeros(cutoff2,1);
deviation = zeros(cutoff2,1);
errexRos = zeros(npredsam,1);

% Randomize Data Points
rand('seed',0)
rand('uniform')

X1m = zeros(cutoff2,nX1);
Gm = zeros(cutoff2,1);
Gnewconm = zeros(cutoff2,1);

```

```

MeasCondrn = zeros(cutoff2,4);
Crm = zeros(cutoff2,1);
K1rn = zeros(cutoff2,1);
otherrn = zeros(cutoff2,4);

RMSEP1 = zeros(nset,1);
RMSEP32 = zeros(nset,1);
RMSEP132 = zeros(nset,1);
RMSEPexRos = zeros(nset,1);

clear Anion Anioneq Anioneqnew Anioneqsort Anionnew Cation Cationeq
Cationeqsort nine data X1Ani X1Cat other cond cond2

l = zeros(cutoff2,nset);

plotmod = 1; % 0 off 1 on

for set=1:nset
    set
    rn = rand(cutoff2,1);
    [rn,l(:,set)] = sort(rn);

    for i=1:cutoff2
        X1rn(i,:) = X1(l(i,set),:);
        Grn(i,1) = Gsort(l(i,set),1);
        MeasCondrn(i,:) = Condsort(l(i,set),:);
        Crm(i,1) = Csort(l(i,set),1);
        K1rn(i,1) = K1sort(l(i,set),1);
        otherrn(i,:) = othersort(l(i,set),:);
    end

    % MLR Models
    % eq.^1

    b1 = X1rn(1:ncalsam,:\MeasCondrn(1:ncalsam,4));

    Cal1 = X1rn(1:ncalsam,:)*b1;
    Pred1 = X1rn(predsam1:cutoff2,:)*b1;
    err1 = (Pred1-MeasCondrn(predsam1:cutoff2,4));
    RMSEP1(set) = sqrt(sum(err1.^2)/(npredsam));

    if plotmod == 1

```

```

plot(MeasCondrn(predsam1:cutoff2,4),MeasCondrn(predsam1:cutoff2,4),'-r',MeasCondrn(predsam1:cutoff2,4),Pred1,'*g')
    title('MLR 1')
    text(200,1400,'- Actual')
    text(200,1300,'* Predicted')
    xlabel('Actual Conductance')
    ylabel('Predicted Conductance')
    pause

    plot(MeasCondrn(predsam1:cutoff2,4),err1,'*g')
    title('MLR 1')
    xlabel('Actual Conductance')
    ylabel('Error')
    pause

end % if plotmod == 1

% eq.^3/2

X32rn = X1rn.^(3/2);

b32 = X32rn(1:ncalsam,:)\MeasCondrn(1:ncalsam,4);

Pred32 = X32rn(predsam1:cutoff2,:)*b32;
err32 = (Pred32-MeasCondrn(predsam1:cutoff2,4));
RMSEP32(set) = sqrt(sum(err32.^2)/(npredsam));

if plotmod == 1

plot(MeasCondrn(predsam1:cutoff2,4),MeasCondrn(predsam1:cutoff2,4),'-r',MeasCondrn(predsam1:cutoff2,4),Pred32,'*g')
    title('MLR 3/2')
    text(200,1400,'- Actual')
    text(200,1300,'* Predicted')
    xlabel('Actual Conductance')
    ylabel('Predicted Conductance')
    pause

    plot(MeasCondrn(predsam1:cutoff2,4),err32,'*g')
    title('MLR 3/2')
    xlabel('Actual Conductance')
    ylabel('Error')
    pause
end % if plotmod == 1

```

```

% eq.^1 eq.^3/2

X132rn = [X1rn X32rn];

b132 = X132rn(1:ncalsam,:)\MeasCondrn(1:ncalsam,4);

Pred132 = X132rn(predsam1:cutoff2,:)*b132;
err132 = (Pred132-MeasCondrn(predsam1:cutoff2,4));
RMSEP132(set) = sqrt(sum(err132.^2)/(npredsam));

if plotmod == 1

plot(MeasCondrn(predsam1:cutoff2,4),MeasCondrn(predsam1:cutoff2,4),'-r',MeasCondrn(predsam1:cutoff2,4),Pred132,'*g')
    title('MLR 1 3/2')
    text(200,1400,'- Actual')
    text(200,1300,'* Predicted')
    xlabel('Actual Conductance')
    ylabel('Predicted Conductance')
    pause

    plot(MeasCondrn(predsam1:cutoff2,4),err132,'*g')
    title('MLR 1 3/2')
    xlabel('Actual Conductance')
    ylabel('Error')
    pause

end % if plotmod == 1

% Extended Rossum Model
% y = B*f(C) where y is the deviations from the Rossum Model and
% f(C) = C.^2(1-K1*C.^1/2)

for i=1:cutoff2
    funC(i) = (Cm(i).^2)*(1-K1rn(i)*sqrt(Crn(i)));
    deviation(i) = MeasCondrn(i,4) - Grn(i);
end

Bi(set) = funC(1:ncalsam)\deviation(1:ncalsam);

for i=predsam1:cutoff2
    exfit(i-predsam1+1) = Grn(i) + Bi(set)*funC(i);
end

```

```

for i=1:ncalsam
    CalEMRM(i) = Grn(i) + Bi(set)*funC(i);
end

if plotmod == 1
    plot(MeasCondrn(:,4),MeasCondrn(:,4),'-r')
    title('Rossum and Extended Rossum')
    text(2000,10000,'- Actual')
    text(2000,9000,'+ Rossum')
    text(2000,8000,'x Extended Rossum')
    xlabel('Actual Conductance')
    ylabel('Predicted Conductance')
    hold
    plot(MeasCondrn(predsam1:(cutoff2),4),exfit,'xb')
    plot(MeasCondrn(predsam1:(cutoff2),4),Grn(predsam1:(cutoff2),1),'+g')
    hold
    pause

    plot(MeasCondrn(:,4),MeasCondrn(:,4),'-r')
    title('Extended Rossum')
    text(2000,10000,'- Actual')
    text(2000,8000,'x Extended Rossum')
    xlabel('Actual Conductance')
    ylabel('Predicted Conductance')
    hold
    plot(MeasCondrn(predsam1:(cutoff2),4),exfit,'xb')
    hold
    pause

    diff = exfit - Grn(predsam1:(cutoff2),1);
    plot(MeasCondrn(predsam1:(cutoff2),4),diff,'*g')
    xlabel('Predicted Conductance')
    ylabel('Difference')
    title(' Difference Between ERM - Rossum')
end % if plotmod == 1

for i=1:npredsam
    errexRos(i) = exfit(i) - MeasCondrn(predsam1+i-1,4);
end

RMSEPexRos(set) = sqrt(sum((errexRos).^2)/(npredsam));

end % for nset
pause

```



```
Dataall = zeros(483,16);  
Dataall(:,1:15)=[X1m otherm MeasCondrn(:,4)];  
Dataall(1:200,16)=CalEMRM';  
Dataall(201:483,16)=exfit;  
Dataall(1,:)   
save dataall.dat Dataall -ascii  
pause  
  
MLRRMSEP = [ RMSEP1 RMSEP32 RMSEP132 ]
```

Appendix E

Program listing for mlrfinal.m

```
% mlrfinal.m
% Written by Stephen Hughes
% Dalhousie University
% Last modified July 1993

load X1.dat
load Condsort.dat
%load othersort.dat

% Sample # 422 Conductivity 1390
% Sample # 423 Conductivity 1430
cutoff2 = 483

x = [X1(1:cutoff2,:); % othersort(1:cutoff2,:)];
y = Condsort(1:cutoff2,4);
x = [x(:,1:9) x(:,10)*2];

[m,nv] = size(x);
%save concions.dat x /ascii
%save conduct.dat y /ascii
ncalsam = 200
%pause
npredsam = m-ncalsam;
nset = 1;

nmd = (2.^nv)-1;
table = zeros(nmd,nv);

for i=1:nmd
    inew = i;
    for j=1:nv
        pow2 = 2.^(nv-j);
        if (inew/pow2) >= 1
            table(i,j) = 1;
            inew = inew - pow2;
        end
    end % for j
end % for i
```

```

% Randomize Data Points

rand('seed',0)
rand('uniform')

ayrn = zeros(m,1);
yrn = zeros(m,1);
xrn = zeros(m,nv);

RMSEP = zeros(nmd,nset);
RMSEPCum = zeros(nmd,nset);

for set=1:nset

    set
    rn = rand(m,1);
    [rn,l] = sort(rn);

    for i=1:m
        yrn(i,1) = y(l(i),1);
        xrn(i,:) = x(l(i),:);
    end

    % Autoscale the data
    mx = mean(xrn(1:ncalsam,:));
    stdx = std(xrn(1:ncalsam,:));

    my = mean(yrn(1:ncalsam,:));
    stdy = std(yrn(1:ncalsam,:));

    for i=1:m
        axrn(i,:) = (xrn(i,:)-mx)./stdx;
        ayrn(i,:) = (yrn(i,:)-my)./stdy;
    end

    for i=1:nmd
        cnt=0;
        for j=1:nv
            if cnt>0
                if table(i,j) == 1
                    xnew = [xnew axrn(:,j)];
                end % if
            end % if
            if cnt==0;

```

```

    if table(i,j) == 1
        xnew = axrn(:,j);
        cnt = 1;
    end
end % if
end

B = xnew(1:ncalsam,:)\aym(1:ncalsam);

ypred = stdy*xnew(ncalsam+1:m,:)*B + my;

if i==992
    MLRoptpr = ypred;
    MLRoptca = stdy*xnew(1:ncalsam,:)*B + my;
end

if i==1023
    MLR10pre = ypred;
    MLR10cal = stdy*xnew(1:ncalsam,:)*B + my;
end
RMSEP(i,set) = sqrt(sum((yrn(ncalsam+1:m)-ypred).^2)/npredsam);
clear B

end % for i

end % for nset

```

Appendix F

Program listing for crfinal.m

```
% crfinal.m
% Written by Stephen Hughes
% Dalhousie University
% Last modified July 1993

% Load in data

load X1.dat
load Condsort.dat

% Sample # 422 Conductivity 1390
% Sample # 423 Conductivity 1430
cutoff2 = 422

x1 = X1(1:cutoff2,:);
y = Condsort(1:cutoff2,4);
x = [x1(:,1:9) x1(:,10)*2];

% Autoscale the data
[m,n] = size(x);
ncalsam = 200;
n,preDSam = m - ncalsam;
nset = 1;
plotmod = 0; % 0 off 1 on

mx = mean(x(1:ncalsam,:));
stdx = std(x(1:ncalsam,:));

my = mean(y(1:ncalsam,:));
stdy = std(y(1:ncalsam,:));

mcx = zeros(ncalsam,n);
mcy = zeros(ncalsam,1);

ax = zeros(ncalsam,n);
ay = zeros(ncalsam,1);

for i=1:ncalsam
```

```

    mcx(i,:) = (x(i,:)-mx);
    mcy(i,:) = (y(i,:)-my);
end

for i=1:ncalsam
    ax(i,:) = (x(i,:)-mx)./stdx;
    ay(i,:) = (y(i,:)-my)./stdy;
end

echo on
% Now we will do PCA analysis
echo off

[scores,loads,ssq1,res,q,tsq] = pca(ax,0);
pause

% Labels
labels = [ 'Na ' ; 'K ' ; 'Ca ' ; 'Mg ' ; 'NH4' ; 'SO4' ; 'Cl ' ; 'NO3' ; 'CO3' ; 'HCO'];

echo on
% Now we can plot the loadings
echo off

%pltscrs(scores)

echo on
% Now we can plot the scores
echo off

%plfloads(loads,labels)

powers = [ 8 5.66 4 2.82 2 1.41 1 0.707 0.5 0.353 0.25 0.177 0.125 ];
[junk,npowers] = size(powers);
nmd = 1+npowers+2;

RMSEPpccr = zeros(nset,n);
RMSEPmiras = zeros(nset,1);
RMSEPmlr = zeros(nset,1);
RMSEPmirmc = zeros(nset,1);
RMSEPcum = zeros(nmd,n);
RMSEPcumave = zeros(nmd,n);
RMSEPind = zeros(nset,n+2);
RMSEP = zeros(nmd,n);

```

```

mcxm = zeros(m,n);
mcyrn = zeros(m,1);
axrn = zeros(m,n);
ayrn = zeros(m,1);
ym = zeros(m,1);
xrn = zeros(m,n);

bmirall = zeros(n,nset);
bmirrcall = zeros(n,nset);
bmirasall = zeros(n,nset);

Ypred = zeros(npredsam,n);
Ypredas = zeros(npredsam,nmd);

rand('seed',0);
rand('uniform')

for set=1:nset
    set
    rn = rand(n,1);
    [rn,l] = sort(rn);

    for i=1:m
        yrn(i,1) = y(l(i),1);
        xrn(i,:) = x(l(i),:);
    end

    % Autoscale the data

    mx = mean(xrn(1:ncalsam,:));
    stdx = std(xrn(1:ncalsam,:));

    my = mean(yrn(1:ncalsam,:));
    stdy = std(yrn(1:ncalsam,:));

    for i=1:m
        mcxm(i,:) = (xrn(i,:)-mx);
        mcyrn(i,:) = (yrn(i,:)-my);
    end

    for i=1:m
        axrn(i,:) = (xrn(i,:)-mx)./stdx;
        ayrn(i,:) = (yrn(i,:)-my)./stdy;
    end
end

```

```

% PCR Analysis

[T,P,bpcr] = pcr1(axrn(1:ncalsam,:),ayrn(1:ncalsam,1),n);

for i=1:n
    r = T(:,1:i)\ayrn(1:ncalsam,1);
    Tpred = axrn(ncalsam+1:m,:)*P(:,1:i);
    Ypredv= Tpred*r;
    Ypred(:,i) = stdy*Ypredv + my;
    RMSEPpcr(set,i) = sqrt((sum((Ypred(:,i) -
yrn(ncalsam+1:m,1)).^2))/npredsam);
end

% Compare to MLR

bmlr = xrn(1:ncalsam,.)\yrn(1:ncalsam);
ymlr = xrn(ncalsam+1:m,:)*bmlr;
RMSEPmlr(set) = sqrt((sum((ymlr - yrn(ncalsam+1:m,1)).^2))/npredsam);

bmlrnc = mcxrn(1:ncalsam,.)\mcyrn(1:ncalsam);
ymlr = mcxrn(ncalsam+1:m,:)*bmlrnc + my;
RMSEPmlrnc(set) = sqrt((sum((ymlr - yrn(ncalsam+1:m,1)).^2))/npredsam);

bmlras = axrn(1:ncalsam,.)\ayrn(1:ncalsam);
ymlr = stdy*axrn(ncalsam+1:m,:)*bmlras + my;
RMSEPmlras(set) = sqrt((sum((ymlr - yrn(ncalsam+1:m,1)).^2))/npredsam);

bmlrall(:,set) = bmlr;
bmlrncall(:,set) = bmlrnc;
bmlrasall(:,set) = bmlras;

bmlrsum = [ bmlr bmlrnc bmlras]

pause(1)

if plotmod ==1
    xplot = 1:1:n;
    plot(xplot,RMSEPpcr(set,:),'-r',xplot,RMSEPpcr(set,:),'og')
    xlabel('Number of Latent Variables')
    ylabel('RMSEP')
    title('MLR +b MLRas *w')
    hold
    plot(n,RMSEPmlras(set),'+b')
    plot(n,RMSEPmlr(set),'*w')

```



```

    hold
    pause
end % for plotmod ==1

RMSEPind = [ RMSEPpccr RMSEPmlras RMSEPmlr];

bcr = powerpls(axrn(1:ncalsam,:),ayrn(1:ncalsam),n,powers);
bees = [ bpcr; bcr; bmlras'; bmlr'];

for i=1:nmd
    for j=1:n
        if i<nmd-1
            Ypredas(:,i) = stdy*(axrn(ncalsam+1:m,:)*(bees(((i-1)*n+j),:))) + my;
            RMSEP(i,j) = sqrt((sum((Ypredas(:,i) -
ym(ncalsam+1:m,1)).^2))/npredsam);
        end
        if i==nmd-1
            Ypredas(:,i) = stdy*(axrn(ncalsam+1:m,:))*bmlras + my;
            RMSEP(i,j) = sqrt((sum((Ypredas(:,i) -
ym(ncalsam+1:m,1)).^2))/npredsam);
        end
        if i==nmd
            Ypredas(:,i) = xrn(ncalsam+1:m,:)*bmlr;
            RMSEP(i,j) = sqrt((sum((Ypredas(:,i) -
ym(ncalsam+1:m,1)).^2))/npredsam);
        end
    end
end

if set==1
    RMSEPcum = RMSEP;
    % save Rmsmat1.dat RMSEP /ascii
end

if set==1
    RMSEPcumave = RMSEP
    pause
end

if set>1
    for i=1:nmd
        for j=1:n
            RMSEPcumave(i,j) = (RMSEPcumave(i,j)*(set-1) + RMSEP(i,j))/set;
        end
    end
end

```

```
end
end % if set >1

end % for set=1:nset
```

Appendix G

Program listing for gavar.m

```
% gabin.m
% Written by Stephen Hughes
% Dalhousie University
% Last modified October 1995

clg
clear
% Conductance and GA
% Conductance Stuff
load x1.dat
load condsort.dat

% Sample # 422 Conductivity 1390
% Sample # 423 Conductivity 1430
cutoff = 422

x = [x1(1:cutoff,:)];
y = condsort(1:cutoff,4);
x = [x(:,1:9) x(:,10)*2 x(:,1:9).^(3/2) (x(:,10)*2).^(3/2) x(:,1:9).^(4/2)
(x(:,10)*2).^(4/2) x(:,1:9).^(5/2) (x(:,10)*2).^(5/2)];

[m,nv] = size(x);
tic
ncalsam = 200;
% GA stuff
% if want new calibration and prediction set each generation then mode = 1
% if want new calibration and prediction set each string then mode = 2
mode = 1;
% objective function criteria
% if criteria 1 set then criteria = 1
% if criteria n set then criteria = 2
criteria = 2;
% if plotstuff==0 plotting off ==1 plotting on
plotstuff = 1;
% if plotting minimum RMSEP then plotm = 1
% if plotting mean RMSEP then plotm = 2
plotm = 1;
```

```

nstr = 100;
sel_repr = 0.2*nstr;
mutation = 0.03;
recomb = 0.30;
max_gen = 100;

% Matrix Declaration
npredsam = m-ncalsam;
yrn = zeros(m,1);
xrn = zeros(m,nv);
tot_nmd = (2.^nv); % Matlab can handle up to 2^1023
toptab = zeros(nstr*max_gen,nv);
toprms = zeros(nstr*max_gen,1);
pop = zeros(nstr,nv);
popsort = zeros(nstr,nv);
RMSEP = zeros(nstr,1);
RMSEPsrt = zeros(nstr,1);
minpred = zeros(max_gen,1);
meanpred = zeros(max_gen,1);
pairs = zeros(sel_repr,nv);
ctpop = ones(nstr,max_gen);
ctrms = ones(nstr,max_gen);

if rem(sel_repr,2) == 1
    disp('You dunderhead! You need a even number of strings to selectively
reproduce')
    pause
end

% Generate Initial Strings
rand('seed',0)
rand('uniform')
Randnum = rand(nstr,1)*tot_nmd;

for i=1:nstr
    inew = Randnum(i);
    for j=1:nv
        pow2 = 2.^(nv-j);
        if (inew/pow2) >= 1
            pop(i,j) = 1;
            inew = inew - pow2;
        end
    end % for j
end % for i

```

```

if mode == 1
    m = rand(m,1);
    [rn,l] = sort(rn);
    yrn(:,1) = y(l,1);
    xrn = x(l,:);
    clear l
end % mode

% Quality Function

for i=1:nstr
    i
    if mode == 2
        m = rand(m,1);
        [rn,l] = sort(rn);
        yrn(:,1) = y(l,1);
        xrn = x(l,:);
        clear l
    end % mode

    % Selecting x variables
    cnt=0;
    for j=1:nv
        if cnt>0
            if pop(i,j) == 1
                xnew = [xnew xrn(:,j)];
            end % if
        end % if
        if cnt==0;
            if pop(i,j) == 1
                xnew = xrn(:,j);
                cnt = 1;
            end
        end % if
    end

    % MLR
    if cnt>0
        B = xnew(1:ncalsam,:)\ym(1:ncalsam);
        ypred = xnew(ncalsam+1:m,:)*B;
        RMSEP(i,1) = RMSEP(i,1) +
sqrt(sum((ym(ncalsam+1:m)-ypred).^2)/npredsam);
    end

```

```

if cnt==0
    RMSEP(i,1) = RMSEP(i,1) + 100;
end

end % for i

% Sorting Strings
[sorted,I] = sort(RMSEP(:,1));

popsort = pop(I,:);
RMSEPsort(:,1) = RMSEP(I,1);

clear sorted xnew

RMSEP = RMSEPsort;
pop = popsort;
popprev = pop;
RMSEPprev = RMSEP;

RMSEP(1)
minpred(1) = RMSEP(1);

toptab(1:nstr,:) = pop;
toprms(1:nstr,1) = RMSEP(1:nstr,1);
meanpred(1) = mean(RMSEP(1:nstr,1));

if plotstuff == 1
    if plotm == 1;
        % Displaying Prediction
        plot(1:1:max_gen,minpred,'og')
        xlabel('Generation')
        ylabel('min RMSEP')
        axis([1 max_gen min(minpred(1))-0.1*min(minpred(1))
max(minpred)+0.1*max(minpred)]);
        pause(0.1)
    end
    if plotm == 2;
        plot(1:1:max_gen,meanpred,'og')
        xlabel('Generation')
        ylabel('mean RMSEP')
        axis([1 max_gen min(meanpred(1))-0.1*min(meanpred(1))
max(meanpred)+0.1*max(meanpred)]);
        pause(0.1)
    end
end

```

```

end

for t=2:max_gen
    t
    RMSEP = zeros(nstr,1);
    best = pop(1:sel_repr,:);

    % Recombination/Mutation
    Randpair = rand(sel_repr,1);
    [Randpairsort,lrandpair] = sort(Randpair);
    Npair = (sel_repr/2);

    for i=1:length(Randpair)
        pairs(i,:) = best(lrandpair(i),:);
    end

    % Recombination
    rn_sel_repr = rand(Npair,nv);
    for i=1:Npair
        for j=1:nv
            if rn_sel_repr(i,j)<=recomb;
                if pairs((i-1)*2+1,j) ~= pairs((i-1)*2+2,j)
                    temp=pairs((i-1)*2+1,j);
                    pairs((i-1)*2+1,j)=pairs((i-1)*2+2,j);
                    pairs((i-1)*2+2,j)=temp;
                end
            end
        end
    end

    % Mutation
    rn_mutation = rand(sel_repr,nv);
    for i=1:sel_repr
        for j=1:nv
            if rn_mutation<=mutation;
                if pairs(i,j) == 1
                    pairs(i,j) = 0;
                else
                    pairs(i,j) = 1;
                end
            end
        end
    end
end

```

```

% Quality Function
pop = [pop(1:nstr-sel_repr,:); pairs;];

if mode == 1
    rn = rand(m,1);
    [rn,l] = sort(rn);
    yrn(:,1) = y(l,1);
    xrn = x(l,:);
    clear l
end

for i=1:nstr
    i
    if mode == 2
        rn = rand(m,1);
        [rn,l] = sort(rn);
        yrn(:,1) = y(l,1);
        xrn = x(l,:);
        clear l
    end

    % Selecting x variables
    cnt=0;
    for j=1:nv
        if cnt>0
            if pop(i,j) == 1
                xnew = [xnew xrn(:,j)];
            end % if
        end % if
        if cnt==0;
            if pop(i,j) == 1
                xnew = xrn(:,j);
                cnt = 1;
            end
        end % if
    end

    % MLR
    if cnt>0
        B = xnew(1:ncalsam,:)\yrn(1:ncalsam);
        ypred = xnew(ncalsam+1:m,:)*B;
        RMSEP(i,1) = RMSEP(i,1) +
sqrt(sum((yrn(ncalsam+1:m)-ypred).^2)/npredsam);
    end % for i

```



```

    if cnt==0
        RMSEP(i,1) = RMSEP(i,1) + 100;
    end
end % for i

if criteria == 2
    for i=1:nstr
        trigger=0;
        for j=1:nstr
            if pop(i,:) == popprev(j,:)
                ctrms(i,t) = ctrms(j,t-1)+1;
                trigger=1;
                RMSEP(i,1)=(RMSEPprev(i,1)*(ctrms(i,t)-1)+RMSEP(i,1))/ctrms(i,t);
            end % if
        end % j
        if trigger == 0 ctrms(i,t) = 1; end
    end % i
end % criteria

%Sorting Strings

[sorted,l] = sort(RMSEP(:,1));

popsort = pop(l,:);
RMSEPsot(:,1) = RMSEP(l,1);

clear sorted l xnew

RMSEP = RMSEPsot;
pop = popsort;

% for i=1:nstr
%   trigger=0;
%   for j=1:nstr
%       if pop(i,:) == popprev(j,:)
%           ctpop(i,t) = ctpop(j,t-1)+1;
%           trigger=1;
%       end % if
%   end % j
%   if trigger == 0 ctpop(i,t) = 1; end
% end % i

popprev=pop;

```

```

RMSEPprev = RMSEP;
RMSEP(1)
minpred(t) = RMSEP(1);

toptab((t-1)*nstr+1:(t-1)*nstr+nstr,:) = pop;
toprms((t-1)*nstr+1:(t-1)*nstr+nstr,1) = RMSEP(1:nstr,1);
meanpred(t) = mean(RMSEP(1:nstr,1));

if plotstuff == 1
    if plotm == 2
        plot(1:1:max_gen,meanpred,'og',1:1:t,meanpred(1:t,1),'-r')
        xlabel('Generation')
        ylabel('mean RMSEP')
        axis([1 max_gen min(meanpred(1:t))-0.1*min(meanpred(1:t))
max(meanpred)+0.1*max(meanpred)]);
        pause(0.1)
    end
    if plotm == 1
        % Displaying Prediction
        plot(1:1:max_gen,minpred,'og',1:1:t,minpred(1:t,1),'-r')
        xlabel('Generation')
        ylabel('min RMSEP')
        axis([1 max_gen min(minpred(1:t))-0.1*min(minpred(1:t))
max(minpred)+0.1*max(minpred)]);
        pause(0.1)
    end
end
end %for t

```

Appendix H

Program listing for gafixed.m

```
% gaperm.m
% Written by Stephen Hughes
% Dalhousie University
% Last modified October 1995

clg
clear
% Conductance and GA
load x1.dat
load condsort.dat

% Sample # 422 Conductivity 1390
% Sample # 423 Conductivity 1430
cutoff = 422

x = [x1(1:cutoff,:)];
y = condsort(1:cutoff,4);
x = [x(:,1:9) x(:,10)*2 x(:,1:9).^(3/2) (x(:,10)*2).^(3/2) x(:,1:9).^(4/2)
(x(:,10)*2).^(4/2) x(:,1:9).^(5/2) (x(:,10)*2).^(5/2)];

[m,nv] = size(x);
tic
ncalsam = 200;
% Scaling Auto scale data? No auto = 0 Yes auto = 1
auto=0;
% GA stuff
% if want new calibration and prediction set each generation then mode = 1
% if want new calibration and prediction set each string then mode = 2
mode = 2;
% objective function criteria (Use RMS from previous sets)
% if criteria 1 set then criteria = 1
% if criteria n set then criteria = 2
criteria = 2;
% if plotstuff==0 plotting off ==1 plotting on
plotstuff = 1;
% if plotting minimum RMSEP then plotm = 1
% if plotting mean RMSEP then plotm = 2
plotm = 1;
```

```

nstr = 100;
sel_repr = 0.5*nstr;
mutation = 0.03;
recomb = 0.3;
max_gen = 100;
n_terms = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 25 30 35]

num_mutation=ceil(mutation*sel_repr);

% Matrix Declaration
nga = length(n_terms);
npredsam = m-ncalsam;
ym = zeros(m,1);
xm = zeros(m,nv);
tot_nmd = (2.^nv); % Matlab can handle up to 2^1023
cs = cumsum(n_terms(1:nga));
toptab = zeros(max_gen,cs(nga));
toprms = zeros(max_gen,nga);

for ct_nga=1:nga

    pop = zeros(nstr,nv);
    RMSEP = zeros(nstr,1);
    best=zeros(sel_repr,nv);
    minpred = zeros(max_gen,1);
    meanpred = zeros(max_gen,1);
    pairs = zeros(sel_repr,nv);
    ctrms = ones(nstr,max_gen);

    if rem(sel_repr,2) == 1
        disp('You dunderhead! You need a even number of strings to selectively
reproduce')
        pause
    end

    % Generate Initial Strings
    if ct_nga==1 rand('seed',0), end
    rand('uniform')

    for i=1:nstr
        pop(i,:) = randperm(nv);
    end

```

```

if mode == 1
    rn = rand(m,1);
    [rn,l] = sort(rn);
    yrn(:,1) = y(l,1);
    xrn = x(l,:);
    clear l
end % mode == 1

% Quality Function

for i=1:nstr

    if mode == 2
        rn = rand(m,1);
        [rn,l] = sort(rn);
        yrn(:,1) = y(l,1);
        xrn = x(l,:);
        clear l
    end % mode == 2

    % Selecting x variables

    xnew = xrn(:,pop(i,1:n_terms(ct_nga)));
    cnt=1;

    % MLR
    if cnt>0
        B = xnew(1:ncalsam,:)\yrn(1:ncalsam);
        ypred = xnew(ncalsam+1:m,:)*B;
        RMSEP(i,1) = RMSEP(i,1) +
sqrt(sum((yrn(ncalsam+1:m)-ypred).^2)/npredsam);
    end

    if cnt==0
        RMSEP(i,1) = RMSEP(i,1) + 100;
    end

end % for i

% Sorting Strings
[sorted,l] = sort(RMSEP(:,1));

clear sorted xnew

```

```

popprev = pop;
RMSEPprev = RMSEP;

minrms=RMSEP(l(1))
minpred(1) = RMSEP(l(1));

for i=1:n_terms(ct_nga)
    if ct_nga ==1 toptab(1,i)=pop(l(1),i); end
    if ct_nga >1 toptab(1,cs(ct_nga-1)+i)=pop(l(1),i); end
end

toprms(1,ct_nga) = RMSEP(l(1,1));
meanpred(1) = mean(RMSEP(1:nstr,1));
meanrms=meanpred(1)

if plotstuff == 1
    if plotm == 1;
        % Displaying Prediction
        plot(1:1:max_gen,minpred,'og')
        str=sprintf('%g',pop(l(1),1:n_terms(ct_nga)));
        title(str)
        xlabel('Generation')
        ylabel('min RMSEP')
        axis([1 max_gen min(minpred(1))-0.1*min(minpred(1))
max(minpred)+0.1*max(minpred)]);
    end
    if plotm==2
        plot(1:1:max_gen,meanpred,'og')
        xlabel('Generation')
        ylabel('mean RMSEP')
        axis([1 max_gen min(meanpred(1))-0.1*min(meanpred(1))
max(meanpred)+0.1*max(meanpred)]);
    end
    if ct_nga == 1 pause; end
end % if plotstuff

for t=2:max_gen
    t
    RMSEP = zeros(nstr,1);
    best = pop(l(1:sel_repr),:);

    Randpair = rand(sel_repr,1);
    [Randpairsort,lrandpair] = sort(Randpair);
    Npair = (sel_repr/2);

```

```

for i=1:length(Randpair)
    pairs(i,:) = best(lrاندpair(i,:);
end

% Recombination/Mutation

% Recombination (developed by PDW)
rn_sel_repr = rand(Npair,1);
for i=1:Npair
    for j=1:n_terms(ct_nga)
        if rn_sel_repr(i,1)<=recomb;
            if pairs((i-1)*2+1,j) ~= pairs((i-1)*2+2,j)
                %pairs((i-1)*2+1,:)
                %pairs((i-1)*2+2,:)
                temp1=pairs((i-1)*2+1,j);
                temp2=pairs((i-1)*2+2,j);
                for k=1:n_terms(ct_nga)
                    if pairs((i-1)*2+1,k) == temp2
                        rn_sel_repr(i,1)=1;
                    end
                end
                for k=1:n_terms(ct_nga)
                    if pairs((i-1)*2+2,k) == temp1
                        rn_sel_repr(i,1)=1;
                    end
                end
                if rn_sel_repr(i,1)<=recomb
                    l1=find(pairs((i-1)*2+1,:)==temp2);
                    l2=find(pairs((i-1)*2+2,:)==temp1);
                    pairs((i-1)*2+1,j)=temp2;
                    pairs((i-1)*2+1,l1)=temp1;
                    pairs((i-1)*2+2,j)=temp1;
                    pairs((i-1)*2+2,l2)=temp2;
                end
                %pairs((i-1)*2+1,:)
                %pairs((i-1)*2+2,:)
                rn_sel_repr(i,1)=1;
                %pause(1)
            end
        end
    end
end
end
end

% Mutation D_TM operator

```

```

for i=1:num_mutation
    m_mutate=rand(sel_repr,1);
    [m,l1]=sort(m_mutate);
    pairs=pairs(l1,:);
    clear l1
    m_swap=rand(n_terms(ct_nga),1);
    [rs,l1]=sort(m_swap);
    l1_1 = l1(1);
    clear l1
    m_swap=rand(nv-n_terms(ct_nga),1);
    [rs,l1]=sort(m_swap);
    l1_2 = l1(1)+n_terms(ct_nga);
    clear l1
    temp=pairs(1,:);
    pairs(1,l1_1)=temp(l1_2);
    pairs(1,l1_2)=temp(l1_1);
    clear l1
%    pause
end

% Quality Function
pop(l(nstr-sel_repr+1:nstr,1),:)=pairs;

if mode == 1
    m = rand(m,1);
    [rn,l] = sort(m);
    yrn(:,1) = y(l,1);
    xrn = x(l,:);
    clear l
end

for i=1:nstr

    if mode == 2
        m = rand(m,1);
        [rn,l] = sort(m);
        yrn(:,1) = y(l,1);
        xrn = x(l,:);
        clear l
    end

% Selecting x variables

xnew = xrn(:,pop(i,1:n_terms(ct_nga)));

```



```

cnt=1;

% MLR
if cnt>0
    B = xnew(1:ncalsam,:\yrn(1:ncalsam));
    ypred = xnew(ncalsam+1:m,:)*B;
    RMSEP(i,1) = RMSEP(i,1) +
sqrt(sum((yrn(ncalsam+1:m)-ypred).^2)/npredsam);
end % for i

if cnt==0
    RMSEP(i,1) = RMSEP(i,1) + 100;
end
end % for i

for i=1:nstr
    trigger=0;
    for j=1:nstr
        if pop(i,:) == popprev(j,:)
            ctrms(i,t) = ctrms(j,t-1)+1;
            trigger=1;
            if criteria == 2
                RMSEP(i,1)=(RMSEPprev(i,1)*(ctrms(i,t)-1)+RMSEP(i,1))/ctrms(i,t);
            end
        end % if
    end % j
    if trigger == 0 ctrms(i,t) = 1; end
end % i

%Sorting Strings

[sorted,I] = sort(RMSEP(:,1));

clear sorted xnew

popprev=pop;
RMSEPprev = RMSEP;

minrms=RMSEP(I(1))
minpred(t) = RMSEP(I(1));

for i=1:n_terms(ct_nga)
    if ct_nga ==1 toptab(t,i)=pop(I(1),i); end

```

```

    if ct_nga > 1 toptab(t,cs(ct_nga-1)+i)=pop(l(1),i); end
end

toprms(t,ct_nga) = RMSEP(l(1,1));
meanpred(t) = mean(RMSEP(1:nstr,1));

if plotstuff == 1
    if plotm==2
        % Displaying Prediction
        plot(1:1:max_gen,meanpred,'og',1:1:t,meanpred(1:t,1),'-r')
        xlabel('Generation')
        ylabel('mean RMSEP')
        axis([1 max_gen min(meanpred(1:t))-0.1*min(meanpred(1:t))
max(meanpred)+0.1*max(meanpred)]);
    end
    if plotm==1
        plot(1:1:max_gen,minpred,'og',1:1:t,minpred(1:t,1),'-r')
        str=sprintf('%g',pop(l(1),1:n_terms(ct_nga)));
        title(str)
        xlabel('Generation')
        ylabel('min RMSEP')
        axis([1 max_gen min(minpred(1:t))-0.1*min(minpred(1:t))
max(minpred)+0.1*max(minpred)]);
        pause(0.1)
    end
end % plotstuff

save tabpm1.mat toptab
save rmspm1.mat toprms

end %for t
pause(1)

end % for ct_nga

total = toc

```

Appendix I

Program listing for gaorder.m

```

% gaorder.m
% Written by Stephen Hughes
% Dalhousie University
% Last modified October 1995

clear
rand('seed',0)
randn('seed',0)
%=====
%Parameters
obj_fun = 2 % 1 eigvalue 2 variance no meas var 3 var with meas var
NumPC = 3;
slice = 12;
%Options
graph_gen = 1; % Generate graph? No 0 Yes 1
auto = 1; % Scale data? No 0 Yes 1
data_set = 2 % Data Simulated Methyl Orange 1
%           Chromatographic 2
%           Real    Methyl Orange 3
%           Pyrocatechol Violet 4
%           From Disk 5
%           Rain water 6
%           Clock Rxn 7
%           PCV buffered 8
% GA
nstr = 100;
sel_repr = 0.5*nstr;
mutation = [0.05 0.3 0.5 0.7 1.0];
recomb = 1;
max_gen = 250;

%=====

if data_set==1
    pKa1 = 3.46;
    Ct=0.102/327.347;
    pH = 0:0.4:7;
    % Generate data

```

```

Ka1 = 10^(-1*pKa1);
C=zeros(length(pH),2);
for i=1:length(pH)
    H=10^(-1*pH(i));
    C(i,1)=(H*Ct)/(H+Ka1);
    C(i,2)=(Ka1*Ct)/(H+Ka1);
end

if graph_gen == 1
    clg
    plot(pH,C(:,1),pH,C(:,2))
    xlabel('pH')
    ylabel('Concentration')
    title('Conc')
    hold
    plot(pH,C(:,1),'o',pH,C(:,2),'o')
    hold
    %pause
end

load meth3.dat
load meth16.dat
S(1,:)=meth3';
S(2,:)=meth16';
S=S(:,81:8:206);
w = 350:16:600;          % w = Wavelengths

if graph_gen == 1
    clg
    plot(w,S(1,:),w,S(2,:))
    xlabel('wavelength')
    ylabel('Absorbance')
    hold
    plot(w,S(1:,:),'o',w,S(2:,:),'o')
    hold
    %pause
end

X = C*S;          % Data matrix
Scale = max(max(X)); % Maximum in X
X = (1/Scale)*X; % Scale X to 1
f = 0.0001;      % Fraction of Random Noise of max X
NR = randn(size(X));
sigma = f*max(max(X));

```

```

R = sigma*sigma;
N = NR*sigma;
X = X + N;          % Add random noise
if graph_gen == 1
    clg
    mesh(w,pH,X)
    xlabel('wavelength')
    ylabel('pH')
    zlabel('Absorbance')
    clear NR Scale
    pause(1)
end
end % data_set == 1

if data_set==2
    t = 1:2:70;          % t = Time
    tmax = [25 35 47];    % m = retention times (#m = #components)
    st = 7.5;
    C = gausssig(t,tmax,st);    % C = concentration matrix
    C = C*diag([1 1 1]);    % relative concentrations
    if graph_gen == 1
        clg
        plot(t,C)
        xlabel('time')
        ylabel('Concentration')
    % print
    pause
    end

    w = 204:8:300;    %4    % w = Wavelengths
    wmax = [230 250 270];    % m = Maximum wavelengths (#w = #
components)
    wt=20;
    S = gausssig(w,wmax,wt);    % S = spectra
    S = diag([1])*S;
    if graph_gen == 1
        clg
        plot(w,S)
        xlabel('wavelength')
        ylabel('Absorbance')
        pause
    end

    X = C*S;          % Data matrix

```

```

Scale = max(max(X)); % Maximum in X
X = (1/Scale)*X; % Scale X to 1
f = 0.0001; % Fraction of Random Noise of max X
NR = randn(size(X));
sigma = f*max(max(X));
R = sigma*sigma;
N = NR*sigma;
X = X + N; % Add random noise
if graph_gen == 1
    clg
    mesh(w,t,X)
    xlabel('wavelength')
    ylabel('time')
    zlabel('Absorbance')
    clear NR Scale
    pause
end
end % data_set == 2

if data_set == 3
    load meth.dat
    meth=meth';
    X=[meth(1:16,:); meth(19,:); meth(18,:); meth(17,:);];
    X=X(:,81:8:206);
    if graph_gen == 1
        clg
        mesh(X)
        pause
    end
end % data_set == 3

if data_set == 4
    load specdat2
    X=specdat2;
    X=X(7:31,1:2:139);

    if graph_gen == 1
        clg
        pH = 3.00:0.25:12.50;
        pH=pH(1,7:31);
        w = 266:4*2:820;
        mesh(w,pH,X)
        view([-127.5 60])
        pause
    end
end

```

```

end

if graph_gen == 1

    pKa1 = 0.2;
    pKa2 = 7.8;
    pKa3 = 9.8;
    pKa4 = 11.7;
    Ka1 = 10^(-1*pKa1);
    Ka2 = 10^(-1*pKa2);
    Ka3 = 10^(-1*pKa3);
    Ka4 = 10^(-1*pKa4);

    for i=1:length(pH)
        H = 10^(-1*pH(i));
        Denom = H*H*H*H + Ka1*H*H*H + Ka1*Ka2*H*H + Ka1*Ka2*Ka3*H +
Ka1*Ka2*Ka3*Ka4;
        C(1,i) = H*H*H*H/(Denom);
        C(2,i) = H*H*H*Ka1/(Denom);
        C(3,i) = H*H*Ka1*Ka2/(Denom);
        C(4,i) = H*Ka1*Ka2*Ka3/(Denom);
        C(5,i) = Ka1*Ka2*Ka3*Ka4/(Denom);
    end

    clg
    plot(pH,C)
    xlabel('pH')
    ylabel('Concentration')
    pause
end % graph_gen
Xerr=(1e-4)*ones(size(X));
end

if data_set == 5
    load camat.dat
    X=camat;
    %13 and 30
    xa=[X(1:4,:); X(6:13,:); X(14:19,:); X(21:33,:); X(34,:); X(36:42,:); X(46:49,:);
X(54:66,:);];
    %18
    xa=[xa(:,1:2) xa(:,4:8) xa(:,11:15) xa(:,18:19) xa(:,21) xa(:,25) xa(:,27) xa(:,30)
xa(:,31)];
    l_xa=length(xa);
    m_num=rand(l_xa,1);

```

```

[Y,la]=sort(m_num);
X=xa(la(1:30),1:17);
% save la.dat la /ascii
% save xa.dat xa /ascii
load caerr.dat
Xerr=caerr;
%Exclude Samples
xaerr=[Xerr(1:4,:); Xerr(6:13,:); Xerr(14:19,:); Xerr(21:33,:); Xerr(34,:);
Xerr(36:42,:); Xerr(46:49,:); Xerr(54:66,:);];
%Exclude Variables
xaerr=[xaerr(:,1:2) xaerr(:,4:8) xaerr(:,11:15) xaerr(:,18:19) xaerr(:,21)
xaerr(:,25) xaerr(:,27) xaerr(:,30) xaerr(:,31)];
Xerr=xaerr(la(1:30),1:17);
end

if data_set==6
load lana1.dat
X=[lana1(2:32,1:13) lana1(2:32,15:20)]; % Exclude Mould Bay
Xerr=ones(size(X));
end

if data_set==7
load osci8.dat
osci8sub=[osci8(1:21,:); osci8(23:97,:);];
X=osci8sub(1:3:96,1:2:84);
load osci8var.dat
osci8sub=[osci8var(1:21,:); osci8(23:97,:);];
Xerr=sqrt(osci8sub(1:3:96,1:2:84));
end

if data_set==8
load pcv7.dat
% 9 19 27 28 33 36 38
pcv7sub=[pcv7(1:9,:); pcv7(10:18,:); pcv7(20:26,:); pcv7(29:32,:);
pcv7(34:35,:); pcv7(37,:); pcv7(39:40,:); ];
X=pcv7sub(:,21:6:316); %6
load pcv7var.dat
% 9 19 27 28 33 36 38
pcv7sub=[pcv7var(1:9,:); pcv7var(10:18,:); pcv7var(20:26,:); pcv7var(29:32,:);
pcv7var(34:35,:); pcv7var(37,:); pcv7var(39:40,:); ];
Xerr=sqrt(pcv7sub(:,21:6:316)); %6
load pcv8.dat
% 6 8 9 11 12 13 18 22 23 31 34 39
pcv8sub=[pcv8(1:5,:); pcv8(7,:); pcv8(10,:); pcv8(14:17,:); pcv8(19:21,:);

```



```

pcv8(24:30,:); pcv8(32:33,:); pcv8(35:38,:); pcv8(40:41,:);];
X=pcv8sub(:,21:6:316);
save pcv8sub.dat X /ascii
load pcv8var.dat
pcv8sub=[pcv8var(1:5,:); pcv8var(7,:); pcv8var(10,:); pcv8var(14:17,:);
pcv8var(19:21,:); pcv8var(24:30,:); pcv8var(32:33,:); pcv8var(35:38,:);
pcv8var(40:41,:);];
Xerr=sqrt(pcv8sub(:,21:6:316));
end

[m,n] = size(X)

if data_set==2
    Xerr=ones(m,n);
end

if auto == 1
    % Autoscale the data
    aX = zeros(m,n);
    aXerr = zeros(m,n);

    sum_X = sum(X');

    for j=1:n
        aX(:,j) = (X(:,j))./(sum_X');
        aXerr(:,j) = (Xerr(:,j))./(sum_X');
    end

    X = aX;
    Xerr = aXerr;

end % auto

num_recombine=ceil(recomb*sel_repr);

if nstr<sel_repr
    disp('You dunderhead! You cannot selectively reproduce more strings than
you have')
    pause
end

if rem(sel_repr,2) == 1
    disp('You dunderhead! You need a even number of strings to selectively
reproduce')

```

```

    pause
end

% Quality Function Optimum Solution

if obj_fun == 1

    eigvalues_ordered=zeros(m,NumPC);
    var_data=zeros(m,1);
    opt_obj_function=zeros(m,1);

    for i=1:m-(slice-1);
        xsub=X(i:slice-1+i,:);
        [vsub,dsub]=eig(xsub*xsub);
        for i=1:NumPC
            eigvalues_ordered(i,j)=dsub(n-j+1,n-j+1);
        end
    end

    term=cumsum(eigvalues_ordered);
    opt_obj_function=term(:,1);
    var_data;

    if graph_gen ==1
        clg
        plot(eigvalues_ordered)
        title('eigvalues')
        pause
    end

    differ_opt=zeros(m-1,1);
    for i=1:m-1
        differ_opt(i,1)=opt_obj_function(i+1,1)-opt_obj_function(i,1);
    end

end % if obj_fun == 1

if obj_fun == 2

    %Objective Function

    var_m_data=zeros(m+(slice-2),1);
    opt_obj_function=zeros(m+(slice-2),1);

```

```

xnew=[X; X(1:slice-1,:)];

for i=2:(slice-1)+(m-1)
    if i<=slice idxi=1; end
    if i>slice idxi=i-(slice-1); end
    if i<=m idxf=i; end
    if i>n1 idxf=m; end
    xsub=xnew(idxi:idxf,:);
    size(xsub)
    var_nx=(std(xsub)); %((idx)/idx+1).^2
    var_m_data(i,1)=sum(var_nx);
    %idxi
    %idxf
    %pause
end

opt_obj_function(:,1)=cumsum(var_m_data(:,1));

if graph_gen ==1
    clg
    plot(var_m_data)
    title('variance')
    pause
end
end % if obj_fun ==2

if obj_fun == 3

    %Objective Function

    chi_m_data=zeros(m+(slice-2),1);
    opt_obj_function=zeros(m+(slice-2),1);

    xnew=[X; X(1:slice-1,:)];
    xnewerr=[Xerr; Xerr(1:slice-1,:)];

    for i=2:(slice-1)+(m-1)
        if i<=slice idxi=1; end
        if i>slice idxi=i-(slice-1); end
        if i<=m idxf=i; end
        if i>m idxf=m; end
        xsub=xnew(idxi:idxf,:);
        size(xsub)
        xsuberr=xnewerr(idxi:idxf,:);
    end
end

```

```

var_nx=(std(xsub)); %((idx)/idx+1).^2
var_meas = mean(xsuberr.^2);
chi_m_data(i,1)=sum(var_nx./var_meas);
%idxi
%idxf
%pause
end

opt_obj_function(:,1)=cumsum(chi_m_data(:,1));

if graph_gen ==1
    clg
    plot(chi_m_data)
    title('chi sq')
    pause(2)
end
end % if obj_fun ==3

tic

t=2;

pop = zeros(nstr,m);
popopt = zeros(max_gen,m);
Fitness = zeros(nstr,1);
pairs = zeros(sel_repr,m);
minpred = zeros(max_gen,1);

% Generate Initial Strings
rand('seed',0)
for i=1:nstr
    pop(i,:)=randperm(m);
end

if obj_fun == 1
    % Testing Initial Strings
    eigvalues=zeros(m,nstr*NumPC);
    obj_function=zeros(m,nstr);

    for k=1:nstr
        xnew=X(pop(k,:),:);
        for i=1:m
            xsub=xnew(1:i,:);
            [vsub,dsub]=eig(xsub'*xsub);

```

```

    for j=1:NumPC
        eigvalues(i,(k-1)*NumPC+j)=dsub(n-j+1,n-j+1);
    end
end

term=cumsum(eigvalues(:,(k-1)*NumPC+1:(k-1)*NumPC+NumPC));
obj_function(:,k)=term(:,1)+term(:,2);
Fitness(k,1)=obj_function(m,k);
end
end % if obj_fun ==1

if obj_fun == 2

    %Objective Function

    obj_function=zeros(m+(slice-2),nstr);
    var_m_data=zeros(m+(slice-2),nstr);

    for k=1:nstr
        xnew=[X(pop(k,:),:); X(pop(k,1:slice-1),:);];
        for i=2:(slice-1)+(m-1)
            if i<=slice idxi=1; end
            if i>slice idxi=i-(slice-1); end
            if i<=m idxf=i; end
            if i>m idxf=m; end
            xsub=xnew(idxi:idxf,:);
            %size(xsub)
            var_nx=(std(xsub)); %((idx)/idx+1).^2
            var_m_data(i,k)=sum(var_nx);
            %idxi
            %idxf
            %pause
        end
        obj_function(:,k)=cumsum(var_m_data(:,k));
        Fitness(k,1) = obj_function(m+(slice-2),k);
    end % k

end % if obj_fun ==2

if obj_fun == 3

    %Objective Function

    chi_m_data=zeros(m+(slice-2),nstr);

```

```

obj_function=zeros(m+(slice-2),nstr);

for k=1:nstr
    xnew=[X(pop(k,:),:); X(pop(k,1:slice-1),:);];
    xnewerr=[Xerr(pop(k,:),:); Xerr(pop(k,1:slice-1),:);];
    for i=2:(slice-1)+(m-1)
        if i<=slice idxi=1; end
        if i>slice idxi=i-(slice-1); end
        if i<=m idxf=i; end
        if i>m idxf=m; end
        xsub=xnew(idxi:idxf,:);
        %size(xsub)
        xsuberr=xnewerr(idxi:idxf,:);
        var_nx=(std(xsub)); %((idx)/idx+1).^2
        var_m_data(i,k)=sum(var_nx);
        var_meas = mean(xsuberr.^2);
        chi_m_data(i,k)=sum(var_nx./var_meas);
        %pause
    end

    obj_function(:,k)=cumsum(chi_m_data(:,k));
    Fitness(k,1) = obj_function(m+(slice-2),k);
end % k

end % if obj_fun ==3

% Sorting Strings
[sorted,l] = sort(Fitness);
pop = pop(l,:);
Fitness(:,1) = Fitness(l);

if obj_fun ==1
    best_obj = obj_function(:,l(1));
    differ=zeros(m-1,1);
    for i=1:m-1
        differ(i,1)=best_obj(i+1,1)-best_obj(i,1);
    end
end % if obj_fun ==1

clear sorted l

popopt(1,:)=pop(1,:);
minpred(1)=Fitness(1);

```

```

if graph_gen==1
    clg
    % Display results
    plot(1:1:max_gen,minpred,'og')
    str=sprintf('%g',pop(1,:));
    title(str)
    xlabel('Generation')
    ylabel('Fitness')
    axis([1 max_gen min(minpred(1))*0.9 max(minpred)*1.1])
    pause(1)
end

while t<=max_gen
    t
    if t<=20 num_mutation=ceil(mutation(1)*sel_repr); end
    if t>20 & t<=30 num_mutation=ceil(mutation(2)*sel_repr); end
    if t>30 & t<=40 num_mutation=ceil(mutation(3)*sel_repr); end
    if t>40 & t<=50 num_mutation=ceil(mutation(4)*sel_repr); end
    if t>50 num_mutation=ceil(mutation(5)*sel_repr); end

    % Recombination
    % Permutation operator P_OX2 as described in Carlos' thesis p.91

    Randpair = rand(sel_repr,1);
    [Randpairsort,lrandpair] = sort(Randpair);
    pairs = pop(lrandpair,:);
    Round05=round(rand(num_recombine,m));

    for i=1:num_recombine/2
        m_recomb=rand(sel_repr,1);
        [rr,l]=sort(m_recomb);
        pairs=pairs(l,:);
        clear l

        for j=1:2
            [l,J] = find(Round05(i,:));

            for k=1:length(J)
                temp(k)=pairs(2*(i-1)+j,J(k));
            end

            ind=zeros(length(J),1);

            if j==1

```

```

for k=1:length(J)
    for l=1:m
        if temp(k)==pairs(2*(i-1)+2,l)
            ind(k,1)=l;
        end
    end
end
temp1=zeros(1,m);
if length(J)~=0
    [lind,Jlind]=sort(ind(:,1));
    for k=1:length(J)
        temp1(1,J(k))=pairs(2*(i-1)+2,lind(k));
    end
end
[Inot,Jnot]=find(Round05(i,:)==1);
for k=1:length(Jnot)
    temp1(1,Jnot(k))=pairs(2*(i-1)+1,Jnot(k));
end
clear l J lind Jlind Inot Jnot ind temp
end

if j==2
    for k=1:length(J)
        for l=1:m
            if temp(k)==pairs(2*(i-1)+1,l)
                ind(k,1)=l;
            end
        end
    end
    temp2=zeros(1,m);
    if length(J)~=0
        [lind,Jlind]=sort(ind(:,1));
        for k=1:length(J)
            temp2(1,J(k))=pairs(2*(i-1)+1,lind(k));
        end
    end
    [Inot,Jnot]=find(Round05(i,:)==1);
    for k=1:length(Jnot)
        temp2(1,Jnot(k))=pairs(2*(i-1)+2,Jnot(k));
    end
    clear l J lind Jlind Inot Jnot ind temp
end
end
end

```



```

    pairs(2*(i-1)+1,:)=temp1;
    pairs(2*(i-1)+2,:)=temp2;
    clear temp1 temp2
end

% Mutation
for i=1:num_mutation
    rn_mutate=rand(sel_repr,1);
    [rm,l]=sort(rn_mutate);
    pairs=pairs(l,:);
    clear l
    rn_mutate=rand(m,1);
    [rm,l]=sort(rn_mutate);
    temp=pairs(1,:);
    pairs(1,l(1))=temp(l(2));
    pairs(1,l(2))=temp(l(1));
    clear l
end

% Quality Function

pop = [pop(1:nstr-sel_repr,:); pairs];
Fitness = zeros(nstr,1);

if obj_fun == 1
    % Testing Initial Strings
    eigvalues=zeros(m,nstr*NumPC);
    obj_function=zeros(m,nstr);

    for k=1:nstr
        xnew=X(pop(k,:),:);
        for i=1:m
            xsub=xnew(1:i,:);
            [vsub,dsub]=eig(xsub'*xsub);
            for j=1:NumPC
                eigvalues(i,(k-1)*NumPC+j)=dsub(n-j+1,n-j+1);
            end
        end
    end

    term=cumsum(eigvalues(:,(k-1)*NumPC+1:(k-1)*NumPC+NumPC));
    obj_function(:,k)=term(:,1)+term(:,2);
    Fitness(k,1)=obj_function(m,k);
end
end % if obj_fun ==1

```

```

if obj_fun == 2

    %Objective Function

    obj_function=zeros(m+(slice-2),nstr);
    var_m_data=zeros(m+(slice-2),nstr);

    for k=1:nstr
        xnew=[X(pop(k,:),:); X(pop(k,1:slice-1),:);];
        for i=2:(slice-1)+(m-1)
            if i<=slice idxi=1; end
            if i>slice idxi=i-(slice-1); end
            if i<=m idxf=i; end
            if i>m idxf=m; end
            xsub=xnew(idxi:idxf,:);
            %size(xsub)
            var_nx=(std(xsub)); %((idx)/idx+1).^2
            var_m_data(i,k)=sum(var_nx);
            %idxi
            %idxf
            %pause
        end
        obj_function(:,k)=cumsum(var_m_data(:,k));
        Fitness(k,1) = obj_function(m+(slice-2),k);
    end % k

end % if obj_fun ==2

if obj_fun == 3
    %Objective Function

    chi_m_data=zeros(m+(slice-2),nstr);
    obj_function=zeros(m+(slice-2),nstr);

    for k=1:nstr
        xnew=[X(pop(k,:),:); X(pop(k,1:slice-1),:);];
        xnewerr=[Xerr(pop(k,:),:); Xerr(pop(k,1:slice-1),:);];
        for i=2:(slice-1)+(m-1)
            if i<=slice idxi=1; end
            if i>slice idxi=i-(slice-1); end
            if i<=m idxf=i; end
            if i>m idxf=m; end
            xsub=xnew(idxi:idxf,:);
            %size(xsub)

```

```

        xsuberr=xnewerr(idxi:idxf,:);
        var_nx=(std(xsub)); %((idx)/idx+1).^2
        var_m_data(i,k)=sum(var_nx);
        var_meas = mean(xsuberr.^2);
        chi_m_data(i,k)=sum(var_nx./var_meas);
        %pause
    end

    obj_function(:,k)=cumsum(chi_m_data(:,k));
    Fitness(k,1) = obj_function(m+(slice-2),k);
end % k

end % if obj_fun ==3

% Sorting Strings
[sorted,l] = sort(Fitness);
pop = pop(l,:);
Fitness(:,1) = Fitness(l);

if obj_fun ==1
    best_obj = obj_function(:,l(1));
    differ=zeros(m-1,1);
    for i=1:m-1
        differ(i,1)=best_obj(i+1,1)-best_obj(i,1);
    end
end % if obj_fun ==1

clear sorted l

popopt(t,:)=pop(1,:);
minpred(t)=Fitness(1);

if graph_gen==1
    % Display results
    plot(1:1:max_gen,minpred,'og')
    str=sprintf('%g',pop(1,:));
    title(str)
    xlabel('Generation')
    ylabel('Fitness')
    axis([1 max_gen min(minpred(1:t))*0.9 max(minpred)*1.1])
    pause(0.1)
end

t=t+1;

```

end % while t

total = toc