UNIVERSITY
of
GLASGOW

Plasma Diagnostic Signal Analysis: a Bayesian based Genetic Algorithm approach

by

Alexander Paul Millar, MPhys.

Thesis
submitted to the
University of Glasgow
for the degree of
Ph.D.

Astronomy and Astrophysics Group
Department of Physics and Astronomy
University of Glasgow
Glasgow, G12 8QQ.

Submitted
December 2000

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Contents

Acknowledgements viii

Summary xi

1 Introduction 1

1.1 Motivation 1

1.2 Plasmas 2

1.3 Thomson scattering 4

1.4 Optimisation 5

1.5 Overview of thesis 7

2 Properties of a Genetic Algorithm 8

2.1 Introduction 8

2.2 Global convergence and Schemata 19

2.3 Mutation rate 20

2.4 Core functionality of the ELGAR library 26

2.5 Genetic Algorithms and NP-completeness 29

2.6 Examples of simple problems solved by a GA 30

2.6.1 Charges on a disc 31

2.6.2 Fitting a sinusoidal signal 33

2.7 Summary 36

3 Bayesian Statistics in signal analysis **37**

3.1 Bayesian Methodology 37

3.2 Simple example 38

3.2.1 'Frequentist' approach 39

3.2.2 Bayesian approach 40

3.3 Prior information in signal analysis 41

3.4 Example of prior information 42

3.5 Errors 43

3.6 Summary 44

4 Case study: Thomson scattering **45**

4.1 Characteristics of Thomson Scattering 46

4.2 Experimental setup at COMPASS-D 51

4.3 Data acquisition and triggering 53

4.4 The noise 61

4.4.1 High frequency region 63

4.4.2 High frequency spikes 63

4.4.3 Low frequency region 64

4.4.4 Noise model 65

4.5 Estimating signal-to-noise ratio 65

4.5.1 Estimating the signal level 65

4.5.2 Estimating the noise level 67

4.6 Obtaining reference signals 70

4.7 Analytic technique of fitting the signals 72

4.8 Quantifying the errors in fits 75

4.8.1 The effects of non Gaussian noise on uncertainty in amplitudes . 76

4.9 Calculating the temperature 78

4.10 Uncertainty in the temperature 79

4.11 Sample results 80

5 Two-temperature Thomson scattering diagnostic analysis 82

5.1 Two-temperature scattering characteristics 82

5.2 Channel responses 84

5.2.1 Response curve graphs 84

5.2.2 Recovering the single-temperature distribution function as limiting case 84

5.3 Normalised outputs 88

5.3.1 Single-temperature distributions 89

5.3.2 Two-temperature considerations 92

5.4 Iso-response curves 94

5.4.1 Theoretical treatment 95

5.4.2 Figures 95

5.5 Data from COMPASS-D 97

5.6 Converting measured responses into iso-response curves 99

5.7 Errors 99

5.8 Conclusions 100

6 Conclusions and future work 102

6.1 Summarising Genetic Algorithms 102

6.2 Genetic Algorithm Signal Analysis 103

6.3 COMPASS-D experiment 103

6.4 Future work 104

A Code listing **107**

A.1 GenePool structure 107

A.2 The Entity structure 112

A.3 Thomson scattering specific code 112

B Thomson scattering analysis scripting language **127**

B.1 Backus-Naur Form 128

B.2 Comments 133

B.3 Variable substitution 133

B.4 Assignments 134

B.5 Aliases 134

B.6 Data storage 135

B.7 Commands 136

B.7.1 Load 136

B.7.2 Ditch 137

B.7.3 Go 138

B.7.4 For 141

B.7.5 Next 142

B.7.6 Array 142

B.7.7 System 143

B.7.8 Echo 143

B.8 Worked example 144

List of Figures

1.1 Diagrammatical representation of one-dimensional optimisation. 5

1.2 Illustration of multidimensional optimisation bracket failing. 6

2.1 The DNA base-pairs. 11

2.2 Cell structure hierarchy. 12

2.3 Codons within mRNA. 13

2.4 Example representation of a GA. 14

2.5 Example of 1-point cross-over. 15

2.6 Steps in progressing from one generation to the next. 16

2.7 Convergence rate for 8 gene problem. 21

2.8 Convergence rate for 12 gene problem. 22

2.9 Convergence rate for 16 gene problem. 22

2.10 Gaussian fit to covergence rate when 62 mutations occur per generation. 23

2.11 Gaussian fit to convergence rate when 63 mutations occur per generation. 23

2.12 Gaussian fit to convergence rate when 64 mutations occur per generation. 24

2.13 Summary of convergence rates. 25

2.14 Comparison between observed best and predicted mutation rate. 25

2.15 A sample of GAME's output. 28

2.16 Minimum energy configurations for Charges on a disc problem. 32

2.17 Contour plots of χ^2 for noiseless sinusoidal data. 35

2.18 Contour plots of χ^2 for noisy sinusoidal data. 35

2.19 Plot of original and recovered sinusoidal signals. 36

3.1 The five platonic solids. 39

4.1 Overview of the scattering geometry in Thomson scattering. 47

4.2 Theoretical Thomson spectrum illustrating pole in correction factor. . . 50

4.3 Theoretical spectra of Thomson scattered light for different electron temperatures. 50

4.4 Schematics of the Thomson Scattering Diagnostics at COMPASS-D. . . 52

4.5 The construction of a polychromator. 52

4.6 Typical spectral response of polychromator filters. 53

4.7 Theoretical channel output as a function of electron temperature. 54

4.8 Example data with best fit Gaussian. 55

4.9 Statistical distribution of the laser's temporal extent. 56

4.10 Statistical distribution of laser's intra-segment delay. 57

4.11 Correlation between offset in channels 1 and 2. 58

4.12 Correlation between offset in channels 1 and 2, separate species are indicated. 59

4.13 Statistical distribution of offsets relative to the mean. 59

4.14 Statistical distribution of offsets relative to the mean, separate species are indicated. 60

4.15 Statistical distribution of offsets relative to the mean for 'species 1' only, split by channel number. 60

4.16 Statistical distribution of offsets relative to the mean for 'species 1' date with Gaussian best fit. 61

4.17 Typical medium to low signal-to-noise ratio data. 62

4.18 Power Density Spectrum of the noise. 63

4.19 Comparison between two different methods of estimating signal level. . . 67

4.20 Comparison of various methods of estimating the noise level for high S/N level signals. 69

4.21 Comparison of the bias of each each estimate of the noise level 69

4.22 An example reference signal. 73

4.23 Comparison of the performance of Genetic Algorithms against a simple Linear Least Squares method. 74

4.24 Correlation between error estimates of low-frequency pass filtered noise. 77

4.25 Theoretical spectral density function 78

4.26 Recovered electron temperature distribution. 81

5.1 Response of the three channels to two-temperature plasma. Density ratio is 50:50. 85

5.2 Response of the three channels to two-temperature plasma. Density ratio is 65:35. 86

5.3 Response of the three channels to two-temperature plasma. Density ratio is 80:20. 87

5.4 Representation of mapping two-temperature parameters to observations. Also illustrates iso-response curves and normalisation. 88

5.5 Normalised channel response for single-temperature plasma. 90

5.6 Iso-temperature curves for normalised channel responses. 92

5.7 Graph relating two-temperature normalised responses to single-temperature curve. 93

5.8 Graph showing degeneracy in normalised channel responses for two-temperature plasma. 94

5.9 Iso-response curves with varying initial T_2. 96

5.10 Iso-response curves with varying initial α. 97

5.11 Normalised channel responses from COMPASS-D data. 98

5.12 Representation of error estimate for temperature extrema. 100

Acknowledgements

After spending nearly four years studying and working at the Astronomy and Astrophysics group, I am left with two striking impressions: that there is an incredible breadth of expertise within the group and that people are always willing to discuss ideas. This thesis would not have be possible without this fertile environment. I try here to express my gratitude to the many people without whom this thesis would not be possible.

Declan Diver was the ideal supervisor. He has an amazing understanding of the physics of plasmas and no matter how busy he was, he was always accessible for a quick chat about how things were going. His faith that I was going to get this thesis in on time was reassuring and he's a formidable proof reader!

As head of the astronomy and astrophysics group, Astronomer Royal for Scotland and due to his involvement with many public out-reach of science projects, John Brown was exceedingly busy; yet as second supervisor he offered support and constructive criticism on progress of this thesis. Both were much appreciated.

The many hours spend in conversation with Richard Barrett about various aspects of this thesis are difficult to tally. He is both open in discussion and has a deep understanding of mathematics. Indeed, I feel somewhat guilty for taking up so much of his time! He is a esteemed college and valued friend.

Darren McDonnald was another tireless source of ideas and help in the research of this thesis. The discussions we held together about various aspects of this research, often at unusual times, resulted in a substantial section of the work, both presented here and in a publication.

Martin Hendry was of enormous help with matters statistical, especially Bayesian related work. Despite working in an unrelated field, he was willing to take time to explain the answers to the questions I had. Thanks Martin!

Many thanks are due to Norman Gray who, as one of his many hats, is the resident TEXpert. His help in persuading TEX to do what I wanted rather than what I was telling it to do resulted in the clean and accessible typesetting of this thesis.

Thanks are also due to Graeme Stewart has kept the computer system running with unprecedented stability. Without this, the task of researching and presenting this thesis would be a considerably harder task.

Hugh Potts was a great encouragement as the thesis writing experience was fresh in his mind. I only hope my thesis process runs more smoothly than his did!

Both Neal Wade and Richard Barrett were great office-mates who never once complained about my choice of music. Literally a winning combination as our office won the sunflower competition.

As quite often is the case, there is someone who works quietly behind the scenes, but who has a pivotal role within the group. That person is Daphne Davison. I owe her a debt of gratitude for all her help over the years.

I must also thank the rest of the astro group: Guillian, Helen, Gail, Stephane, Esther, Lyndsay, Lida, Eve, Aidan, Chris, Graham, Paul and Doug. You have all been wonderful friends. I have really enjoyed taking tutorial and undertaking planetariums shows together as well as the numerious parties and trips to the Rubi.

I'd also like to thank Sara Hunter for her support and encouragement and her time in proof reading this thesis. Many, many renegade commas were displaced to their correct position thanks to her diligence.

And finally a big thanks to my family, especially my Mum, Dad and my brother Stephen. Quite literally, without their help and support over the years none of this would be possible.

To Scott and Margaret Millar
for their love, support and
encouragement.

Summary

Plasma physics is rich in phenomena, occurrences and applications: many instabilities exist due to the relatively long-range Coloumb forces that mediate constituent particle dynamics, most of the visible universe is in the plasma state, and plasmas have been used from lighting to computer chip manufacturing to (attempted) fusion power generation.

Within plasma fusion research, Thomson scattering is a commonly used diagnostic. It allows the temperature and density of the plasma electrons to be measured without distorting the plasma. However, the scattering cross-section is small. Thomson scattering signals can be difficult to detect against the background emission of the plasma.

In this thesis, the Thomson scattering diagnostic data from the COMPASS-D experiment is analysed. Several aspects of the diagnostic are presented along with detailed explanation of the inference procedure for determining the plasma's electron temperature.

This temperature analysis was achieved by utilising a Bayesian inference model that allowed prior information about likely values to be systematically included. This prior information was found to remove the degeneracy present due to the low signal-to-noise ratio of the data.

A genetic algorithm (GA) library, called ELGAR, was developed and used to solve the minimisation problem resulting from the Bayesian inference. The GA proved to be a reliable method of solving such problems. ELGAR was also used to investigate certain characteristics of the GA such as optimal choice of key parameters. These were found to be in disagreement with theoretical results but the difference was explained by the different mode of operation of ELGAR.

The Thomson scattering analysis was extended to include two-temperature considerations. The set of observations consistent with an n-temperature distribution function was found to be bounded by a curve. Some data from the COMPASS-D experiment lay

outside this boundary, but was bounded by a similar curve. This suggested that some systematic error had occurred. Some explanations of possible causes of this bias were suggested.

Constraints were found for interpretations of any observations. These indicated that, for some observations, a set of temperatures is unavailable (as either to hotter or colder component) for a distribution function which is consistent with that observation. For certain observations, the least-squares temperature estimate is contained within the set of impossible temperatures. This indicates that the presence of a hotter species of electrons can bias the observations towards higher temperatures.

The thesis concludes with a summary and a discussion of possible future work.

Chapter 1

Introduction

This chapter gives an introduction to the various elements that are combined within the thesis proper. The topics are covered in greater depth in later chapters but sufficient information is presented in this chapter to appreciate the overall structure of the research programme.

The first section of this chapter discusses diagnostics in general and presents the motivation for Bayesian based analysis. Section two briefly introduces plasmas stating the relevance of plasma research. Section three discusses Thomson scattering, although a more detailed description is presented in §4.1. Section four discusses optimisation and some of the problems within this field. The final section brings these concepts together to form the framework that the following chapters will build on.

1.1 Motivation

It is rare that an experiment is conducted with no knowledge of possible outcomes. Often experiments on some physical process are conducted in environments in which that physical process is carefully controlled. The experimentalist is trying to tease out the difference between two or more different theorems. Therefore, it is important that experiments and the analysis of results are conducted without prejudice from any information known to the investigator, such as 'likely outcomes' derived from previous results.

Physical processes can also be used to measure underlying quantities such as pressure, temperature, velocity, etc. Generally, when the physical process is used to diagnose some

parameter it is because the complexity of the experiment prevents precise control of that parameter. Often, this uncertainty arises from the nature of the experiment itself or from some uncertainty in calibration.

There is a subtle distinction between experimental investigation of some phenomenon and using that phenomenon as a diagnostic of some physical property. In the former the rigors of scientific investigation require an unbiased measurement of the phenomenon, as stated above. For the diagnostic case, however, we require the best estimate of the physical property. This estimate may include previous results or other 'external' knowledge. For example, if we are in the unlikely position of knowing, *a priori*, the exact state of the physical property the diagnostic is set to measure, then the best estimate would be to ignore the measurements (which would be subject to experimental errors) and use our infallible information. Likewise, if we knew nothing about likely outcomes, then our best estimate of the physical property would be based only on the observed phenomenon.

In general, some information is known *a priori* but usually insufficient to adequately constrain the required physical property of the system. Bayesian inference provides a systematic framework within which this knowledge can be introduced. In the language of Bayes' theorem, this knowledge represents 'prior information'. The inclusion of such prior information biases the inference towards 'more likely' at the expense of 'less likely' ones.

1.2 Plasmas

The archetypal plasma is an ionised gas. Instead of the usual gas-like short range collisional interactions, the charged particles move in response to the local electric and magnetic fields. The local electric field only arises from Maxwell's equations; the plasma will move to oppose externally applied electric fields. The local magnetic field arises from both Maxwell's equations and from any externally applied field. Thus any part of the plasma responds to all activity elsewhere in the plasma.

Plasma physics encompasses the study of both naturally occurring and artificially created plasmas. Almost all of the mass of the visible universe, from stars to the sparse interstellar medium, is in a plasma state. Laboratory plasmas are used in a diverse range of manufacturing processes from computer chip fabrication to coating metalised plastic

crisp packets. The study of laboratory plasmas also includes the current attempts to achieve controllable nuclear fusion as the basis for power generation.

The simplest model of plasma dynamics considers a single particle which experiences some local electric and magnetic field. This model is sufficient to explain dynamics for sufficiently short time-scales that the particle does not interact strongly with other particles. Key plasma phenomena are explained such as the gyro-rotation of the particles. This is the rotation of an charged particle in a magnetic field, with charge e and mass m, about the magnetic field lines, of magnitude B, with cyclotron frequency $\omega_c = eB/m$ and Larmor radius $r_L = v_\perp/\omega_c$, where $v_\perp$ is the component of velocity perpendicular to the magnetic field lines.

Although this simple model reveals a wealth of observed phenomena it fails to account for the inter-particle interactions. A full treatment would require accounting for six degrees of freedom (three spatial and three velocity) per particle. A typical tokamak plasma density is 10^{19} m^{-3} rendering the exact approach untenable. Since the exact method is unapproachable an approximation must be made. One simple approximation is to describe the plasma by a statistical distribution function, f, where the value of $f(\mathbf{r}, \mathbf{u})$ at some location $\mathbf{r}$ describes the number of particles in the infinitesimal volume element $\mathrm{d}^3\mathbf{r}$ and some infinitesimal velocity element $\mathrm{d}^3\mathbf{u}$.

In common with the kinetic theory of gases, the evolution of the distribution function is given by the Boltzmann equation:

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \frac{\partial f}{\partial \mathbf{r}} + \mathbf{a} \cdot \frac{\partial f}{\partial \mathbf{u}} = \left(\frac{\partial f}{\partial t}\right)_c \tag{1.1}$$

where the left hand side is the convective derivative of f and the right hand side discusses the effect of collisions on the distribution function.

To form a more tractable approximation the acceleration is assumed to be from any local electric and magnetic fields and collisions are ignored. Ignoring collisions is justified by the long-range nature of the electric and magnetic fields. These allow particles to 'miss each other' without strong distortions in f. Such fields are referred to as the self-consistent fields and contain the long-range interaction effects. This form of the Boltzmann equation is known as the Vlasov equation:

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \frac{\partial f}{\partial \mathbf{r}} + \frac{q}{m}(\mathbf{E} + \mathbf{u} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{u}} = 0 \tag{1.2}$$

If the collisional term is replaced then some approximate value must be derived. If the distribution function is considered a small perturbation of a Maxwellian then the

distribution function can be expanded as a Taylor series. If the first term of the Taylor series is retained then this is the Krook collision term:

$$\left(\frac{\partial f}{\partial t}\right)_c = (f - f_0)/\tau \tag{1.3}$$

where τ is the time-scale for collisions.

A less crude approximation is the basis for the Fokker-Planck collisional term. This assumes some transitional probability function $\psi(\mathbf{u}, \Delta\mathbf{u})$ of some particle initially at velocity $\mathbf{u}$ obtaining, through a collision, an increase in velocity of $\Delta\mathbf{u}$ in time Δt. The collision term is derived by forming a Taylor series expansion of the expectation value of f at time t.

1.3 Thomson scattering

Thomson scattering is the non-quantum scattering of light off a charged particle. The assumption that the scattering can be described without quantum effects requires that the object suffers no recoil from the photon. This assumption is valid provided the energy of the incident photon ($E_{\text{photon}} = \hbar\omega$) is much smaller that the 'rest mass' energy of the scattering object (mc^2). In the case of a plasma consisting of electrons and heavier positively charged ions the scattering is almost exclusively due to the electrons. This is because, under the classical viewpoint, the scattering object vibrates in response to the electric field of the incident light. The electrons, being far lighter than a typical ion, will vibrate more easily than the ions and so are responsible for most of the Thomson scattering.

Electrons have a rest mass of approximately 511 keV. Most Thomson scattering diagnostics use optical or near optical frequency lasers as the light source due to availability of lasers in that frequency, the relative power of those lasers and, for the near infrared, the relative lack of spectral lines. Infrared wavelengths are often used as there is typically less line-emission in lower wavelengths. A typical near infrared photon energy is 1 eV so any scattered radiation will be as a result of Thomson scattering.

The measured frequency of any scattered light will depend on the electron's motion. The ensemble effect of measuring many scattered photons from a monochromatic source will be a distribution of the scattered light's frequency that will depend on the electron's velocity distribution. By assuming a Maxwellian distribution of electron velocities and

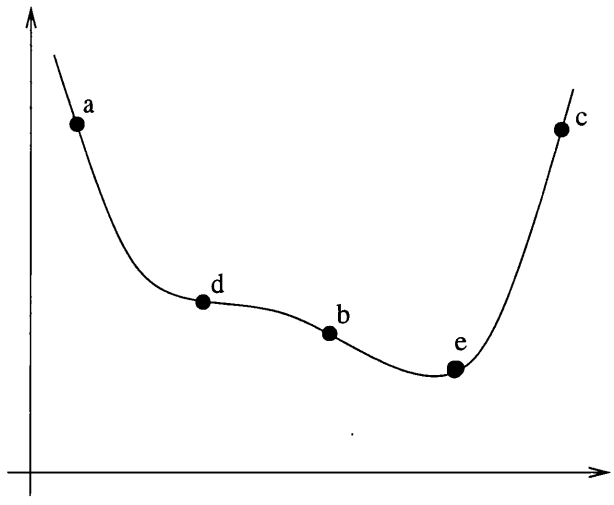


Figure 1.1: Diagrammatical representation of one-dimensional optimisation. Triplet (a, b, c) bracket the minimum. Point d then e were chosen to further constrain the minimum. This process proceeds until require accuracy is achieved.

comparing the observed distribution with predicted scattering profiles, the electrons' temperature can be derived.

1.4 Optimisation

Optimisation is the search for parameters that minimise (or equivalently maximise) a particular function. There are a number of techniques for finding a local minimum, *i.e.* a minimum in the neighbourhood of some initial point. With one-dimensional optimisation of a function it is possible to bracket the minimum with three points. For example, a minimum in the function f could be bracketed by the points a, b and c if $f(b) < f(a)$ and $f(b) < f(c)$. A further point d can then be selected, *e.g.* $d = (a + b)/2$. The minimum bracket (a,b,c) can be updated to (a,d,b) or (d,b,c) depending on whether $f(d) < f(b)$. Since the interval is always smaller, this method will always converge towards the minimum. This is shown diagrammatically in figure 1.1.

For multi-dimensional minimisation, minimising a function with more than one parameter, it is impossible to bracket a minimum. Intuitively, this is because there is always a direction along which the function could have a narrow valley but to which the bracketing points (however many) are insensitive. Figure 1.2 demonstrates an example of this problem. Instead of bracketing, multi-dimensional optimisation methods work by taking a 'good guess' and hopefully improving upon it. Because of this, multi-dimensional minimisation involves (usually many) iterations.

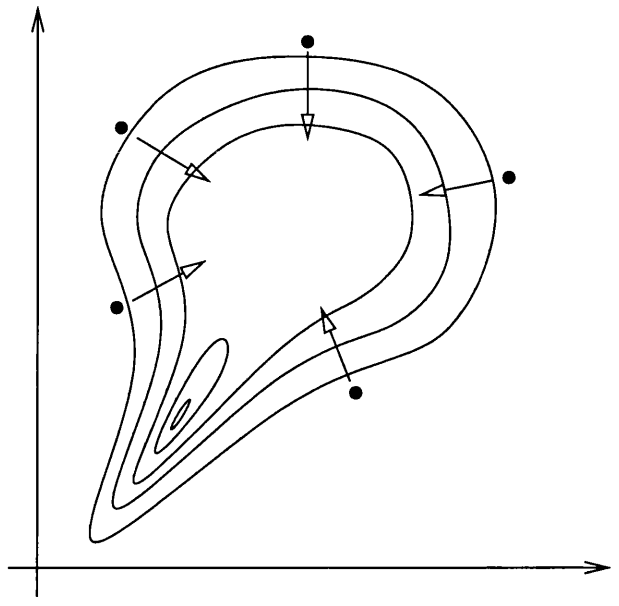


Figure 1.2: Illustration of the futility of attempting to bracket a minimum in more than one dimension. Contour lines illustrate the function's shape. Five 'bracketing' points are located on a circle. The arrows point in the direction of steepest descent, which is toward the circle's centre. However, the true extremum lies outside this circle.

As well as distinguishing between one-dimensional and multi-dimensional optimisation problems there is also a distinction between local and global optimisation. Local optimisation is finding the extremum of a function usually in the vicinity of a point. Multi-dimensional local optimisation usually starts with this point and iteratively finds more optimal points until an extremum is found.

Global optimisation is searching for the extremum for all possible parameters, for example the smallest of the local minimum. In practice, the procedure is is usually limited to some given range of parameter space. Finding the global optimum is a difficult task and guaranteeing that the discovered extremum is the global extremum is, in general, impossible. Various techniques exist for finding a reasonable estimate of the global extremum such as Simulated Annealing and Genetic Algorithms. These techniques are generally iterative in nature and not guaranteed to work in all cases but usually provide a good estimate of the vicinity of the global extremum. A local optimisation finding routine can then be employed to locate the true global extremity.

1.5 Overview of thesis

This thesis discusses the inclusion of prior information and the analytical form for such analysis. This is used to improve results from diagnostic equipment in poor signal-to-noise conditions by systematically including prior information.

Genetic Algorithms are discussed. These are used as a 'work horse' to solve the multi-dimensional non-linear minimisation problems. These problems arise due to the extra, prior information included in the analysis. Using Bayesian statistics to solve non-linear problems has been considered in other fields: Aoki et al. [1999] details using a Genetic Algorithm to cluster documents based on a Baysian measure of their similarity.

The thesis also presents analysis of data from the Thomson Scattering Diagnostic used at COMPASS-D, the UKAEA Fusion Division's test tokamak reactor. The signal-to-noise ratio for Thomson scattering makes it amenable to the Bayesian inference method.

The Thomson scattering analysis is then extended to consider results from a two-temperature distribution. Although there is insufficient data to fully constrain the degrees of freedom some constraints can be placed on the available values of parameters.

Chapter 2

Properties of a Genetic Algorithm

This chapter discusses the Genetic Algorithm and its relevance and importance in optimisation problems. Chapter 4 will discuss a specific example of an optimisation problem that uses both the algorithm discussed in this chapter and the formalisation developed in chapter 3.

The first section of this chapter gives a general introduction to Genetic Algorithms on which the following section expands giving more details on global optimisation. Section three discusses the effect of different mutation rates, an adjustable parameter of the GA, on the convergence rate. Section four details how the Genetic Algorithm was constructed and section five illustrates some standard problems that were tackled with a Genetic Algorithm. The key points are then summarised.

2.1 Introduction

Unlike a human, a computer cannot take an overview of a problem. Heuristical problems, such as spotting the global maximum[1], are generally easy for a human if the data is suitably presented; but it is difficult to describe the sequence of steps the human observer uses to derive the answer.

With vast collections of data it is impractical to require a human to sift through each dataset and locate the overall maximum. Such drudgery is to be avoided and, in

[1] Although throughout this chapter, the objective of the algorithms discussed will be to locate the maximum of a function, it should be remembered that the minimum of the function f is the maximum of the function $-f$.

any case, is prone to mistakes. For 'complex' data, where the maximum could be the subject of debate, an automated scheme would remove the possibility of unwarranted human bias. However, in such cases, a human observer can bring experience to bear and bias the answer towards more likely solutions. A method for encoding experience, in the form of 'prior information', is discussed in chapter 3; but for the moment, we shall consider the problem of locating some extremum of the supplied data without being influenced by previous results.

Many routines exist for locating the maximum of a function in the neighbourhood of some initial point in parameter space. For examples of such algorithms see chapter 10 of Press et al. [1996]. These routines are collectively referred to as 'local maximisation routines'. Perhaps the simplest local maximisation routine is 'hill climbing': an iterative algorithm where the next point is selected by stepping in the direction of steepest ascent.

Local maximisation routines have the property that they take a point in parameter space (or in the case of simplex optimisation, $n+1$ points for n dimensional parameter space) and conjure up a new point usually based on a variant of the hill climbing algorithm. This process is then repeated forming an iterative scheme that is terminated once a suitable stopping criteria is reached. These routines use different methods to decide on the new, better point. Some use information about the function's gradient whilst others rely upon only the evaluated function values. Some use a single method whereas others are hybrids and tentatively try a selection of different methods before committing to a new point and proceeding to the next iteration. However the iterative scheme proceeds one factor is common to all local maximisation routines: after finding a region where the function's first derivative vanishes they will all stop.

It is possible to construct a global maximum searching algorithm from the simpler local maximisation routines. One method of achieving this is to create a set of initial trial solutions: points in parameter space where it is hoped that for one trial solution the local maximum is the global maximum. By finding the local maximum near each trial solution and retaining the largest local maximum the global maximum may have been discovered. In many real-life problems there is no way to know *a priori* the 'smoothness' of a given solution, *i.e.* how large a volume of parameter space is considered the neighbourhood of the global maximum. For most functions this means that one of the starting points must be 'close enough' to the global maximum that the function is smooth and the maximum can be arrived upon.

One method of finding a point in an n dimensional parameter space close enough to the global maximum is to try a large number of points in different locations. These initial points can be selected randomly, sub-randomly such as a Halton or Sobel sequence (see section 7.7 of Press et al. [1996]), or from a n dimensional grid of $m \times m \times m \times \cdots = m^n$ points.

However the points are selected the multiple-point local maximum algorithm is ineffective. The probability of one randomly chosen point lying within the neighbourhood of the global maximum is $P_1 = \frac{v}{V}$ where v is the volume of the neighbourhood and V is the total volume of the parameter space under investigation. For k points, the probability of at least one lying within the neighbourhood is $P_k = 1 - \left(1 - \frac{v}{V}\right)^k \approx k\frac{r}{R}^l$ where l is the dimensionality of the parameter space and r and R are typical length-scales for the neighbourhood and the parameter space under investigation respectively. It is clear that an exponentially increasing number of points are required to maintain the same probability of finding the global maximum for increasing dimensionality of the problem.

Despite the inefficiency of the multiple-point local maximisation routine it is impossible to guarantee that a computer will find the global maximum of a function. Moreover, if the function we wish to optimise is in some sense 'expensive' to calculate (*e.g.* in terms of computer resources) then a more efficient algorithm is required. Such algorithms are often inspired by natural phenomena. One such method is simulated annealing, which gradually 'cools' towards a solution in an analogy to a metal being annealed to remove grain boundaries and achieve a minimum internal energy. For more information about simulated annealing see §10.9 of Press et al. [1996], Kirkpatrick et al. [1983], Wille and Vennik [1985b] and references therein. Another method, deriving inspiration from natural selection and evolution, is the Genetic Algorithm.

A Genetic Algorithm (GA) uses concepts borrowed from genetics and Darwin's theory of evolution. As an algorithm classification the GA is quite broad. In the most abstract sense a GA consists of a collection (tens or hundreds) of points in parameter space and a collection of operators that are applied iteratively to the collection. The effect is to create an environment in which multiple entities (the points in parameter space) vie with each other to include their 'genetic information' as part of the 'gene pool' of the next iteration.

Each entity can be considered analogous to a single creature forming part of a particular species. Therefore, the collection of such entities used by a GA at some particular

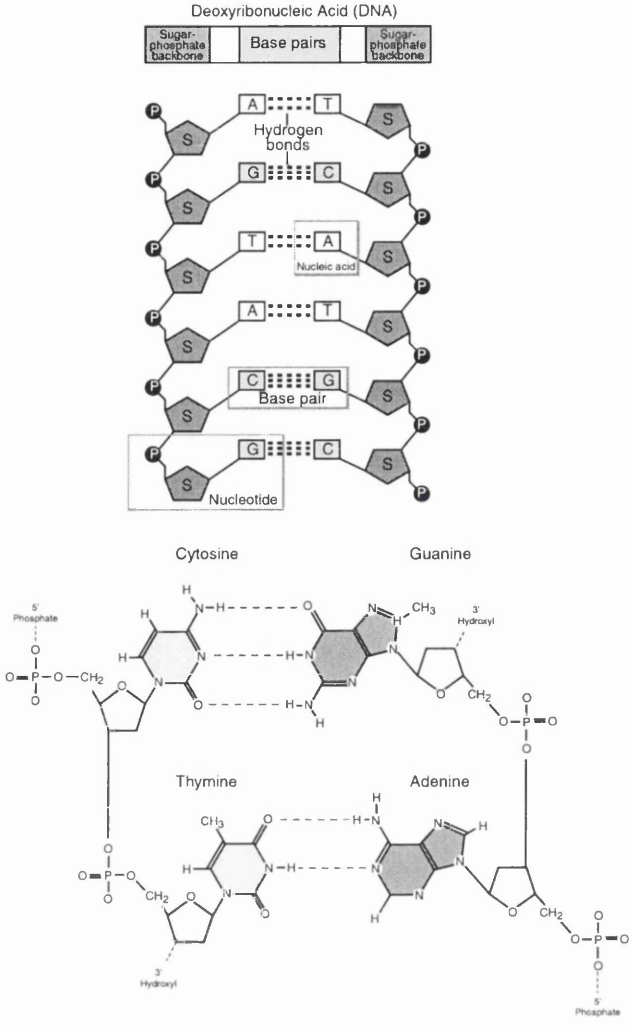


Figure 2.1: The DNA base-pairs that encode amino acids.

iteration is analogous to the bio-diversity of the species at that particular time.

In a further analogy to natural systems, the terms genotype and phenotype are used to describe the process of evaluating the entity's fitness. In biological systems, the genotype refers to the sequence of DNA base-pairs that 'code up' the entity (see figure 2.1) whilst the phenotype refers to the physical result of the coding, which includes any environmental effects. The creature's phenotype has to survive until maturity so it can propagate it's particular genetic code onto the next generation.

In humans, the complete genotype is present in most cells as 23 pairs of chromosomes. A pair of chromosomes contain the same number of DNA base-pairs, which are nearly identical. Each chromosome consists of two connected chomatids that contain the genes. A gene is the region of DNA responsible for a particular trait (*e.g.* hair colour). This hierarchy of structure is shown in figure 2.2.

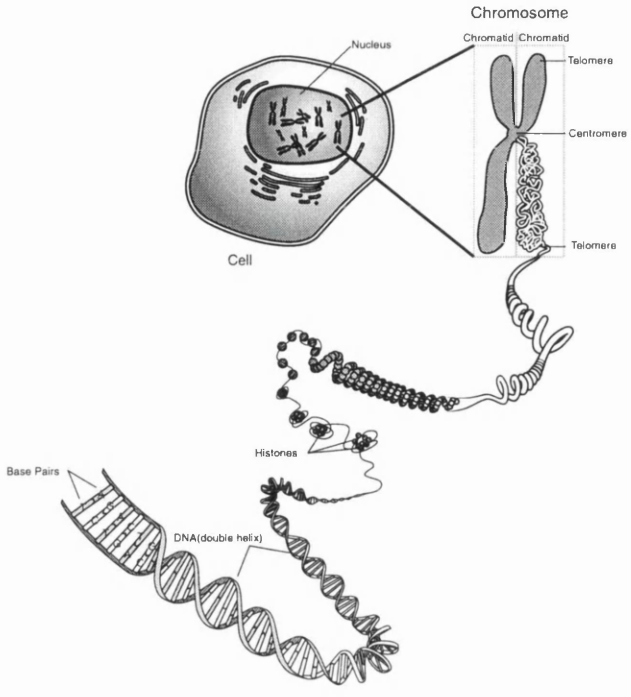


Figure 2.2: The hierarchy of structure within a cell's nucleus from chromosome to the base pairs.

A gene typically consisting of many codons which are groups of three base-pairs that code a particular amino acid. Part of the cell's manufacturing process involves copying a section of DNA to RNA (called messenger RNA or mRNA). The multiple codons in a gene specify the sequence of amino acids that join to form a particular protein that then causes the particular trait. An example set of codons is shown in figure 2.3.

The gene's location within its chromosome is referred to as the gene's locus. Some genes' loci are not fixed and can change from individual to individual. This is achieved because the vast majority of genetic material within a chromosome is insignificant. As the chromosome is 'read' (*i.e.* copied to mRNA) specific codons turn on the cell's manufacturing process enabling later codons to be 'decoded' into protein. The different expressions of the gene's trait are referred to as the allele of that gene. For example, the gene for the eye colour trait results in different proteins being produced due to different codons within the gene. Blue eye colour is an example of the eye colour gene's alleles.

The pairing of chromosomes allows complex behaviour such as dominant and recessive genes. At a particular locus, the gene from one chromosome might take precedence over the other resulting in the phenotype only developing the trait associated with the dominant gene. However, a recessive gene that is not been expressed (due to the presence of a dominant gene) can be passed on to an offspring where it can be expressed

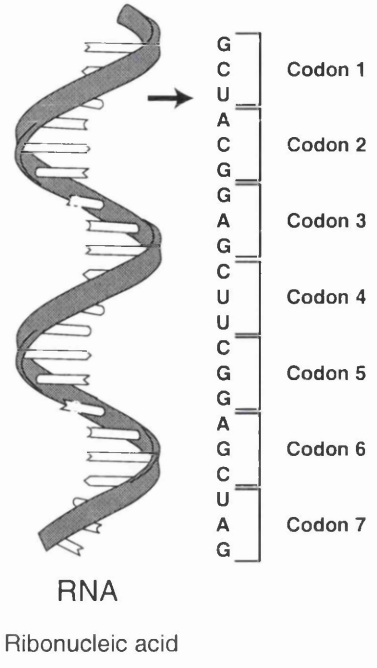


Figure 2.3: Codons within mRNA copied from DNA (with U instead of T).

if no equivalent dominant gene is present. It is worth noting that the phenotype also includes environmental effects so that two different phenotypes can derive from the same genotype. This level of indirection (phenotype is used for fitness evaluation whilst only genotype is passed on) promotes hardiness by encouraging genotypes that are flexible enough to cope with different environments.

With genetic algorithms the genotype usually consists of a fixed length string from an alphabet. The alphabet is usually a fixed set of symbols (such as binary or decimal digits) but some GAs work with a genotype consisting of floating point numbers. The 'meaning' attributed to each element (the alleles) of the genotype string is usually fixed although some work (Bethera and Nanjundiah [1997]) suggests some benefits from allowing some variation in gene alleles.

To illustrate the definitions discussed above in the context of optimisation consider the following curve-fitting problem. Data d_i is supplied with estimates of the uncertainties σ_i assuming the errors are normally distributed. The model is two independent Gaussian curves and a constant offset as shown in (2.1).

$$f_i = a_1 \exp\left[-\frac{(i-\mu_1)^2}{2w_1^2}\right] + a_2 \exp\left[-\frac{(i-\mu_2)^2}{2w_2^2}\right] + l \tag{2.1}$$

If we take $\chi^2 = \sum_i (f_i - d_i)^2/\sigma_i^2$ as our likelihood measure one method of solving this

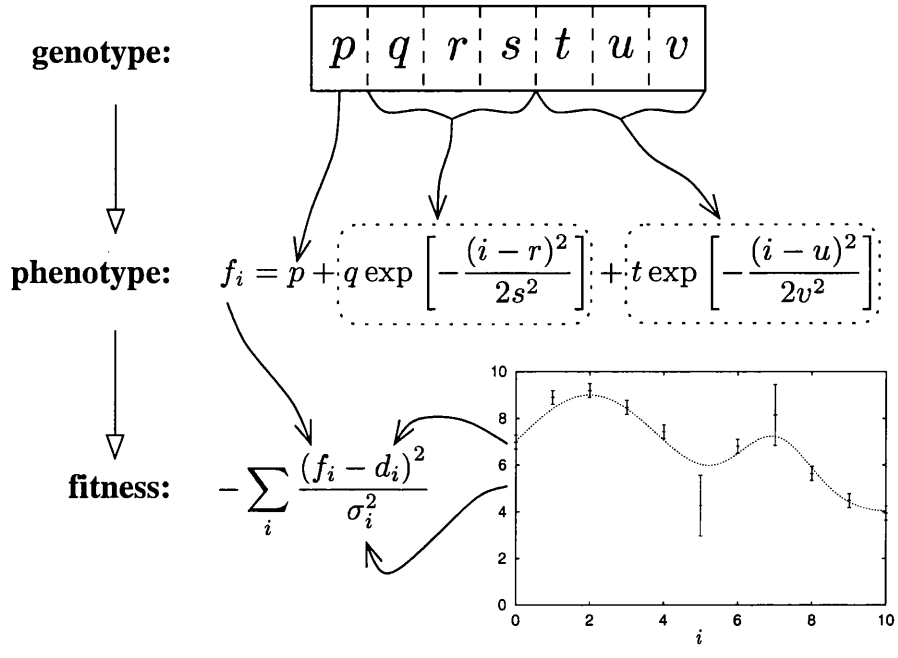


Figure 2.4: Representations and information pertaining to a single entity of a GA.

problem is to guess a set of initial values for the seven parameters $\{a_1, \mu_1, w_1; a_2, \mu_2, w_2; l\}$ and to use some multi-dimensional optimisation method to iteratively improve this solution. As stated earlier, this will work only if the initial guess is close enough to the true minimum. Figure 2.17 on page 35 demonstrates that for a similar fitting problem, even in the absence of noise, the fraction of the total parameter space 'close enough' to the global minimum can be small.

An alternative method of solving this curve fitting problem is via a GA. For this example, the GA's genotype consists of five genes: one for each parameter. Whilst the GA is running each entity in the GA's current population will have specific numbers allocated to each gene. The phenotype is the function we wish to fit, so a specific entity's phenotype is a curve given by the values of the genes that describes (to a greater or lesser extent) the data. The fitness function is taken to be $-\chi^2$. This is illustrated in figure 2.4.

The genetic operators are used to generate new entities which then form the next iteration of the algorithm. Although implementations of GAs differ in their choice of which operator to implement they will generally include selection, cross-over, mutation and some form of survival operators. The selection and cross-over operators are used to generated the required number of entities: a user supplied parameter of the algorithm.

The mutation operator is then applied to the set of child solutions. The survival operator is analogous to Darwin's survival of the fittest and ensures the algorithm 'evolves' towards the global maximum.

The selection operator finds two entities from which new entities are formed. The three common selection operators are 'unbiased', 'roulette' and 'tournament'. Unbiased is the simplest and selects two entities from the collection of available (without replacement). The roulette operator differs from unbiased in that it weights the selection towards more optimal solutions. The weighting is usually taken to be the fitness function, requiring it to be positive definite. Tournament selection works in two stages: it first randomly selects two pairs of entities then the fittest of each pair then proceeds to the breeding stage.

The cross-over operator takes two selected entities ('parents') and combines them to produce two variants ('children') so that each contain some information from one parent and some from the other. There are three variants on the cross-over operator: single-point, two-point and n-point. Single-point cross-over operator duplicates the two parent entities' chromosomes and chooses some 'splice point' along the length of the chromosome between two genes. The fragment of genetic material after the splice point are swapped between the child entities to form distinct entities. An example of this is shown in figure 2.5. Two-point cross-over is similar to one-point but two splice points are selected. The genetic material between the two splice points in the child solutions are then swapped. The n-point cross-over considers each point along the chromosome in turn and allocates the position a splice point with probability p_{splice}. Genetic material between alternate sections is copied.

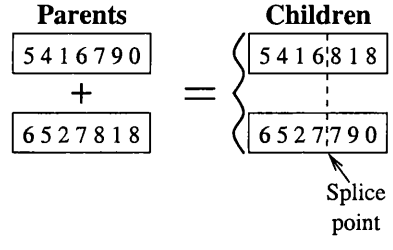


Figure 2.5: An example of a 1-point cross-over operator for a 7 gene problem.

Once all the children have been bred the mutation operator is applied. This alters a randomly chosen genes to new, random values. This serves two purposes. First it gives the GA a method of escaping a local maximum and second it allows fine adjustment to

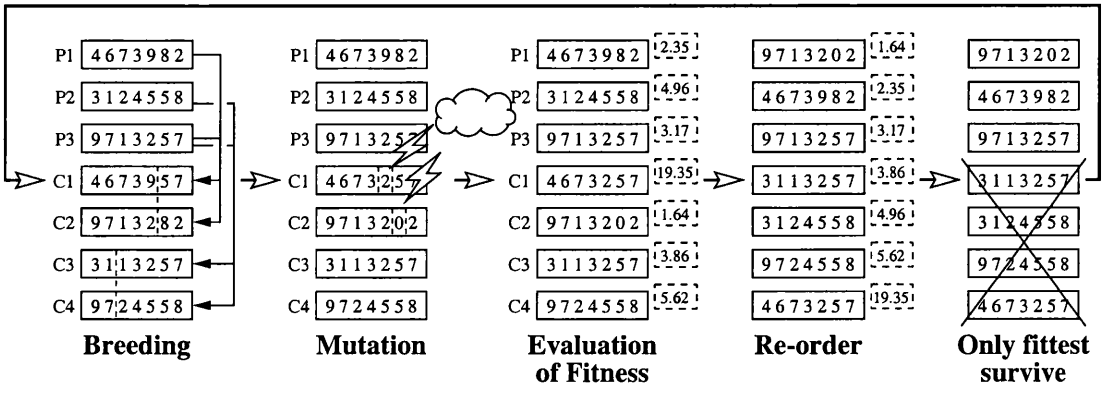


Figure 2.6: Steps in progressing from one generation to the next.

a solutions close to a local (perhaps global) maximum.

After mutation, a merit or fitness function is used to assign a numerical 'fitness' to each new solution and the survival operator is applied. One survival operator is: rank entities (parents and children together) based on their fitness function and select the top fraction to proceed to become the potential parents for the next generation. This is illustrated diagrammatically in figure 2.6. Another common survival operator is to have the number of children equal the number of parents and replace all parents with their children. In this case, evolution is encouraged by using roulette or tournament selection and cross-over is altered to no longer be automatic but to operate with probability p_{cr}.

It is vital to note that all the problem specific information is contained within the fitness function and that the only assumption the algorithm makes is that this function is single valued. This is a weak assumption. It suggests that GAs are robust: the algorithm will still converge on the global maximum if the fitness function varies sharply or is discontinuous — provided a limiting value of the function can be defined.

The following list illustrates some advantages gained from using a Genetic Algorithm:

- A genetic algorithm is self-organising. It requires no problem specific strategy or length-scale information. A generic GA should solve most problems eventually. For discussion on minimum conditions for a GA to converge on the global minimum see Eiben et al. [1991].

- Genetic algorithms are a robust method. For example, in the arena of data reduction 'extra information', such as extra peaks and noise in spectral data, does not confuse a GA. It will still arrive at a 'reasonable answer', *i.e.* within the

neighbourhood of the global minimum.

- GAs are very fast algorithms for finding approximately the global minimum: a solution that is nearly correct.

Despite these advantages listed above there are disadvantages and unresolved issues related to Genetic Algorithms.

The improvement in successive generations' best solution decays exponentially. Eventually it become ineffective to continue with a GA. The best solution from the GA could then be further improved by some local maximum finding procedure, provided the true global maximum has been located.

If the fitness function is noisy then this can disrupt the progress of a GA. This has been demonstrated for a binary GA using roulette wheel selection (Fogel and Ghozeil [2000]). Other forms of GAs may better deal with such problems.

A genetic algorithm will improve a good solution via mutations. But this method is ineffective if the 'ridge' of the maximum lies diagonally (*i.e.* off-axis) in parameter space. Diagonal maxima require multiple mutations of the same solution for any improvement. This is increasingly unlikely as the dimensionality of the valley increases.

It is unclear how to implement a genetic algorithm that uses floating-point numbers. The floating-point analogue of cross-over and mutation is not obvious. For example, if the numbers are encoded as decimal digits then there is a problem associated with rounding: 0.999 would require four co-incident mutations to become 1.000, where as 0.998 would require only one to become 0.999. The result of this effect is floating-point GAs tends to get stuck at rounding points.

Most numerical optimisation problems use a rectangular region of parameter space. For the operators discussed, mutation and cross-over will always produce children whose parameters lie within this region of parameter space. If the bounded region of parameter space is irregular, or if parameter space is, in some sense, non-orientable, cross-over may produce invalid solutions. Invalid solutions may be rejected automatically, which greatly hinders the genetic algorithm, or could be penalised with a decrease in the solution's fitness. See Goldberg [1989] for specific examples.

For non-continuous, combinatory problems, such as scheduling a series of order-dependent tasks to take the minimum time on a parallel computer as in Corrêa et al.

[1999], the parameter space is non-Euclidean and no distance-metric can be applied as there is no direction which is 'uphill'. For the purpose of developing suitable genotypes such spaces can be embedded within a 'larger' space that also includes invalid solutions. For example, a scheduling problem could be expressed as an ordered list of tasks but this would then permit possibilities that are invalid. Simple operators, such as cross-over, cannot be used as they might generate an invalid point in parameter space, such as a schedule that does not include all the required tasks. Care must be taken in developing new operators that have similar properties to the simpler versions. New operators should permit only correct solutions that do not disrupt the operation of the GA, apply some minimum fix-up operation to invalid solutions or use a penalty system such as described above.

Perhaps a surprising result from Wolpert and Macready [1997] is the No Free Lunch (NFL) theorems for optimisation. These state that the average performance of an algorithm, when considering all possible problems (*i.e.* over all 'algorithm space'), is independent of the algorithm: all algorithms average out the same. If, for two algorithms a_1 and a_2, a_1 out performs a_2 for some subset of all problems then it must, on average, under perform a_2 for all other problems.

The GA's robustness can be viewed under this context. GAs have a broad peak in algorithm space. For any given narrow problem-set of interest, other algorithms may exist that are faster. However, because of the breadth of the GA's peak the GA will still work when, for example due to noise, the specific problem has wandered outside the region in which the fast algorithm works.

There are a number of methods of improving GAs. A GA can be restarted after some criteria such as lack of progress or after a specific time (see Fukunaga [1998]). The mutation rate (and cross-over probability) can be altered to reflex the GAs progress as described in Sriniva and Patnaik [1994] or Neubauer [1997]. Mutations can also be 'directed' towards the (hopefully global) minimum at hand (see Bhandari et al. [1994]). However, the effectiveness of these techniques generally have been shown on a few problem-sets. Accelerating convergence through one of these techniques increases the risk of premature convergence of the population (lack of bio-diversity) around some non-global minimum. This goes back to the NFL theorems. Nothing is free!

2.2 Global convergence and Schemata

Most of the analysis on how Genetic Algorithms work is based on the concept of schemata. A schema is a string with the same length as the GA's genotype and populated with entries from either the GA's alphabet or an extra 'wild card' character. For example, if the GA's alphabet was the binary digits, `0` and `1`, then schemata could be written using the symbols `0`, `1` and `#` where `#` is the wild card character.

A particular genome is consistent with a schema if it matches all non-wild card entries. For example, the genotype `01011` matches the schema `01##1` whereas the genotype `00011` does not. In general, a schema with k `#` symbols can be thought of as a $d - k$ hyper-plane within the d dimensional parameter space.

According to schema theory, any string in a GA contains partial information about many schemata simply by replacing an element of the genotype with the wild-card symbol `#`. For example in a binary GA, the function value or 'fitness' of the string `01011` contains information about the schemata `0101#`, `010#1`, `010##`, `01#11`, and so on, where `#` represents the extra 'wild card' character. In turn, each schema matches many other possible strings, for example `010##` matches `01000`, `01001` and `01010`. In this way, the function evaluation of `010111` provides information about many other strings. This is called implicit parallelisation.

Breeding, especially the cross-over operation, on average will preserve short-order schemata. This effect can be shown (see Goldberg [1989]) to improve the average fitness of the population by increasing the number of genotypes that have good-looking schemata. This works well for certain class of problems: one where the problem can be split into two or more mostly orthogonal sub-problems.

Recently, there has been some debate as to the effectiveness of schema theory for best describing the dynamics of genetic algorithms. Alternative methods have been proposed including statistical mechanics (Rattray and Shapiro [1996]) and Markov chain analysis. Markov chain analysis is a powerful description allowing expressions for minimum conditions for convergence (in the sense of tending towards the optimal solution). Eiben et al. [1991], Agapie [1998] and Schmitt et al. [1998] discuss these findings.

2.3 Mutation rate

One of the major benefits of GAs is their speed in finding an answer close to the global maximum. This convergence relies on the choice of the various elements that make up the genetic algorithm. One of these elements required by a GA is an operator that mutates solutions. Typically, this operator changes solution genes and operates at a fixed probability p_{M}. This is equivalent to mutating a fixed number $m = (Ml)p_{\mathrm{M}}$ (for small values of p_{M}) of genes. The genes to be mutated are selected randomly (with replacement) from a population of M solutions each having l genes. The mutation rate is usually kept constant throughout the optimisation process although adaptive mutation rates have been successfully applied (see Neubauer [1997]).

The mutation rate m is a key variable in determining the convergence rate of a GA. If it is set too high then too much of the information learned from previous generations is lost. Taken to extreme enough mutations would occurred that all child genes were overwritten. This would reduce the method to a simple hit or miss with the current best solution retained.

If the mutation rate is too low then the algorithm's exploration of parameter space is hindered. If the mutation is turned off then the best solution of all permutations of the initial genes would be found. As this is likely to be a sub-optimal maximum, the GA would not be able to find the global maximum.

The expected value of the optimum mutation rate for a simple test problem (*OneMax*) are presented in Hesser and Männer [1991]. This states that the optimum mutation rate is one with probability p^*_{M} given by (2.2).

$$p^*_{\mathrm{M}} = \frac{1.76}{M\sqrt{l}} \tag{2.2}$$

To investigate the effect of different mutation rate of the efficiency of the GA, and compare the results with (2.2), the charges on a disk problem (see §2.6.1) was considered with different values of mutation rate. This problem was chosen as it is close to a 'real-life' optimisation problem but the dimensionality can be altered easily.

The analytical solution of the minimum energy problem for n charges ($n < 12$) can be calculated as discussed in §2.6.1. For each value of the mutation rate 10 parents were used to generate 20 children and the process was stopped when the best solution is less

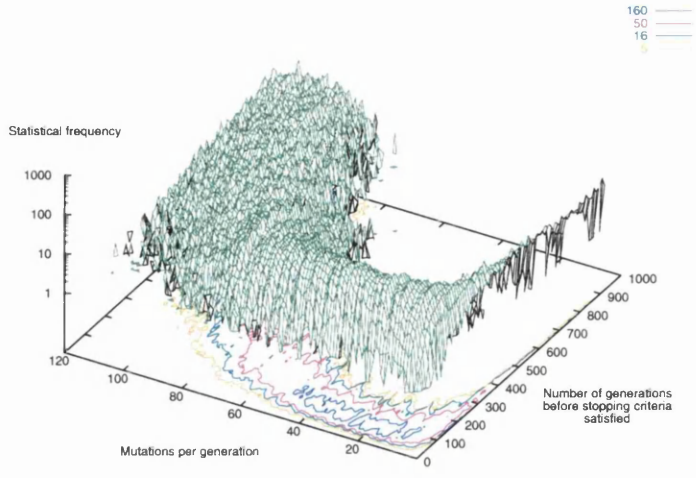


Figure 2.7: The convergence rate for the Charges on a Disc problem with different mutations per generation for the 8 gene problem (4 charges).

than 1% greater than the true minimum or more than 10000 generations had occurred. This process was repeated 2000 times and the results binned based on the number of generations before achieving the 1% tolerance required. A selection of results are shown in figures 2.7, 2.8 and 2.9.

These figures show three dimensional representations of the log of the probability distribution function as estimated by the Monte Carlo simulation along with contour lines. For very small values of mutation rate, m, there is a wide range of convergence rates. This is consistent with the GA being largely dominated by initial population with the lack of mutation hindering the effective search of parameter space. As m increases the middle and width of the distribution of convergence rates decreases to a minimum before increasing again. These increases are symptomatic of over-mutating and the resulting loss of information reducing the GA to a random hit-or-miss method whilst retaining the best solution.

For each value of mutations rate a Gaussion fit was conducted using the Levenberg-Marquardt method (see §15.5 Press et al. [1996]). To overcome the problem of choosing initial conditions the minimisation was repeated with each datapoint as the initial centre of the Gaussian. The best resulting minimisation was retained. Figures 2.10, 2.11 and 2.12 show sample results from this fitting procedure.

Figure 2.13 shows the resulting Gaussian's centre as a function of mutations rate for different dimensioned problems. For some of the data the 1000 generation cut-off truncated the distribution too greatly to recover the Gaussian curve. These points have

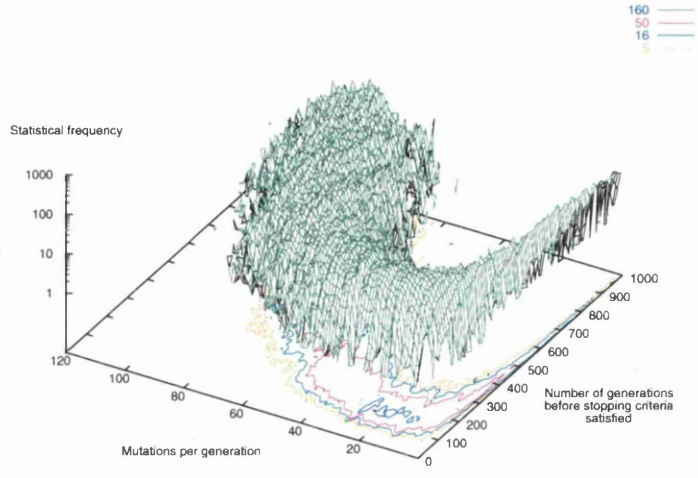


Figure 2.8: The convergence rate for the Charges on a Disc problem with different mutations per generation for the 12 gene problem (6 charges).

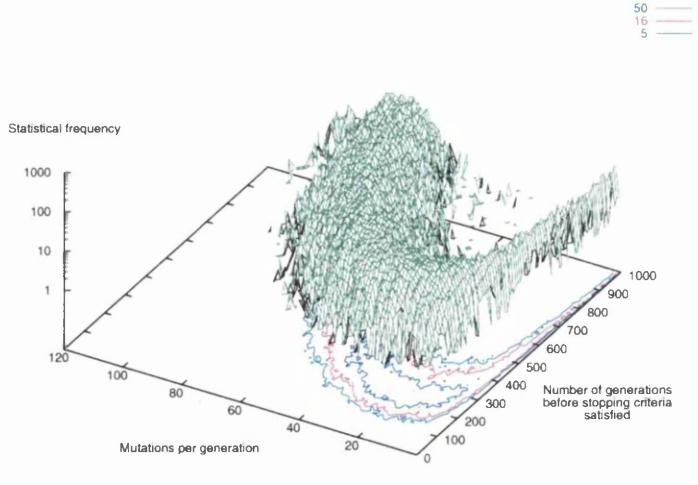


Figure 2.9: The convergence rate for the Charges on a Disc problem with different mutations per generation for the 16 gene problem (8 charges).

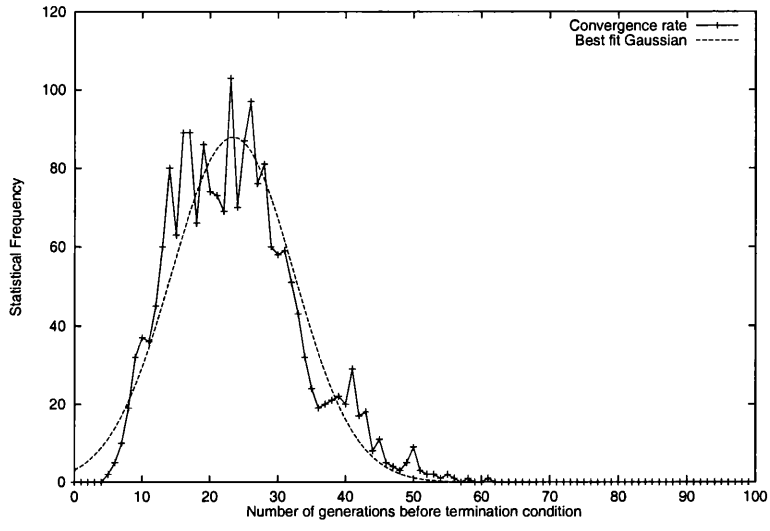


Figure 2.10: Fitting of a Gaussian to the convergence rate curve for the 8 gene (four charges) problem with 62 mutations per generation.

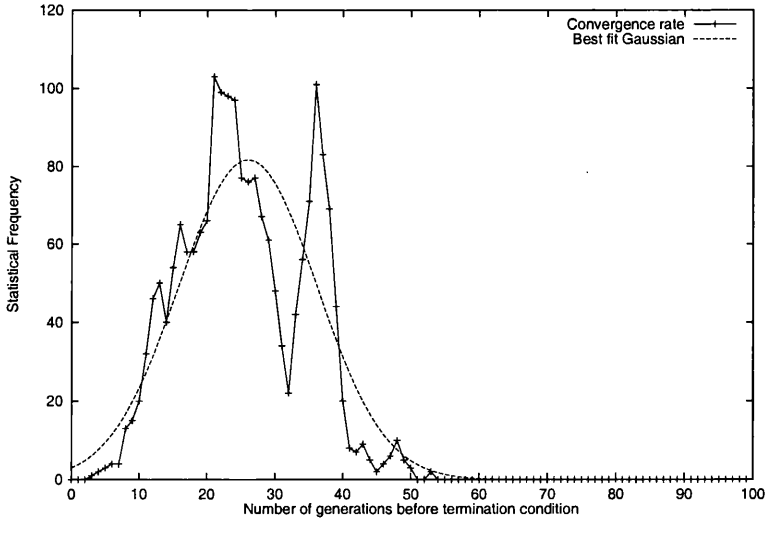


Figure 2.11: Fitting of a Gaussian to the convergence rate curve for the 8 gene (four charges) problem with 63 mutations per generation.

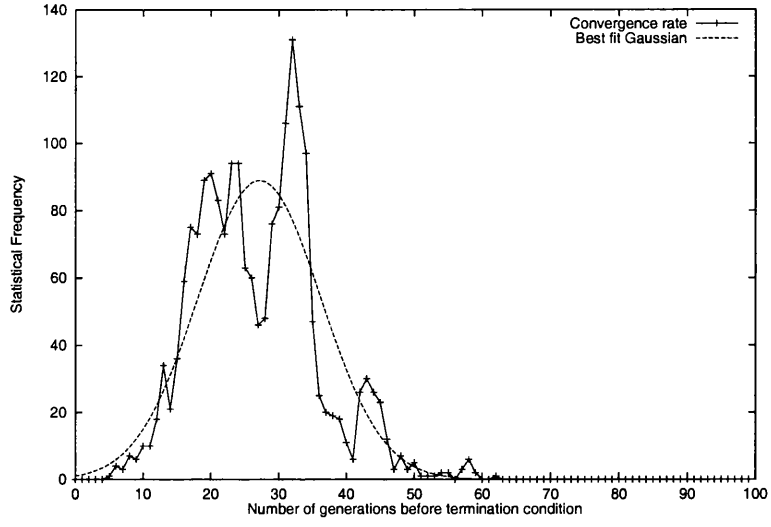


Figure 2.12: Fitting of a Gaussian to the convergence rate curve for the 8 gene (four charges) problem with 64 mutations per generation.

been removed.

To establish the optimum mutation rate a simple function (2.3) was fitted to each dimensionality where x is the mutation rate and α, β, γ, δ and ζ are fitting parameters.

$$y = \alpha \left(\frac{x}{\beta}\right)^{-\frac{1}{2}} + \gamma \left(\frac{x}{\delta}\right)^{2} + \zeta \tag{2.3}$$

The minimum, $x_{\min}$, is shown in (2.4).

$$x_{\min} = \left(\frac{\alpha^2 \delta^2 \beta^3}{16\gamma^2}\right)^{\frac{1}{5}} \tag{2.4}$$

This allows an objective measure of the optimum mutation rate for some specified problem. This procedure was used to calculate the observed optimum for the 3, 4, 5, 6, 7 and 8 charge problems. The results are plotted in figure 2.14 along with the predictions from Hesser and Männer [1991] stated in (2.2).

The observed values of optimum fitness are higher than the predicted value. This is not too surprising as the binary GA described in Hesser and Männer [1991] allows the splice point to occur within the binary coding for the parameters. This has an effect similar to mutating as there is no guarantee that, say, the lower 4 bits of a floating point number will retain the same meaning when swapped from one number to another. This implicit mutation could account for the discrepancy.

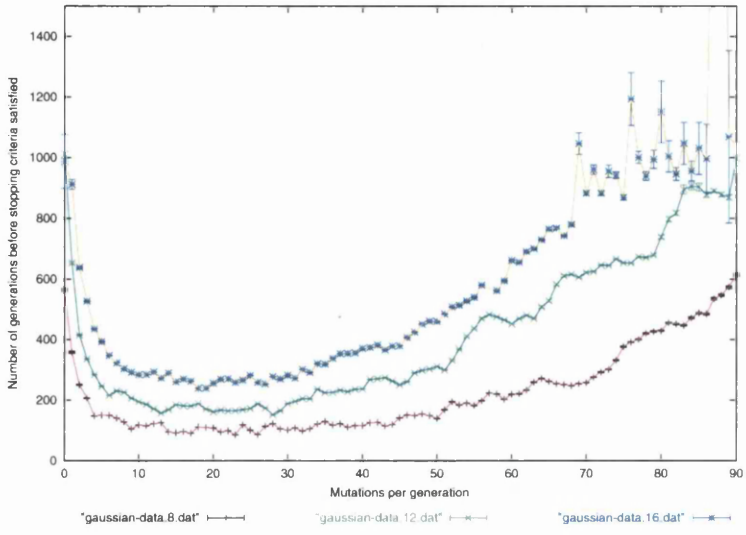


Figure 2.13: Summary of the convergence rate for the Charges on a Disc problem by fitting Gaussians to each mutation rate.

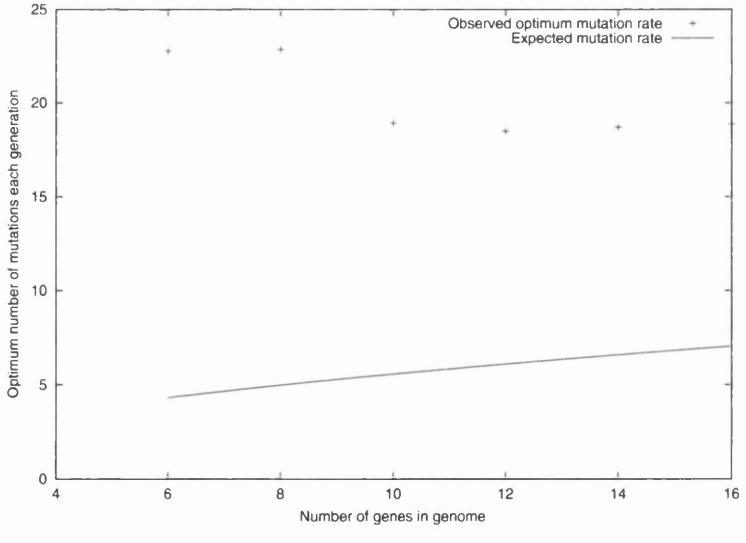


Figure 2.14: Comparison between observed best and predicted mutation rate.

It is also possible that the geometry of the global minimum changes in a problem-specific way. This is unlikely as the solution for the problems considered were of the same form.

2.4 Core functionality of the ELGAR library

In order to study various problems associated with a genetic algorithms a GA was written using the C computer language. It was designed as a set of library routines. These routines could be embedded with the problem-specific code at the linking stage. This allows the code to be reused for different problems without recompiling. The set of GA routines was called Embedded Library Genetic Algorithm Resource (ELGAR).

ELGAR was written for the Unix operating system model and ran under both the GNU/Linux and Sun Solaris operating systems but should run without major alterations on any platform with an ANSI-C compiler. ELGAR was designed to be an experimental library. In addition to the problem-specific fitness function the two genetic operators (cross-over and mutation) could be altered by replacing the existing default code with routines designed with a particular problem in mind.

The data structure `Entity` within ELGAR holds information about each of the points in parameter space. These are, in turn, used to evolve into a new potential solutions. The structure is defined as:

File: GA/Elgar/elgar.h

```
49 typedef struct {
50         float           ent_Error;
51         unsigned        ent_Age;
52         GeneSequence    ent_Genes;
53 } Entity;
```

The collection of entities that ELGAR manipulates is an array of `Entities` as defined above. It is the responsibility of the error function to assign a numerical value to the `ent_Error` element when it is presented with a `Entity` structure.

The `ent_Age` is included so that more advanced evolutionary schemes could be included such as automatically culling entities that have become too old. This feature would allow ELGAR to mimic the behaviour of some other GAs in which all parent solutions are replaced by their children.

The `ent_Genes` stores an array of genes. These are defined as `unsigned chars`, which allows each gene to take an integer value between 0 and 255 inclusively.

File: GA/Elgar/elgar.h

```
44 typedef unsigned char * GeneSequence;
```

The key data element in ELGAR is the `GenePool` structure. This holds all the problem specific information about a particular problem so several problems can be solved simultaneously or hierarchically: a GA to optimise GA parameters. The structure is displayed in Appendex A.1.

To optimise a function, a `GenePool` structure must be allocated and initialised with the required parameters (all those before "AUTOINITIALISE" above). After the values have been set, the function `GimiGenePool` completes the initialisation process, including the generation of a random initial population. Thereafter, a call to `FindMinimum` will start the minimisation proper.

The key element of the structure is `gp_CalculateError`. This contains a pointer to the error function, which must be defined to take an `Entity` pointer as its single argument and return an integer value.

The error function should take the values of the `ent_Genes` array and use these values to obtain some floating point number, placed in `ent_Error`, that ELGAR tries to minimise. Note that although all functions described herein divide each element of `ent_Genes` by 256 to obtain a number in $[0, 1)$, there is no requirement to use the `unsigned chars` in this fashion: any encoding of the problem is valid but consideration should be given to prevent invalid encodings and to facilitate preservation of schemata when breeding.

The return value is used to indicate the success or failure of the error calculating routines; for example, if the error function required some equipment external to the computer that became unavailable during the minimisation process the error function could indicate this by returning a non-zero value.

This library was further extended by the Genetic Algorithm Monitoring Extension (GAME), which graphically displayed the current progress of the GA. This was used to assess the progress of the GA when choosing a mutation rate and also to gives a quick visual indication of any problems with the GA. GAME was designed to run under the X

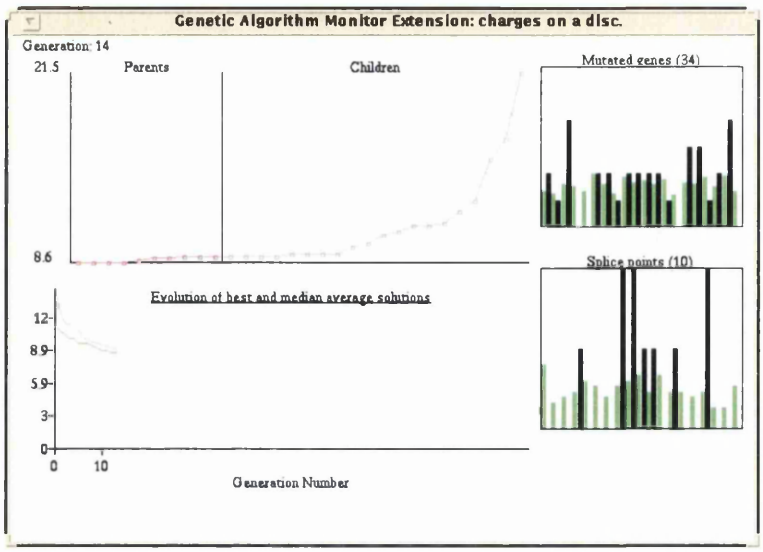


Figure 2.15: A sample of GAME's output.

windowing system but is modular so that if ELGAR was compiled for another computer architecture it would be easier to implement a new version of GAME.

A typical view of the output from GAME is shown in figure 2.15. The top left of the GAME output shows the current generation number and a plot of the entities' fitness functions. Below this is a graph showing the evolution of the best and median average fitness functions as the system evolves. The two boxes to the right of the output show representations of the statistical distribution of splice points and selected genes for mutation; the numbers in brackets indicates the number of random calls per generation. The black bars indication this generation's selected values whereas the lighter colour indicates the average values so far. This checks that the cross-over and mutation operators are not biased towards certain values.

One requirement of the problem specific code is that it contained the function `CalcErr`. This function contains all the problem specific information ELGAR requires and is called with an entity as a parameter. The function must calculate the fitness of this entity and store the result in the entity structure.

ELGAR used `unsigned chars` to store the gene information. Each byte takes a numerical value between 0 and 255. A typical `CalcErr` function first divides each gene by 256 to obtain a number in $[0, 1)$ and then multiply that number by some scaling factor. This would then be used as the parameters for the fitness function.

The error function for the charges on a disc problem discussed in §2.6.1 is a useful

illustration of these concepts. The functions used within the algorithm are presented without explanation at this stage but are descriptively named and commented so should be readily understood.

File: GA/Disc/disc.c

```
407 int CalcErr( Entity *myentity)
408 {
409   Charges *soln;
410
411   /* Convert byte-string into something meaningful */
412   soln = CharToCharges( myentity->ent_Genes);
413
414   /* if soln is NULL, then an error has occured. */
415   if( soln == NULL)
416     return 1;
417
418   /* Calculate the corresponding error */
419   myentity->ent_Error = WorkOutEnergy( soln, charges);
420
421   /* All ok */
422   return 0;
423 } /* CalcErr */
```

Instead of searching for the maximum of the function, the ELGAR library found the solution with the minimum of the function as this is more often required in data reduction problems. However, as stated earlier, any maximisation problem can be rewritten as a minimisation problem.

2.5 Genetic Algorithms and NP-completeness

Binary decision problems (*i.e.* problems where the solution is either 'yes' or 'no') can be classified based on the complexity of the best algorithm that can solve them. The complexity of an algorithm describes how the computational burden (measured by the time taken to find the answer) increases with more difficult problems; for example, a problem that has a complexity of order N^2, usually written simply as $O(N^2)$, will take four times longer to complete if the problem is twice as 'big'. The variable N measures the size of the problem in usually an intuitive but problem-dependent fashion.

Problems that can be solved by algorithms that have a complexity described by a polynomial function of N are in the class P. Also included in P are problems solved by algorithms that have complexities that are not polynomial but grow at a slower rate

than an exponential, such as $O(N \log N)$. Problems that allow the answer to be verified in polynomial time are in the class NP. All problems in P are in NP but it is believed that not all problems in NP are in P or NP$\neq$P. Perhaps surprising, it can be shown that certain NP problems can encode all other NP problems through some polynomial time algorithm. All problems with this property are called NP-complete. The importance of this is that if a polynomial time algorithm were found for any of the NP-complete problems then all NP problems would be solvable in polynomial time. In some sense the NP-complete problems are the most difficult problems in NP.

Problems that are not binary decision, such as optimisation, but that can encode an NP-complete problem through a polynomial time algorithm are called NP-hard. These have the same property that they can be reduced to each other through some polynomial algorithm so that being able to solve one in polynomial time would imply being able to solve all.

At present, no algorithm exists that can solve any NP-complete problem in polynomial time. It is believed (see Garey and Johnson [1979] for more details) that NP-hard problems (and the complementary problem-set: NP-complete) are somehow intrinsically difficult and that no algorithm can exist that solves these problems in polynomial time. It is, therefore, unfortunate and ironic that proving the intractability of NP-hard and NP-complete problems at present seems to be as intractable as the problems themselves.

Various people have attempted various NP-complete and NP-hard problems using genetic algorithms. The satisfiability problem (SAT) and travelling salesman problem (TSP) are perhaps the two most common. Due to the combinatorial nature of the parameter space, the simple genetic algorithm described above generally is inefficient. Different methods have been attempted, such as adding a fixing-up stage to mend broken solutions (see Watson et al. [1998], Ulder et al. [1991] and Gottlieb and Voss [1998] for examples) with excellent results, or by introducing redundancy in the genotype (as in Gerrits and Hogeweg [1991]), but this continues to be an area of ongoing research.

2.6 Examples of simple problems solved by a GA

This section discusses some problems with which ELGAR was used. These were used as either a test problem to check ELGAR was working correctly or to investigate how well ELGAR performs. They are included here as they illustrate some of the benefits

and difficulties associated with a GA. They also give practical examples of how to solve a problem with ELGAR.

Also used with some of the programs is a small (approximately 1700 lines of code) library called ReadData. Various projects were required to read user supplied data in a variety of formats. This library unifies the reading procedure to completing a 'request' structure `DataRequest` and a single function call: `ProcessDataRequest()`. The resulting structure (`DataList`) contains details about the `NumberOfSeries` series each containing `NumberOfEntries` elements.

2.6.1 Charges on a disc

The problem of minimising the electrostatic energy of n discrete charges embedded within some conducting medium was first posed by Berezin [1985]. For the three dimensional spherical case the form of the minimum energy configuration is independent of the number of charges and has all charges confined to the sphere's surface. However, for the two dimensional circular case the minimum energy configuration is dependent on the number of charges. For cases where the number of charges is less than 12 the minimum energy is with the charged particles located at the vertices of a regular n-sided polygon inscribed within the disc. For $n = 12$ to 15 charges the minimum energy configuration is $n - 1$ charges forming a regular polygon as previously and the last charge in the centre. Figure 2.16 illustrates these different configurations. For the energy minimisation problem with n greater than 15 the ground state has a more complex pattern.

The problem of determining the ground-state of an atomic cluster, which is related to the charges on a disk problem, has been shown in Wille and Vennik [1985a] to be NP-hard. And, whilst the 2 dimensional charges on a disc problem has not be shown to be NP-hard it is still difficult due to the small neighbourhood in which solutions converge on the global minimum. This is seen clearly by considering the case for moderate n, say 10, where the global minimum configuration is a regular decagon. If the charges are placed on the circumference (the global minimum configuration) and one charge is displaced slightly then a local optimisation method would expel that charge to the centre resulting in a sub-optimal minimum.

ELGAR defaults to 10 parents breeding 20 children each generation. With these values the correct configurations for up to 8 charged particles was found. However, the

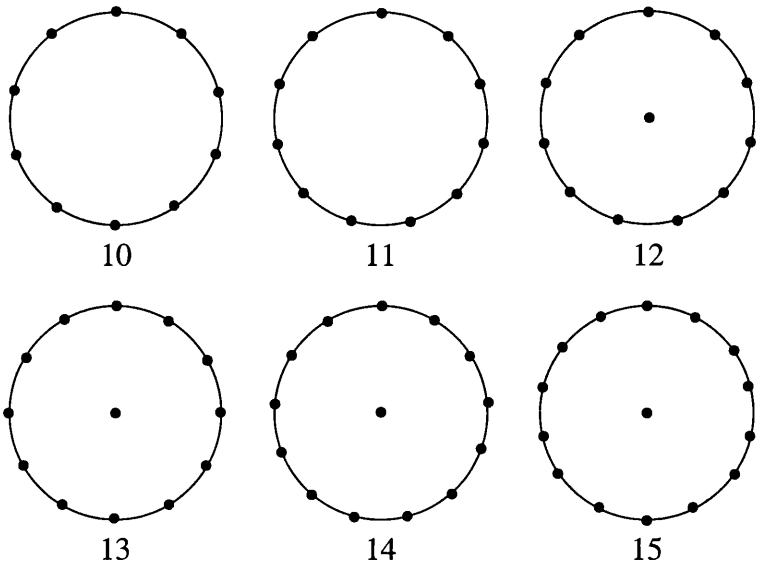


Figure 2.16: Minimum energy configurations for the Charges on a disc problem. Diagram shows $n = 10$ to 15.

configuration found for 9 particles was a local minimum: a regular octagon with the ninth charge in the centre.

Increasing the number of parents and children improves the performance of a GA. With 20 parents and 50 children the GA found the correct configuration for 9 charges. 50 parents and 200 children were required before ELGAR found the correct configuration for 10 charges and 500 parents and 1000 children were required before the correct configuration of 11 charges was found.

The increasingly large resources required to solve these problems are symptomatic of off-axis valleys in parameter space. Whereas in §2.6.2 the off-axis valley hinders further improvement after the neighbourhood of the minimum is found in this problem a local minimum is present that is far easier for the GA to fall into: the global minimum is off-axis whereas the local minimum is not. Consider the case of 9 charges arranged at the circumference of the disc but with one charge offset towards the centre. Moving the offset charge towards the centre reduces the energy, and requires just one mutation, but to move it back to the correct position on the circumference would require moving all the charges by the correct amount: nine coincident mutations all of the correct magnitude.

The efficiency of the GA in solving this problem is adequate compared to existing methods (such as simulated annealing discussed in Wille and Vennik [1985b]) but improvements are likely to be possible. The coding scheme for the genotype and the genotype to phenotype conversion could be improved and alternative forms of genetic

operators could be investigated.

2.6.2 Fitting a sinusoidal signal

Another problem attempted with ELGAR was extracting an arbitrary sinusoidal signal from noisy sampled data. The sinusoidal signal was taken to be the sine function with arbitrary amplitude, phase offset and frequency. It was assumed that there may be some DC offset. Although this problem is trivially solved (for example, the Fourier transform of the data reveals this information) it could be any non-linear function sampled at arbitrary intervals.

One consideration was deciding the limits for each parameter. Both frequency and phase offset have natural boundaries. For the DC offset and amplitude realistic limits can be imposed without overly increasing the search space.

Sample theory limits the maximum frequency of reconstructible signals to the Nyquist frequency, which is half the sampling frequency: ν_{sam}. The non-dimensional variable used, $\nu = \nu_{\text{sig}}/\nu_{\text{sam}}$, varied between 0 and 0.5.

The phase offset can have any value in $[0, 2\pi)$. The non-dimensional variable used, Φ, varied between 0 and 1.

The amplitude, a, is taken to lie between 0 and the range of the data (the difference between the data maximum and minimum). The DC offset, b, was constrained to lie between the largest and smallest data values.

The phenotype for the problem is:

$$f_i(a, b, \nu, \Phi) = a \sin\left[2\pi\left(\nu i + \Phi\right)\right] + b \tag{2.5}$$

The code section to implement this is:

File: GA/Sines/fitsine2.c

```
112 void CharsToSolution( unsigned char *byte, Solution *mySoln)
113 {
114         mySoln->Frequency = MAX_FREQ * byte[0]/256.0;
115         mySoln->Offset = byte[1]/256.0;
116         mySoln->Amplitude = Globals.Max_Amplitude*byte[2]/256.0;
117         mySoln->DC_Level = Globals.Min_DC +\
118                 byte[3]*(Globals.Max_DC-Globals.Min_DC)/256.0;
119 } /* CharsToFloats */
```

where the `Solution` structure is defined as:

File: GA/Sines/fitsine2.c

```
18 typedef struct {
19         float   Amplitude;
20         float   Offset;
21         float   DC_Level;
22         float   Frequency;
23 } Solution;
```

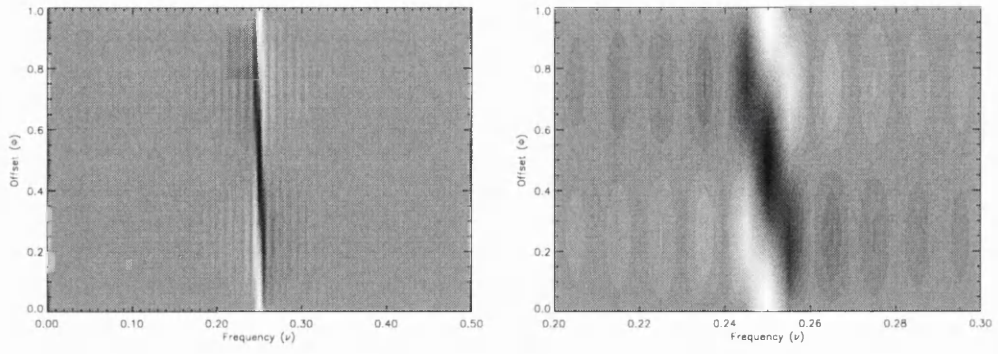
The function ELGAR minimised (*i.e.* the negative fitness) was the equal error χ^2 function as stated in (2.6). y_i is the data, each with uncertainty σ. However, it is trivial to adapt the method to deal with data with uneven uncertainties.

$$\chi^2 = \sum_{i=1}^{n} \frac{(y_i - f_i)^2}{\sigma^2} \tag{2.6}$$

This is implemented by the following section of code:

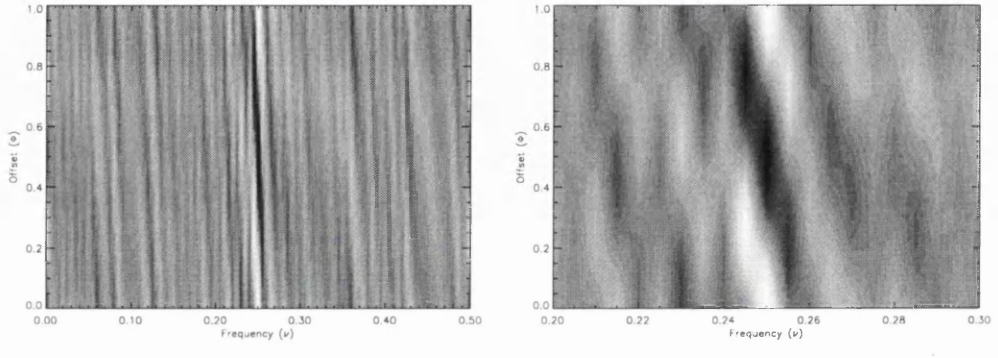
File: GA/Sines/fitsine2.c

```
122 int ErrFn( Entity *Ent)
123 {
124         Solution mySoln;
125         float X, Y, diff, chi_s=0.0;
126         unsigned n;
127
128                 /* Convert byte sequence into meaningful floats */
129         CharsToSolution( Ent->ent_Genes, &mySoln);
130
131                 /* For each data point */
132         for( n=0; n < theData.Entries; n++) {
133
134                 X = theData.X [n];
135                 Y = mySoln.DC_Level + mySoln.Amplitude*sin( TWO_PI *\
136                                         (mySoln.Frequency*X + mySoln.Offset));
137                 diff = (theData.Y [n] - Y)/Globals.Error;
138                 chi_s += diff*diff;
139         }
140
141         Ent->ent_Error = chi_s/n;
142
143                 /* Everything ok */
144         return 0;
145 } /* ErrFn */
```



(a) (b)

Figure 2.17: Grey-scaled contour plots of χ^2(a=1.0, ν, Φ, $b = 0.0$) for a noiseless sample.(a) shows the full range of Φ and ν, (b) shows detail of central region.



(a) (b)

Figure 2.18: Grey-scaled contour plots of $\chi^2(a = 1.0, \nu, \Phi)$ for the noisy data ($\sigma = 1.0$).

Perhaps surprisingly, the χ^2 function shows detail for sampled data with no noise. Figure 2.17 show a χ^2 plot for offset and frequency for a signal $\sin 2\pi(\frac{1}{4}x + \frac{1}{2})$. Plot (a) shows χ^2 for the whole range of ν and Φ whereas plot (b) shows the detail closer to the global minimum at $\nu = 0.25, \Phi = 0.5$. Note the number of local minima close to the global minimum. Any local minimisation routine with initial parameters too far away from the global minimum would 'get stuck' in these regions.

Artificial noise was generated using the Gaussian distribution function `gasdev` (see chapter 7 of Press et al. [1996] for details). Figure 2.18 shows the same χ^2 plots but for the noisy data. The noise has $\sigma = 1.0$.

ELGAR found the region of the global minimum within 100 generations using very modest resources: 10 parents breeding 40 children each generation. However, the global minimum's valley is off-axis. Once a solution close to the original signal is found, further improvement is slow. Figure 2.19 shows the original signal, the noisy sampled data and

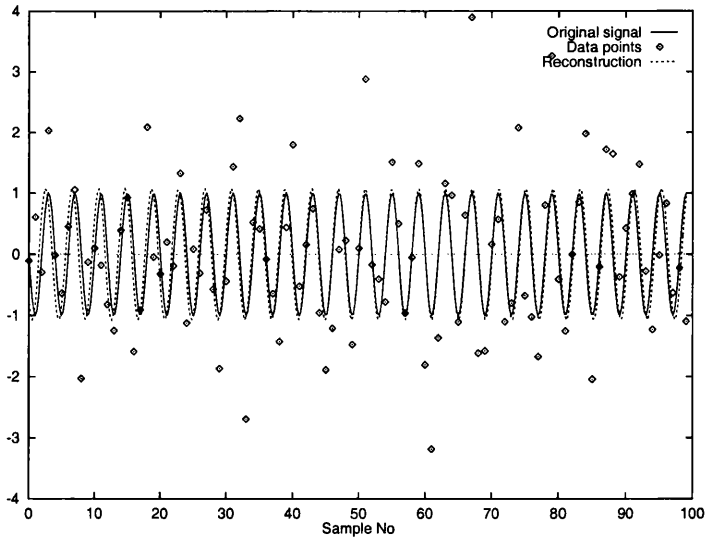


Figure 2.19: Plot of the original signal, the sampled data and the reconstructed signal.

the reconstructed signal.

Further improvements to the algorithm could be achieved by rephrasing the problem so the global minimum appears on-axis. Alternatively, a local optimisation routine (such as Powell's method in §10.5 of Press et al. [1996]) could be employed once the GA has found the global minimum.

2.7 Summary

The Genetic Algorithm is a technique for solving a wide range of difficult optimisation problems. Difficult may mean that the parameter space has lots of local extrema (such as fitting to noisy data) or that it cannot easily be searched (such as combinatory optimisation).

GAs track and use many points in parameter space rather than trying to improve one point. These many points are iteratively subjected to several operators which mimic some of the processes that a biological species undergoes. The effect is to encourage the points to 'evolve' to increase their fitness thus solving the optimisation problem.

Problems with off-axis valleys around the global minimum prove to be a difficulty for GAs but the effect can be reduced by constructing a hybrid method that uses a local minimisation routine after the GA has finished, by using more computer power, or by rephrasing the problem so that the minimum valley is aligned with the axis.

Chapter 3

Bayesian Statistics in signal analysis

This chapter describes the method used to introduce 'prior information' using abstract information known *a priori* to analysing data that then biases the inference procedure (see Kendall and Stuart [1963] and Hoel and Craig [1978] for indepth discussion). This is useful when analysing noisy data: the prior information biases the analysis towards more likely values. This is achieved via Bayesian Inference: using Bayes' equation to choose between different possible hypotheses.

The first section of this chapter introduces Bayes' equation and discusses the basic premise of Bayesian Inference which the following section illustrates with a simple example without any prior information. Section three formulates the Bayesian concept of prior information in a useful form for the next chapter. An abstract example of this formulation is given in section four. Section five discusses methods of estimating the uncertainties and section six summaries the key points.

3.1 Bayesian Methodology

Many data reduction problems involve fitting a model described by a set of parameters, $\boldsymbol{\lambda}$, to some experimental data, $\mathbf{D}$. Typically, this is done by maximising a statistical measure of likelihood: a measure of how likely the data is for a given value of $\boldsymbol{\lambda}$. One commonly used likelihood is the χ^2 statistic: $\chi^2 = \sum_i (f_i - d_i)^2/\sigma_i^2$ where f_i is the model, d_i (which depends on $\boldsymbol{\lambda}$) is the data and σ_i^2 is the variance of the noise. The method of maximising this likelihood (by minimising χ^2) is called the 'least squares'

method and, in the case when $\boldsymbol{\lambda}$ is a set of linearly independent parameters, can be solved analytically.

The Bayesian model for inference works on the concept that each value of $\boldsymbol{\lambda}$ has some 'acceptability' assigned to it. This acceptability is expressed by a number between 0 and 1 where 0 indicates the value of $\boldsymbol{\lambda}$ is not acceptable and 1 indicates that the value is certainly the correct one. The acceptability of a particular value of $\boldsymbol{\lambda}$ is denoted $P(\boldsymbol{\lambda})$ so the most likely value of $\boldsymbol{\lambda}$ is the one with the maximum $P(\boldsymbol{\lambda})$.

This is similar to the concept of probability in 'normal' or 'frequentist' statistics where $P(\alpha)$ is the probability of event α occurring. The usual interpretation of $P(\alpha)$ in the frequentist approach is that for an infinite number of trials the fraction of trials consistent with event α occurring is $P(\alpha)$ or that $P(\alpha)$ is the limiting value for the fraction of event α occurring for increasing number of trials. This is in contrast to the Bayesian interpretation which treats $P(\boldsymbol{\lambda})$ as the plausibility of the particular set of values $\boldsymbol{\lambda}$. This is a significant difference as $P(\boldsymbol{\lambda})$ does not appear as a limiting value for an increasing number of trials but rather as an estimate of plausibility based on the available data.

Each value of $\boldsymbol{\lambda}$ has some *a priori* acceptability before any measurements are made. Once data has been observed it can be employed to update the plausibility of each hypothesis. The method of doing this is described in Bayes' equation:

$$P(\boldsymbol{\lambda}|\mathbf{D}) = \frac{P(\mathbf{D}|\boldsymbol{\lambda})P(\boldsymbol{\lambda})}{P(\mathbf{D})} \tag{3.1}$$

This gives us a method of calculating $P(\boldsymbol{\lambda}|\mathbf{D})$, the posterior distribution, which is the probability that a set of parameters explains the data. $P(\mathbf{D}|\boldsymbol{\lambda})$ is the likelihood function detailing how likely the observed data values are for a given set of parameters $\boldsymbol{\lambda}$ and $P(\boldsymbol{\lambda})$ is our prior assumptions about the likelihood of any given set of parameters. $P(\mathbf{D})$, the probability of observing a given set of data, is a constant for those data and so can be ignored in our optimisation problems—in this context it is a normalisation constant.

3.2 Simple example

In this section, we will consider a simple inference problem to illustrate the different approach taken by a 'Bayesian' statistician when compared to the approach taken by a

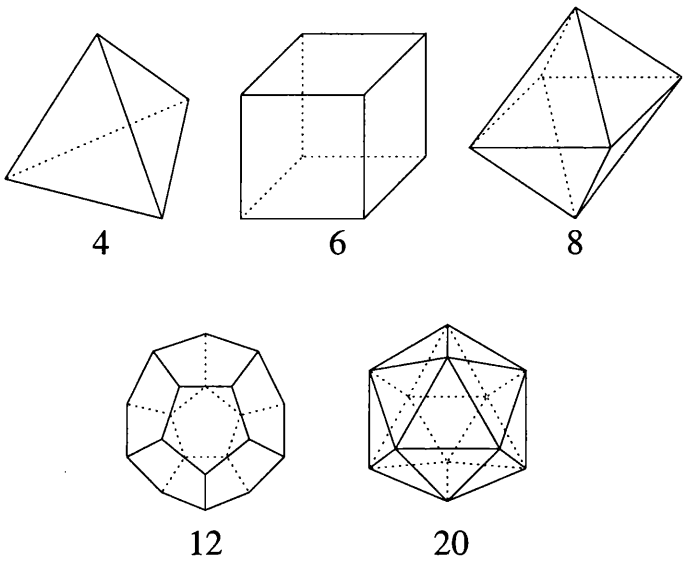


Figure 3.1: The five platonic solids.

'frequentist' statistician. The approach adopted by the imaginary frequentist statistician should be considered pedagogical and not a recommended method. The purpose is to illustrate the differences between the process of inference for the two methods which, in this case, leads to a sub-optimal choice of frequentist statistic.

The problem is to ascertain the number of sides of a die when only given the results of several throws of the die. Assuming the die is fair and has regular polygon sides of equal area, there are geometric constraints that restrict the number of sides of the die to be from the set $\{4, 6, 8, 12, 20\}$: the platonic solids (see figure 3.1). For the purposes of this illustration, the data is taken to be $D = \{4, 1, 2, 7, 3, 8\}$.

3.2.1 'Frequentist' approach

The process of inferring a hypothesis from a selection of hypotheses takes the following steps:

1. Given the problem, we choose a statistic that will differentiate between the contending hypotheses. In this example, the statistic can be defined: let s be the frequency of the number '1' in the data.

2. For each hypothesis, the expected value of the statistic is calculated, based on some theoretical (usually infinite) dataset. For example, a four sided die would have the statistic $s = \frac{1}{4}$.

3. For the observed data, calculate the value of the statistic. For this example: $s_D = \frac{1}{6}$

4. Compare each model against the data by comparing each model's expected statistic value with the data's statistic value. This is typically achieved by disproving some 'null hypothesis'. Hopefully, this will rejecting all but one hypothesis. For this example, the most likely solution is a six-sided die.

3.2.2 Bayesian approach

The Bayesian approach starts with a statement of how likely each model is before any data is known. This is expressed by the prior probability distribution $P(\boldsymbol{\lambda})$. A flat distribution implies no prior bias towards any one solution (H_n).

The steps in the inference are as follows:

1. Calculate the prior information. Dice with six sides are more common than dice with other number of sides and this information could be included in the analysis (*e.g.* we could take $P(H_n) \propto A_n$ where A_n is the sales of n sided dice). However, we shall assume no such prior information so that we simply take $P(H_4) = P(H_6) = \ldots = P(H_{20}) = \frac{1}{5}$. In fact, in this example, the inclusion of such prior information would not make much difference to the outcome of the inference.

2. The likelihood function, $P(d|\boldsymbol{\lambda})$, is considered for each hypothesis. If we assume that the die is fair then the following likelihoods are appropriate.

$$P(d|H_i) = \frac{1}{i} \qquad (d \leq i)$$
$$P(d|H_i) = 0 \qquad (d > i)$$

3. For each element of available data, update the probability of each hypothesis according to Bayes' theorem. Note that under Bayes' equation, data can be analysed either all at once or datum by datum without any consequence. Here we analyse the data roll-by-roll using the likelihood function described above.

D	$P(H_4\|D)$	$P(H_6\|D)$	$P(H_8\|D)$	$P(H_{12}\|D)$	$P(H_{20}\|D)$
$\emptyset$	0.20	0.20	0.20	0.20	0.20
{4}	0.37	0.25	0.19	0.12	0.07
{4,1}	0.54	0.24	0.14	0.06	0.02
{4,1,2}	0.68	0.20	0.09	0.025	0.005
{4,1,2,7}	0	0	0.83	0.15	0.02
{4,1,2,7,3}	0	0	0.89	0.11	0.01
{4,1,2,7,3,8}	0	0	0.9903	0.0093	0.0004

From the above table it is clear that the the die cannot be either four or six sided as there is at least one roll that cannot be generated by either sided die (the '7' for example). There are two interesting points to note. Firstly, this deduction was not imposed from *ad hoc* modeling of the system: we did not have to explicitly state that because there was a '7' that four and six sided dies were impossible. Instead, it arose naturally because if one datum excludes a hypothesis ($P(H_4|7) = 0$) then any data including that datum must equally be impossible ($P(H_4|4) \times P(H_4|1) \times P(H_4|2) \times 0 \times \ldots = 0$). Secondly, this effect is not limited to restricting impossible hypotheses: a collection of data that is unlikely but not impossible under one model can effect the overall inference procedure towards more likely models.

After completing the Bayesian analysis we see that a better frequentist statistic would be: smallest $n \in \{4, 6, 8, 12, 20\}$ such that $\forall d \in D \quad n \geqslant d$.

3.3 Prior information in signal analysis

We shall develop some terminology using the following example. In doing this, we shall place more restrictions on the class of problems that can be dealt with. However, despite these restrictions, the methodology is still applicable to a large class of problems and is likely to be of use in physically realistic analysis.

We will consider a set of I distinct problems which are labelled $i = 1, \cdots, I$. These problems are unrelated except for any constraints created by prior information. Each problem i involves fitting J pieces of data $\mathbf{d}_i = \{d_{ij}\}$ (where $j = 1, \cdots, J$) to a model $\mathbf{g}_i(\boldsymbol{\lambda}) = \{g_{ij}(\boldsymbol{\lambda})\}$. Although each problem could have a different number of data (J_i) we shall restrict our attention to problems that have the same number of data.

For each problem i, the model is described by the set of parameters λ_{ik} $(k = 1, \cdots, K)$. The overall task is to find the value of the parameters, $\boldsymbol{\lambda}$, that 'best describe' the data $\mathbf{D} = \{\mathbf{d}_i\}$ where $\boldsymbol{\lambda} = \{\lambda_{ik}\}$.

For ease of notation, $\boldsymbol{\lambda}$ will sometimes be written indexed by a single index: λ_m. The translation between this notation and the two-index form is simply $\boldsymbol{\lambda} = \{\lambda_{ik}\} = \{\lambda_m\}$ where $m = i + I(k-1) = 1, \cdots, M$ and $M = IK$.

If the data is assumed to be produced from the models $\mathbf{g}_i(\boldsymbol{\lambda})$ (for some $\boldsymbol{\lambda}$) obscured by Gaussian noise with a standard deviation of σ_D then the probability of any given set of data is:

$$P(\mathbf{D}|\boldsymbol{\lambda}) = \left(2\pi\sigma_D^2\right)^{-IJ/2} \exp\left\{-\sum_{i=1,j=1}^{I,J} \frac{[d_{ij} - g_{ij}(\boldsymbol{\lambda})]^2}{2\sigma_D^2}\right\} \tag{3.2}$$

$$= \left(2\pi\sigma_D^2\right)^{-IJ/2} \exp\left\{-\chi_D^2/2\right\} \tag{3.3}$$

where

$$\chi_D^2 = \sum_{i=1,j=1}^{I,J} \frac{[d_{ij} - g_{ij}(\boldsymbol{\lambda})]^2}{\sigma_D^2}. \tag{3.4}$$

If we have no prior knowledge of the parameters (that is, $P(\boldsymbol{\lambda})$ is constant) our task is to maximise equation (3.3). More usually, we think of this as minimising:

$$-\ln\left[P(\mathbf{D}|\boldsymbol{\lambda})\right] \tag{3.5}$$

which is equivalent to minimising χ_D^2. Thus, we have a Bayesian justification for the 'least squares' method.

However, if we do have some prior information about the parameters we must include the $P(\boldsymbol{\lambda})$ term in our analysis. We would then have to minimise:

$$-\ln\left[P(\mathbf{D}|\boldsymbol{\lambda})\right] - \ln\left[P(\boldsymbol{\lambda})\right] \tag{3.6}$$

which would amount to adding an extra χ^2 term to represent the prior information.

3.4 Example of prior information

Consider in each problem i, one of the parameters (λ_{i1}, say) may be most likely to lie around some value λ^* with a standard deviation σ_λ: that is,

$$P(\boldsymbol{\lambda}) = \left(2\pi\sigma_\lambda^2\right)^{-I/2} \exp\left[-\sum_{i=1}^{I} \frac{(\lambda_{i1} - \lambda^*)^2}{2\sigma_\lambda^2}\right] \tag{3.7}$$

$$= \left(2\pi\sigma_\lambda^2\right)^{-I/2} \exp\left[-\chi_\lambda^2/2\right] \tag{3.8}$$

where

$$\chi^2_\lambda = \sum_{i=1}^{I} \frac{(\lambda_{i1} - \lambda^*)^2}{\sigma^2_\lambda}. \tag{3.9}$$

The optimisation problem of equation (3.6) becomes simply the minimisation of:

$$\chi^2_D + \chi^2_\lambda. \tag{3.10}$$

Therefore equation (3.10) is the full minimisation problem including the prior information for this example.

In general, the probability distribution of the parameters will be more complicated than in the above example and may well involve cross terms between the different sets of data ($\mathbf{d}_i$). When this happens the number of dependent parameters, and hence the complexity, of the individual optimisation problems rises dramatically. Often what we are left with is a multi-dimensional nonlinear minimisation problem. Such problems are notoriously difficult to solve numerically due to problems of convergence and the existence of many local minima.

3.5 Errors

Confidence intervals are the correct way of dealing with uncertainty. For one-dimensional distributions, the relationship between error-bars and confidence intervals is that a $1 \times \sigma$ error-bar corresponds to a 68% confidence interval for normally distributed errors, $2 \times \sigma$ errors bracket a 95.4% confidence interval and so on.

With non-normally distributed errors, there will be a 68% confidence interval, a 95.4% confidence interval and so on, but they may be asymmetric with respect to the distribution's centre. Moreover, the extent of the 95.4% confidence interval may be different than twice the 68% confidence interval. Usually an error bar can be drawn but in general the interpretation of this interval will be different than for normal error bars.

To find an interval for a particular variable the full parameter space must be bracketed using some chosen shape (such as a ellipsoid). This region then contains the correct solution to the required probability. For the uncertainty in a single variable this region can be projected onto the required axis to produce the confidence interval in that variable.

This method of estimating the uncertainty in some variable requires knowledge of the distribution function around the solution (or at least around the best-fit point). In general this is difficult to obtain analytically and methods such a Monte Carlo simulation are required. However, a Monte Carlo simulation requires repeating the fitting procedure many times to obtain an estimate of the distribution function. Although Genetic Algorithms are an efficient global optimisation method they generally impose a non-trivial computational burden. The Monte Carlo simulation would be required for each recovered signal, which, for an acceptable uncertainty in the distribution function, would require solving at least 100 further fitting problems.

As the computational burden for the full uncertainty method is too great, the errors are estimated by computing the information matrix (see Kendall and Stuart [1963]). This compromise is equivalent to approximating the errors to being normally distributed. Sufficiently close to the solution, this approximation will be valid. A normal distribution is more compact than other distributions. This implies that a normal estimate of errors will over- rather than under-estimate the width of a confidence interval.

3.6 Summary

Bayesian Statistics allows systematic including of prior information. This prior information is introduced through Bayes' equation. With the more general understanding of probability as a measure of plausibility for some non-repeatable event, Bayes' equation can be viewed as a method of inference.

For the problem of model fitting, there is an increased computational burden as a result of this prior information. This is because the hitherto separate inference problems have been linked by a statistical constraint.

Chapter 4

Case study: Thomson scattering

In this chapter the problem of analysing the data produced by the Thomson Scattering Diagnostic at the COMPASS-D test reactor is discussed. This topic forms a case study of how the Genetic Algorithm with Bayesian Statistics method described in the previous chapter can be implemented.

There are several aspects of this problem that are specific to this case study and that simplify the analysis (such as the linear dependency of certain parameters), but the main concept, prior information resulting in improved parameter estimation, is applicable to a wide range of problems.

The first section of this chapter describes the physical arrangement of the experimental rig at the COMPASS-D test reactor. These details are expanded in section two, which describes the practicalities of how data is acquired at the reactor. Section three discusses the nature of the noise and section four describes how the signal to noise ratio can be estimated. This estimate is used in later statistical analysis of the signal. Section five describes a method of obtaining reference signals, which are used later for deducing the temperature.

The theory described in the previous chapter is applied to the scattering problem in section six and section seven discusses how the errors are quantified. Section eight outlines the method of converting between measured amplitudes and the inferred temperature and section nine describes how the errors in the signal-fitting amplitudes can be mapped into an error in the final temperature. Section ten contains recovered electron distributions.

4.1 Characteristics of Thomson Scattering

Thomson scattering is the scattering of light off mobile charged particles such as electrons. Classically, this scattering process takes into account no quantum or relativistic effects. Thus, for the treatment to be valid, the scattering particle must be moving slowly and the light must have a long wavelength. The condition for 'moving slowly' is $v \ll c$, where v is particle's speed and c is the speed of light in vacuum and the condition for long wavelength is $h\omega \ll mc^2$ where h is Planck's constant and m is the mass of the particle.

In the following sections, only the electrons of the plasma are considered. The electron will be far less massive than the plasma's ion species. Classically, the scattering process can be considered in terms of an incident oscillating electric field (from the incident light) causing the electron to oscillate. An oscillating charge will radiate, implying the electron oscillating as a result of the incident light will, by virtue of that oscillation, emit light. This is the scattered radiation. Since the electron's motion will have to overcome the effect of inertia, the more massive ions will oscillate to a far lesser extent. This results in an insignificant scattering off ions.

Although little light is scattered off the ions directly they may have an indirect effect on scattered radiation. A cloud of increased electron density, a Debye sphere with typical radius λ_D, surrounds each positive ion. These 'shield' the positive ion from the rest of the plasma resulting in a stable equilibrium configuration.

If the wavelength is comparable to the length-scale of these spheres then the fluctuations in electron density will result in scattering and the ions will affect the resulting spectrum. However, if $k_i\lambda_D \gg 1$, where k_i is the magnitude of the incident wave vector, then each Debye sphere will experience many wavelengths of the incident light and the scattering can be approximated by considering no ion correlation effects: the scattering is incoherent.

The scattering geometry is shown in Figure 4.1. The vectors $\mathbf{k}_i$ and $\mathbf{k}_s$ are the wave vectors of the incident and scattered light respectively. It is useful to define $\mathbf{k} = \mathbf{k}_s - \mathbf{k}_i$. The scalar values k_i, k_s and k are the magnitudes of $\mathbf{k}_i$, $\mathbf{k}_s$ and $\mathbf{k}$ respectively.

The total Thomson scattering cross-section (*e.g.* see Hutchinson [1987]) integrated over all solid angles is:

$$\sigma_T = \frac{8\pi}{3} r_e^2 \tag{4.1}$$

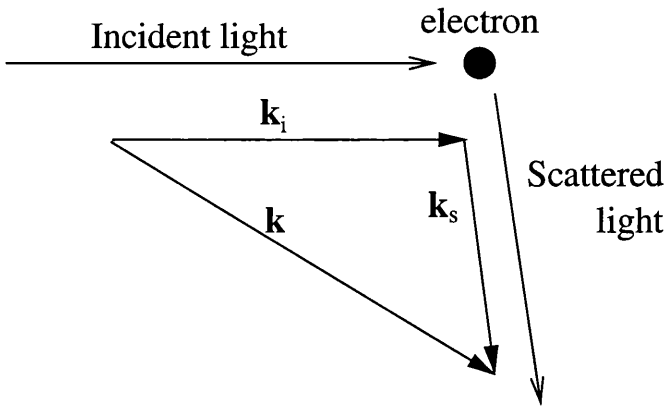


Figure 4.1: Overview of the scattering geometry in Thomson scattering.

where r_e is the classic electron radius $r_e \equiv e^2/(4\pi\epsilon_0 mc^2)$.

The oscillating electron behaves analogously to a dipole and the non-relativistic limit is referred to as the dipole approximation. The total scattered power from this approximation is given by (4.2), where $\langle S_i \rangle$ is the mean incident Poynting vector, which can be taken as the total beam power divided by the beam's cross-section area. $\hat{\mathbf{E}}_0$ is the unit vector for the incident electric field indicating the plane of polarisation. $f_k(v_k)$ is the projection of the distribution function onto the vector $\mathbf{k}$, *i.e.* $f_k(v_k) = \int f(\mathbf{v}_\perp, v_k)\mathrm{d}\mathbf{v}_\perp$. The projected distribution function for a (non-relativistic) Maxwellian distribution is given by (4.3).

$$\frac{\partial^2 P}{\partial\omega_s \partial\Omega} = r_e^2 \int \mathrm{d}^3 r \langle S_i \rangle |\hat{\mathbf{k}}_s \times \hat{\mathbf{k}}_s \times \hat{\mathbf{E}}_0|^2 f_k \left(\frac{\omega}{k}\right) \frac{1}{k} \tag{4.2}$$

$$f_k = n_e \left(\frac{m_e}{2\pi T}\right)^{\frac{1}{2}} \exp\left(-\frac{m_e v_k^2}{2T}\right) \tag{4.3}$$

Pechacek and Trivelpiece [1967] was the first correct relativistic treatment of inhomogeneous Thomson scattering. In this extension of the classic Thomson scattering process, the electrons are no longer required to be travelling slowly with respect to the speed of light. The power spectrum of scattered radiation is given by (4.4) where β is the electron velocity as a fraction of the speed of light, N is the number of scattering electrons and $\gamma = (1-\beta^2)^{-1/2}$. The tensor $\mathbf{I}$ is the identity tensor whilst $\mathbf{M}$ and $\mathbf{N}$ are given by (4.5) and (4.6) respectively.

$$\frac{\partial^2 P(\omega)}{\partial\omega\partial\Omega} = N\frac{cr_e^2}{8\pi}\int \mathrm{d}^3\beta\, f(\beta)\frac{[\mathbf{M}\cdot\mathbf{N}\cdot\mathbf{E}_0]^2}{(1-\boldsymbol{\beta}\cdot\hat{r})^5}\delta(\omega-\omega_d) \tag{4.4}$$

$$\mathbf{M} = \mathbf{I} - \hat{\mathbf{k}}_s\hat{\mathbf{k}}_s - \mathbf{I}\hat{\mathbf{k}}_s.\boldsymbol{\beta} + \boldsymbol{\beta}\hat{\mathbf{k}}_s \tag{4.5}$$

$$\mathbf{N} = \gamma^{-1}(\mathbf{I} - \boldsymbol{\beta}\boldsymbol{\beta})\cdot[\mathbf{I}(1-\boldsymbol{\beta}\cdot\hat{\mathbf{k}}_i) + \hat{\mathbf{k}}_i\boldsymbol{\beta}] \tag{4.6}$$

In (4.4), $f(\beta)$ is the distribution function for the electrons in the scattering region. For thermally relaxed electrons moving at relativistic speeds, this distribution function is given by (4.7) where $\alpha = mc^2/2k_BT$ and $K_2(x)$ is the modified Bessel function.

$$f(\beta) = [2\pi K_2(2\alpha)]^{-1}\alpha\gamma^5\exp(-2\alpha\gamma) \tag{4.7}$$

An approximate analytical solution to (4.4) with a electron distribution function given by (4.7) is derived in Zhuravlev and Petrov [1979]. The resulting form is given by (4.8).

$$\frac{\partial\sigma}{\partial\omega\partial\Omega} = \frac{r_0^2\omega^2\alpha^{-1}K_2^{-1}(2\alpha)}{2(1-2\omega\cos\theta+\omega^2)^{\frac{1}{2}}}\exp\left\{-2\alpha\left[1+\frac{(\omega-1)^2}{4\omega\sin^2(\theta/2)}\right]^{\frac{1}{2}}\right\} \tag{4.8}$$

The approximation used by Zhuravlev and Petrov [1979] is that the 'depolarisation term' is taken as constant over the velocity-space integration. This may be corrected for by introducing a scattering angle dependent correction faction $q(\theta)$. This correction can be calculated exactly for specific angles (as illustrated in Selden [1980]), but a more convenient form is the rational approximation discussed in Naito et al. [1993].

Several approximations are discussed in Naito et al. [1993]. The lowest order is:

$$q = 1 - 4\eta\zeta + O(\eta^2) \tag{4.9}$$

where $\eta = (2\alpha)^{-1}y$, $\zeta = xy$, $y = (x^2+u^2)^{\frac{1}{2}}$, $u = \sin\theta/(1-\cos\theta)$, $x^2 = 1+\epsilon^2/(2(1-\cos\theta)(1+\epsilon)$ and $\epsilon = \lambda_s/\lambda_i - 1$.

The (1,1) approximation is:

$$q = 1 - 4\eta\zeta\frac{2\zeta-(2-3\zeta^2)\eta}{2\zeta-(2-15\zeta^2)\eta} + O(\eta^4) \tag{4.10}$$

and the (2,2) approximation is:

$$q = 1 - 4\eta\zeta\frac{p_0 + p_1\eta + p_2\eta^2}{q_0 + q_1\eta + q_2\eta^2} + O(\eta^6) \tag{4.11}$$

where

$$\begin{aligned} p_0 &= q_0 = 4 + 30\zeta^2 - 55\zeta^4 \\ p_1 &= -\zeta(24 - 545\zeta^2 + 720\zeta^4) \\ p_2 &= 2(33 - 165\zeta^2 + 240\zeta^4 + 100\zeta^6) \\ q_1 &= 25\zeta^3(29 - 42\zeta^2) \\ q_2 &= 5(18 - 66\zeta^2 + 630\zeta^4 - 805\zeta^6) \end{aligned} \tag{4.12}$$

The resulting approximation, consisting of (4.8) multiplied by a correction term, q, was compared to results from numerically integrating (4.4) in Naito et al. [1993]. The relative errors for the (2,2) approximation with 180° scattering is less than 10^{-5}% at 10 keV, 10^{-3}% at 20 keV and less than 0.1% at 100 keV. The (1,1) approximation gives poorer results with relative errors of 10^{-3}% at 10 keV, 0.1% at 20 keV and $\approx$ 1% at 100 keV.

The (2,2) approximation has excellent agreement with numerical work for all reasonable temperatures. However, for the 90° scattering configuration, the (2,2) rational approximation has a pole. This is illustrated in figure 4.2. The (1,1) approximation does not have a pole at this location (also indicated in figure 4.2). To alleviate this problem, the (2,2) solution is used unless the correction term q is starting to diverge. If it does, then the (1,1) approximation is used instead.

Figure 4.3 shows the spectrum of the Thomson scattered light for several electron temperatures for monochromatic incident light. As the temperature is increased the scattered light's distribution is both broadened and the centre of the distribution is shifted towards higher frequencies. This blue shift is a result of the relativistic beaming effect, where light is scattered preferentially towards the observer when the electron is moving relativistically.

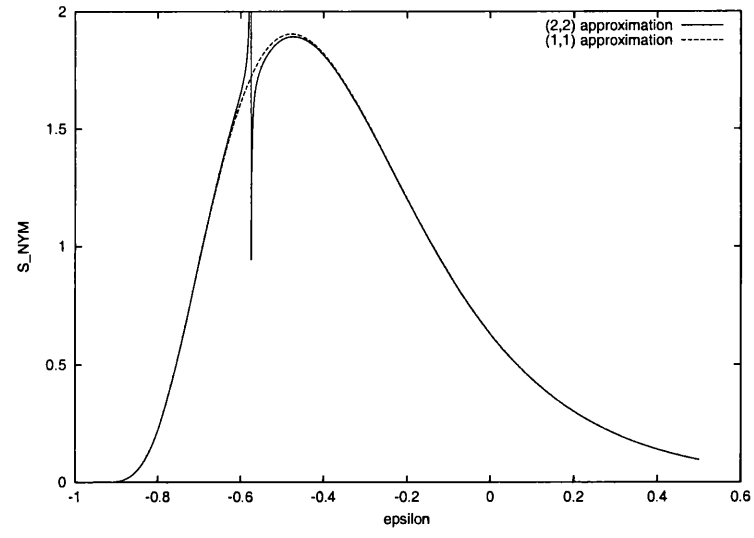


Figure 4.2: Theoretical Thomson scattering spectrum. (2,2) rational approximation contains a pole for $\theta = 90°$ whereas the (1,1) approximation does not.

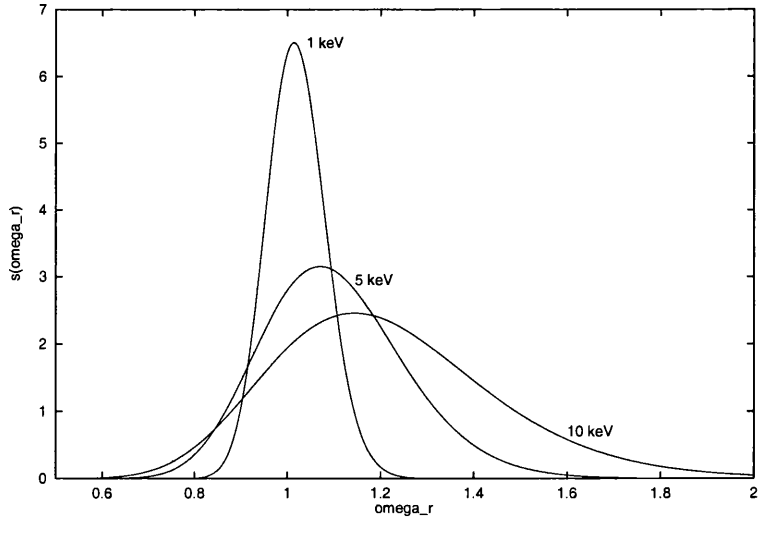


Figure 4.3: Theoretical spectra of Thomson scattered light for different electron temperatures.

4.2 Experimental setup at COMPASS-D

COMPASS-D is an experimental fusion reactor that contains plasma by the applied magnetic field configuration. The vacuum vessel is toroidal in shape and the magnetic field is set to have a strong, directly induced toroidal component with a secondary, current-generated poloidal component. This design is typical of tokamak devices.

At COMPASS-D, a neodymium doped yttrium aluminum garnet (Nd:YAG) laser is employed to produce the incident light (with a wavelength of 1064 nm) for Thomson scattering. This is used as a diagnostic that measures electron temperature. Nd:YAG lasers are preferable for performing Thomson scattering as they can be repeatedly fired at a high enough frequency that the evolution of the plasma can be studied. In addition there are fewer impurity lines around the 1064 nm wavelength than at the 694.3 nm regime at which ruby lasers operate; for example the H_α line is 656 nm and there are He lines at 587.6 nm and 656.3 nm (see Barth et al. [1997] and page 258 of Hutchinson [1987]). As discussed in §4.3, the laser is set to pulse with a pulse width of approximately 10 ns. This is repeated at 50 ms intervals allowing the laser to produce a typical total pulse energy of 1 J.

The laser pulses produce 'snap-shots' of information about the electron temperature distribution along the laser beam path within the tokamak. Each snap-shot is hereafter referred to as a segment. In each plasma shot, the laser fires test pulses before the plasma has formed. Segment 91 (counting the first segment as segment 0) is typically the first segment with useful results. The plasma typically lasts for several hundred milliseconds so that the plasma normally quenches before segment 99: the last recorded segment.

The physical setup of the laser is shown in Figure 4.4. The laser beam passes vertically through the plasma's centre and onto a beam dump, which absorbs the majority of any unscattered light. The laser beam path is imaged onto an array of 16 optical fibre ends via a lens within a view port in the vacuum vessel. Each fibre optic images a section of the laser beam approximately 24 mm in length and is hereafter referred to as a view.

The fibre-optics of selected views are connected to polychromators. These are located outside the immediate vicinity of the toroidal coils to reduce interference. Each polychromator consists of three detector channels, each of which has a filter that transmits a narrow spectrum of light and reflects the remainder. With each channel, the trans-

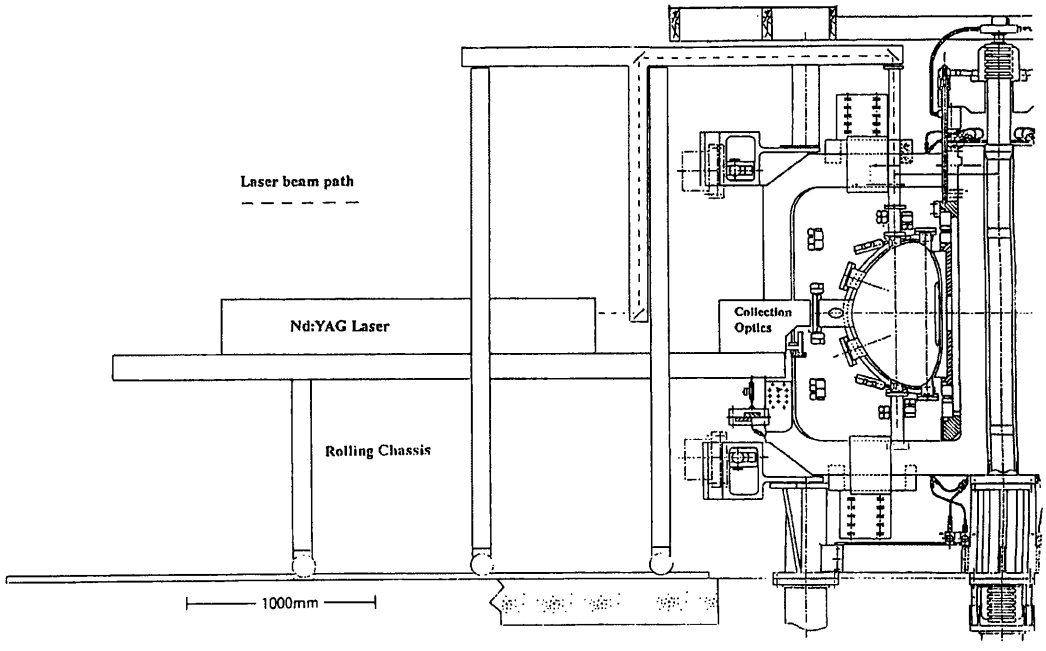


Figure 4.4: Schematics of the Thomson Scattering Diagnostics at COMPASS-D.

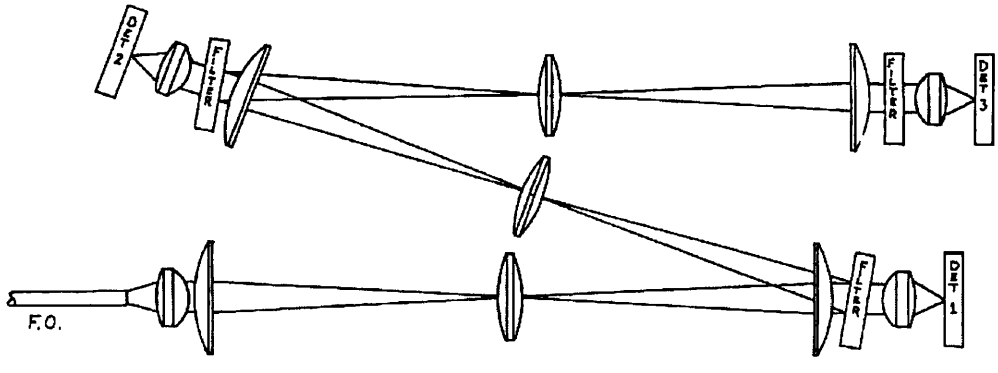


Figure 4.5: The construction of a polychromator.

mitted light is focused onto photodetector-amplifier assembly. The light reflected from one channel is directed towards the next channel. Figure 4.5 illustrates the construction of a polychromator.

The three channels of each polychromator produces three independent signals that are proportional to the integrated intensity over that channel's filter response curve. A typical frequency response of three channel filters is shown in Figure 4.6. Note that the filters transmit light to the shorter wavelength ('blue-shifted') side of the laser light's wavelength, which is marked "Nd:YAG" on the ordinance axis.

The scattered light's distribution, as illustrated in Figure 4.3, will pass through the

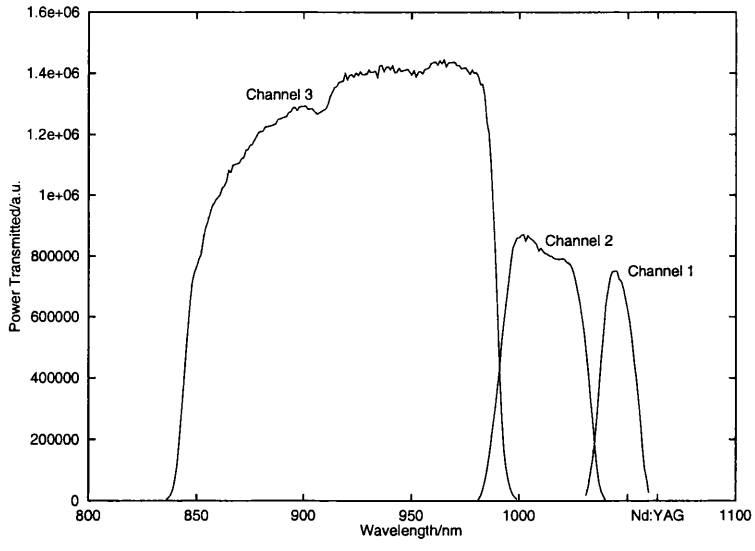


Figure 4.6: The spectral response of typical polychromator filters.

set of three filters of a polychromator. The theoretical output from these filters, as a function of electron temperature, can be computed. For filters with the spectral response shown in Figure 4.6, Figure 4.7 shows the expected form of the output from each channel as a function of electron temperature.

4.3 Data acquisition and triggering

The output from each channel will rise and fall over the temporal extent of the laser's pulse. In addition to recording any scattered light the detectors will also measure a background 'noise' consisting mainly of Bremsstrahlung radiation and line emission from the plasma. To make best use of the available information, the output from each channel is connected to a channel of a fast, digitising oscilloscope. Each laser pulse triggers the oscilloscope data-acquisition system which is operated in sequence mode with memory segmented and triggered to capture individual scattering pulses. The oscilloscopes sample every 2 ns for 504 ns each segment and are triggered slightly before the laser pulse. The triggering is to ensure the measured pulse activity in the channel's output occurs approximately in the middle of the segment's data.

The triggering of the oscilloscope is not exact. There is some 'jitter' in the exact position of the pulse. In order to analyse the distribution of 'jitter' we considered only data with a high signal to noise ratio and assumed the profile of the laser's pulse is approximately Gaussian. Data typical of this criterion is shown in Figure 4.8 along

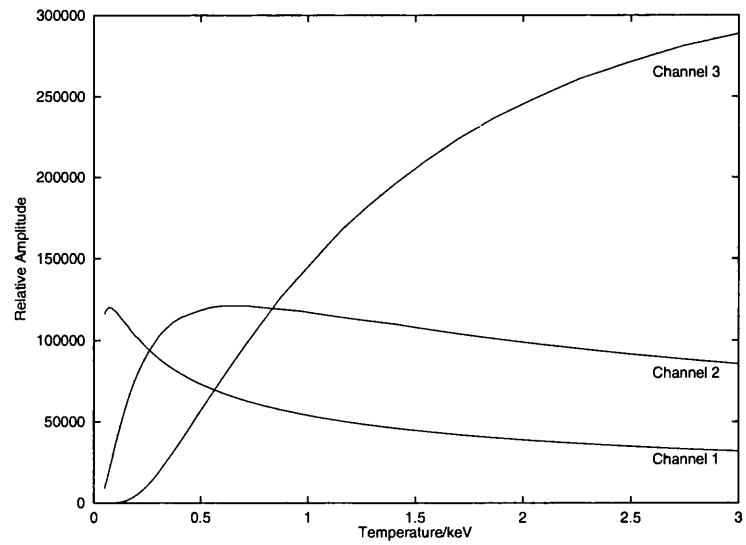


Figure 4.7: Theoretical channel output as a function of electron temperature.

with a Gaussian best fit. For sufficiently high signal-to-noise ratio, the position of the pulse can be found by fitting a Gaussian model described by (4.13) where A is the amplitude of the Gaussian, B is the offset to the data due to, for example, a constant DC current, i_0 is the centre of the distribution and σ_i is a measure of the signal's width.

$$f_i = A \exp\left[\frac{-(i-i_0)^2}{2\sigma^2}\right] + B \tag{4.13}$$

For some dataset g_i, the χ^2 statistic (4.14) is used to measure the extent particular values of A, B, i_0 and σ are inconsistent with the data: a smaller χ^2 indicates a model which is more consistent with the data.

$$\chi^2 = \sum_{i=1}^{I} \frac{(f_i - g_i)^2}{\sigma^2} \tag{4.14}$$

To find the values of A, B, i_0 and σ that best describe the data the χ^2 function is minimised. This minimisation problem is complicated by the fact that the Gaussian model is non-linear. The method used to achieve this minimisation is the Levenberg-Marquardt (LM) method. This method for solving non-linear minimisation problems is discussed in chapter 15 of Press et al. [1996].

The LM method returns the parameters that minimise χ^2 and an estimate of the uncertainty of each parameter. The minimum value of χ^2 is used to estimate the 'goodness of fit', *i.e.* to ascertain at what probability a value of χ^2 as poor as the recovered

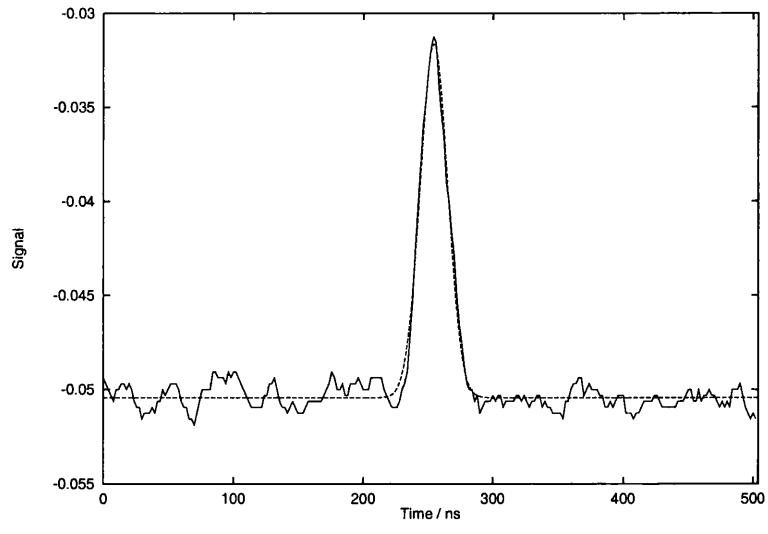


Figure 4.8: Shot 26921 Polychromator 9 Channel 1 Segment 93 and best fit Gaussian.

minimum might occurred by chance. This probability is given by the incomplete gamma function $Q(\frac{N-\nu}{2}, \frac{\chi^2}{2})$, where N is the number of data-points, ν is the number of degrees of freedom and $Q(a, x)$ is given by (4.15) where $\Gamma(a)$ is the gamma function.

$$Q(a,x) = \frac{1}{\Gamma(a)} \int_x^\infty e^{-t} t^{a-1} \mathrm{d}t \tag{4.15}$$

The LM method returns only the local minimum, so the initial value of the parameters must be sufficiently close to the correct answer for the algorithm to succeed. The initial values for A and B, as defined in (4.13), are the difference between the minimum and maximum data values and the minimum data value respectively. The width is estimated by considering the first and last points where the graph crosses the value halfway between the maximum and minimum data values. If the distance between these two points is less than half the total ordinance length of the data then it is used as the estimate of the width. If the distance between the two points is greater then the width defaults to a third of the ordinance length of the data. The method depends somewhat critically on the initial value of the Gaussian's centre. An initial value of the data's ordinance centre is the logical choice; however, in the presence of noise, there may be a local minimum in that vicinity. To alleviate this problem, each datapoint in turn is tried as the initial value of the Gaussian's centre. The Gaussian fit with the lowest χ^2 is used.

The method of estimating signal-to-noise ratio described in §4.5 was used to select

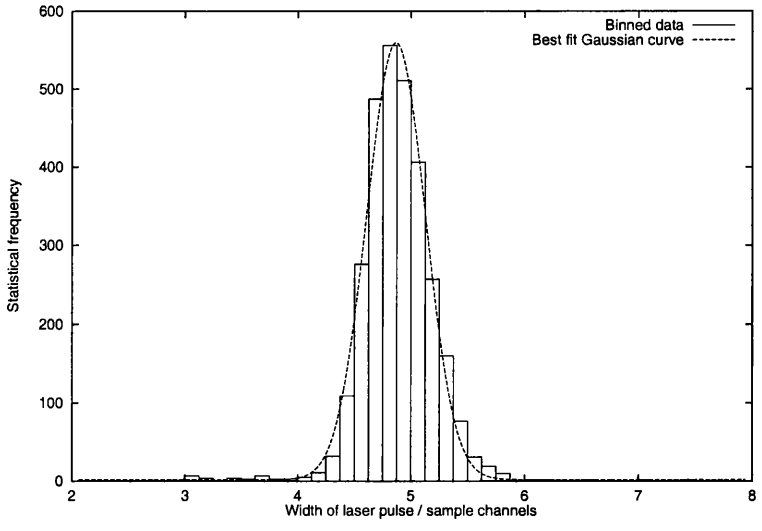


Figure 4.9: The distribution of σ_i measuring the width of the laser's temporal profile for the 3000 segments (of all polychromator channels) that have the highest signal-to-noise ratio. The best-fit Gaussian has width 0.52 ns and centre 9.7 ns.

the 3000 segment datasets with the highest signal-to-noise ratio from all polychromator channels. These segments were taken from data collected between 24 November 1997 (shot 25659) and 4 June 1998 (shot 27490). For each segment, the values of σ_i and i_0 that minimised the χ^2 statistic were calculated.

The distribution of laser widths (σ_i above) is shown in figure 4.9 and has been modelled by a Gaussian with centre of 4.867 and width of 0.2651. These values correspond to a laser width of 9.73 ± 0.53 ns.

For the offsets the situation is more complicated. The distribution of offsets is clearly non-Gaussian as can be seen from figure 4.10. Instead of fitting a Gaussian, the model fitted was a convolution of square pulse and a Gaussian. The normalised convolved distribution is:

$$f(y;a,b,\sigma) = \frac{1}{2}(b-a)^{-1}\left[\Phi\left(\frac{b-y}{\sigma\sqrt{2}}\right) - \Phi\left(\frac{a-y}{\sigma\sqrt{2}}\right)\right] \tag{4.16}$$

where Φ is the Error Function given by:

$$\Phi(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\mathrm{d}t \tag{4.17}$$

Scaling to reflect the integrated area of the data and solving a, b and σ using the LM method the best-fit values are $a = 118.9$, $b = 127.2$ and $\sigma = 0.91$.

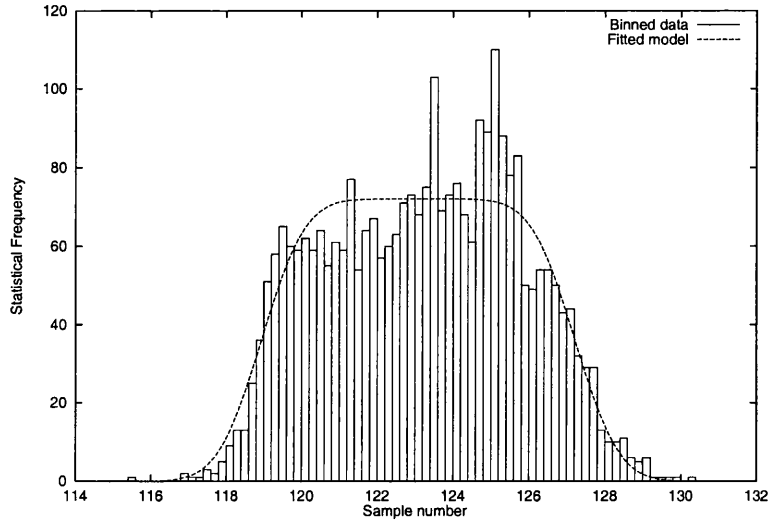


Figure 4.10: The distribution of sample number containing the laser's peak power for the 3000 segments (of all polychromator channels) that have the highest signal-to-noise ratio along with model fit.

The method described above analyses the three channel offsets independently: each channel's offset is with respect to the start of the segment. This is an optimal analysis if there is one source of the jitter that (by virtue of different channels having different temporal offsets) occurs after the beam passes through the polychromator. If there is some source of jitter before the polychromator (for example, some uncertainty about when the laser fires) then all three channels would be offset by the same amount. In this case the offsets would be distributed about some time μ^* (jitter after the polychromator) and μ^* is distributed about some point μ_0 (jitter before the polychromator).

The two sources of jitter can be illustrated by plotting the offset of channel 1 peak's centre against the offset of channel 2 peak's centre as shown in figure 4.11. The strong trend along the line $y = x$ is consistent with variability in the laser's firing time. The uncorrelated scatter around this trend is consistent with uncertainty due to oscilloscope timing and also in the Gaussian fitting procedure. In addition to the central trend data (species 1), there is a large number of 'outliers' (species 2): points that are sufficiently far away from the central trend that they constitute a subset of the total data with different origins.

The mean of the three offsets ($\overline{\mu} = (\mu_1 + \mu_2 + \mu_3)/3$) is an estimator of μ^*: the laser's true temporal offset. As there are only three data-points, this estimate will, in general, be poor. If $\overline{\mu}$ is a good estimate of μ^* then the offsets relative to $\overline{\mu}$ would become

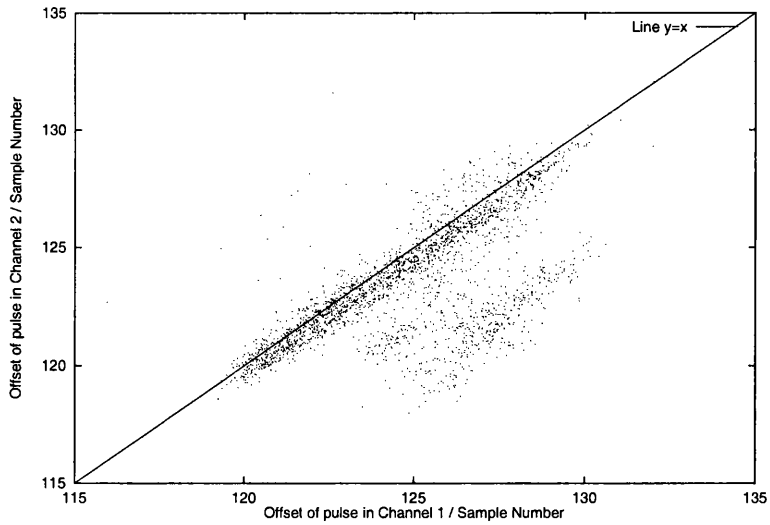


Figure 4.11: Scatter plot showing correlation between the pulse's offset in Channel 1 and its offset in Channel 2.

uncorrelated. Figure 4.12 shows a similar plot to figure 4.11, but using only species 1 data.

For each polychromator and segment, the sum of the signal-to-noise ratios for the three channels was used as a quality measure for that segment. The 3000 best overall signals were used to build the distribution of the relative offsets. This distribution is shown in Figure 4.13

The data was largely Gaussian, with small 'bumps' at relative offset values 3 and -2. Figure 4.14 shows the same data is in figure 4.13 but split into separate channels.

The peak at relative offset 2 is clearly evident in channel 1 only whilst the peak at relative offset -2 is from offsets derived from channel 2 only. Data lying within 1.5 of line $y = x$ in figure 4.11 are considered species 1. By selecting only this subset of data the extra channel-dependent bumps of figure 4.14 are eliminated, as shown in figure 4.15.

Recombining the three channels gives our prior: that the three channel offsets, relative to the mean offset of the three channels, are normally distributed with standard deviation of 0.6099 samples or 1.220 ns. This is illustrated in figure 4.16.

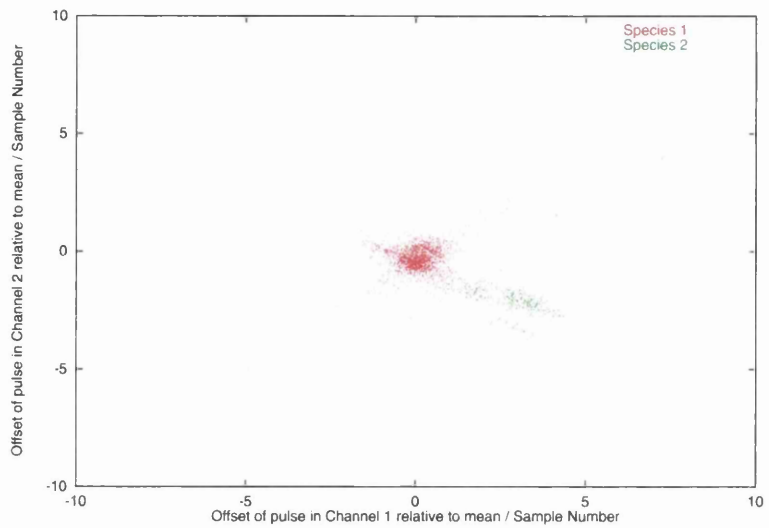


Figure 4.12: Scatter plot showing correlation between the pulse's offset in Channel 1 and its offset in Channel 2. Species are separated for clarity.

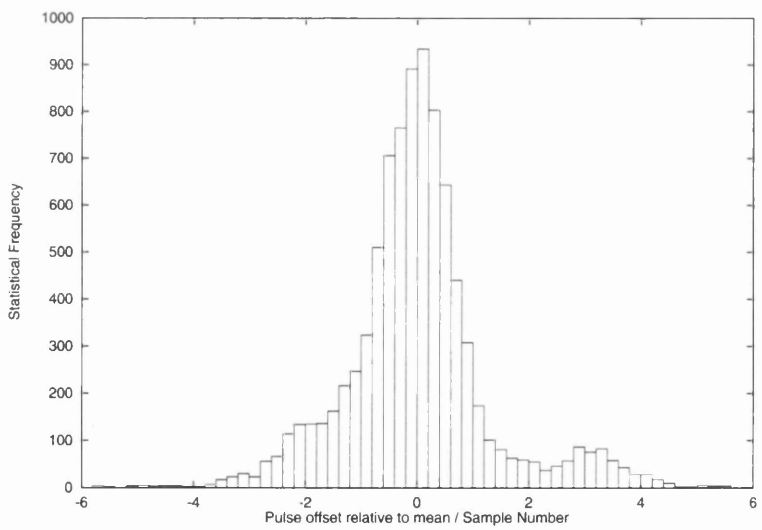


Figure 4.13: The distribution of $\mu_i - \overline{\mu}$ (where $i \in \{1, 2, 3\}$) for the 3000 best polychromator segments.

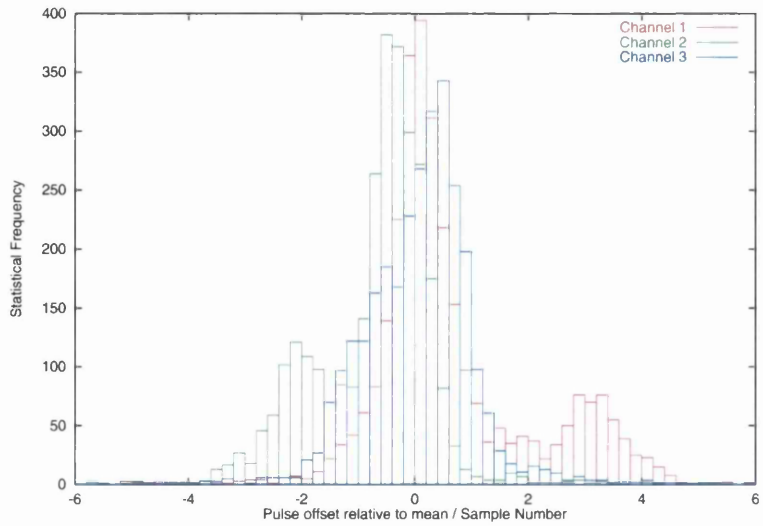


Figure 4.14: The distribution of $\mu_i - \overline{\mu}$ (where $i \in \{1, 2, 3\}$) for the 3000 best polychromator segments, split into the three channels.

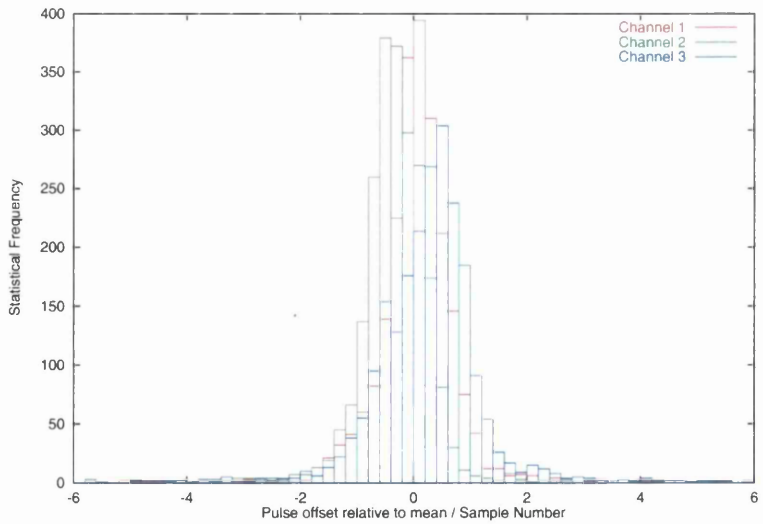


Figure 4.15: The distribution of $\mu_i - \overline{\mu}$ (where $i \in \{1, 2, 3\}$) for 'species 1' data selected from the 3000 best polychromator segments, split into the three channels.

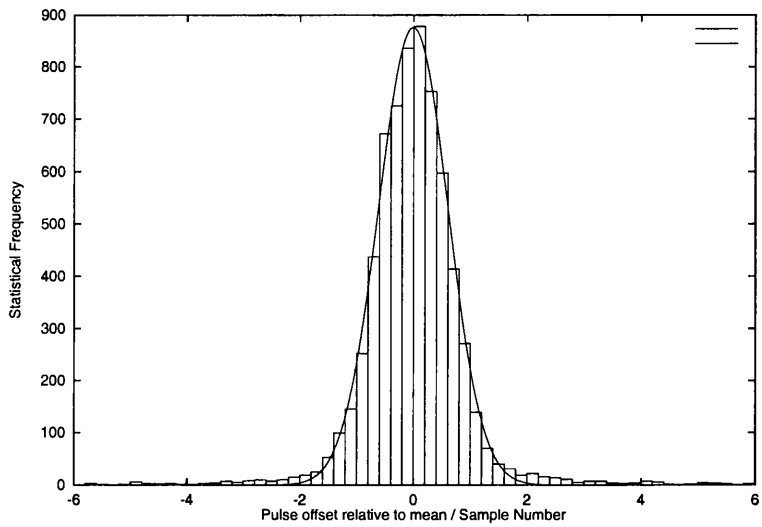


Figure 4.16: The distribution of $\mu_i - \overline{\mu}$ (where $i \in \{1, 2, 3\}$) for 'species 1' data selected from the 3000 best polychromator segments along with best fit Gaussian curve.

4.4 The noise

With all data, there is some noise that contributes to uncertainty in the measurements. The analysis of this noise is important as it allows the uncertainty in the recovered amplitudes to be estimated, either analytically or via a Monte Carlo simulation.

From typical low signal-to-noise ratio data (see figure 4.17), it is clear that the noise is derived from a non-Gaussian distribution since the noise varies smoothly from point to point. To discover the nature of the distribution, an estimate of the Power Density Spectrum (PDS) of the data was obtained. The PDS of a dataset describes to what extent the data is a result of various frequencies. A pure sinusoidal signal would have a sharp spike at the relevant frequency in its PDS whereas random noise appears as a flat PDS spectrum.

To obtain the PDS estimate, the datasets were windowed using a Welsh window, (4.18). The j^{th} datapoint is multiplied by the weighting given by (4.18), where N is the number of datapoints to be considered. Windowing reduces the 'leakage' from far away frequency bins at the expense of (slightly) broadening the central response. The Discrete Fourier Transform (DFT) is then taken using the Fast Fourier Transform (FFT) method (see chapter 13 of Press et al. [1996] for further details). This was repeated for several segments of low signal-to-noise ratio data and the results were averaged. The resulting PDS is shown in Figure 4.18.

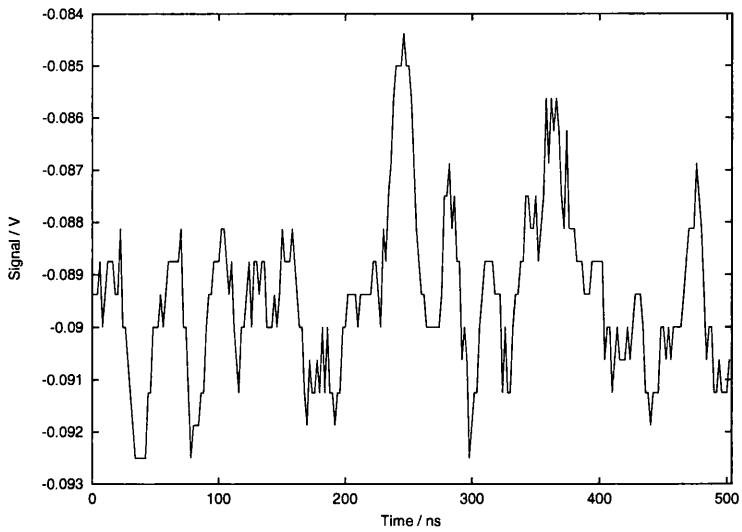


Figure 4.17: Typical medium to low signal-to-noise ratio data. Data from shot 26521 spectrometer 7 channel 3 segment 96.

$$w_j = 1 - \left(\frac{j - \frac{1}{2}N}{\frac{1}{2}N} \right)^2 \tag{4.18}$$

White noise is an uncorrelated random signal with a Gaussian probability density function: $(2\pi\sigma^2)^{-\frac{1}{2}} \exp[-x^2/2\sigma^2]$. The PDS of this is a flat level with amplitude of σ. When a signal passes through some bandwidth limiting device (such as an amplifier), higher frequency components of the signal are increasingly attenuated.

The overall PDS shown in figure 4.18 is not flat. This would imply a correlation between successive data-points. To test this hypothesis, the statistic described by (4.19) was calculated for sample data.

$$D = 2 \frac{\sum_{i=0}^{N-1} (x_i - \overline{x})^2}{\sum_{i=1}^{N-1} (x_i - x_{i-1})^2} \tag{4.19}$$

The expectation value of D for Gaussian based noise is 1. A value of D greater than 1 is consistent with noise in which each data point places a constraint on the successive data point. The value of D was calculated for some data and the values of D were drawn from a distribution where $D = 17.8 \pm 6.8$.

The following subsections discuss various aspects of the PDS which together form a model that can be used to generate 'fake' data by adding noise generated by the model to a reference signal.

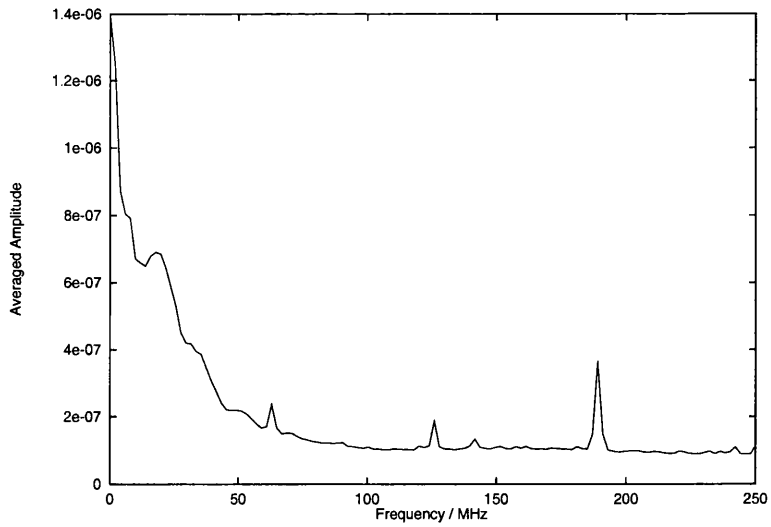


Figure 4.18: Power Density Spectrum estimate for the 'noise' in poor signal-to-noise data.

4.4.1 High frequency region

Figure 4.18 shows a flat region (except for the sharp features discussed in §4.4.2) beyond 100 MHz. This response is characteristic of uncorrelated or white noise. It is remarkable that the signal is flat up to the Nyquist frequency (ν_N) of 250 MHz.

The most likely cause of this flat response is 'quantisation noise'. The oscilloscope takes samples with 8-bit (binary digits) detail level, thus the signal is stored as an integer between 0 and 255. By imposing this constraint an error is introduced into the signal. This error is effectively uncorrelated as successive segments (*i.e.* sampling different signals) will have different quantisation error signature.

Provided the signal's dynamic range (difference between maximum and minimum values) is spread over sufficiently large number of sample bins, the error introduced by quantisation should be small. However, the dynamic range for many segments is sampled by few data-bins leading to the visible quantisation signal.

4.4.2 High frequency spikes

There are peaks at around 190 MHz, 125 MHz and 60 MHz. To establish if these are artifacts from the experiment several 'dark' segments were taken. A 'dark' segment is a dataset taken where there is no plasma and with no laser pulse. The peaks were present

in these segments. This then eliminates the plasma and laser as possible origins of the peaks.

For the special case of a flat spectrum with a single elevated data-point (*i.e.* an additional sinusoidal signal) adding a windowing function will smooth out that signal whereas the FFT without any window (effectively a top-hat window function) will preserve the signal (see chapter 13 of Press et al. [1996] for more in depth discussion on windowing functions). For the above spectrum, a non-windowed FFT reveals the spikes as having a width of a single frequency bin. The spikes are located at $\frac{3}{4}\nu_{\mathrm{N}}$, $\frac{1}{2}\nu_{\mathrm{N}}$ and $\frac{1}{4}\nu_{\mathrm{N}}$.

It seems highly likely that these peaks are artifacts introduced into the signals by the oscilloscopes. The effects of these peaks are largely negligible as the power in the peaks is small and their frequency is much higher than the typical frequencies contained in the reference signal.

4.4.3 Low frequency region

It is clear that the majority of the signal's power occurs below 60 MHz. This is consistent with some noisy source's signal that passed through a system that has a limited bandwidth. Suspicion immediately falls on the amplifiers that increase the signal level from the photodiodes. However, for this analysis, the source of the bandwidth limitation is immaterial.

The first Fourier channel contains the mean of the signal. As each shot has an arbitrary DC voltage offset, this channel is large. In order to prevent leakage from this channel into nearby channels, an estimate of the DC offset level (the median) was subtracted from each segment. The small signal that persists is indicative of a skewed distribution in the data. One possible cause of skew is the presence of a small (but non-zero) signal.

Ignoring the first two data-points (from signal contamination and window related leakage), the lower frequency response can be modeled as a Gaussian centred on zero with width σ_{d}. This is equivalent to white noise smoothed by a window with Gaussian weights.

4.4.4 Noise model

The ideas in the preceding subsections can be combined to form a model for the noise. If the quantisation noise and the $\frac{3}{4}\nu_{\mathrm{N}}$ spikes are ignored, the noise can be modeled by (4.20).

$$g * N(\sigma) \equiv \mathcal{F}^{-1}\left[\, g\, \mathcal{F}[N(\sigma)]\right] \tag{4.20}$$

where $*$ is the convolution operator, $\mathcal{F}$ and $\mathcal{F}^{-1}$ represent the Fourier and inverse Fourier transforms, σ is a measure of the intensity of noise before passing through the amplifiers and g is the amplifier response which, for simplicity, is taken to be a Gaussian centred at $\nu = 0$ and with width σ_{d}.

4.5 Estimating signal-to-noise ratio

A useful and intuitive statistic when dealing with signal analysis is the signal-to-noise ratio (SNR). The SNR is a measure of how much of the data contains information about the signal and is some measure of the uncertainties in measuring the signal. A typical high SNR dataset is shown in Figure 4.8. The data consists of a central peak with noise superimposed.

The method used to estimate the signal-to-noise ratio is to simply estimate the signal level and the noise level and to then take ratio of the two. The two estimates are discussed in the following two sections.

4.5.1 Estimating the signal level

The difficulty associated with estimating the signal level is due to the presence and nature of the noise within the measured signal.

Methods exist for removing noise whilst leaving some quantities of the signal intact. Whilst sources indicate that this is a dubious activity (for discussion see chapter 14 of Press et al. [1996]), the ability to emphasise any signal present suggests that that signal can, in principle, be measured. A primitive example of emphasising a signal in noisy data is the moving average: replacing each value with the average value of the four neighboring values. The moving average preserves the signal's zeroth order moment

and, if the signal is symmetric about some time, the first order moment; however, it distorts higher order moments. There are more sophisticated techniques (*e.g.* Savitzky-Golay smoothing filters and various Kalman filters) that preserve higher order features of the signal. However, these techniques work on the premise that the signal is slowly varying whilst the noise is random and rapidly changing. This is not the case with these datasets.

Clearly then, it is impossible to remove the noise from the signal; the estimate must work with the raw data. This becomes a problem for low SNR data where any slight signal is swamped by the similar noise and becomes indistinguishable.

Two methods for estimating the signal level are described below. Both make assumptions about the signal's form, but the first method is more general. The two methods are:

1. Find the maximum value of the dataset. Let the difference between the maximum value and the median value be the signal's strength. The median is a 'robust' estimator of a distribution's centre (see §15.7 of Press et al. [1996]). Is is less sensitive to outliers and differences in overall distribution shape than the mean and it is used to estimate the DC offset. This method assumes the signal is, at some point, greater than zero.

2. Assume the signal is of the form described in (4.13) where A is the amplitude of the signal, B is the DC offset, i is the time of the i^{th} data point, i_0 is the centre of the peak and σ_i is the width of the peak. This assumes a specific form of the distribution but provided the assumptions are valid, linear least squares is applicable to finding values of A and B. The analysis is similar to above. However, the value of σ_i is assumed constant and each data-point is tried as the centre of the Gaussian. This simplifies the minimisation and reduces the computational effort involved.

Both methods were tested using a Monte Carlo calculation of simulated data using the model described in §4.4. The results are plotted in Figure 4.19.

As both estimators will overestimate the signal level in the presence of noise, the minimum of both methods was taken as the estimate of the signal level.

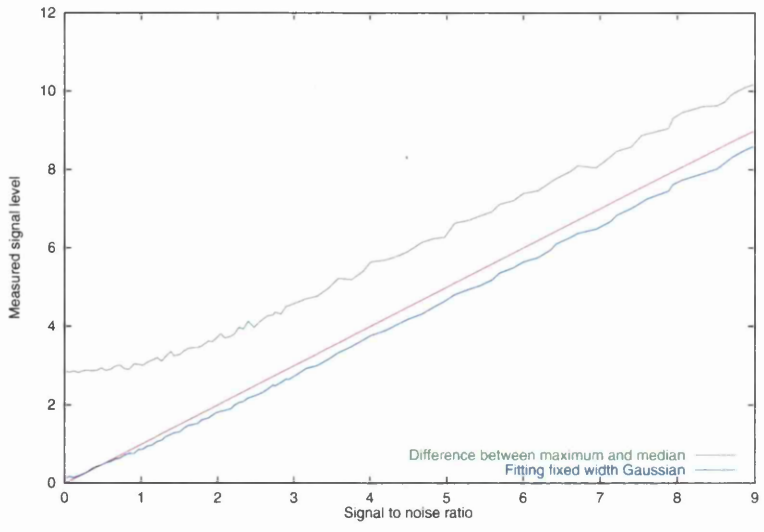


Figure 4.19: Comparison between two different methods of estimating signal level.

4.5.2 Estimating the noise level

Noise is usually taken to be derived from a Gaussian distribution, often with no formal justification that noise is truly from such a distribution. In general, as stated by the Central Limit theorem, large quantities of small random events tend towards a Gaussian distribution. In the case of data from the Thomson Scattering Diagnostic at COMPASS-D, the noise is clearly non-Gaussian. However, we shall take the noise as Gaussian. This has the effect of biasing the distribution width estimator of the noise towards higher values, but, as we are interested in the SNR mainly for ranking data, this does not matter.

With data from the COMPASS-D reactor, the segments typically will have some signal which is positive with respect to the noise (see Figure 4.8 for a sample segment). The presence of the signal will introduce a distorted 'tail' in the data's distribution in the positive half. This will bias the normal estimator of distribution width (the variance) towards larger values by an amount dependent on the signal strength.

With that in mind, six methods of estimating the noise-distribution's width were tested:

1. Using the Standard Deviation. This is the statistic σ where $\sigma^2 = \frac{1}{N-1}\sum_{j=1}^{N}(x_j - \overline{x})^{\frac{1}{2}}$ for the N data x_j. This is the maximal unbiased estimator of the distribution's width, provided the distribution is Gaussian. The presence of the tail in the positive half of the distribution will, therefore, be most noticeable to this estimate.

2. Using the Average Deviation. This is the statistic $\frac{1}{N}\sum_{j=1}^{N}|x_j - \overline{x}|$ for the N data x_j. The median x_{med} was substituted for $\overline{x}$ as it is a more robust estimator of the distribution's centre. The average deviation is a more robust estimate than the standard deviation as it weights more importance to points near the centre of the distribution.

3. Using the median as an estimator of the distribution's centre, the standard deviation of data less than the median is calculated, *i.e.* σ' where $\sigma'^2 = \frac{1}{N'}\sum_{j=1}^{N'}(x'_j - x_{\text{med}})^2$ where x'_j is the N' data less than the median x_{med}. If a signal is present in the data as additional values of zero or greater, then the data with value of less than the distribution's centre should be unaffected by the signal.

4. Removing the middle section of data points. The signal generally occurs in the middle of the segment. By simply removing the 30 middle data-points (those most likely to have some signal content) the standard deviation should yield a more accurate estimate at the expense of uncertainty due to the fewer data-points present.

5. Adaptively removing the middle section of data points. This is similar to the previous method, but with two steps. First the middle 30 data-points are removed and the maximum value of the remaining noise found, then all data points greater than this value are removed and the standard deviation calculated. This is to try and reduce the number of data-points removed from the segment.

6. Minimising the Kolmogorov-Smirnov (K-S) statistic to fit a Gaussian to the signal's distribution. The K-S statistic measures the 'difference' between two distributions and is defined as $D = \max_{-\infty<x<\infty}|S_{N_1}(x) - S_{N_2}(x)|$ where S_{N_1} and S_{N_2} are the cumulative frequency distributions for the two distributions to be compared, *i.e.* $S_{N_1} = \int_{-\infty}^{x} P(N_1)\mathrm{d}x$. For this case, the cumulative frequency of the data is calculated and compared against that of a Gaussian. The variance of the Gaussian is altered so that the statistic is minimised. The minimisation was done via an algorithm due to Brent; see chapter 10 of Press et al. [1996] for implementation discussion.

The results of a Monte Carlo simulation are shown in figure 4.5.2. The model of the noise described in §4.4 was used to generate the noise at a prescribed level and a reference signal was used to construct the signal element.

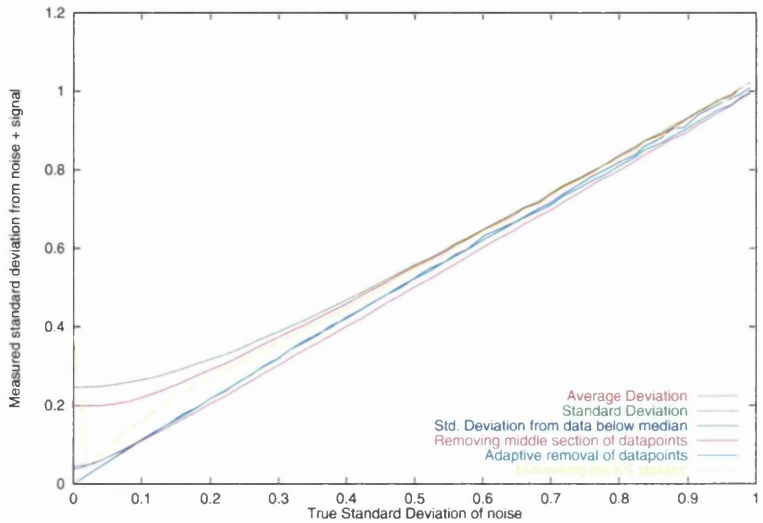


Figure 4.20: Comparison of various methods of estimating the noise level for high S/N level signals.

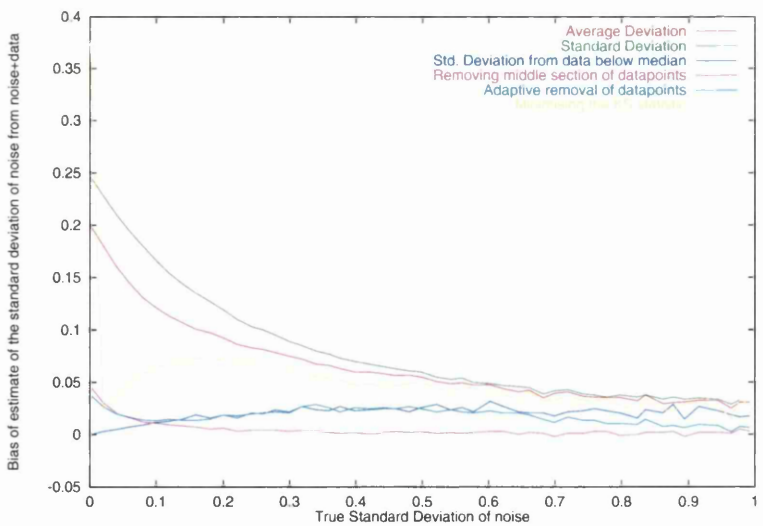


Figure 4.21: Comparison of the bias of each each estimate of the noise level

The Standard Deviation deviates by the greatest amount, as expected. The Average Deviation is recommended as a robust estimator of the distribution's width when the distribution has interference in its tails. However, other methods provided a better estimate of the noise level in this case.

The minimum of methods 3 and 4 was taken as the estimate of noise level.

4.6 Obtaining reference signals

When first analysing the data from the oscilloscopes, reference signals for each channel of each polychromator were supplied. These reference signals were either one 'good' dataset or the average of a few such segments.

Reference signals were used for fitting instead of using an analytical function. This was to allow for any instrumentation effects in the electronics or optics of the diagnostic without having to model the total system.

There were several problems with the reference signals as supplied. The reference signals were 'hand made'. This may have lead to bias in the reference signal (for example, what makes a signal 'good'?). Also, for certain channels of particular polychromator, no discernible signal was present in the reference signal! Because of these limitations, reference signals were produced via the GA driven Bayesian Statistics method.

The problem associated with extracting the reference signal is that the signals occur at different times: they are not aligned. The problem of finding a reference curve can therefore be split into two parts: aligning the datasets and extracting the reference signal.

If the datasets were aligned then intuitively, averaging the data would produce an estimate of the reference signal provided the distribution the noise is taken from is symmetric about zero. This can easily be seen by imagining some collection of datasets $(\mathbf{d}_j)$ that contain the reference signal with an arbitrary amplitudes, an arbitrary offset and some noise $(a_j\mathbf{r} + \mathbf{1}(b_j + \epsilon_j))$. Averaging these signals gives $\sum[a_j\mathbf{r} + (b_j + \epsilon_j)\mathbf{1}]/I$ which is $\mathbf{r}\sum a_j/I + \mathbf{1}\sum b_j/I + \mathbf{1}\sum \epsilon_j/I$. With a symmetrical distribution about zero, we would expect $\sum \epsilon_j$ to be less that $I\langle\epsilon\rangle$ in general.

More rigorously, the optimal I-point reference signal (r_i) derived from a set of J noisy datasets (d_{ij}) can be derived via the linear least squares method. Considering all fitting

problems together, (4.21) describes the simple model for the reference fitting in terms of a set of J amplitudes and offsets, a_j and b_j respectively.

$$d_{ij} = a_j r_i + b_j \tag{4.21}$$

The total χ^2 for this fitting problem is defined in (4.22) where σ_j^2 is the variance of the j^{th} dataset's noise.

$$\chi^2 = \sum_{ij} (d_{ij} - a_j r_i - b_j)^2 / \sigma_j^2 \tag{4.22}$$

Requiring the reference signal to have mean zero and unit variance and assuming the datasets have had their means subtracted, the values of a_j and b_j are given by (4.23) and (4.24) respectively.

$$a_j = \sum_i d_{ij} r_i \tag{4.23}$$

$$b_j = \sum_i d_{ij} / I \tag{4.24}$$

Substituting (4.23) and (4.24) into (4.22) and solving $\frac{\partial}{\partial r_i}\chi^2 = 0$ gives (4.25). This indicates that a weighted average (using the signal-to-noise ratio as the weight) yields the optimum estimate of the reference signal, provided the noise is normally distributed. As stated in §4.4, the noise is not drawn from a normal distribution. Therefore, (4.25) is not the optimum estimator for the reference signal. However, it will be an estimator and probably not too bad.

$$r_i = \sum_j \frac{a_j}{\sigma_j^2} d_{ij} / \sum_j \frac{a_j^2}{\sigma_j^2} \tag{4.25}$$

The problem of aligning the datasets can be tackled by Bayesian analysis. Using the normal Bayesian formulation, we see that:

$$P(\boldsymbol{\mu}|D) = P(D|\boldsymbol{\mu}) P(\boldsymbol{\mu}) / P(D) \tag{4.26}$$

where $\boldsymbol{\mu}$ is the set of alignment offsets for the data and D is the set of datasets. $P(\boldsymbol{\mu}|D)$ is the posterior probability, *i.e.* how likely the value of $\boldsymbol{\mu}$ is correct. $P(\boldsymbol{\mu})$ is the prior

information, in this case that the offsets should be from some Gaussian distribution (see Figure 4.10). $P(D|\boldsymbol{\mu})$ is the likelihood function. This can be estimated by first deriving the reference curve for the value of $\boldsymbol{\mu}$, then calculating some statistic that measures 'goodness of fit,' *e.g.* the χ^2 statistic. $P(D)$ is a constant with respect to the offsets and therefore can be ignored.

The problem was split into the two parts described above. The non-linear part (aligning the datasets) was solved by a GA whilst Linear Least Squares was employed to fit the resulting reference curve to each dataset so the value of $P(D|\boldsymbol{\mu})$ could be calculated.

After the GA has chosen some set of offsets and the corresponding reference signal was found, an estimate of the likelihood of this set of offsets is evaluated by fitting the reference signal to all the aligned signals, as described above. Any uncertainty in the reference signal will be manifest in the distribution of the residuals after fitting the reference signal as described by:

$$\{\delta_i\} = \{d_{ij} - (a_j r_i + b_j) : j \in [1, J]\} \tag{4.27}$$

where a_j and b_j are defined above.

Consider one point, i, along the reference curve. If the reference curve fits each equivalent point exactly then the set of residuals, $\{\delta_i\}$, will contain J zeros. This implies that r_i is exact. However, if the values in $\{\delta\}$ are scattered in some distribution then the width of this distribution is an estimate of the error of r_i: the standard deviation of $\{\delta_i\}$ measures the uncertainty of the reference point.

4.7 Analytic technique of fitting the signals

Using the terminology of section §3.1, we identify the number of problems (I) with the number of channels. The number of data (J) in each channel is 252 so the set of data for a given segment and polychromator is d_{ij} where $i = 1, 2, 3$ and $j = 1, \cdots, 252$. For each channel we have a reference signal $\mathbf{f}_i = \{f_{ij}\}$ derived from previous shots (see §4.6). The received signal in channel i ($\mathbf{d}_i$) has the same shape as the reference signal but with an amplitude a_i, a DC bias b_i, and a time offset μ_i: hence our model has the form:

$$g_{ij}(\boldsymbol{\lambda}) = \begin{cases} a_i f_{i(j-\mu_i)} + b_i & \text{if } 1 \leqslant j - \mu_i \leqslant J \\ b_i & \text{otherwise.} \end{cases} \tag{4.28}$$

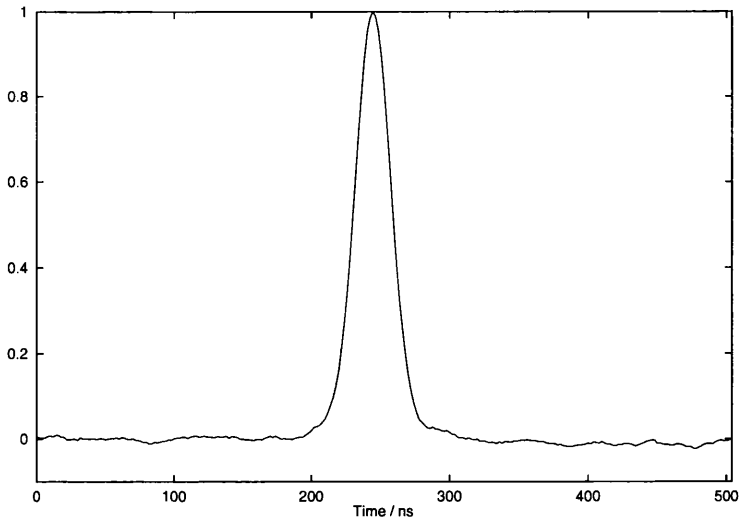


Figure 4.22: An example reference signal.

For a given set of data $\mathbf{d}_i$ we have three curves, $\mathbf{g}_i(\boldsymbol{\lambda})$, to fit, described by 9 parameters (the three dimensional vectors: $\mathbf{a} = \{a_i\}$, $\mathbf{b} = \{b_i\}$ and $\boldsymbol{\mu} = \{\mu_i\}$). Assuming Gaussian noise with standard deviation σ_D, the overall task becomes the minimisation of the Least Squares function of (3.1), where:

$$\boldsymbol{\lambda} = \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \boldsymbol{\mu} \end{pmatrix}. \tag{4.29}$$

Without prior information, the analysis is simply to minimise (3.4), with $g_{ij}(\boldsymbol{\lambda})$ given by (4.28), *i.e.* the standard Least Squares method.

This minimisation can be achieved whilst treating the three channels separately so the problem reduces to three separate ones, each with only three parameters. Of these three parameters, a_i and b_i appear linearly in (4.28) and so may be minimised by matrix inversion (see Mood and Graybrel [1974]). Only μ_i appears nonlinearly so we are left with a rather trivial one dimensional minimisation problem.

However, this method fails badly when the signals amplitude becomes comparable to the background noise. To produce a better fit for the data given, we include prior information in the problem using the Bayesian formulation of section 3.1. In the Thomson Scattering Diagnostic, for a given segment, the pulse should appear in the three channels of the data at the same time. However, there is a spread in arrival times as discussed in §4.3. Our prior information is that the relative spread of the three peak offsets will

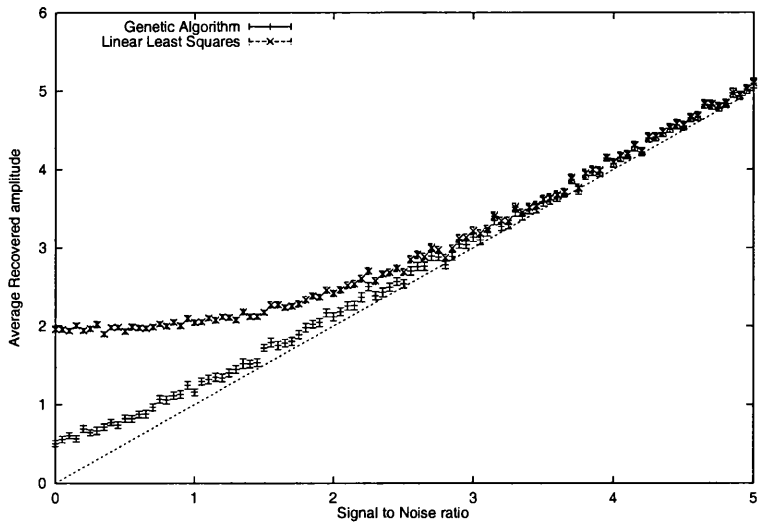


Figure 4.23: Comparison of the performance of Genetic Algorithms against a simple Linear Least Squares. The line $y = x$ is included for reference.

be normally distributed about the mean offset of the three peaks ($\overline{\mu} = [\mu_1 + \mu_2 + \mu_3]/3$) with standard deviation $\sigma_\mu = 3.0$ ms:

$$P(\boldsymbol{\lambda}) = (2\pi\sigma_\mu^2)^{-3/2} \exp\left\{-\sum_{i=1}^{3} \frac{[\mu_i - \overline{\mu}]^2}{2\sigma_\mu^2}\right\} \tag{4.30}$$

$$= (2\pi\sigma_\mu^2)^{-3/2} \exp\left\{-\chi_\mu^2/2\right\} \tag{4.31}$$

where

$$\chi_\mu^2 = \sum_{i=1}^{3} \frac{[\mu_i - \overline{\mu}]^2}{\sigma_\mu^2}. \tag{4.32}$$

This is a precise statement of our prior information, based on physical observation, but without over constraining the solution: the pulse must occur 'near-simultaneously' in each channel but not necessarily at the same time.

Applying the Bayesian method of §3.1, (3.1) gives us a χ^2 of:

$$\chi^2 = \chi_D^2 + \chi_\mu^2. \tag{4.33}$$

The inclusion of the prior information gives a significant improvement in data analysis, especially when signal to noise ratio is low. To illustrate this, figure 4.23 shows the performance of the complete analysis (minimising $\chi_D^2 + \chi_\mu^2$) by the Genetic Algorithm technique compared to that of Least Squares (minimising just χ_D^2). At all points, noise was generated at a constant level using the model discussed in §4.4. A reference signal was added with the required amplitude to achieve the signal to noise ratio. Three sets

of such data were generated which the Genetic Algorithm technique (full analysis) and a simple Linear Least Squares technique were required to fit. The average of the three amplitude was recorded and the process was repeated 100 times to obtain a reasonable estimate of this value. The line $y = x$ is included in the graph to show the actual signal level. The Linear Least Squares curve deviates markedly from this line for low signal to noise ratios whereas the complete analysis deviates by a lesser extent.

The resulting increase in complexity from including the prior information is manifest in that χ^2 is no longer separable into the χ^2_D of the three separate channels. The overall task is now to minimise a nonlinear function of nine variables, which is readily handled by our Genetic Algorithm technique.

4.8 Quantifying the errors in fits

Errors were calculated using the information matrix: the inverse of the covariance matrix. The information matrix, α, is given by (4.34).

$$\alpha_{mn} = \frac{\partial^2 \chi^2}{\partial \lambda_m \partial \lambda_n} \tag{4.34}$$

Considering the information matrix amounts to assuming that data points are taken from a normal distribution centred on some 'correct value' $\boldsymbol{\lambda}^0$, thus:

$$P(\boldsymbol{\lambda}) = (\pi)^{-\frac{1}{2}} \det(\alpha)^{\frac{1}{2}} \exp\left[- \sum_{m=1,n=1}^{9,9} \left(\lambda_m - \lambda_m^0\right) \alpha_{mn} \left(\lambda_n - \lambda_n^0\right) \right]. \tag{4.35}$$

The information matrix α represents a set of correlated errors for the nine parameters of the minimisation task. We then marginalise this matrix to one for only the three amplitudes, a_i, by integrating over the other six parameters. The result is a 3×3 information matrix, $\alpha^{(\mathbf{a})}$, that describes the correlated errors for the amplitudes alone. For uncorrelated errors in the three amplitudes $\alpha^{(\mathbf{a})}$ has the form:

$$\alpha^{(\mathbf{a})} = \begin{bmatrix} 1/2\sigma_{a_1}^2 & 0 & 0 \\ 0 & 1/2\sigma_{a_2}^2 & 0 \\ 0 & 0 & 1/2\sigma_{a_3}^2 \end{bmatrix} \tag{4.36}$$

where σ_{a_1}, σ_{a_2} and σ_{a_3} are the standard deviations of the three measured amplitudes. Thus the Probability Density Function is reduced to:

$$P(\mathbf{a}) = (\pi)^{-\frac{1}{2}} \det(\alpha^{(\mathbf{a})})^{\frac{1}{2}} \exp\left[- \sum_{m=1,n=1}^{3,3} \left(a_m - a_m^0\right) \alpha_{mn}^{(\mathbf{a})} \left(a_n - a_n^0\right) \right]. \tag{4.37}$$

For data with normal errors, the uncertainty σ_{a_j} in the amplitude a_j is given by:

$$\sigma_{a_j}^2 = \sum_{i=1}^{N} \sigma_i^2 \left(\frac{\partial a_j}{\partial y_i} \right)^2 \tag{4.38}$$

where σ_i^2 is the variance of the i^{th} data-points' normal error, y_i is the i^{th} data-point and N is the number of data-points.

For this fitting problem and assuming constant normal errors across each segment of σ_j^2, the estimate is given by:

$$\sigma_{a_j}^2 = \frac{\sigma_j^2}{\sum_{i=1}^{N} f_i^2} \tag{4.39}$$

4.8.1 The effects of non Gaussian noise on uncertainty in amplitudes

As discussed in §4.4, the noise was found to be non-Gaussian. The 'correct' method of dealing with non-Gaussian errors is to conduct a Monte Carlo simulation. For each set of recovered amplitudes, a statistically large number of synthetic data would be generated: for example, by using the recovered amplitudes, the reference signals and the model described in §4.4. The fitting procedure would be run against each set of 'fake' data. By looking at the statistical behaviour of the recovered amplitudes (for example) the nature of the errors of the recovered amplitude is gleaned.

If the Monte Carlo simulation contains too few fake datasets then the resulting distribution of recovered amplitudes will be highly and randomly distorted: the measure of uncertainty would be unreliable. Therefore, the Monte Carlo process requires a large number of datasets which, in turn, requires a large number of GA runs to fit each fake dataset. The GA fitting procedure is expensive (in terms of processing time) compared to simple local minimisation routines and, in general, Monte Carlo simulation cannot be used because it would take too long.

Instead of using the Monte Carlo method for each GA fit one can assume that, for sufficiently large signal-to-noise ratios, the recovered fit will have approximately normal errors. There are two points to note. First, these errors will, in general, be different from the linear estimate given in (4.39). Second, as the signal-to-noise ratio decreases the signal would eventually become 'lost in the noise' and (4.39) will cease to give any meaningful information about the uncertainty in the signals' amplitudes.

To analyse this approximation, a Monte Carlo simulation of the error in recovered amplitude was conducted. For each of the 41 selected data-sets the amplitude of the

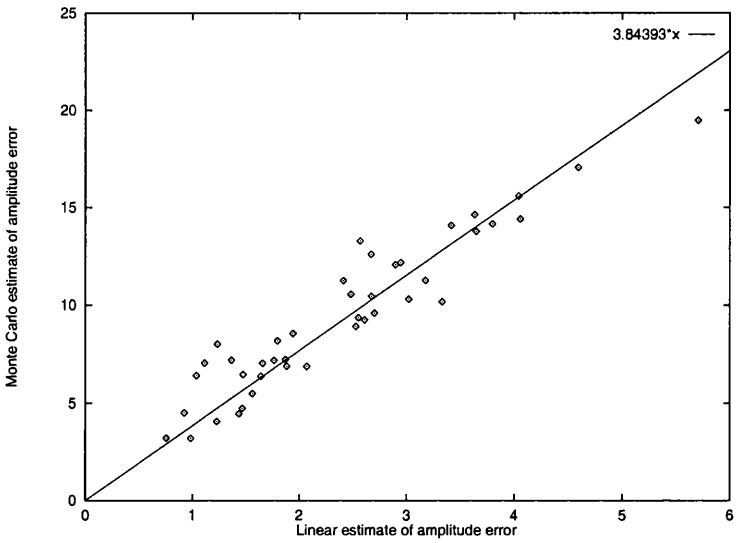


Figure 4.24: Correlation between error estimates of low-frequency pass filtered noise.

signal was determined using the GA technique and the uncertainty in the amplitude was estimated (assuming uncorrelated Gaussian noise). Once the amplitude was determined, consistent noise was constructed using the limited band-width model (see §4.4) and the reference curve (scaled to match the determined amplitude) was added. The amplitude of the GA fit to the resulting data was noted. This was repeated 1000 times and the standard deviation of the distribution of amplitudes was taken. Figure 4.24 shows the correlation between the Monte Carlo estimate of the uncertainty and the uncorrelated estimate.

The results suggest a simple numerical correction allows the correlated, non-Gaussian noise to be treated as uncorrelated Gaussian noise.

The cut-off after which the linear estimate gives no information about the signal (when the signal is lost in the noise) is less of a problem. This occurs when a signal is faint. By considering figure 4.25, this is most likely a problem for channel 3 for electron temperatures of less than 0.3 keV. However, the actual value of channel 3's signal amplitude does not matter much as the graph is relatively flat for that region. Thus any error in channel 3 (including an incorrectly estimated uncertainty) will not effect the final value of the electron temperature.

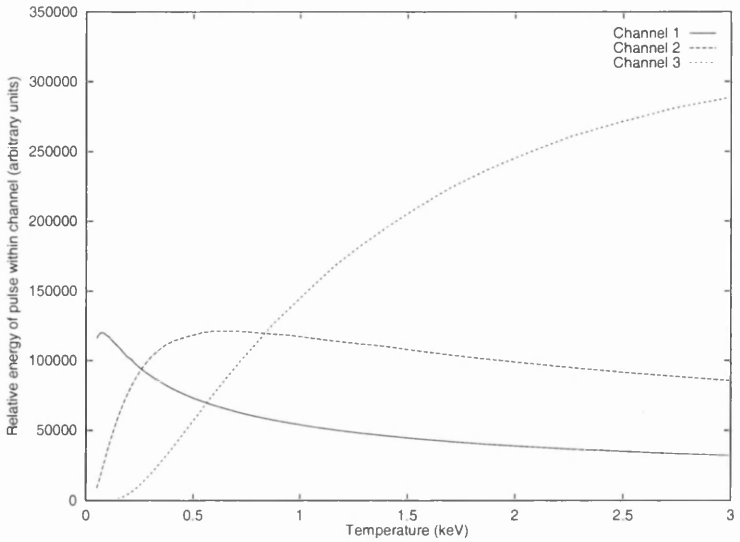


Figure 4.25: Theoretical spectral density function, calculated for incident wavelength of 1064 nm scattered through 87.7°.

4.9 Calculating the temperature

To derive the electron temperature the amplitudes of the three channels of the polychromator, a_i, are fitted to the theoretical scattering functions, $S_i(T_e)$, for a given electron temperature (T_e), assuming a Maxwellian velocity distribution for the electrons. The analytical form of $S_i(T_e)$ is discussed in Selden [1980].

Finding the correct temperature for a given set of amplitudes, $\mathbf{a}$, is equivalent (see Bindslev [1999]) to minimising the function:

$$\chi^2_{T_e}(\mathbf{a}, \boldsymbol{\xi}) = \sum_{i=1,i'=1}^{I,I} (a_i - \Phi S_i)^{\mathrm{T}} \alpha^{(\mathbf{a})}_{ii'} (a_{i'} - \Phi S_{i'}) \tag{4.40}$$

with,

$$\boldsymbol{\xi} = \begin{pmatrix} T_e \\ \Phi \end{pmatrix} \tag{4.41}$$

where Φ is a parameter that is proportional to the plasma's electron number density. The minimisation problem discussed in (4.40) is linear in Φ. The linear least squares method can be employed to determine the optimum value of Φ for a given electron temperature:

$$\Phi_{\mathrm{min}}(T_e) = \frac{\sum_{i=1,i'=1}^{I,I} S_i(T_e) \alpha^{(\mathbf{a})}_{ii'} a_{i'}}{\sum_{i=1,i'=1}^{I,I} S_i(T_e) \alpha^{(\mathbf{a})}_{ii'} S_{i'}} \tag{4.42}$$

For uncorrelated errors in amplitudes, $\alpha^{(\mathbf{a})}$ is given by the diagonal matrix:

$$\alpha^{(\mathbf{a})} = \mathrm{diag}(1/2\sigma^2_{a_1}, 1/2\sigma^2_{a_2}, 1/2\sigma^2_{a_3})$$

For this case (4.42) reduces to:

$$\Phi_{\min}(T_e) = \sum_{i=1}^{I} \frac{a_i S_i(T_e)}{2\sigma_{a_i}^2} \Bigg/ \sum_{i=1}^{I} \frac{S_i^2(T_e)}{2\sigma_{a_i}^2} \tag{4.43}$$

Substituting this into (4.40) reduces the problem to a one-dimensional minimisation problem:

$$\chi^2_{T_e}(\mathbf{a}, T_e) = \sum_{i=1,i'=1}^{I,I} (a_i - \Phi_{\min} S_i)\, \alpha^{(\mathbf{a})}_{ii'}\, (a_{i'} - \Phi_{\min} S_{i'}) \tag{4.44}$$

This was solved using the one-dimensional minimisation routine due to Brent (see Chapter 10 of Press et al. [1996] for details).

4.10 Uncertainty in the temperature

To calculate the uncertainty in the temperature, the uncertainty in $\boldsymbol{\xi}$ (see (4.41)) is calculated from the underlying distribution. If we assume that measured value of $\boldsymbol{\xi}$ (that is, $\boldsymbol{\xi}^0$) is drawn from some normal distribution, then estimating the uncertainty in $\boldsymbol{\xi}$ involves converting the trivariant normal distribution described in (4.37) into some bivariant normal distribution:

$$P(\boldsymbol{\xi}) = \pi^{\frac{1}{2}} |\alpha^{(\boldsymbol{\xi})}|^{\frac{1}{2}} \exp\left[-\delta\boldsymbol{\xi}^{\mathrm{T}} \alpha^{(\boldsymbol{\xi})} \delta\boldsymbol{\xi}\right] \tag{4.45}$$

where $\alpha^{(\boldsymbol{\xi})}$ is the information matrix for the $\boldsymbol{\xi}$ distribution, $\delta\boldsymbol{\xi} = \boldsymbol{\xi} - \boldsymbol{\xi}^0$ and $\boldsymbol{\xi}^0 = \{T_e^0, \Phi^0\}$ is the centre of the distribution: the values that minimise (4.44).

In general, if there exists some linear transformation, *i.e.*

$$\delta\boldsymbol{\xi} = M \delta\mathbf{a} \tag{4.46}$$

then the information matrices transform as in:

$$\alpha^{(\mathbf{a})} = M^{\mathrm{T}} \alpha^{(\boldsymbol{\xi})} M \tag{4.47}$$

where M^{T} denotes the transpose of matrix M. Consider the Taylor series of $f(\mathbf{a}, \boldsymbol{\xi}) = \chi^2_{T_e}(\mathbf{a}, \boldsymbol{\xi})$ expanded to second order, *i.e.*

$$\begin{aligned} f(\mathbf{a}, \boldsymbol{\xi}) = f_0 \quad &+ \quad \frac{\partial f}{\partial \mathbf{a}} \delta\mathbf{a} + \frac{1}{2} (\delta\mathbf{a})^{\mathrm{T}} \frac{\partial^2 f}{\partial \mathbf{a}^2} \delta\mathbf{a} + (\delta\mathbf{a})^{\mathrm{T}} \frac{\partial^2 f}{\partial \mathbf{a} \partial \boldsymbol{\xi}} \delta\boldsymbol{\xi} \\ &+ \quad \frac{1}{2} (\delta\boldsymbol{\xi})^{\mathrm{T}} \frac{\partial^2 f}{\partial \boldsymbol{\xi}^2} \delta\boldsymbol{\xi} \end{aligned} \tag{4.48}$$

Considering an arbitrary shift in origin, $\boldsymbol{\zeta} = \delta\boldsymbol{\xi} + \mathbf{C}$, the Taylor series expansion becomes:

$$\begin{aligned} f(\mathbf{a},\boldsymbol{\xi}) = f_0 \;&+\; \frac{\partial f}{\partial \mathbf{a}}\delta\mathbf{a} + \frac{1}{2}(\delta\mathbf{a})^{\mathrm{T}}\frac{\partial^2 f}{\partial \mathbf{a}^2}\delta\mathbf{a} + \frac{1}{2}\boldsymbol{\zeta}^{\mathrm{T}}\frac{\partial^2 f}{\partial \boldsymbol{\xi}^2}\boldsymbol{\zeta} \\ &+\; \frac{1}{2}\mathbf{C}^{\mathrm{T}}\frac{\partial^2 f}{\partial \boldsymbol{\xi}^2}\mathbf{C} + \left[(\delta\mathbf{a})^{\mathrm{T}}\frac{\partial^2 f}{\partial \mathbf{a}\partial\boldsymbol{\xi}} - \mathbf{C}^{\mathrm{T}}\frac{\partial^2 f}{\partial \boldsymbol{\xi}^2}\right]\boldsymbol{\zeta} \end{aligned} \tag{4.49}$$

If $\mathbf{C}$ is chosen to be $\left(\frac{\partial^2 f}{\partial \boldsymbol{\xi}^2}\right)^{-1}\frac{\partial^2 f}{\partial \boldsymbol{\xi}\partial \mathbf{a}}\delta\mathbf{a}$, then the Taylor series expansion of f has no linear terms in $\boldsymbol{\zeta}$, *i.e.* $\boldsymbol{\zeta} = \mathbf{0}$ for all values of $\delta\mathbf{a}$. From a direct comparison with (4.46), the value of M is:

$$M = \left(\frac{\partial^2 f}{\partial \boldsymbol{\xi}^2}\right)^{-1}\frac{\partial^2 f}{\partial \boldsymbol{\xi}\partial \mathbf{a}} \tag{4.50}$$

Substituting (4.50) into (4.47) and rearranging leads to $\alpha^{(\boldsymbol{\xi})} = T^{\mathrm{T}}\alpha^{(\mathbf{a})}T$, where T is:

$$T = \frac{\partial^2 f}{\partial \mathbf{a}\partial \boldsymbol{\xi}}\left(\frac{\partial^2 f}{\partial \boldsymbol{\xi}\partial \mathbf{a}}\frac{\partial^2 f}{\partial \mathbf{a}\partial \boldsymbol{\xi}}\right)^{-1}\frac{\partial^2 f}{\partial \boldsymbol{\xi}^2} \tag{4.51}$$

The uncertainty in the recovered electron temperature σ_{T_e} is found by marginalising the matrix $\alpha^{(\boldsymbol{\xi})}$. Consider the probability density function for T_e:

$$P(T_e) = \frac{|\alpha^{(\boldsymbol{\xi})}|^{\frac{1}{2}}}{\pi}\exp\left[-(\delta T_e)^2\alpha_{11}^{(\boldsymbol{\xi})}\right]\int \mathrm{d}\Phi \exp\left[-(\delta\Phi)^2\alpha_{22}^{(\boldsymbol{\xi})} - 2\delta\Phi\delta T_e\alpha_{12}^{(\boldsymbol{\xi})}\right] \tag{4.52}$$

where $\delta T_e = T_e - T_e^0$, $\delta\Phi = \Phi - \Phi^0$ and T_e^0 and Φ^0 are the recovered electron temperature and electron number density respectively.

By the change of coordinate $y = \Phi + \frac{\alpha_{12}^{(\boldsymbol{\xi})}\delta T_e}{\alpha_{11}^{(\boldsymbol{\xi})}}$, (4.53) is obtained.

$$\begin{aligned} P(T_e) &= \frac{|\alpha^{(\boldsymbol{\xi})}|^{\frac{1}{2}}}{\pi}\exp\left[-(\delta T_e)^2\left(\frac{|\alpha^{(\boldsymbol{\xi})}|}{\alpha_{22}^{(\boldsymbol{\xi})}}\right)\right]\int \mathrm{d}y \exp\left[-y^2\alpha_{22}^{(\boldsymbol{\xi})}\right] \\ &= \left(\frac{|\alpha^{(\boldsymbol{\xi})}|}{\pi\alpha_{22}^{(\boldsymbol{\xi})}}\right)^{\frac{1}{2}}\exp\left[-(\delta T_e)^2\left(\frac{|\alpha^{(\boldsymbol{\xi})}|}{\alpha_{22}^{(\boldsymbol{\xi})}}\right)\right] \end{aligned} \tag{4.53}$$

Therefore, the uncertainty in T_e is:

$$\sigma_{T_e} = \left\{\frac{\alpha_{22}^{(\boldsymbol{\xi})}}{2|\alpha^{(\boldsymbol{\xi})}|}\right\}^{\frac{1}{2}}$$

4.11 Sample results

To illustrate the recovery of data in poor signal-to-noise conditions, figure 4.26 shows the reconstructed plasma electron temperature distribution as a function of height above the torus mid-plane for part of shot 26522.

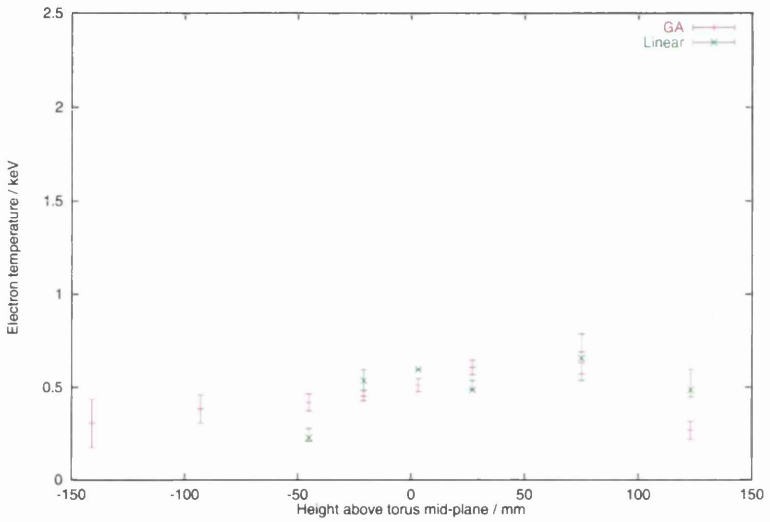


Figure 4.26: Recovered electron temperature distributions for segment 91 of shot 26522. GA is distribution recovered using techniques described in this chapter, Linear uses a pre-existing method.

Chapter 5

Two-temperature Thomson scattering diagnostic analysis

This chapter extends the concepts and methodology of the previous chapter to a more complicated model: a two electron temperature plasma. As the available data consists of three channel responses at each segment it is insufficient to fully constrain the two temperatures. However, some constraints can be placed on the model parameters.

In this chapter, the first section gives an overview of the model stating the motivation for adopting a two-temperature distribution function. Section two illustrates the effect of the second temperature on the channel responses. The significance of normalising the channel outputs is discussed in section three. Section four interprets the degeneracy in the solution. Section five illustrates data taken from COMPASS-D whilst section six discusses observed data in terms of the degeneracy. Section seven discusses the effects of observational errors.

5.1 Two-temperature scattering characteristics

Whenever something is described as having a specific temperature there is a tacit assumption that the distribution function (the number of particles with a particular velocity at a particular location) is Maxwellian. As the distribution function of the gas will, over time, evolve towards a Maxwellian distribution most objects will be close to this distribution.

Often an ensemble under investigation will have a non-Maxwellian distribution function. For example, there might be some process that excites a subset of the particles resulting in a fast-moving species. In order to model these subspecies, perturbations to the Maxwellian distribution must be considered.

Non-thermal distribution functions have been observed at the Alcator tokamak (Coppi et al. [1976] and Pieroni and Segre [1975]), at the TORTUR tokamak (Kluiver et al. [1988] and van Lammeren et al. [1992]), the L-2 stellarator (Blokh and Larionova [1981]) and the Rijnhuizen Tokamak Project (RTP) (Box [1999]). Non-thermal signatures in the Thomson scattered spectrum can occur due to neutral beam injection, electron-cyclotron resonance heating (ECRH) or deuterium pellet injection (Box [1999]), electron streams travelling at the Alfvén speed (as described in Kluiver et al. [1988]) or from a trapped electron population (Blokh and Larionova [1981]).

The simplest natural generalisation of the single-temperature Maxwellian is to add another Maxwellian distribution of a smaller amplitude. This corresponds to the majority of electrons existing at one temperature whilst a smaller subsection of the electron population exists at some hotter temperature. Thomson scattering spectra consistent with a two-temperature electron distribution have been observed at the RTP (Box [1999]) and Alcator tokamak (Coppi et al. [1976] and Pieroni and Segre [1975]). It is worth emphasising that, although a two-temperature distribution will be discussed throughout this chapter, without two extra measurements no quantitative analysis of the validity of this model can be undertaken. Moreover, non two-temperature deviations from a Maxwellian distribution have been observed at tokamaks such as TORTUR (van Lammeren et al. [1992]).

If the two species are considered separate and non-interacting over the duration of observation (approximately 10 ns) then the combined Thomson scattering is the linear sum of the light scattered from each separate species given by

$$f_i = \Phi_1 S_i(T_1) + \Phi_2 S_i(T_2) \qquad i = 1, 2, 3 \tag{5.1}$$

where f_i is the combined scattered intensity from the two species, Φ_1 and Φ_2 are proportional to the plasma electron densities for the two species and $S_i(T_1)$ and $S_i(T_2)$ are the two spectral density functions for the i^{th} filter (see §4.2 for more details) with electron temperatures T_1 and T_2 for the two species respectively.

There are four degrees of freedom for the two-temperature model: Φ_1, Φ_2, T_1 and T_2.

However, as only three filter responses f_i are available an observation is only sufficient to constrain the solution to lie on some line. Therefore, there will be a 1-parameter family of two-temperature solutions consistent with each set of filter responses. All of these solutions are possible in the absence of further information.

5.2 Channel responses

In order to get an idea of the effect of a second temperature, we investigate how the filter responses change with the presence of a second temperature species.

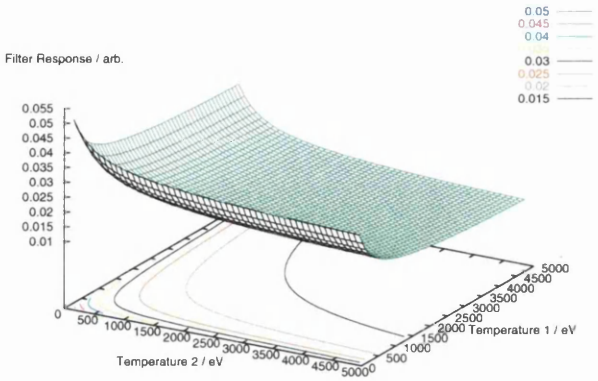
5.2.1 Response curve graphs

The response to two-temperature distributions for the channels of spectrometer 5 are shown in figures 5.1–5.3. For figure 5.1, the plasma densities are equal and the corresponding response curves are symmetric about the plane $T_1 = T_2$. In the sequence of figures 5.1–5.3 the electron density becomes increasingly dominated by the first species, which has a temperature of T_1. This can be clearly seen in figure 5.3 where the curve becomes stretched along the T_2 axis: at any point on the surface the response is roughly constant along the T_2 axis whereas it varies markedly along the T_1 axis.

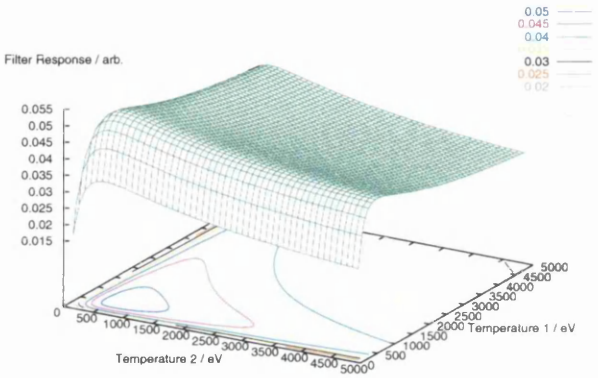
5.2.2 Recovering the single-temperature distribution function as limiting case

The single-temperature response function appears from the two-temperature distribution function as several variables are taken to limiting values. Irrespective of the method of measuring the scattered spectra, the response of a two-temperature distribution will tend to that of a single-temperature distribution as the electron density of one of the two species tends to zero.

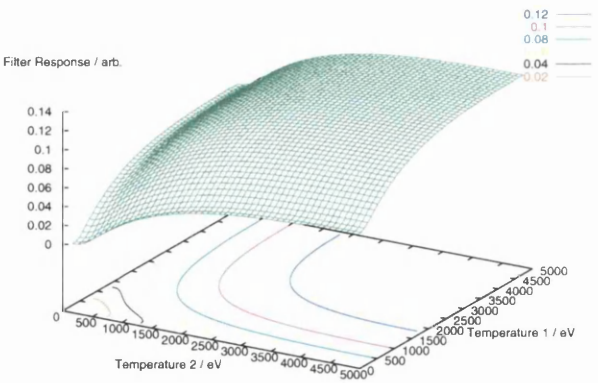
The method of Thomson-scattered-spectrum analysis used at COMPASS-D involves measuring the response at three spectral channels. As can be seen in figure 4.6, these channels do not cover all frequencies. For example, channel 1 (measuring the longest wavelengths) does not measure wavelengths very close to the laser's wavelength. If the electron temperature is sufficiently low then the filters will detect none of the scattered light. Likewise, if the temperature is sufficiently high, then although the detectors



(a) Channel 1

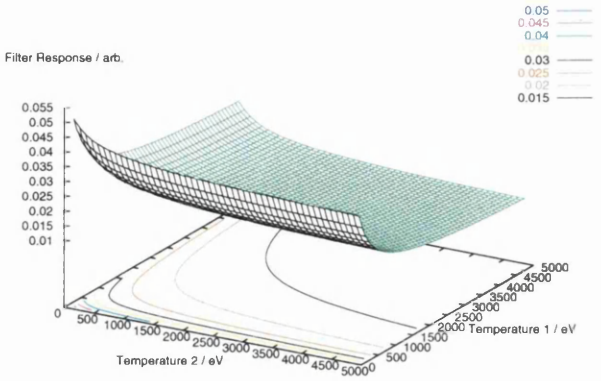


(b) Channel 2

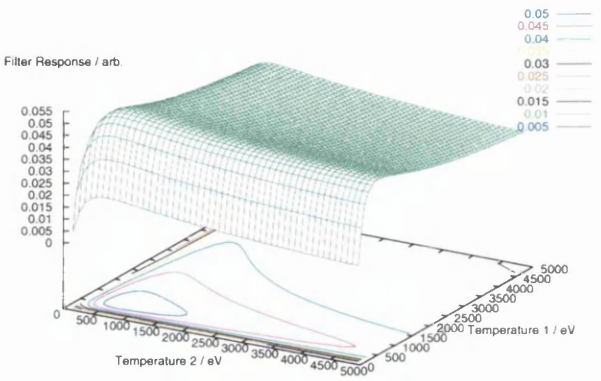


(c) Channel 3

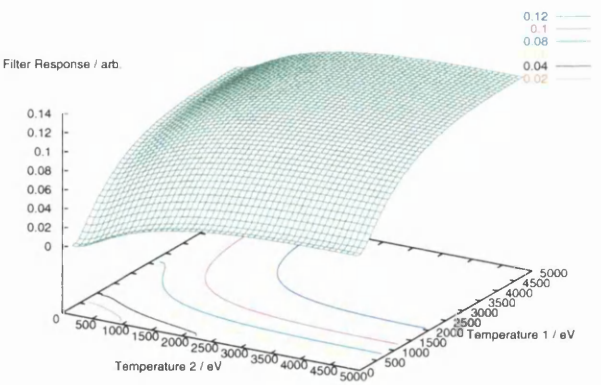
Figure 5.1: Response of the three channels to a two-temperature distribution function. The electrons have densities $n_1 = n_2 = 10^{20}\mathrm{m}^{-3}$; the ratio $n_1 : n_2$ is $50 : 50$.



(a) Channel 1

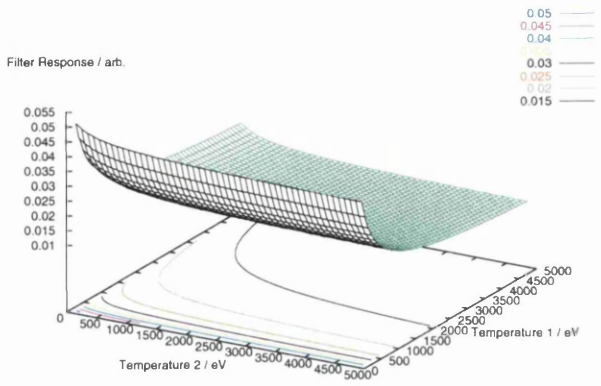


(b) Channel 2

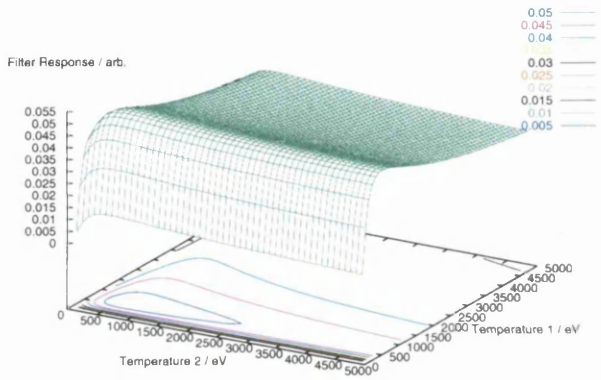


(c) Channel 3

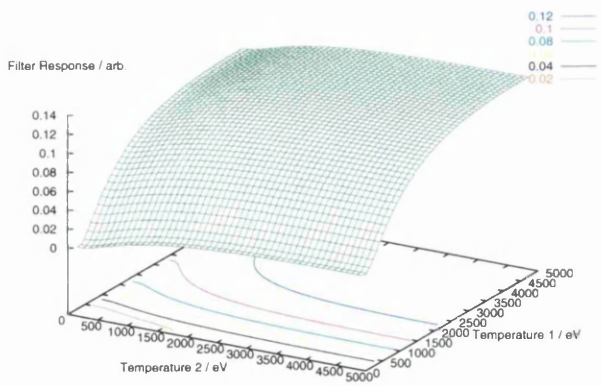
Figure 5.2: Response of the three channels to a two-temperature distribution function. The electrons have densities $n_1 = 1.3 \times 10^{20}\mathrm{m}^{-3}$ and $n_2 = 0.7 \times 10^{20}\mathrm{m}^{-3}$; the ratio $n_1 : n_2$ is $65 : 35$.



(a) Channel 1



(b) Channel 2



(c) Channel 3

Figure 5.3: Two-temperature distribution function response for the three channels. Electrons densities are $n_1 = 1.6 \times 10^{20}\mathrm{m}^{-3}$ and $n_2 = 0.4 \times 10^{20}\mathrm{m}^{-3}$; the ratio $n_1 : n_2$ is $80 : 20$.

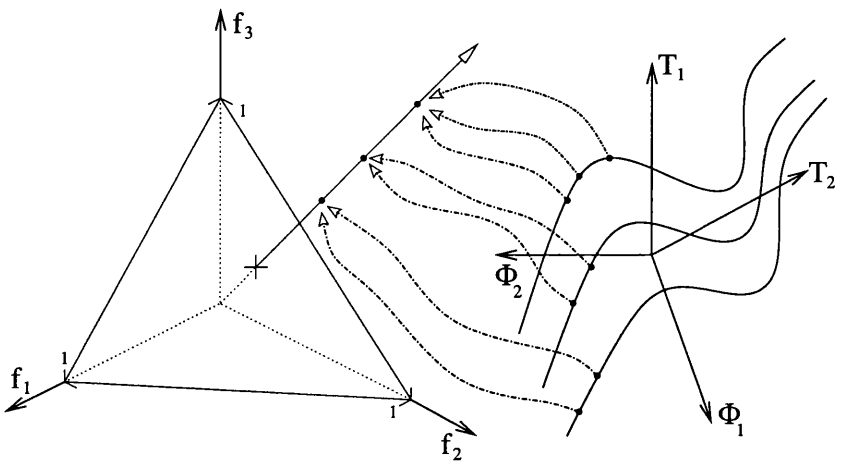


Figure 5.4: Various mappings from $\{T_1, T_2, \Phi_1, \Phi_2\}$ space into observation space $\{f_1, f_2, f_3\}$ are shown. Solid curves in $\{T_1, T_2, \Phi_1, \Phi_2\}$ space are iso-response curves. Certain iso-response curves are equivalent when the channel responses are normalised by projection onto the plane.

pick up some scattered light, the majority is scattered at too high a frequency for the spectrometer channels to detect. Although these temperature limits exist they are unlikely to be encountered in practice.

A one-temperature response curve is recovered in the limit as n_2 tends to zero, as T_2 tends to zero or as T_2 tends to ∞.

5.3 Normalised outputs

In order to present and understand the data, it is convenient to reduce the number of degrees of freedom. One can observe that there is a degree of freedom that corresponds simply to scaling the overall density of the plasma, which can be expressed by multiplying (5.1) by some arbitrary number. This gives a different measured triplet, $\{f_1, f_2, f_3\}$, but the two temperatures remain unaltered. Geometrically, this degree of freedom is manifest as straight lines emanating from the origin in (f_1, f_2, f_3) space. Each point along these lines is identified with exactly the same two-temperature distribution. This is shown in figure 5.4.

In order to remove this 'uninteresting' degree of freedom, the responses were normalised by dividing the i^{th} observed channel response f_i by the sum of all three channels. The resulting normalised channel responses, $\tilde{f}_i$, are described by (5.2). This places the constraint on the triplet $\{\tilde{f}_1, \tilde{f}_2, \tilde{f}_3\}$ that $\tilde{f}_1 + \tilde{f}_2 + \tilde{f}_3 = 1$. This normalisation can be viewed as projecting all points in $\{f_1, f_2, f_3\}$ space onto the plane that intercepts the

three axes at 1 as shown in figure 5.4.

$$\tilde{f}_i = \frac{f_i}{\sum_j f_j} = \frac{\Phi_1 S_i(T_1) + \Phi_2 S_i(T_2)}{\Phi_1 \sum_j S_j(T_1) + \Phi_2 \sum_j S_j(T_2)} \tag{5.2}$$

If the responses from the three channels are f_1, f_2 and f_3 as defined in (5.1), then we can define a normalised spectral density function for the i^{th} channel, $\tilde{S}_i(T)$, as in (5.3).

$$\tilde{S}_i(T) = \frac{S_i(T)}{\sum_j S_j(T)} \tag{5.3}$$

This leads to an alternative definition of the overall filter response, analogous to (5.1) but using the normalised spectral density function as in (5.4).

$$f_i = \tilde{\Phi}_1 \tilde{S}_i(T_1) + \tilde{\Phi}_2 \tilde{S}_i(T_2) \tag{5.4}$$

$\tilde{\Phi}_1$ is again proportional to the plasma electron density n_1 but now also takes into account the normalisation factor $\sum_i S_i(T)$.

We can express (5.2) in terms of this normalised spectral density function (5.3) as in (5.4), so that (5.2) becomes:

$$\begin{aligned} \tilde{f}_i &= \frac{\tilde{\Phi}_1}{\tilde{\Phi}_1 + \tilde{\Phi}_2} \tilde{S}_i(T_1) + \left(1 - \frac{\tilde{\Phi}_1}{\tilde{\Phi}_1 + \tilde{\Phi}_2}\right) \tilde{S}_i(T_2) \\ &= \alpha \tilde{S}_i(T_1) + (1 - \alpha)\, \tilde{S}_2(T_2) \end{aligned} \tag{5.5}$$

This form of the normalised data depends on three variables: T_1, T_2 and α where α is the relative density of the first species compared to the second species.

It will be useful later to define a vector of normalised observed responses as defined in (5.6). The third normalised channel response is not included as it is not independent of the first two normalised channel responses.

$$\tilde{\mathbf{f}} = \begin{pmatrix} \tilde{f}_1 \\ \tilde{f}_2 \end{pmatrix} = \begin{pmatrix} \alpha \tilde{S}_1(T_1) + (1 - \alpha)\, \tilde{S}_1(T_2) \\ \alpha \tilde{S}_2(T_1) + (1 - \alpha)\, \tilde{S}_2(T_2) \end{pmatrix} \tag{5.6}$$

5.3.1 Single-temperature distributions

A single-temperature electron distribution has a unique set of observed channel responses and so has a unique set of normalised channel responses. Figure 5.5 shows the

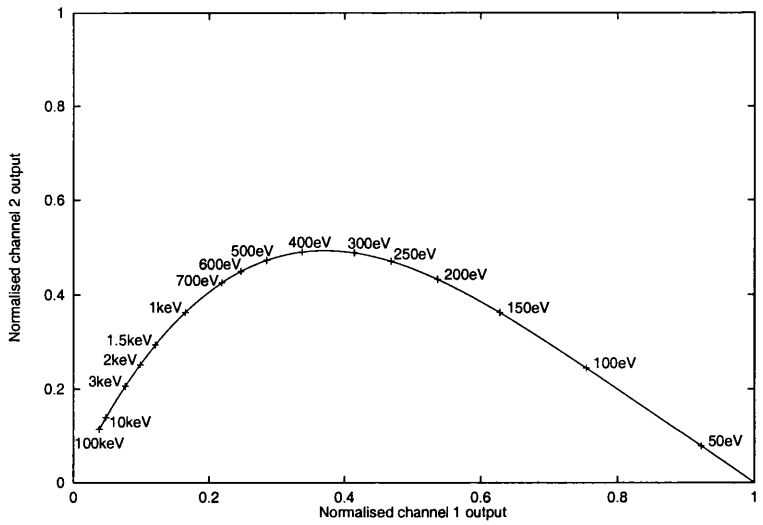


Figure 5.5: Plot showing the pairs of normalised channel responses for single-temperature distribution functions. The scattering angle is 90°. Selected points have their temperature in electron volts labeled.

set of normalised channel responses consistent with a single-temperature distribution function. This curve was generated using the measured spectral responses for spectrometer 5 (see figure 4.6). For a given temperature, the theoretical Thomson scattered signal was integrated over each channel's spectral response in turn. The resulting three signals were then normalised and plotted.

It is the nature of experimental error that the measured response from each channel will have an element of noise. This means that observations from a plasma precisely described by a single-temperature distribution will not, in general, lie exactly on the curve shown in figure 5.5. Instead, there will some scatter about this curve. To assign a best-fit temperature to such deviant data we find the point on the single-temperature curve (figure 5.5 for the normalised case) that is 'closest' to the observed data. Under §4.9, this closeness is taken to be the least-squares measure. This is usual, as χ^2 is the minimum variance unbiased estimator or the maximum likelihood estimator for Gaussian errors. The likelihood function for estimating the temperature of a single-temperature distribution in the presence of noise is given by (4.40) on page 78.

For any given temperature, T, there will be a curve, $\Omega_{\text{iso}-T}$, consisting of the set of points (*i.e.* the set of observations) for which the best measure of temperature is T. These curves will be referred to as iso-temperature curves hereafter. This degeneracy arises from forcing data to fit a model, which in this case is a single-temperature distribution function. The distance from the single-temperature curve to the observation along the iso-temperature curve is a measure of how inconsistent the observation is with

a true single-temperature distribution.

The 'closeness' is measured with the observed signals before they are normalised. Under the full set of observations the iso-temperature curve will be normal to the single-temperature surface: this is from our least-squares definition of 'closest' or 'best'. The normalisation has the geometric interpretation of a projection on the plane $f_1+f_2+f_3 = 1$. This projection will introduce apparent distortions to the iso-temperature curves: they will no longer appear normal to the single-temperature surface.

To analyse these iso-ture curves, the normalised responses for a single-temperature distribution were considered. For such a distribution, normalising the three channel responses is equivalent to removing any Φ (*i.e.* plasma electron density) dependency, with $\tilde{f}_i(T) = \tilde{S}_i(T)$. Therefore, the 'best fit' procedure described in §4.9 assigns a temperature to each set of normalised observations: $T(\tilde{\mathbf{f}})$. This mapping was then expanded into a Taylor series as shown in (5.7), where $\mathbf{A}$ is the 1×2 matrix $(\mathrm{d}T/\mathrm{d}\tilde{f}_1\ \mathrm{d}T/\mathrm{d}\tilde{f}_2)$.

$$T(\tilde{\mathbf{f}}) = T_0 + \mathbf{A}\,\mathrm{d}\tilde{\mathbf{f}} + O(\mathrm{d}\tilde{f}^2) \tag{5.7}$$

Iso-temperature curves in $\tilde{\mathbf{f}}$ are those curves along which the temperature does not vary, *i.e.* $\forall\,\tilde{\mathbf{f}} \in \boldsymbol{\Omega}_{\mathrm{iso}-T}\quad T(\tilde{\mathbf{f}}) = T_0$. This must be true for an arbitrary point along the line, requiring (5.8) to hold along $\boldsymbol{\Omega}_{\mathrm{iso}-T}$.

$$\mathbf{A}\,\mathrm{d}\tilde{\mathbf{f}} = 0 \tag{5.8}$$

Single value decomposition (SVD) of matrix $\mathbf{A}$ (an $N \times M$ matrix) finds the three matrices $\mathbf{U}$ ($N \times M$), $\mathbf{W}$ ($M \times M$) and $\mathbf{V}$ ($M \times M$) for which $\mathbf{A} = \mathbf{U}\cdot\mathbf{W}\cdot\mathbf{V}^{\mathrm{T}}$. The three matrices also have the property that $\mathbf{W}$ is diagonal, $\mathbf{V}$ is orthogonal and, if $M \geqslant N$, $\mathbf{U}^{\mathrm{T}} \cdot \mathbf{U} = \mathbf{1}$. If $M < N$ then $\mathbf{U}^{\mathrm{T}} \cdot \mathbf{U}$ will still be diagonal but $N - M$ elements will be zero instead of one. The diagonal elements in $\mathbf{U}$ that are zero will have corresponding elements in $\mathbf{W}$ that are also zero. Columns of $\mathbf{V}$ (or rows of $\mathbf{V}^{\mathrm{T}}$) corresponding to zero diagonal elements of $\mathbf{W}$ are the null-vectors of $\mathbf{A}$. For the above problem there is one null-vector. This vector is unbounded by 5.8, so it is tangential to the iso-temperature curve.

Following the null-vector (as given by SVD) to form the complete iso-temperature curve is a problem in integrating an Ordinary Differential Equation (ODE). By using the boundary condition that the integration have an initial value for $\tilde{f}_1$ and $\tilde{f}_2$ this

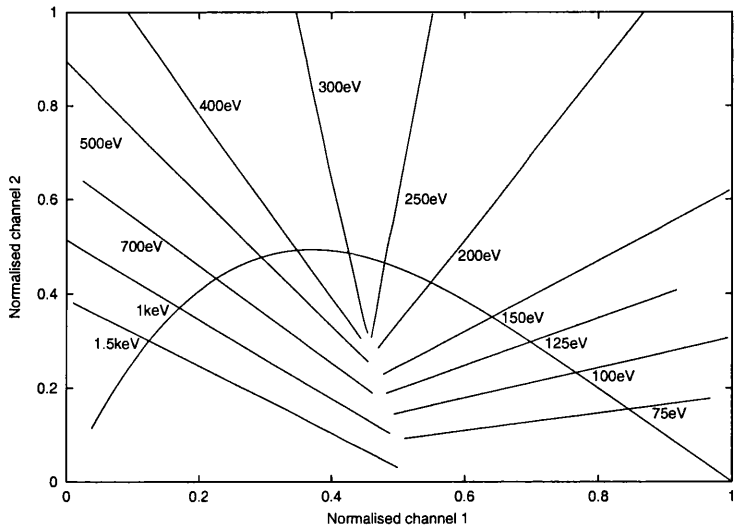


Figure 5.6: A selection of iso-temperature curves are shown along with the exact single-temperature curve as in figure 5.5. All points along an iso-temperature curve have the sample 'best estimate' temperature. The lines are not normal to the single-temperature curve because of the projection due to the normalisation.

ODE can be classified as an initial value problem (IVP). Various methods are available for solving IVPs. A commonly used robust method is the fourth-order Runge-Kutta method (see chapter 16 of Press et al. [1996]). The step size can be either fixed or adaptive. Because of the flexibility associated with the adaptive step-size, this method was used for the integration. The routines used were taken from Press et al. [1996].

The results from the Runge-Kutta method is shown in figure 5.6 along with the single-temperature curve. The iso-temperature curves are straight lines crossing the single-temperature curve at the relevant temperature. As can be seen clearly for the low temperature cases, the iso-temperature curves are generally not normal to the single-temperature curve.

5.3.2 Two-temperature considerations

From (5.5), a two-temperature distribution is a linear superposition of two single-temperature distributions. In this simple model, the electron density of each species cannot be negative. This positivity constraint implies that a line connecting two points on the single-temperature curve in figure 5.5 contains all points consistent with a two-temperature distribution function, as shown in figure 5.7. T_1 and T_2 for this two-temperature distribution function are given by the two intercepts between the line and the single-temperature curve.

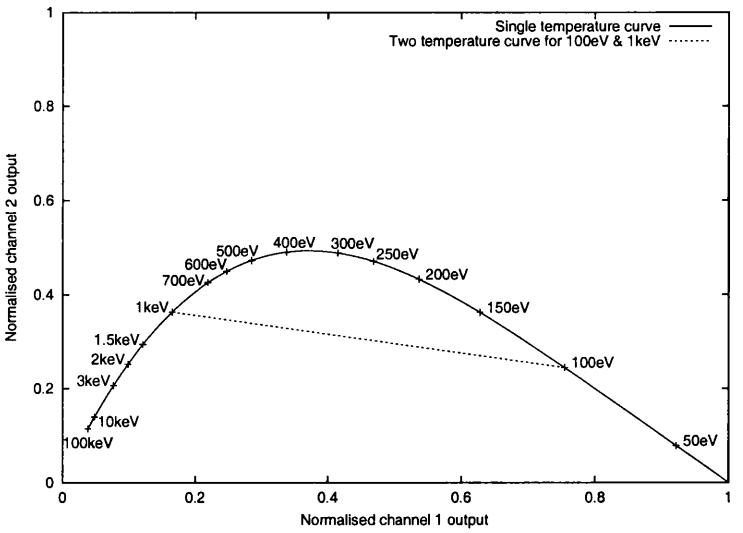


Figure 5.7: Graph showing the pairs of normalised channel responses for all two-temperature distributions (dashed) with $T_1 = 100\text{eV}$ and $T_2 = 1\text{keV}$. The scattering angle is 90°. The single-temperature response curve of figure 5.5 is shown for reference.

The positivity constraint on plasma electron density requires that all observations must lie underneath the curve for two-temperature distribution functions. This is obvious as the observation must lie on the line connecting some T_1 and some other T_2. If the model is relaxed to allow an n-temperature distribution function and those n temperatures are known *a priori* then the observation must lie within the n sided polygon with vertices touching the single-temperature curve at T_1, T_2, ..., T_n. Irrespective of the values of T_1, T_2, ..., T_n, again positivity requires that the observation lie beneath the single-temperature curve.

There is a natural, geometric interpretation of the degree of freedom in the two temperatures. Consider a point on the line connecting T_1 and T_2. This point corresponds to a specific observation: a set of three observed channel responses that have been normalised (by dividing by their sum) to produce the two normalised channel responses. By rotating the line about that point the data remains consistent with the observation whilst the values of T_1, T_2 and α are altered. Any line that passes through that point gives rise to a valid two-temperature interpretation where T_1 and T_2 are the intercepts with the single-temperature response curve and α is the length of the line from T_2 to the observation point as a fraction of the total line's length.

Requiring $T_1 > 0$ places a lower bound ($T_2^{\min}$) on T_2 whilst stipulating $T_2 < \infty$ places an upper bound ($T_1^{\max}$) on T_1. For all points that lie off the single-temperature curve, there is a range of temperatures $T_2^{\min} > T > T_1^{\max}$ that the electron distribution cannot

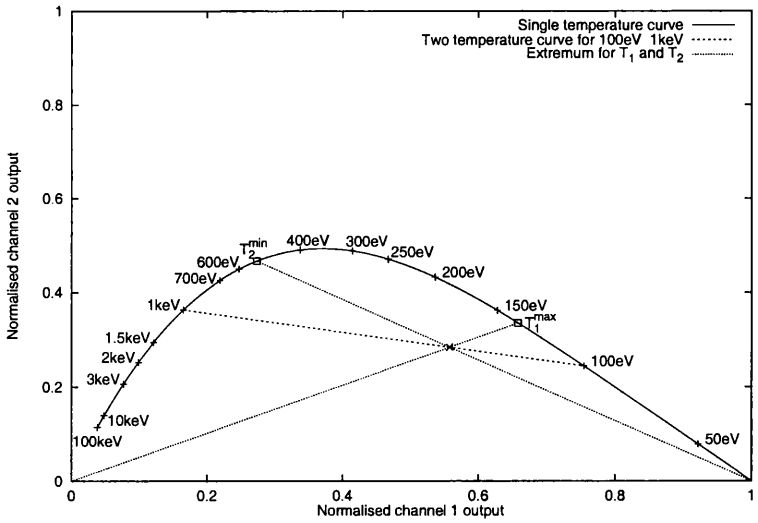


Figure 5.8: Graph showing the single-temperature response curve (solid) and three two-temperature lines consistent with $T_1 = 100\text{eV}$, $T_2 = 1\text{keV}$ and $\alpha = 2/3$. Two of the lines represent the limiting cases; $T_1^{\max}$ and $T_2^{\min}$ are labelled.

contain. The two limits are shown in figure 5.8 along with a third intermediate case.

It is worth emphasising that the best fit single-temperature distribution may have a temperature that is inconsistent with the two-temperature model. By comparing figures 5.6 and 5.8 it is easy to find points (*i.e.* observations) that have a best fit single-temperature distribution, with a temperature T say, and have a two-temperature distribution with $T_2^{\min} > T > T_1^{\max}$. This inconsistency is not surprising as any non-thermal component will distort the best fit single-temperature away from the underlying Maxwellian distribution.

In addition to the limits placed on T_1 and T_2, there are constraints on the value of α. For the case where $T_1 = 0$ and $T_2 = T_2^{\min}$ α then has a minimum value: $\alpha^{\min}$. There is a corresponding maximum value of α, $\alpha^{\max}$, appearing as the limit as T_1 tends to $T_1^{\max}$ whilst T_2 tends to ∞. Therefore the value of α must lie between $\alpha^{\min}$ and $\alpha^{\max}$.

5.4 Iso-response curves

As illustrated in the previous section, each observed measurement corresponds to a family of solutions. It is impossible to find the correct value without further information but the family of solutions may be plotted and external information, such as the likely fraction of the minority hot component, can be used to constrain this family of solutions.

5.4.1 Theoretical treatment

We defined $\tilde{\mathbf{f}}$ in (5.6) as the vector of normalised observed responses. Also, we define $\tilde{\mathbf{g}}$ as the vector of model parameters of the normalised distribution function as:

$$\tilde{\mathbf{g}} = \begin{pmatrix} T_1 \\ T_2 \\ \alpha \end{pmatrix} \tag{5.9}$$

By considering a Taylor series expansion of $\tilde{\mathbf{f}}(\tilde{\mathbf{g}})$ and retaining the first-order term, the gradient of the normalised responses, the matrix $\mathbf{A}$, is defined as:

$$\mathrm{d}\tilde{\mathbf{f}} = \mathbf{A}\,\mathrm{d}\tilde{\mathbf{g}} \tag{5.10}$$

The iso-response curves are lines in $\tilde{\mathbf{g}}$ along which the normalised response, $\tilde{\mathbf{f}}$, does not change. A tangent to the iso-response curve is the direction in which the gradient of the response is zero:

$$\mathrm{d}\tilde{\mathbf{f}} = \mathbf{A}\,\mathrm{d}\tilde{\mathbf{g}} = 0 \tag{5.11}$$

$\mathrm{d}\tilde{\mathbf{g}}$ thus lies in the null space of $\mathbf{A}$, *i.e.* it is an eigenvector of $\mathbf{A}$ with eigenvalue of zero. This null vector, described by (5.11), was obtained using a method similar to §5.3.1. SVD was used to find the three matrices $\mathbf{U}$, $\mathbf{W}$ and $\mathbf{V}$. The null vector is the column of $\mathbf{V}$ corresponding to the zero singular value in $\mathbf{W}$. These vectors were integrated using an adaptive step-size fourth-order Runge-Kutta method.

5.4.2 Figures

This methodology was used to produce the iso-response curves illustrated. It is crucial to appreciate that in the following graphs the third axis, α, has been suppressed. Because of this projection, the curves may appear to cross where in the full 3-space, they do not.

In figure 5.9, several iso-response curves are plotted on the same graph. The initial points chosen from which to build the curves are $T_1 = 10\text{eV}$, $\alpha = 0.001$ and T_2 selected from 250eV to 4keV in 250eV steps.

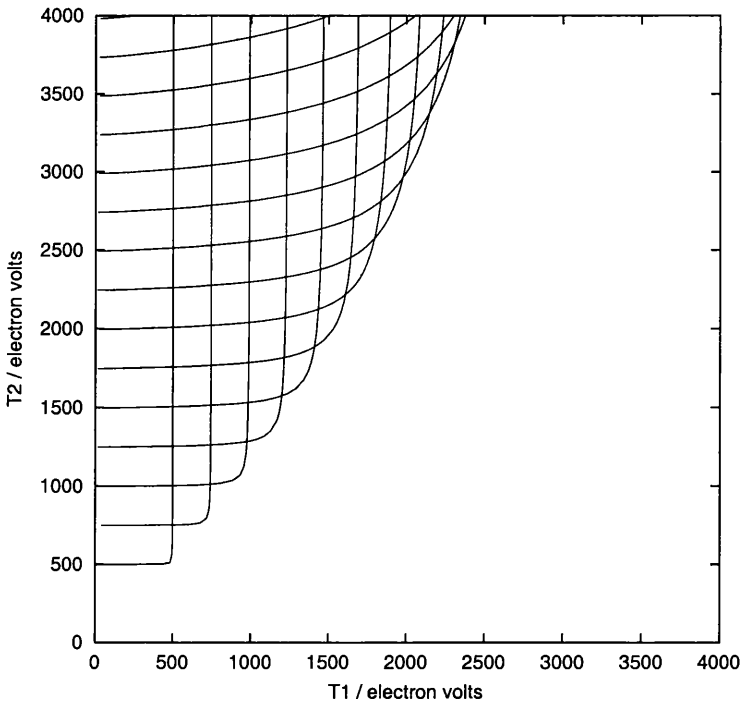


Figure 5.9: Iso-response curves (α axis suppressed) for initial points with differing values of T_2, where $T_2 > T_1$.

The initial points of T_1 are 'close' to the axis but not at zero. This is because the solutions become degenerate if T_1 is exactly zero. A complementary set of curves can be generated that are the mirror image through the line $T_1 = T_2$. This simply corresponds to swapping T_1 and T_2 and taking $1 - \alpha_0$ instead of α_0 as the initial value of α.

The value of α increases along the curve as T_1 increase. For the tight curves with low initial T_1 the value of α increase slowly up to the sharp corner. At this point, the value of α rapidly increases until a value close to one is achieved and the curve has turned the corner. After this, α increases slowly as T_2 increase.

The limiting case is an iso-response curve with an initial value of α of zero. As such an iso-response curve is followed, the initial direction will be a straight line from $T_1 = 0$ to $T_1 = T_2$ with α remaining zero. As the point where $T_1 = T_2$ is reached, α will rise without any change in T_1 or T_2 until α is unity. Further following the curve reveals T_1 and α remaining constant and T_2 increases. This case corresponds to an observation exactly on the single-temperature curve of figure 5.5.

Figure 5.10 shows the effect of varying the initial value of α. The iso-curves have an initial conditions of $T_1 = 10\text{eV}$, T_2 is either 1keV or 3keV and α is from the set $\{5 \times 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0.1\}$.

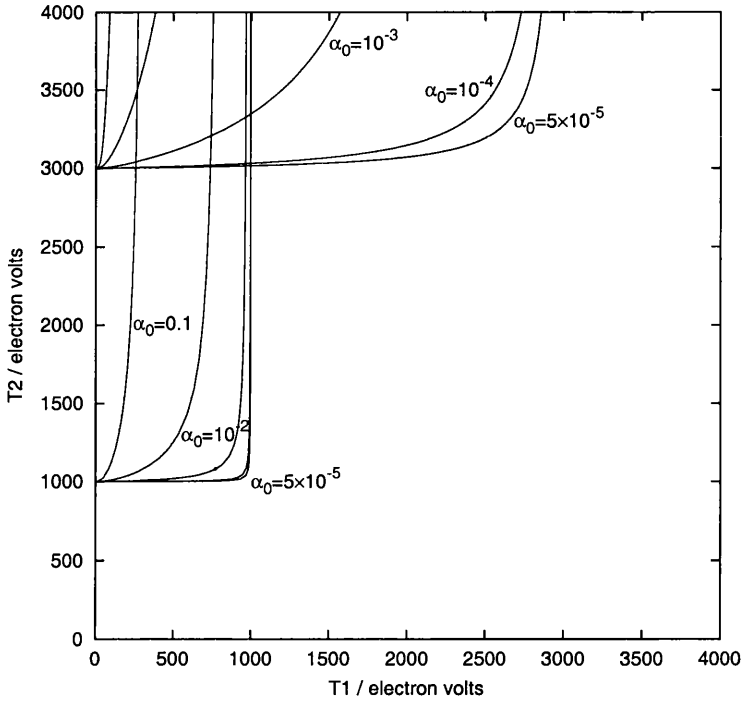


Figure 5.10: Iso-response curves (α axis suppressed) for initial points with differing values of α. The initial value of α (α_0) is taken from the set $\{5 \times 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0.1\}$.

5.5 Data from COMPASS-D

It is interesting to consider a large collection of data and plot these observations on a normalised response graph. To do this, the data for all plasma shots between 24 November 1997 and 4 June 1998 (some 1831 shots) were catalogued. For the data due to spectrometer 5, the combined signal-to-noise ratio for the three channels was calculated for each of segment range 90 to 99 (those where plasma is likely to have been present) of each shot, as described in §4.5. The 1000 observations with the greatest combined signal-to-noise ratio were then plotted in figure 5.11 with error bars for selected data.

It is clear that the majority of data in figure 5.11 lie above the single-temperature distribution function curve. This is in contrast to the expectation that all data would lie below the curve. Although the majority of data are above the curve, there are data-points which lie inside the curve by many error-bar lengths. In addition, the data-points that lie outside the curve appear to be bounded by some curve similar to the single-temperature curve. This suggests that the two-temperature distribution model is applicable and that there is some systematic bias in the analysis.

One example of a systematic bias that would account for the difference is if the ref-

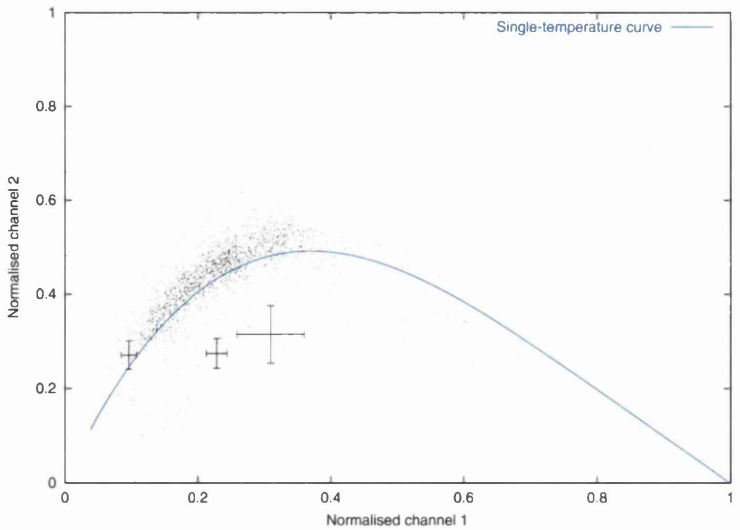


Figure 5.11: Observated normalised responses for the 1000 data from spectrometer 5 with the greatest combined signal-to-noise ratio. The single temperature curve of figure 5.5 is also shown. Selected points have error bars.

erence signal was not correct for a particular channel. This would reduce the measured amplitude for that channel and normalisation would then increase the measured amplitude for the other two channels. If the reference signal for channels one and three were slightly incorrect then the observations would be biased diagonally towards increased channel two and decreased channel one normalised responses, which is the observed behaviour.

Another possible cause is a film forming on the vacuum vessel's view port from deposition of impurities. Barth et al. [1997] states that, at the RTP, transmission may decrease by 78% for wavelengths of 700 to 850 nm and 70% for 550 nm light. This chromatic effect would affect channel one (longest wavelength) the most, channel two to a lesser extent and channel three the least. After normalising, this would introduce a systematic bias in the channel responses: the response for channel one would be suppressed whilst the response for channel two enhanced.

It is worth emphasising that there are unquantifiable uncertainties in the single-temperature curve. This curve was constructed using supplied frequency response data for each of the channels. If this information were incorrect, due to the chromatic effect of the deposited film for example, then the single-temperature curve will not describe the upper limit for data.

5.6 Converting measured responses into iso-response curves

Iso-response curves are values of T_1, T_2 and α that have identical observed responses. Any point on the line is sufficient information to construct the iso-response curve as any point can be the initial value for the IVP. Therefore, the iso-response curve for a set of normalised observations can be obtained by ascertaining one valid set of T_1, T_2 and α. Since there is a degree of freedom, there is freedom in choice of initial T_1.

It is possible to choose $T_1 = 0$ and discover T_2 and α from inspection against the single-temperature curve. However, as previously stated, the solution is degenerate for any temperature at or sufficiently close to zero. Instead some small, non-zero value of T_1 could be used (*e.g.* 10eV) or some another value which is more convenient.

Once the iso-response curve has been established, it may be possible to place any further constraints on the system. Any measurement will yield two (independent) normalised channel responses. As described above, these can be converted into a iso-response curve containing all values of $\{T_1, T_2, \alpha\}$ that are consistent with the observed data. The values of T_1, T_2 and α vary along this curve. Sections of the curve may correspond to values of T_1, T_2 or α that can be ruled out by 'external' criteria.

This should be done carefully as the extra information may be inapplicable for the specific shot under consideration. For example, evidence from van Lammeren et al. [1992] suggests the minority (in this case non-thermal) species at TORTUR is less than 10% of the electron density, placing a strong constraint on valid values of α. However, Pieroni and Segre [1975] reports that the density of the minority hotter species at Alcator was 40% of the cooler species. Undoubtably, this differences arises from the different operational regimes of the two tokamaks. However, since both the upper limit of α and the cause of the minority species are not known for COMPASS, imposing upper-boundary restrictions on α would be dubious.

5.7 Errors

An uncertainty in the $(\tilde{f}_1, \tilde{f}_2)$ will be manifest as an uncertainty in $T_1^{\max}$ and $T_2^{\min}$. To obtain the true probability distribution function for the uncertainties the probability density function of the point must be integrated over. For example, consider the probability density for $T_1^{\max}$ being some temperature T_1^*, denoted $p(T_1^*)$. If the line

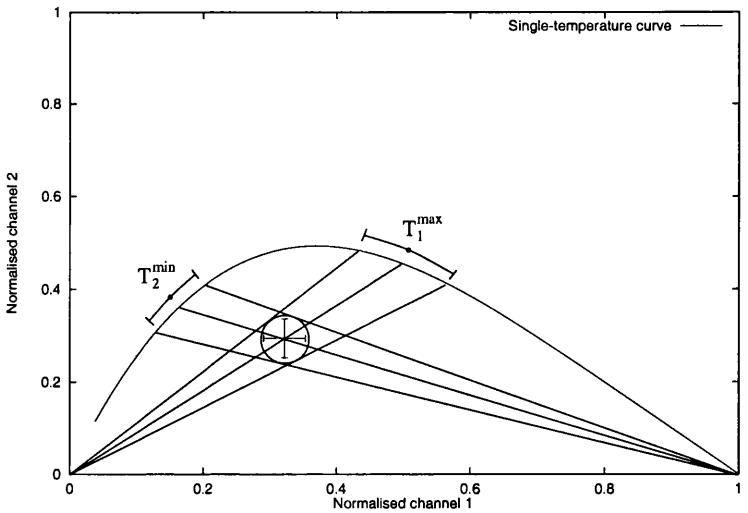


Figure 5.12: Error estimate for $T_1^{\max}$ and $T_2^{\min}$ by projecting the $1 \times \sigma$ surface onto the single-temperature curve.

connecting $\tilde{\mathbf{f}}(T=\infty)$ (the origin in figure 5.5) to the single-temperature response point $\tilde{\mathbf{f}}(T = T_1^*)$ is parameterised by l so that $\tilde{\mathbf{f}}(l = 0) = (0, 0)$ and $\tilde{\mathbf{f}}(l = 1) = \tilde{\mathbf{f}}(T = T_1^*)$ then $p(T_1^*)$ is given by (5.12) where $p(\tilde{\mathbf{f}})$ is the probability density associated with the uncertainty in the observation.

$$p(T_1^*) = \int_0^1 p(\tilde{\mathbf{f}})\, \mathrm{d}l \tag{5.12}$$

In general this integral is intractable as the form of the curve will depend on the channel responses. An approximation to this is to project the $1 \times \sigma$ ellipse onto the curve as shown in figure 5.12.

5.8 Conclusions

Fitting a two-temperature model when supplied with three channel responses is an ill-conditioned problem. As the data provides insufficient constraints on the model, a family of solutions (a curve) exists for each observation. In the absence of further information, any solution along this curve is equally likely.

Despite the degeneracy in the two-temperature parameters, analysis of the channel responses provides constraints for the two temperatures and the ratio of the two species' electron densities. For a given set of observations and without further 'external' information, the data is sufficient to place upper and lower bounds on the cooler and hotter

components respectively and both upper and lower boundaries on the ratio of electron densities for the two temperature species.

With additional information, it is possible to constrain the solutions further. However, this information must be from a source other than Thomson scattering results. An extra channel would enable precise measures of the two temperatures without the degeneracy. Whilst if five channels were present then an objective measure of the validity of the two-temperature model would be possible.

Increasing the number of channels in the spectrometers becomes progressively more difficult due to alignment problems and losses. An alternative method of analysing the scattered light is to use a grating or prism. This splits the one-dimensional beam path into a two-dimensional planar representing the scattered light spectrum at each point along the beam path. This two-dimensional plane can then be imaged onto some recording device.

The original form of this diagnostic, as described in Bretz et al. [1978], used a television camera as the recording device. From this design, the generic term Television Thomson Scattering (or TVTS) diagnostic was coined.

Modern TVTS uses image intensifiers with a Charge-Coupled Device (CCD) as the method of data acquisition (see Barth et al. [1997] and van Lammeren et al. [1992] for example configurations). A CCD uses the photoelectric effect to generate a charge proportional to the number of incident photons at each picture element (pixel). CCDs provide a more flexible and higher resolution method of data acquisition than analogue television camera technology.

The major disadvantage with TVTS is a low repetition rate. Diagnostics using TVTS utilise ruby lasers. These are high power devices, up to 25 J, that produce light in the visible spectrum at a wavelength of 964.3 nm. However, these diagnostics suffer from a number of disadvantages. The laser frequency is close to the H_α line and some He lines so a regions of the spectra are unusable. Ruby lasers do not have the high repetition rate of the Nd:YAG lasers, limiting TVTS to analysing the signal at one time during the plasma discharge. Recently, the TVTS system at RTP has been upgraded to achieve some temporal measurement. See Beurskens et al. [1997] and Beurskens et al. [1999] for more details.

Chapter 6

Conclusions and future work

This chapter summarises the results and conclusions presented in this thesis. Possible future developments are suggested.

The first section of this chapter recapitulates the main advantages and disadvantages of the Genetic Algorithm (GA) approach to optimisation. In section two these points are framed within the context of signal analysis, especially with Bayesian methodology in mind. A retrospective view of the Thomson scattering diagnostic is presented in section three with the final section describing future work and unanswered questions.

6.1 Summarising Genetic Algorithms

A set of generic library routines, collectively referred to as called ELGAR, was developed to allow solutions of optimisation problems. Although the principle use of this library was signal analysis, the library is flexible enough to permit investigations of other problems.

By default, each parameter in ELGAR can take on one of 256 different values. Due to the biological inspiration of the algorithm, these parameters are usually referred to as genes. This allows sufficient resolution without encountering 'boundary problems' associated with true floating point representations.

In addition to the signal analysis, some properties of GAs in general (such as the optimum mutation rate) were investigated using ELGAR. A discrepancy between the

results observed and those predicted was observed. This was accounted for by the different operational mode between ELGAR and typical binary genetic algorithms.

GAs proved a valuable method of solving the optimisation problems investigated. However, the improvement of the solution was found to be severely limited if the minimum 'valley' lies off-axis. This seems to be a fundamental limitation of GA techniques as further improvement requires multiple mutations of the required magnitude. This problem would be lessened if variable mutation rate scheme were employed.

6.2 Genetic Algorithm Signal Analysis

Signal analysis for models with non-linear dependent variables was investigated. GAs were found to be able to fit such models efficiently. An example of this is fitting a sinusoidal signal to some time sequence data (§2.6.2). It is important to note that the GA fitting procedure is via a purely forward modeling paradigm. Although fitting periodic signals can be achieved via Fourier decomposition, the GA does not use this information and therefore can be applied to a non-periodic or nonlinear signal.

Augmenting the likelihood function via Bayesian Inference model was found to greatly improve the fitting procedure. In the presence of high levels of noise, the addition of prior information can reduce any degeneracy and was found to reduce the error from fitting to the noise. In general, the optimisation problem then becomes more complex, typically there will be many local minima, but the GA technique was found to cope with this problem. Complex problems, such as extracting a reference signal from many poor examples that are not perfectly aligned (as described in §4.6), became tractable.

6.3 COMPASS-D experiment

An analysis of the diagnostic output was conducted. The method of solving the Bayesian augmented likelihood estimation via application of a Genetic Algorithm proved to be valuable in analysing results from the Thomson scattering diagnostic at COMPASS-D.

The analysis was further extended by considering a two-temperature distribution function. This form of perturbation of a Maxwellian distribution was chosen due to its simplicity and observational evidence. Several constraints on the observations were

found, despite the available information being insufficient to full constrain the two-temperature distribution function.

The region of 'observation space' consistent with an n-temperature distribution function was found to be bounded by the single-temperature response curve. Data from COMPASS-D diagnostic was analysed and found to like outside this region of valid solutions. However, the data appeared to be bounded by a similar curve suggesting systematic bias at some point in the analysis process. Several possible explanations for this bias were suggested.

For any valid observation, and assuming a two-temperature distribution function, let T_1 be the lower temperature and T_2 the upper temperature. Let the set of T_1 consistent with an observation be A and the set of T_2 consistent with the same observation be B. It was found that these two sets, A and B, are disjoint. For many possible observations, the set of temperatures in neither A or B (*i.e.* the intersection of the complements of A and B) contained the least-squares best-fit single-temperature solution. If we assume that the plasma is consistent with the two-temperature model with bulk temperature T_1 with some minority species at T_2 then the presence of these 'hot' electrons was found to bias the least-squares estimate towards higher temperatures.

6.4 Future work

The obvious extension for the two temperature analysis is to consider a many channel dataset and to recover a full electron distribution function. This recovery procedure would require a dataset with good spectral resolution. One possible source of such data is from a Television Thomson Scattering (TVTS) based diagnostic.

The problem of recovering the distribution function is an inverse problem. The form of the problem is given by Pechacek and Trivelpiece [1967] as stated in (4.4) on page 48. This is an inhomogeneous Fredholm equation of the first kind. Although the equation appears linear in $f(\beta)$, due to positivity constraints, the full problem will be non-linear. It is possible to solve (4.4) treating it as linear and forcing the positivity after the inversion.

Various methods exist for solving Fredholm equations. Most use regularisation to remove the degeneracy from the ill-posed nature of the inversion. Such regularisation

could be from imposing a (relativistic) Maxwellian-like distribution function or from Maximum Entropy constraint.

Stability is another concern in inversion. Sharp features in the response of one speed (considering all directions) indicates stability in the recovered distribution. The relativistic beaming effect implies that electrons approaching the observer are scattered preferentially to those receding. This introduces a skew in the observed spectrum which would improve the stability of the solution.

A GA will utilise many points in the attempt to solve the specified problem. When the GA finishes, the final generation will contain some best-fit solution as well as many other solutions. In general, these other solutions will be close to or identical to the best solution. If identical points are disallowed (an option in ELGAR) then the final generation will contain $n-1$ distinct points close to the best solution.

It may be possible to use this distribution of points to estimate the distribution function near to the best-fit solution. This would alleviate the necessity of a full Monte Carlo simulation for recovering confidence intervals without the normal distribution assumption. However, there would have to be careful analyses of GA dynamics (*e.g.* via Markov chain analysis) as the final points are very likely to be correlated.

The code for fitting the reference signal to observed data could be extended. At present it considers the reference signals to be perfect. However, after constructing reference signals there will inevitably be some uncertainty in the final result. This uncertainty is easily included by considering (3.4) on page 42. To consider errors on the reference signal of $\sigma_{i,j}$ the new likelihood function is:

$$\chi^2_D = \sum_{i=1,j=1}^{I,J} \frac{[d_{ij} - g_{ij}(\boldsymbol{\lambda})]^2}{\sigma^2_{i,j} + \sigma^2_D}. \tag{6.1}$$

The most obvious element that could be included to improve ELGAR is the addition of a local optimisation routine. Such routines would allow 'fine tuning' of the fitting parameters after the GA had finished. A standard multi-dimensional algorithm that only use function evaluations, such as Powell's method, could be employed. One such method could be adapted to work for the library. This code should then operate as an option for backward compatibility (see Appendix B for discussion).

ELGAR could also be extended by adding more advanced options such as aging, variable mutation rate and directed mutations. It is possible to simulate these effects

without altering ELGAR through carefully written client-side software (the problem-specific code). However, code that extends GA-based functionality belongs within the library. Inclusion of these extra features and operators would allow testing of the library against other GA implementations.

The overall method, using GAs to solve Bayesian inference problems, is applicable to a wide range of problems not limited to plasma diagnostic analysis. The method provides a systematic method of introducing previous experience whilst not requiring a initial point close enough to the true minimum. This identifies the algorithm as a highly suitable method of solving a wide range of problems. Therefore, applications in unrelated fields should be investigated.

Appendix A

Code listing

There are 207 program files (*i.e.* ending in `.c`) and 76 header files (*i.e.* files ending in `.h`). These contain 49670 lines of code which would take up 1035 pages to list. Because displaying all the code is impractical, only selected pieces are listed. The whole source code is available on a CDROM at the end of this document.

A.1 GenePool structure

The C programming language allows many variable types to be grouped together to form a 'structure'. Structures usually contain many related items and grouping these items together allows easy handling of often abstract objects. For example, C has no concept of a complex number but a structure consisting of two floating point numbers (labelled real and imaginary) would allow a function to take a complex number as a single argument.

The most important structure within ELGAR is GenePool. This contains all the information about the optimisation. As all the information about the optimisation in contained within the structure, there can be any number of instances of ELGAR running concurrently or hierarchically.

The following code listing is as present on cdrom, including all comments.

File: GA/Elgar/elgar.h

```
59 typedef struct gp_struct{
60
61 /* ALL values from here to the AUTOINITALISE section *MUST* be
```

```
62   * filled in before calling the GimiGenePool() routine.
63     */
64  /* These are the parameters to main. They are required
65               * by the X interface so that X-like options can
66     * be processed. */
67  int gp_argc;
68  char **gp_argv;
69
70  /* Number of parents and number of children should be
71                   * obvious. Note that these values must be greater than
72                   * the minimum above. Also, number of children must be
73   * an even number.*/
74  unsigned gp_NoOfParents;
75  unsigned gp_NoOfChildren;
76
77  /* This is the number of genes needed to describe each
78   * entity. A byte is allocated for each Gene. Note, it
79                   * is purely the responsibility of the gp_CalculateError()
80   * routine to assign some kind of interpretation on these
81              * bytes. */
82  unsigned gp_NoOfGenes;
83
84  /* The number of byte-wise mutations to impose per
85         * generation. At the moment this is a fixed number. */
86  unsigned gp_MutationsPerGeneration;
87
88
89  /* Boolian value:
90   * TRUE => display X window
91                * FALSE => don't use X interface */
92  unsigned gp_GAMEOn;
93
94  /* This is the pointer to the routine for calculating
95   * the error for a specified entity. This is the problem
96   * specific area. It must be a routine with a prototype
97   * like:
98   * #include "elgar.h"
99   *
100  * int CalcErr( Entity *);
101  *
102  * and assigned within c-code thus:
103  *
104  *  myGenePool.gp_CalculateError = CalcErr;
105  *
106  * It is the responsibility of this routine to fill in
107  * the element ent_Error within the entity, based on
108  * the gene sequence. */
109 int (*gp_CalculateError)( Entity *);
110
```

```
111
112 /* These entries *MUST* be filled in, but may contain blanks/NULL. NB
113  *  Don't assume that they are blanks: initalise them! */
114
115 /* If this is not NULL then the routine is called after
116  * each generation. This allows the calling routine(s)
117  * to update some information, watch progress, etc */
118 int (*gp_NewGeneration)( struct gp_struct *);
119
120
121 /* Sometimes a different method of breeding is required
122  * This entry allows the "hooking" of a different breeding
123  * routine. The first two Entities are the parents and the
124  * second two are the children. The whole GenePool structure
125  * is also passed as a parameter (in case it is needed).
126  * To disable this option, assign the value NULL to it. */
127 int (*gp_NewBreed)( struct gp_struct *, Entity *, Entity *,\
128 Entity *, Entity *);
129
130 /* Another hook entry, this time for the mutation operation
131  * Set to NULL to use the built-in one. */
132 int (*gp_NewMutate)( struct gp_struct *, Entity *);
133
134 /* If this value is non-zero, then the optimisation
135  * routines will terminate if the lowest error (best
136  * solution) is less than this value. */
137 float gp_Tolerance;
138
139 /* If the range of parent error is less than the best error
140  * divided by this number, then all but the best parent are
141  * replaced by random values. */
142 unsigned gp_CatastrophicFraction;
143
144 /* The maximum number of catastrophes before "giving up" and
145  * returning from FindMinimum(). Set to zero for an infinite
146  * number. NB this option is ignored if gp_CatastrophicRange
147  * is set to -1.0. */
148 int gp_MaxNoCatastrophes;
149
150 /* If this value is greater than zero, then FindMinimum() will
151  * give up after this number of generations. NB other criteria
152  * may cause FindMinimum() to return before this. */
153 unsigned gp_MaxNoOfGenerations;
154
155
156                 /* If this is non-zero, then parents will die after gp_MaxAge
157  * generations. */
158         unsigned gp_MaxAge;
159
```

```
160 /* If non-NULL, this string will appear as part of the title
161  * of the GAME window. The memory pointed to by gp_Title
162  * is copied, so it need not have extent for the
163  * duration of the optimisation. */
164 char *gp_Title;
165
166
167 /* Boolean value:
168  * TRUE  = All (completely) identical entities in the gene
169  * pool are removed. The method of removal is that
170  * they are moved from the parent area to the far
171  * end of the child area. They are then lost in the
172  * next lot of breading.
173  * FALSE = Multiple copies of gene-identical entities
174  *  are allowed. */
175 int gp_RemoveIdenticals;
176
177 /* Elgar will pause for gp_SleepFor milliseconds after each
178  * generation. The sleep occures after GAME has refreshed (if
179  * active) and after calling gp_NewGeneration() (if defined). */
180 unsigned gp_SleepFor;
181
182 /* The seed for the random number generator. Set to zero for the
183  * default. */
184 unsigned gp_Seed;
185
186 /* Elgar _can_ produce output to STDERR as a quick error-tracing
187  * facility. The output is subdivided into three classes:
188  * 1. Critical errors - Something has gone
189  * very wrong and Elgar
190  * cannot recover from this.
191  * 2. Recoverable errors - Something has gone
192  * wrong but Elgar can use
193  * a default value, etc.
194  * 3. Warnings - Something unexpected happened
195  * which has no direct consequence.
196  * 4. Information - General information (there's an
197  * awful lot of this!).
198  * Set gp_Verbosity to 0 for no output, 1 for just critical errors, 2
199  * for Critical and Recoverable errors etc. */
200 unsigned gp_Verbosity;
201
202 /*       ************************************
203  *    *****          AUTOINITALISE          *****
204  *       ************************************
205  *
206  * ... from here onwards the variables do not need to be specificed before calling
207  * GimiGenePool().
208  */
```

```
209
210 /* These four variables are used for the two relevent
211  * graphs. */
212 unsigned long *gp_SpliceSpread;
213 unsigned long *gp_SpliceTotals;
214 unsigned long *gp_MutateSpread;
215 unsigned long *gp_MutateTotals;
216
217
218 /* Here we store the evolution of the GA
219    * gp_BestEvolution: time-evolution of the best solution.
220  * gp_MedianEvolution: evolution of the middle solution.
221  * gp_EvolutionCount: total number of entries taken. Increases
222  *       from zero to EVOLUTION_SIZE.
223  * gp_NextPoint: where in the arrays we should insert the next
224  *  data. */
225 float *gp_BestEvolution;
226 float *gp_MedianEvolution;
227 unsigned gp_EvolutionCount;
228 unsigned gp_NextEvolutionPoint;
229
230 /* Calculated automatically from gp_NoOfParents and
231  * gp_NoOfChildren. Also includes the two spares required
232  * by the QuickSort algorithm. */
233 unsigned gp_NoOfEntities;
234
235 /* The current generation's number. Initalised to
236  * zero */
237 unsigned gp_GenerationNo;
238
239 /* Elgar's current "problem". If it's zero, then there is no
240  * problem. */
241 unsigned gp_ErrorLevel;
242
243 /* The collection of parents, which is allocated automatically */
244 Entity *gp_Parents;
245
246 /* This collection of children, which is allocated automatically */
247 Entity *gp_Children;
248
249 /* We need a couple of spare entities for the QuickSort
250  * algorithm (and other things too). These, too, are
251  * automatically allocated *BUT* these spare entities do
252  * NOT have gene sequences allocated. */
253 Entity *gp_Spare1;
254 Entity *gp_Spare2;
255
256 /* Used by the Numerical Recipes in C random number generator
257  * ran1() (see chapter 7 of NRC) */
```

```
258 long gp_idum;
259
260
261 /* These are specified by whichever GAME's being used */
262 GameDetails gp_GAME;
263
264 } GenePool;
```

A.2 The Entity structure

Another important structure is 'Entity'. This contains all information about a particular entity within the genetic algorithm (see §2.1). It includes the creature's genes, it's error value and its age. The definition is:

File: GA/Elgar/elgar.h

```
49 typedef struct {
50         float           ent_Error;
51         unsigned        ent_Age;
52         GeneSequence    ent_Genes;
53 } Entity;
```

where the GeneSequence is defined as

File: GA/Elgar/elgar.h

```
44 typedef unsigned char * GeneSequence;
```

Although all the programs consider GeneSequence as a collection of numbers in either base 256 or base $(256)^2$, the only parts of ELGAR that make that distinction are the breeding and mutation routines. These can be replaced via the `gp_NewBreed` and `gp_NewMutate` elements of the GenePool structure. If new breeding and mutation routines are written, the interpretation (*i.e.* conversion into phenotype) of GeneSequence is arbitrary.

A.3 Thomson scattering specific code

The code specifically for analysing the Thomson scattering diagnostic channels is included here. The function DoJob() is the main entry point for this code. It is called by the parser when it encounters the Go command (see Appendix B, especially §B.7.3). The Job structure contains all information about a particular fitting problem.

File: GA/Thomson/noglobal.c

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <math.h>
4  #include "elgar.h"
5  #include "random.h"
6  #include "display.h"
7  #include "noglobal.h"
8
9  /* Routines to calculate temperature from areas */
10 #include "temp.h"
11
12 #define GOFFSET_LOWER (-50.0)
13 #define GOFFSET_RANGE (100.0)
14
15 #define DOUBLE_GENES
16
17
18 /* History:
19  *  29/11/98 Relaxed condition that dT (uncertainty in temperature) must lie
20  *           at least one standard deviation away from zero in StoreBestSoln().
21  */
22
23
24 /* A generalised collection of statistical information */
25 typedef struct {
26   float Mean;
27   float AverageDeviation;
28   float StandardDeviation;
29   float Variance;
30   float Skewness;
31   float Kurtosis;
32 } StatsInfo;
33
34
35 /* A collection of all the parameters required
36  * to specify a solution */
37 typedef struct {
38   float Offsets [3];
39   float Amplitudes [3];
40   float DCLevels [3];
41 } Solution;
42
43
44 typedef struct {
45   float WaveMaximum[3], WaveMinimum[3];
46   Job *myJob;
47   unsigned Generation;
48   float *fits[3], *residuals[3];
```

```
49    float area[3];
50    DisplayCtrl *win;
51    float ResidalsVarianceEstimate[3];
52  } GlobalVariables;
53
54
55  GlobalVariables Globals;
56
57  /* This is the error function, as required by the GA. */
58  int ErrFn( Entity *);
59  void ConvertToNumbers(unsigned char *, Solution *);
60  void BuildFits( Solution *);
61  void StoreBestSoln( GenePool *, Job *);
62  char *StrDup( char *);
63  void CalcStats( float *, unsigned, StatsInfo *);
64  int SoFar( GenePool *);
65  void PlotGraph( unsigned, unsigned, unsigned, float *, float *, unsigned);
66  int Initalise( Job *);
67  void CleanUp( Job *);
68  int compare( const void *, const void *);
69
70
71  int Initalise( Job *myJob)
72  {
73    float squares, value, *sorted, median, area, minimum, maximum;
74    unsigned i, RefNo, wave;
75    StatsInfo stats;
76
77    DisplayRequest reqst;
78
79    /* Open window to display best solution after each generation. */
80    reqst.Width = 520;
81    reqst.Height = 300;
82    reqst.Title1 = "Best Solution";
83    reqst.Title2 = "Soln";
84    reqst.WaitAtEnd = FALSE;
85    if( myJob->DoGraphics) {
86      Globals.win = OpenWindow( myJob->argc, myJob->argv, &reqst);
87      if( Globals.win == NULL)
88        fprintf( stderr, "Unable to open X-Windows\n");
89    }
90    else
91      Globals.win = NULL;
92
93
94    /* Point the global variable myJob to myJob. */
95    Globals.myJob = myJob;
96
97    /* Normalise reference pulse shapes. First we work out a good(ish)
```

```
98      * approximation to the DC offset. Note we use the median instead of
99      * the mean since the median is a more robust estimate of the DC
100     * Offset (see section 15.7 in Numerical Recipes in C) */
101    for( RefNo = 0; RefNo < 3; RefNo++) {
102
103      /* Calculate the median. We use the standard library function qsort()
104       * which aught to be portable. */
105      sorted = (float *) malloc( myJob->nReference[RefNo]*sizeof(float));
106      if( sorted == NULL) {
107        printf( "Out of memory trying to sort Reference[%u].\nTerminating.\n", RefNo);
108        return 1;
109      }
110      for( i = 0; i < myJob->nReference[RefNo]; i++)
111        sorted [i] = myJob->Reference[RefNo] [i];
112      qsort( sorted, myJob->nReference[RefNo], sizeof( float), compare);
113      median = sorted [myJob->nReference[RefNo]/2];
114      maximum = sorted [myJob->nReference[RefNo]-1]-median;
115      free( sorted);
116
117      /* Reset Reference[i] so it goes through the zero. */
118      for( i = 0; i < myJob->nReference[RefNo]; i++)
119        myJob->Reference [RefNo] [i] -= median;
120
121      /* Normalise the pulse shape so that it has maximum height of 1.0. */
122      for( i = 0, area = 0.0; i < myJob->nReference[RefNo]; i++)
123        myJob->Reference [RefNo] [i] /= maximum;
124
125      /* Code to calculate the sum of the squares in the (normalised) function. */
126      for( i=0, area=squares=0.0; i < myJob->nReference [RefNo]; i++) {
127        area += (value = myJob->Reference [RefNo] [i]);
128        squares += value * value;
129      }
130      myJob->Factor[RefNo] = sqrt( squares);
131      Globals.area[RefNo] = area;
132    } /* for */
133
134
135
136    /* Calculate the maximum and minimum values for each Waveform. */
137    for( wave = 0; wave < 3; wave++) {
138
139      /* Find minimum and maximum for this waveform. */
140      maximum = myJob->Waveform [wave] [0];
141      minimum = maximum;
142      for( i = 1; i < myJob->nWaveform[wave]; i++) {
143        if( myJob->Waveform [wave] [i] < minimum)
144  minimum = myJob->Waveform [wave] [i];
145        if( myJob->Waveform [wave] [i] > maximum)
146  maximum = myJob->Waveform [wave] [i];
```

```
147     } /* for */
148
149     /* Store the results. */
150     Globals.WaveMinimum[wave] = minimum;
151     Globals.WaveMaximum[wave] = maximum;
152
153     sorted = (float *) malloc( myJob->nWaveform [wave] * sizeof(float));
154     if( sorted == NULL) {
155       printf( "out of memory.\n");
156       return 1;
157     }
158     for( i = 0; i < myJob->nWaveform[wave]; i++)
159       sorted [i] = myJob->Waveform[wave] [i];
160     qsort( sorted, myJob->nWaveform[wave], sizeof( float), compare);
161
162     /* Remove the upper 5 percentile and take the average deviation:
163      * a more robust statistic. */
164     CalcStats( sorted, (unsigned) (myJob->nWaveform [wave] * .95), &stats);
165     Globals.ResidalsVarianceEstimate [wave] = stats.AverageDeviation * \
166                                               stats.AverageDeviation;
167     free( sorted);
168
169   } /* for */
170
171
172   /* Allocate memory for storing fits and residuals. */
173   for( i = 0; i < 3; i++) {
174     Globals.fits[i] = (float *) malloc( myJob->nWaveform[i] * sizeof( float));
175     if( Globals.fits[i] == NULL) {
176       printf( "Initalise: out of memory.\n");
177       return 1;
178     }
179   }
180   for( i = 0; i < 3; i++) {
181     Globals.residuals[i]=(float *)malloc( myJob->nWaveform[i] * sizeof( float));
182     if( Globals.residuals[i] == NULL) {
183       printf( "Initalise: out of memory initalising storage.\n");
184       return 1;
185     }
186   }
187
188   return 0;
189 } /* Initalise. */
190
191
192 void CleanUp( Job *myJob)
193 {
194   unsigned i;
195
```

```
196   /* Fool (cc) into not warning that myJob is not used */
197   myJob = myJob;
198
199   /* Free memory allocated for storing residuals and fits. */
200   for( i = 0; i < 3; i++) {
201     free( Globals.fits[i]);
202     free( Globals.residuals[i]);
203   }
204
205   if( Globals.win)
206     CloseWindow( Globals.win);
207
208 } /* CleanUp */
209
210
211 /* This function acts as a simple comparitor for the stdlib library
212  * quick sort algorithm qsort() */
213 int compare( const void *first, const void *second)
214 {
215   if(*(float*)first==*(float*)second)return 0;
216   return (*(float*)first<*(float*)second)?-1:1;
217 } /* compare */
218
219
220
221
222
223 int DoJob( Job *myJob)
224 {
225   GenePool myGA;
226
227
228   if( Initalise( myJob))
229     return 1;
230
231
232   /* Build Genetic Algorithm information. */
233   InitaliseGenePool( &myGA);
234   myGA.gp_argc = myJob->argc; myGA.gp_argv = myJob->argv;
235   myGA.gp_NoOfParents = myJob->NoParents;
236   myGA.gp_NoOfChildren = myJob->NoChildren;
237 #ifdef DOUBLE_GENES
238   myGA.gp_NoOfGenes = 18;
239 #else
240   myGA.gp_NoOfGenes = 9;
241 #endif
242   myGA.gp_MutationsPerGeneration = myJob->NoMutations;
243   myGA.gp_MaxNoOfGenerations = myJob->NoOfGenerations;
244   myGA.gp_GAMEOn = myJob->DoGAME;
```

```
245   myGA.gp_CalculateError = ErrFn;
246   myGA.gp_NewGeneration = (myJob->DoGraphics) ? SoFar : NULL;
247   myGA.gp_Tolerance = -1;
248   myGA.gp_CatastrophicFraction = 100000; /* 100000;*/
249   myGA.gp_MaxNoCatastrophes = 0;
250   myGA.gp_Title = "Thomson Signal fitting";
251   myGA.gp_RemoveIdenticals = TRUE;
252   myGA.gp_SleepFor = 0;
253   myGA.gp_Seed = 0; /*myJob->RndSeed;*/
254   myGA.gp_Verbosity = VERBOSE_WARNING_OUTPUT;
255
256   Globals.Generation = 0;
257
258   Randomise( &myGA);
259
260   /* Allocate GA resources */
261   if( GimiGenePool( &myGA))
262     return -1;
263
264
265   /* Find the answer. */
266   if( FindMinimum( &myGA))
267     printf( "GA returned an error code.\n");
268
269   /* Store answer. */
270   StoreBestSoln( &myGA, myJob);
271
272
273   /* Clean up. */
274   FreeGenePool( &myGA);
275   CleanUp( myJob);
276
277
278   /* That's us */
279   return 0;
280 } /* DoJob */
281
282
283
284
285
286 void StoreBestSoln( GenePool *myGA, Job *myJob)
287 {
288   Areas myAreas;
289   Temperature myTemp;
290   Calibration myCalib;
291   Solution bestSolution;
292   unsigned i, wave;
293   float TotalChiS = 0.0;
```

```
294   StatsInfo stats[3];
295
296   /* be optimistic. */
297   myJob->InvalidSoln = 0;
298
299   /* Build the best fits */
300   ConvertToNumbers(myGA->gp_Parents[0].ent_Genes, &bestSolution);
301   BuildFits( &bestSolution);
302
303
304   /* Build the residuals. */
305   for( wave = 0; wave < 3; wave++)
306     for( i = 0; i < myJob->nWaveform [wave]; i++) {
307       Globals.residuals [wave] [i] = myJob->Waveform [wave] [i] -\
308                                           Globals.fits[wave] [i];
309       TotalChiS += Globals.residuals [wave] [i] * Globals.residuals [wave] [i];
310     }
311
312
313   /* Calculate the stats on each of the residuals. */
314   for( wave = 0; wave < 3; wave++)
315     CalcStats( Globals.residuals [wave], myJob->nWaveform [wave],&stats[wave]);
316
317
318   for( wave = 0; wave < 3; wave++){
319     myJob->BestDC [wave] = bestSolution.DCLevels [wave];
320     myJob->BestOffset [wave] = bestSolution.Offsets [wave];
321
322     myJob->BestAmp [wave] = bestSolution.Amplitudes [wave];
323     myAreas.A[wave]=myJob->BestArea [wave] = bestSolution.Amplitudes [wave] *\
324                                           Globals.area [wave];
325     myJob->AreaError [wave] = stats[wave].StandardDeviation *\
326                                     Globals.area[wave] / myJob->Factor[wave];
327     myJob->AreaError [wave] *= myJob->MultiError;
328     myAreas.dA[wave] = myJob->AreaError [wave];
329     myAreas.dA2[wave] = myAreas.dA[wave] * myAreas.dA[wave];
330     myJob->ResidualMean [wave] = stats[wave].Mean;
331     myJob->ResidualSD [wave] = stats[wave].StandardDeviation;
332     memcpy( myJob->Fit [wave], Globals.fits[wave],
333             myJob->nWaveform [wave]*sizeof( float));
334     memcpy( myJob->Residuals [wave], Globals.residuals[wave],
335             myJob->nWaveform [wave]*sizeof( float));
336   }
337
338   /* Point the entries in myCalib to the correct bits of the calibration
339    * data passed to this section from the parser. */
340   myCalib.NoOfEntries = myJob->nCalibration;
341   myCalib.T = myJob->Calibration[0];
342   myCalib.C[0] = myJob->Calibration[1];
```

```
343   myCalib.C[1] = myJob->Calibration[2];
344   myCalib.C[2] = myJob->Calibration[3];
345
346   /* Attempt to calculate the temperature. */
347   if( AreasToTemp( &myAreas, &myTemp, &myCalib)) {
348     myJob->InvalidSoln = 1;
349   }
350
351   /* Store results. */
352   myJob->Temp = myTemp.T;
353   myJob->TempError = myTemp.dT;
354
355   myJob->ChiS = TotalChiS;
356
357   /* We relax the condition that the result must lie one sigma away from zero.
358    *
359    * if( myTemp.dT >= myTemp.T)
360    *   myJob->InvalidSoln = 1;
361    */
362
363 } /* StoreBestSoln */
364
365
366
367 /* Function for calculated the error. Required for the GA. */
368 int ErrFn( Entity *myEntity)
369 {
370   Solution mySolution;
371   float diff, goffset;
372   float chi1_sqr, chi2_sqr, chi3_sqr, chi_sqr;
373   unsigned i, wave;
374
375
376   /* Convert our genes to significant numbers. */
377   ConvertToNumbers( myEntity->ent_Genes, &mySolution);
378
379   /* Build our fit, based on these numbers. */
380   BuildFits( &mySolution);
381
382   /* Calculate chi squared for just the data. */
383   chi1_sqr = 0.0;
384   for( wave = 0; wave < 3; wave++) {
385     for( i = 0; i < Globals.myJob->nWaveform [wave]; i++) {
386       diff = (Globals.myJob->Waveform [wave] [i] - Globals.fits [wave] [i]);
387       chi1_sqr += diff*diff / Globals.ResidalsVarianceEstimate[wave];
388     }
389   }
390
391   /* Calculate the global offset. This is just the average offset */
```

```
392   for( wave = 0, goffset=0.0; wave < 3; wave++)
393     goffset += mySolution.Offsets [wave];
394   goffset /= 3;
395
396
397   /* Now calculate the sum of the differences squared */
398   for( wave = 0,chi2_sqr = 0.0; wave < 3; wave++) {
399     diff = goffset - mySolution.Offsets [wave];
400     chi2_sqr += diff * diff / (2*0.61*0.61);
401   }
402
403   /* N/(N-1) Finite population factor */
404   chi2_sqr *= 3/2;
405
406
407   /* This is the "pulse roughly in the middle" bit */
408   /*  chi3_sqr = goffset * goffset / (1.5*1.5);*/
409   diff = goffset - 123.819;
410   chi3_sqr =  diff*diff / (2*2.96074*2.96074);
411
412   /* The overall chi squared is the weighted sum of these two components
413    * The weighting factor comes from (s_D^2/s_O^2) where s_D^2 is the
414    * square of the standard deviation of typical noise and s_O^2 is the
415    * square of the standard deviation of the offsets. s_D^2 was found to
416    * be approx 84.4 and s_O^2 is approx 1.2. */
417   chi_sqr = chi1_sqr  + chi2_sqr + chi3_sqr;
418
419
420   /* Store this as the error. */
421   myEntity->ent_Error = chi_sqr;
422
423   return 0;
424 } /* ErrFn */
425
426
427
428
429 void ConvertToNumbers(unsigned char *genes, Solution *mySoln)
430 {
431   unsigned i;
432   signed mumin, murange;
433   float rep[10], range;
434
435   for( i = 0; i < 9; i++){
436 #ifdef DOUBLE_GENES
437     rep[i] = genes[2*i]/256.0 + genes[2*i+1]/(65536.0);
438 #else
439     rep[i] = genes[i]/256.0;
440 #endif
```

```
441   }
442
443   for( i = 0; i < 3; i++) {
444     range = Globals.WaveMaximum [i] - Globals.WaveMinimum [i];
445     mumin = -(Globals.myJob->nReference[i]/2);
446     murange = (Globals.myJob->nReference[i]+Globals.myJob->nWaveform[i])/2;
447
448     mySoln->Offsets[i] = mumin + rep[  i]*murange;
449
450     mySoln->Amplitudes[i] = rep[3+i]*range;
451
452     mySoln->DCLevels [i] = Globals.WaveMinimum [i] + \
453                                     rep[6+i]*range;
454
455   }
456
457 } /* ConvertToNumbers */
458
459
460
461
462
463 void BuildFits( Solution *mySoln)
464 {
465   unsigned PulseIndex=0, FitIndex=0;
466   unsigned PulseLength, FitLength, i, waveno;
467   signed offset;
468
469   for( waveno = 0; waveno < 3; waveno++) {
470
471     PulseIndex = 0;
472     FitIndex = 0;
473     offset = (signed) mySoln->Offsets [waveno];
474
475     PulseLength = Globals.myJob->nReference [waveno];
476     FitLength = Globals.myJob->nWaveform [waveno];
477
478     /* Fill in the blanks before PulseShape */
479     if( offset < 0) {
480       PulseIndex = -offset;
481     }
482     else {
483       if( offset >= Globals.myJob->nWaveform [waveno])
484 offset = Globals.myJob->nWaveform [waveno];
485
486       for( i=0; i < offset; i++)
487 Globals.fits [waveno] [i] = mySoln->DCLevels [waveno];
488       FitIndex = offset;
489     }
```

```
490
491     /* Copy PulseShape with correct amplitude and DC offset. */
492     for(;(PulseIndex<PulseLength)&&(FitIndex<FitLength);PulseIndex++\
493                                                         ,FitIndex++)
494       Globals.fits [waveno] [FitIndex] = Globals.myJob->Reference [waveno] [PulseIndex]\
495                                          * mySoln->Amplitudes [waveno] \
496                                          + mySoln->DCLevels [waveno];
497
498
499     /* Pad out after PulseShape with the DC level. */
500     for(;FitIndex < FitLength;FitIndex++)
501       Globals.fits [waveno] [FitIndex] = mySoln->DCLevels [waveno];
502   } /* for */
503
504 } /* BuildFit */
505
506
507
508 /* Routine to see how the optimisation is progressing. Displays problem specific
509  * information. */
510 int SoFar( GenePool *GP)
511 {
512   unsigned WinHeight, i;
513   Solution bestSoln;
514
515   if( !Globals.win)
516     return 0;
517
518   /* Build best solution */
519   ConvertToNumbers( GP->gp_Parents[0].ent_Genes, &bestSoln);
520   BuildFits( &bestSoln);
521
522   /* Graph these solutions */
523   ClearPad( Globals.win);
524
525   /* Height of each graph, 5 pixel gaps */
526   WinHeight = (Globals.win->d_Height - 5 * (3+1))/3;
527
528   for( i = 0; i < 3; i++)
529     PlotGraph( 5, 5+i*(5+WinHeight), (i+1)*(5+WinHeight),
530        Globals.myJob->Waveform [i], Globals.fits[i],\
531        Globals.myJob->nWaveform [i]);
532
533   /*
534   PlotGraph( 5,  50, 95, Globals.myJob->Waveform [1], Globals.fits[1],\
535      Globals.myJob->nWaveform [1]);
536   PlotGraph( 5, 100,145, Globals.myJob->Waveform [2], Globals.fits[2],\
537      Globals.myJob->nWaveform [2]);
538   */
```

```
539
540   SwapBuffers( Globals.win);
541
542   return 0;
543 } /* sofar */
544
545
546
547 /* This function emulates the function strdup() for portability. */
548 char *StrDup( char *string)
549 {
550   char *result;
551   unsigned length;
552
553   if( string == NULL)
554     return NULL;
555
556   if( (length = strlen( string)) == 0)
557     return NULL;
558
559   result = (char *) malloc( length+1);
560   if( result == NULL)
561     return NULL;
562
563
564   strcpy( result, string);
565
566   return result;
567 } /* DupStr */
568
569
570
571 /* Calculate various statistics on data. This routine is a modified version
572  * of the one appearing in Numerical recipies in C (see section 14.1 in NRC). */
573 void CalcStats( float *data, unsigned n, StatsInfo *info)
574 {
575   int j;
576   float ep=0.0,s,p;
577
578   if( n <= 1) {
579     printf( "CalcStats: there must be at least 2 data pts\n");
580     return;
581   }
582   s=0.0;
583   for(j=0; j<n; j++) s += data[j];
584   info->Mean = s/n;
585   info->AverageDeviation = info->Variance = info->Skewness = info->Kurtosis = 0.0;
586   for(j=0; j<n; j++) {
587     info->AverageDeviation += fabs(s=data[j]-info->Mean);
```

```
588       ep += s;
589       info->Variance += (p = s*s);
590       info->Skewness += (p *= s);
591       info->Kurtosis += (p *= s);
592     }
593
594     info->AverageDeviation /= n;
595
596     info->Variance = (info->Variance - ep*ep/n)/(n-1);
597     info->StandardDeviation = sqrt( info->Variance);
598     if( info->Variance) {
599       info->Skewness /= (n * info->Variance * info->StandardDeviation);
600       info->Kurtosis = info->Kurtosis/(n*info->Variance*info->Variance)-3.0;
601     }
602     else {
603       /*printf( "CalcStats: no skewness/kertosis when Variance = 0.0\n");*/
604       info->Skewness = 0.0;
605       info->Kurtosis = 0.0;
606     }
607
608   } /* CalcStats */
609
610
611
612   void PlotGraph( unsigned x, unsigned y1,unsigned y2,\
613   float *data1, float *data2, unsigned num)
614   {
615     signed dy, i, yp, yn, y_1, y_2;
616     float y_scale, largest, smallest;
617
618     y_1 = y1;
619     y_2 = y2;
620
621     /* calculate the y_scale */
622     dy = y2-y1;
623     for( i = 1, smallest=largest=data1[0]; i < num; i++) {
624       if( smallest > data1 [i])
625         smallest = data1 [i];
626       if( largest < data1 [i])
627         largest = data1 [i];
628     }
629     y_scale = dy / (largest - smallest);
630
631
632     /* Plot data. */
633     for( i = 1; i < num; i++) {
634       XSetForeground( Globals.win->d_Display, Globals.win->d_gc,\
635       Globals.win->d_Foreground);
636       XDrawLine( Globals.win->d_Display,Globals.win->d_DDO,Globals.win->d_gc,\
```

```
637         x + 2*(i-1), y2 - y_scale*(data1 [i-1] - smallest),\
638         x + 2*i,   y2 - y_scale*(data1 [i] - smallest));
639
640       yp = y2 - y_scale*(data2[i-1]-smallest);
641       if( yp < y_1)
642         yp = y1;
643       else
644         if( yp > y_2)
645 yp = y2;
646       yn = y2 - y_scale*(data2[i]-smallest);
647       if( yn < y_1)
648         yn = y1;
649       else
650         if( yn > y_2)
651 yn = y2;
652       XSetForeground( Globals.win->d_Display, Globals.win->d_gc,\
653       Globals.win->d_blue);
654       XDrawLine( Globals.win->d_Display,Globals.win->d_DDO,Globals.win->d_gc,\
655          x + 2*(i-1), yp, x + 2*i,   yn);
656    }
657
658
659 } /* PlotGraph */
```

Appendix B

Thomson scattering analysis scripting language

As quite often happens with code development, the design specification changed after the code was produced. Such things as the format that the data was supplied in changed as did the final output format and the nature of the analysis.

When faced with the problem of changing design, a decision is needed whether to extend the limited existing system to cover both the old and new requirements, or to retire the existing code and rewrite (*e.g.* copy and alter) the code to support only the new requirements. The latter option is generally quicker in the short term but suffers many problems: future improvements must be applied to each different implementation, bugs fixed in one must be applied to different source trees, potentially introducing further bugs, etc.

Extending the code to cover new cases will generally take longer to program, at least initially. However, this is offset by the increased ease with which future improvements can be included. Other benefits include any bugs are fixed in the one source tree and improvements can be implemented whilst retaining backwards compatibility. For these reasons, the code to analyse the Thomson scattering problem was extended over time. Because of this, there is one large program rather than many smaller specialised programs.

In order to obtain the required level of flexibility, a scripting language was developed. This was initially a simple set of commands and registers but has been extended to solve

alternative problems as the need arose. The language now includes support for global and local assignments, arrays, loops and a collection of commands including the ability to call external programs.

B.1 Backus-Naur Form

When defining a computer language it is usual to use the Backus-Naur Form (usually referred to as BNF). A logical element of the grammar is denoted by a label contained within angular brackets. These logical elements are broken down into one or more possible simpler expressions. Alternative expressions are separated by a vertical line character. Literal strings are contained within single quote marks. This is true except for space, tab and end of line . An exclamation mark in front of a set indicates a logical not and a dot represents the empty set.

The first three definitions in this grammar are irreducible elements of the language, usually referred to as tokens. The remaining elements are comprised of these simpler elements.

⟨*space char*⟩ ::= '␣'

⟨*tab char*⟩ ::= '\t'

⟨*EOL char*⟩ ::= '\n'

⟨*digit*⟩ ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

⟨*white space char*⟩ ::= ⟨*space char*⟩
| ⟨*tab char*⟩

⟨*white space*⟩ ::= ⟨*white space char*⟩ ⟨*white space*⟩ | ⟨*white space char*⟩

⟨*opt white space*⟩ ::= ⟨*white space*⟩ | .

⟨*req white space*⟩ ::= ⟨*white space char*⟩ ⟨*opt white space*⟩

⟨*int*⟩ ::= ⟨*digit*⟩ ⟨*int*⟩ | ⟨*digit*⟩

⟨*opt int*⟩ ::= ⟨*int*⟩ | .

⟨*mantissa*⟩ ::= ⟨*int*⟩ | ⟨*opt int*⟩ '.' ⟨*int*⟩

⟨*e*⟩ ::= 'e' | 'E'

⟨*exponent*⟩ ::= ⟨*e*⟩ ⟨*int*⟩

⟨*float*⟩ ::= ⟨*mantissa*⟩ | ⟨*mantissa*⟩ ⟨*exponent*⟩

⟨*text*⟩ ::= !⟨*EOL char*⟩ ⟨*text*⟩ | !⟨*EOL char*⟩

⟨*opt text*⟩ ::= ⟨*text*⟩ | .

⟨*non white space char*⟩ ::= !(⟨*EOL char*⟩ | ⟨*white space char*⟩)

⟨*non white space text*⟩ ::= ⟨*non white space char*⟩ ⟨*non white space text*⟩
| ⟨*non white space char*⟩

⟨*empty statement*⟩ ::= .

⟨*comment statement*⟩ ::= '#' ⟨*opt text*⟩

⟨*string*⟩ ::= ⟨*non white space text*⟩
| '"' ⟨*text*⟩ '"'

⟨*format type*⟩ ::= 'ASCII_Column'
| 'ASCII_FreeForm'
| 'LeCroy_Waveform'

⟨*full numeric range*⟩ ::= ⟨*int*⟩ ⟨*opt white space*⟩ '..' ⟨*opt white space*⟩ ⟨*int*⟩

⟨*numeric range*⟩ ::= ⟨*full numeric range*⟩
| '..' ⟨*opt white space int*⟩
| ⟨*int*⟩ ⟨*opt white space*⟩ '..'

⟨*series number*⟩ ::= ':' ⟨*opt white space*⟩ ⟨*int*⟩

⟨*data series*⟩ ::= ⟨*text*⟩
| ⟨*text*⟩ ⟨*opt white space*⟩ ⟨*series number*⟩

⟨*numerical range ref*⟩ ::= '[' ⟨*opt white space*⟩ ⟨*numeric range*⟩ ⟨*opt white space*⟩ ']'

⟨*data reference*⟩ ::= ⟨*data series*⟩
| ⟨*data series*⟩ ⟨*opt white space*⟩ ⟨*numerical range ref*⟩

⟨*true*⟩ ::= 'y' | 'yes' | 'true' | '1'

⟨*false*⟩ ::= 'n' | 'no' | 'false' | '0'

⟨*boolean*⟩ ::= ⟨*true*⟩ | ⟨*false*⟩

⟨*range*⟩ ::= ⟨*full numeric range*⟩ | ⟨*string*⟩

⟨*non white space list*⟩ ::= ⟨*non white space text*⟩ ',' ⟨*non white space list*⟩
| ⟨*non white space text*⟩

⟨*white space list*⟩ ::= ⟨*text*⟩ ',' ⟨*list*⟩ | ⟨*text*⟩

⟨*list*⟩ ::= ⟨*non white space list*⟩
| '"' ⟨*white space list*⟩ '"'

⟨*string assignment*⟩ ::= ⟨*opt white space*⟩ '=' ⟨*opt white space*⟩ ⟨*string*⟩

⟨*format assignment*⟩ ::= ⟨*opt white space*⟩ '=' ⟨*opt white space*⟩ ⟨*format type*⟩

⟨*integer assignment*⟩ ::= ⟨*opt white space*⟩ '=' ⟨*opt white space*⟩ ⟨*int*⟩

⟨*data assignment*⟩ ::= ⟨*opt white space*⟩ '=' ⟨*opt white space*⟩ ⟨*data reference*⟩

⟨*boolean assignment*⟩ ::= ⟨*opt white space*⟩ '=' ⟨*opt white space*⟩ ⟨*boolean*⟩

⟨*range assignment*⟩ ::= ⟨*opt white space*⟩ '=' ⟨*opt white space*⟩ ⟨*range*⟩

⟨*list assignment*⟩ ::= ⟨*opt white space*⟩ '=' ⟨*opt white space*⟩ ⟨*list*⟩

⟨*assignment statement*⟩ ::= 'Filename' ⟨*string assignment*⟩
| 'Format' ⟨*format assignment*⟩
| 'Skip' ⟨*integer assignment*⟩
| 'Preamble' ⟨*string assignment*⟩
| 'ErrorMultiplier' ⟨*string assignment*⟩
| 'References' ⟨*data assignment*⟩
| 'Reference1' ⟨*data assignment*⟩
| 'Reference2' ⟨*data assignment*⟩
| 'Reference3' ⟨*data assignment*⟩
| 'Response1' ⟨*data assignment*⟩
| 'Response2' ⟨*data assignment*⟩
| 'Response3' ⟨*data assignment*⟩
| 'Temperatures' ⟨*data assignment*⟩
| 'Waveforms' ⟨*data assignment*⟩
| 'Waveform1' ⟨*data assignment*⟩
| 'Waveform2' ⟨*data assignment*⟩
| 'Waveform3' ⟨*data assignment*⟩
| 'Parents' ⟨*integer assignment*⟩
| 'Children' ⟨*integer assignment*⟩
| 'Mutations' ⟨*integer assignment*⟩

| 'Generations' ⟨*integer assignment*⟩
| 'RandomSeed' ⟨*integer assignment*⟩
| 'OutputLogFilename' ⟨*string assignment*⟩
| 'ErrorLogFilename' ⟨*string assignment*⟩
| 'OutputLogFormat' ⟨*string assignment*⟩
| 'DumpFilename' ⟨*string assignment*⟩
| 'Graphics' ⟨*boolean assignment*⟩
| 'GAME' ⟨*boolean assignment*⟩
| 'MakeAllValid' ⟨*boolean assignment*⟩
| 'Range' ⟨*range assignment*⟩
| 'StepSize' ⟨*integer assignment*⟩
| 'Values' ⟨*list assignment*⟩
| 'Start' ⟨*integer assignment*⟩

⟨*register*⟩ ::= 'Filename'
| 'Format'
| 'Skip'
| 'Preamble'
| 'ErrorMultiplier'
| 'References'
| 'Reference1'
| 'Reference2'
| 'Reference3'
| 'Response1'
| 'Response2'
| 'Response3'
| 'Temperatures'
| 'Waveforms'
| 'Waveform1'
| 'Waveform2'
| 'Waveform3'
| 'Parents'
| 'Children'
| 'Mutations'
| 'Generations'

| 'RandomSeed'
| 'OutputLogFilename'
| 'ErrorLogFilename'
| 'OutputLogFormat'
| 'DumpFilename'
| 'Graphics'
| 'GAME'
| 'MakeAllValid'
| 'Range'
| 'StepSize'
| 'Values'
| 'Start'

⟨*command expression*⟩ ::= 'Load' ⟨*req white space*⟩ ⟨*string*⟩
| 'Ditch' ⟨*req white space*⟩ ⟨*string*⟩
| 'Go'
| 'For' ⟨*req white space*⟩ ⟨*string*⟩
| 'Next'
| 'Array' ⟨*req white space*⟩ ⟨*string*⟩
| 'System' ⟨*req white space*⟩ ⟨*string*⟩

⟨*local assignments*⟩ ::= ⟨*assignment statement*⟩
| ⟨*assignment statement*⟩ ⟨*req white space*⟩ ⟨*local assignments*⟩

⟨*command statement*⟩ ::= ⟨*command expression*⟩
| ⟨*command expression*⟩ ⟨*req white space*⟩ ⟨*local assignments*⟩

⟨*statement*⟩ ::= ⟨*empty statement*⟩
| ⟨*comment statement*⟩
| ⟨*assignment statement*⟩
| ⟨*command statement*⟩

⟨*line*⟩ ::= ⟨*opt white space*⟩ ⟨*statement*⟩ ⟨*opt white space*⟩ ⟨*EOL char*⟩

⟨*file*⟩ ::= ⟨*line*⟩ ⟨*file*⟩ | .

⟨*array element*⟩ := ⟨*text*⟩ ':' ⟨*opt white space*⟩ ⟨*int*⟩ ⟨*opt white space*⟩

⟨*loop label*⟩ ::= ⟨*text*⟩

⟨*variable*⟩ ::= ⟨*loop label*⟩ | ⟨*array element*⟩

⟨*variable ref*⟩ ::= '{' ⟨*opt white space*⟩ ⟨*variable*⟩ ⟨*opt white space*⟩ '}'

The grammar describes how to build a complete script from simpler elements, but it does not describe the meaning of each element. Semantics are crucial to correct usage of the scripting language and the following sections briefly describe key elements of the language giving examples where appropriate.

B.2 Comments

Any line where the first ⟨*non white space char*⟩ is a hash symbol '#' is treated as a comment. Lines identified as comments are read by the parser and ignored. However, if the '#' is used within a line then it is treated as any other symbol.

Comments allow annotation to exist within the script. They also allow code to be temporarily disabled.

B.3 Variable substitution

It is often useful to refer to the current value of a variable, *e.g.* to index an array. Variable substitution allows this as well as simple substitution of the current value of a loop label or array element.

A variable appears as either a ⟨*loop label*⟩ or as an ⟨*array element*⟩ identifier surrounded by braces: '{' and '}'. For example, if the array view has an element with an index of 5, then the value of this element can be substituted at any point by the string '`{view:5}`'.

Variables are expanded as the line is processed. Therefore, an assignment with a variable in the argument takes a fixed value, even if the value of the variable subsequently changes.

Variable substitution can be nested, so that '`{view: {i} }`' returns the i^{th} element of array '`view`'.

B.4 Assignments

An assignment alters the value of a ⟨*register*⟩ to the supplied value. If the value contains ⟨*white space*⟩, then the value must be protected by placing double quote marks, '"', on either side.

There is a fixed set of registers each with particular meaning. Most are connected with a particular command, some of which must be set for the command to work. An explanation of these registers is presented with the related command. However, there are some registers that apply globally. These are defined in the following subsections.

OutputLogFilename Normal output consists of results from the GA fitting routines, including the `preamble`, and the output from the `echo` command. By default, all normal output is sent to `stdout`, which is normally displayed in the terminal. If this register is set, then all output is appended to the file given by this register.

ErrorLogFilename Any problems, for example due to an incorrect script or missing datafile, results in error messages. These error messages are normally sent to `stderr`, which is displayed in the terminal by default. If this register is set then all warning or error messages are appended to the file given by this register.

The value of registers can be assigned on the command line. For example including a command-line option of '`GAME=True`' would assign '`True`' to register '`GAME`'. These assignments allows the program to be run within a batch environment with different parameters for each run.

B.5 Aliases

Certain registers do not store data themselves but form a short-hand method of assigning the same data to many registers. This is useful when a single reference curve is used to fit all three data-channels.

The aliases are:

References Any assignment made to this alias is made to Reference1, Reference2 and Reference3.

Waveforms Any assignment made to this alias is made to Waveform1, Waveform2 and Waveform3.

Example:

The aliases assignment:

```
Waveform = data:1
```

is equivalent to:

```
Waveform1 = data:1
Waveform2 = data:1
Waveform3 = data:1
```

B.6 Data storage

Any data that has been loaded into memory is treated as a collection of many series. A series contains a sequential list of numbers and each series of a datafile is required to have the same number of data.

When data is loaded, a label is assigned by the load command. This label is used when identifying the data to a ⟨*register*⟩. In addition, the specific series and range can be specified, although only the label is required. If no series is specified then the first series is used. The default range is the whole series.

Example:

If data has been loaded, labelled by '`wave`', then all of the first series can be referred to by:

```
wave
```

To refer to the third series, the following construction can be used:

```
wave:3
```

To refer to data elements 10–35 of this series, the following construction is used:

```
wave:3[10..35]
```

To refer to the twenty fifth element through to the last element, the following construction is used:

```
wave:3[25..]
```

B.7 Commands

Commands take either zero or one argument. If an argument is required, then it must be the next word after the command. If the argument contains ⟨*white space*⟩ then it can be encased within double quotation marks.

It is possible to alter the value of registers for the duration of a command by placing the assignments after either the argument, if present, or after the command. Assignment lines can be thought of as altering global settings whilst assignments on the same line as a command are local to the command.

The following sections describe the various commands and the registers they inspect.

B.7.1 Load

This command attempts to load data from a storage medium into memory. It requires an argument which is the label with which the data can be referred to later. If the load fails, then an error message is generated and the script attempts to continue.

Example:

```
load reference_data filename="some file.dat" format=ASCII_Column
```

Registers:

The load command accepts three registers:

filename (*mandatory*) The filename register to decide which file to load. It should be the correct format for the computer's file system.

format (*mandatory*) The format that the data is stored in. The system understands three formats:

ASCII_column Data in this format is contained in columns separated by one or more non-numeric characters. The exact alignment of the columns does not matter and it is assumed that data exists for all columns in each row. Missing data generates a warning message and a zero is substituted.

ASCII_freeform Data in ASCII_freeform is read sequentially. Multiple numbers on the same line are treated as sequential numbers from the same series. A blank line separates successive series. All series are made up to the same total number elements with zeros replacing missing data.

LeCroy_waveform This is the raw output format from the LeCroy oscilloscopes. Data is segmented time series. Typically for the COMPASS-D configuration, there will be 100 series corresponding to the 100 segments recorded by a spectromer's channel.

skip (*optional*) For ASCII_column and ASCII_freeform this register informs the data parser to skip the first n lines of the datafile. This register is ignored if the format is LeCroy_waveform.

B.7.2 Ditch

Frees memory associated with a particular label. It requires an argument which is the data to free. It also relinquishes the label so future load commands may use this label.

This command is optional. If a load command uses an existing label, then a warning message is generated and the old data is ditched. At the end of the script all loaded data is ditched.

Example:

The following commands load some data and label it '`mydata`' then free resources associated with the dataset:

```
load mydata filename="some data.dat" format=ASCII_FreeForm

ditch mydata
```

Registers:

This command uses no registers.

B.7.3 Go

This command is used to start the GA. It takes no arguments but uses many registers that affect the optimisation process.

Registers:

This command uses 22 registers:

Preamble (*optional*) If this register is set, then the contents of the register is sent to the current normal output, *i.e.* either ⟨*OutputLogFilename*⟩ or `stdout`. The exact format of the preamble depends on the output format setting (see below).

ErrorMultiplier (*optional*) If set, the resulting uncertainty in temperature is multiplied by this factor.

Reference1, Reference2, Reference3 (*mandatory*) The three data series containing the reference signal for channels one, two and three respectively.

Temperatures (*mandatory*) Data containing a list of temperatures in electron volts. This corresponds to the ordinance axis in figure 4.7. It must have the same number of elements as Response1, Response2 and Response3.

Response1, Response2, Response3 (*mandatory*) Data containing the response of channels one, two and three respectively at the temperature given by the Temperature dataset at the same index. Must have the same number of elements as Temperature.

Waveform1, Waveform2, Waveform3 (*mandatory*) The observed data from channels one, two and three respectively. This is the data that will be analysed. The three waveforms must have the same number of data.

Parents (*mandatory*) The number of parents to use in the Genetic Algorithm.

Children (*mandatory*) The number of children to use in the Genetic Algorithm. This must be an even number

Mutations (*mandatory*) The number of mutations to undertake after breeding is completed.

Generations (*mandatory*) The maximum number of generations to undertake.

RandomSeed (*mandatory*) An ⟨*int*⟩ with which the random number generator is seeded. This allows for identical runs, if required.

OutputLogFormat (*optional*) The output format for the analysis. There are six options:

verbose This is the default output. It gives detailed information about the optimisation. The output has the following format:

`Preamble:` ⟨*text*⟩
`Results (after "Go" on line` ⟨*int*⟩`) are as follows`
`Temperature =` ⟨*float*⟩ `+/-` ⟨*float*⟩
`Line 1 Area =` ⟨*float*⟩ `+/-` ⟨*float*⟩
`DC Level =` ⟨*float*⟩
`Time base offset =` ⟨*float*⟩
`Residuals have mean` ⟨*float*⟩ `and standard deviation` ⟨*float*⟩
`Line 2 Area =` ⟨*float*⟩ `+/-` ⟨*float*⟩
`DC Level =` ⟨*float*⟩
`Time base offset =` ⟨*float*⟩
`Residuals have mean` ⟨*float*⟩ `and standard deviation` ⟨*float*⟩
`Line 3 Area =` ⟨*float*⟩ `+/-` ⟨*float*⟩
`DC Level =` ⟨*float*⟩
`Time base offset =` ⟨*float*⟩
`Residuals have mean` ⟨*float*⟩ `and standard deviation` ⟨*float*⟩

temperature This output option restricts the recorded information to just a single line containing the preamble (if any), the recovered temperature and the uncertainty in the temperature. The format is:

⟨*text*⟩⟨*float*⟩⟨*float*⟩

areas This output format stores the results from fitting the reference signals to the data. The result of fitting the three areas to the temperature response curves is still calculated but is ignored. The output format is three lines containing the preamble, the area and the uncertainty in the areas, *i.e.*

⟨*text*⟩⟨*float*⟩⟨*float*⟩
⟨*text*⟩⟨*float*⟩⟨*float*⟩
⟨*text*⟩⟨*float*⟩⟨*float*⟩

terse This output, like verbose, contains all generated information but condenses the information to make further analysis easier. The preamble, if present, is displayed on its own line, followed by the temperature and uncertainty in temperature. In the following three lines the results for each channel are stored. The area and uncertainty in the area is followed by the DC level and (channel) offset. The last two entries in each line are the mean and standard deviation of the residuals. The format is:

⟨*text*⟩
⟨*float*⟩⟨*float*⟩
⟨*float*⟩⟨*float*⟩⟨*float*⟩⟨*int*⟩⟨*float*⟩⟨*float*⟩
⟨*float*⟩⟨*float*⟩⟨*float*⟩⟨*int*⟩⟨*float*⟩⟨*float*⟩
⟨*float*⟩⟨*float*⟩⟨*float*⟩⟨*int*⟩⟨*float*⟩⟨*float*⟩

TempChiS This option is just an extension of the Temperature option. It includes a final field with the value of χ^2 for the three channels. The format is:

⟨*text*⟩⟨*float*⟩⟨*float*⟩

amplitudes This option outputs the amplitude for each channel rather than the integrated areas. There is no direct physical meaning to the amplitude, so this option is mainly for debugging purposes. The format is:

⟨*float*⟩
⟨*float*⟩
⟨*float*⟩

DumpFilename (*optional*) This option stores the raw result of the GA fitting procedure. This data is stored in a file given by the register. The data is formatted as ten columns: the first column is an index and it is followed by nine columns consisting of the raw data, the GA's best fit of the reference signal and the residues for each of the three channels.

Graphics (*optional*) If this register is set to true then a window is opened and the data (in black) and the results of the best-fit parameters (in blue) are displayed for each channel. The graphs are updated at the end of each generation. This provides an easy, visual way of checking on the GA's progress.

GAME (*optional*) If this register is set to true then the GA monitoring window is

opened. This maintains information about basic GA statistics as shown in figure 2.15.

MakeAllValid (*optional*) If this register is set to true then all data is stored. Normally, if something goes wrong (*e.g.* the matrix inversion required to obtain the uncertainty in temperature is singular) then a warning message is generated and that datapoint is ignored. This option overrules that behaviour and all datapoints are stored.

B.7.4 For

This command establishes a point in the program to which a loop will return. The command takes an argument which is a variable. The variable changes value in each iteration of the loop as described by the Range register.

Example:

To load four files from files 'data1.dat', 'data2.dat', 'data3.dat' and 'data4.dat' which we wish to label 'first', 'second', 'third' and 'fourth', the following code may be used.

```
Array labels start=1 values=first,second,third,fourth

Format=ASCII_Column

For i range=1..4
  Load {label:{i}} filename=data{i}.dat
Next
```

Registers:

Range (*mandatory*) This register describes the start and end values for the loop. If the argument of the assignment is a ⟨*numerical range*⟩ then the loop will be between the first number and the second number. If the argument of the register is an array, then the loop variable will be assigned to the elements of the array in turn.

StepSize (*optional*) For a loop where the range register is set to a ⟨*numerical range*⟩ this register describes what number to increment the loop variable by after each loop.

B.7.5 Next

Loops back to the most resent 'For' statement. This command takes no argument or registers.

Example:

```
For i range=1..3
   Ditch data{i}
Next
```

Registers:

This command uses no registers.

B.7.6 Array

Establishes an array of values. The command takes an argument which is the label used to reference this array.

Example:

```
Array labels start=1 values=first,second,third,fourth

Format=ASCII_Column

For i range=1..4
  Load {label:{i}} filename=data{i}.dat
Next
```

Registers:

Values (*mandatory*) This register contains a comma separated list of entries. If any of the list contains ⟨*white space*⟩ then the whole list must be enclosed within double quotes ('"'s).

Start (*optional*) The first element of the array is usually assigned to index 1. If this register is set then the first element will be the numerical value of this register.

B.7.7 System

This command run a external program. It requires an argument which is the command to run.

Example:

If an external program creates data, for example for a Monte Carlo simulation, then the following code section could be used.

```
For i range=1..1000
  System buildData

  For j range=1..3
    Load data{j} filename=file{j}.dat
  Next

  Go

  For j range=1..3
    Ditch data{j}
  Next
Next
```

Registers:

This command uses no registers.

B.7.8 Echo

This command places some arbitrary text in the normal output. It requires an argument: the text to be displayed.

Example:

This command can be used to store the current value of a particular variable:

```
echo "variable i: {i}   j th label: {label:{j}}"
```

Registers:

This command uses no registers directly, but the OutputLogFilename is used indirectly.

B.8 Worked example

The following script analyses the data from a single plasma shot although many shots can be analysed by appending the definition of the allshots array. The output is in a series of files each containing lines with the height above torus mid-plane followed by a temperature and uncertainty. The code contains numerous comments that, along with the above description of the scripting language, should fully explain how this script works.

File: GA/Thomson/datdata/persegment.ctrl

```
1  # Control file for second set of data.
2  # See test.ctrl for explainitory information.
3
4  # Global definitions.
5  Parents        = 40
6  Children       = 100
7  Mutations      = 110
8  Generations    =  400
9  RandomSeed     =  0
10 ErrorMultiplier = 3.8
11 ErrorLogFilename = persegment.err
12
13 #  Choice of Temperatures, Areas, Verbose or Terse.
14 OutputLogFormat = Temperatures
15 # Graphics = TRUE
16 # GAME = True
17
18
19 # Define an array which maps spectrometer to view. First element in
20 # view is index 1.
21 Array view start=1 values=9,3,5,,7,12,8,14,10
22
23 # We also want an array of what height each view looks at (in mm).
24 Array height start=1 values=171,147,123,99,75,51,27,3,-21,-45,-69,-93,-117,-141,-165,-189
25
26 # Also an array of all the spectrometer. This is for a For loop: starting value
27 # does not matter.
28 Array SpectrometersUsed values=2,3,5,7,1,9,6,8
29
30 Array allshots values=26933
31
32
33 # Specify the default format (for the rest of the data files)
34 format   = ASCII_FreeForm
35
36
```

```
37 # Load in all the reference spectra. Skip over initial blank data.
38 load ref filename=tsd7_ref.dat skip=127
39
40 for shot range=allshots
41
42    # Main loop for calculating each segment. This refers to segments so the
43    # first one is '1' and the last one is '100'.
44    for segment range=90..100
45
46       OutputLogFilename = s{shot}seg{segment}.temp.dat
47
48       for spec range=SpectrometersUsed
49
50          # Load in spectrometer specific data.
51          load temp filename=s{spec}v{view:{spec}}_r9_040298.cal format=ASCII_Column skip=10
52
53          # Define our temperature calibration curves.
54          temperatures = temp:1
55          response1    = temp:2
56          response2    = temp:3
57          response3    = temp:4
58
59          # Break up the series to form the reference data.
60          reference1 = ref:{spec}[0..251]
61          reference2 = ref:{spec}[252..503]
62          reference3 = ref:{spec}[504..755]
63
64  # Output the height (above torus mid-plane) for this spectrometer.
65          Preamble = {height:{view:{spec}}}
66
67  # Load the data. Deal with ith channel.
68          for i range=1..3
69             load data{i} filename=tsr{shot}.0{spec}{i} format=LeCroy_Waveform
70             waveform{i} = data{i}:{segment}
71          next
72
73          # Save raw data.
74          dumpfilename = s{shot}seg{segment}spec{spec}.dump.dat
75
76  # Run Genetic Algorithm.
77          go
78
79          ditch temp
80          for i range=1..3
81             ditch data{i}
82          next
83
84       next
85    next
```

```
86 next
87
88 ditch ref
```

References

A. Agapie. Genetic Algorithms: Minimum conditions for convergence. *Lecture Notes in Computer Science*, 1363:183–193, 1998.

K. Aoki, K. Matsumoto, K. Hoashi, and K. Hashimoto. A study of Bayesian clustering of a document set based on GA. *Lecture Notes in Computer Science*, 1585:260–267, 1999.

C. J. Barth, M. N. A. Beurskens, C. C. Chu, A. J. H. Donné, N. J. L. Cardozo, J. Herranz, H. J. van der Meiden, and F. J. . Pijper. A high resolution multiposition Thomson scattering system for the Rijnhuizen Tokamak Project. *Review of Scientific Instruments*, 68(9):3380–3392, September 1997.

A. A. Berezin. An unexpected result in classical electrostatics. *Nature*, 315(6015):104, 1985.

N. Bethera and V. Nanjundiah. *trans* gene regulation in adaptive evolution: a genetic algorithm model. *Journal of Theoretical Biology*, 188:153–162, 1997.

M. N. A. Beurskens, C. J. Barth, N. J. L. Cardozo, and H. J. van der Meiden. A high spatial resolution double-pulse Thomson scattering diagnostic; description, assessment of accuracy and examples of applications. *Plasma Physics and Controlled Fusion*, 41:1321–1348, 1999.

M. N. A. Beurskens, C. J. Barth, C. C. Chu, A. J. H. Donné, J. A. Herranz, N. J. L. Cardozo, H. J. van der Meiden, and F. J. Pijper. Double pulse Thomson scattering system at rtp. *Review of Scientific Instruments*, 68(2):721–724, January 1997.

D. Bhandari, N. R. Pal, and S. K. Pal. Directed mutation in Genetic Algorithms. *Information Sciences*, 79:251–270, 1994.

H. Bindslev. Methods for optimizing and assessing diagnostic capability, demonstrated for collective Thomson scattering. *Review of Scientific Instruments*, 70(1):1093–1099, 1999. Part 2.

M. A. Blokh and N. F. Larionova. *Soviet Journal of Plasma Physics*, 7:31, 1981.

F. M. A. Box. Non-Maxwellian Thomson scattering spectra at the RTP tokamak as a new diagnostic tool. *Nuclear Fusion*, 39(9):1193–1203, 1999.

N. Bretz, D. Dimock, V. Foote, D. Johnson, D. Long, and E. Tolnas. Multichannel Thomson scattering apparatus. *Applied Optics*, 17(2):192–202, January 1978.

B. Coppi, F. Pegoraro, R. Pozzoli, and G. Rewoldt. Slide-away distributions and relevant collective modes in high-temperature plasmas. *Nuclear Fusion*, 16(2):309–328, 1976.

R. C. Corrêa, A. Ferreira, and P. Rebreyend. Scheduling multiprocessor tasks with Genetic Algorithms. *IEEE Transactions on parallel and distributed systems*, 10(8): 825–837, August 1999.

A. E. Eiben, E. H. L. Aarts, and K. M. van Hee. Global convergence of Genetic Algorithms: a Markov chain analysis. *Lecture Notes in Computer Science*, 496, 1991.

D. B. Fogel and A. Ghozeil. Schema processing, proportial selection, and the misallocation of trials in genetic algorithms. *Information Sciences*, 122:93–119, 2000.

A. S. Fukunaga. Restart scheduling for Genetic Algorithms. *Lecture Notes in Computer Science*, 1498:357–366, 1998.

M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-Completeness.* W.H. Freeman and Co, 1979.

M. Gerrits and P. Hogeweg. Redundant coding of a NP-complete problem allows effective Genetic Algorithm search. *Lecture Notes in Computer Science*, 496:70–74, 1991.

D. E. Goldberg. *Genetic algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

J. Gottlieb and N. Voss. Representations, fitness functions and genetic operators for the satisfiability problem. *Lecture Notes in Computer Science*, 1363:55–68, 1998.

J. Hesser and R. Männer. Towards an optimal mutation probability for Genetic Algorithms. *Lecture Notes in computer science*, 496:23–32, 1991.

P. Hoel and A. Craig. *An introduction to Mathematical statistics.* MacMillan, 1978.

I. H. Hutchinson. *Principles of Plasma Diagnostics.* Cambridge University Press, 1987.

M. Kendall and A. Stuart. *The advanced theory of statistics vol 1.* Haffner Publ Co., NY, 1963.

S. Kirkpatrick, C. D. Gellat, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

H. D. Kluiver, C. J. Barth, and A. J. H. Donné. Current driven turbulence and microturbulent spectra in the TORTUR tokamak. *Plasma Physics and Controlled Fusion*, 30(6):699–719, 1988.

A. M. Mood and A. F. Graybrel. *Introduction to the Theory of Statistics.* McGraw-Hill, 1974.

O. Naito, H. Yoshida, and T. Matoba. Analytic formula for fully relativistic Thomson scattering spectrum. *Physics of Fluids B*, 5(11):4256–4257, November 1993.

A. Neubauer. Adaptive non-uniform mutation for Genetic Algorithms. *Lecture Notes in Computer Science*, 1226:24–34, 1997.

R. E. Pechacek and A. W. Trivelpiece. Electromagnetic wave scattering from a high-temperature plasma. *The Physics of Fluids*, 10(8):1688–1696, August 1967.

L. Pieroni and S. E. Segre. Observation of non-Maxwellian electron distribution functions in the Alcator device by means of Thomson scattering and their interpretation. *Physical Review Letter*, 34(15):928–, April 1975.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 2nd edition, 1996.

M. Rattray and J. Shapiro. The dynamics of a Genetic Algorithm for a simple learning problem. *Journal of Physics A: Mathematical and general*, 29:7451–7473, 1996.

L. M. Schmitt, C. L. Nehaniv, and R. H. Fujii. Linear analysis of genetic algorithms. *Theoretical Computer Science*, 200:101–134, 1998.

A. C. Selden. Simple analytic form of the relativistic Thomson scattering spectrum. *Physics Letters*, 79A(5,6):405–406, 1980.

M. Sriniva and L. M. Patnaik. Adaptive probability of crossover and mutation in Genetic Algorithms. *IEEE transactions on systems, man and cybernetics*, 24(4):656–666, April 1994.

N. L. J. Ulder, E. H. L. Aarts, H. Bandelt, P. J. M. van Laarhoven, and E. Pesch. Genetic local search algorithms for the travelling salesman problem. *Lecture Notes in Computer Science*, 496:109–116, 1991.

A. C. A. P. van Lammeren, C. J. Barth, Q. C. van Est, and F. C. Schüller. Non-Maxwellian electron velocity distributions observed with Thomson scattering in the TORTUT tokamak. *Nuclear Fusion*, 32(4):655–665, 1992.

J. Watson, C. Ross, V. Eisele, J. Denton, J. Bins, C. Guerra, D. Whitley, and A. Howe. The traveling salesrep problem, edge assembly crossover, and 2-opt. *Lecture notes in Computer Science*, 1498:823–832, 1998.

L. Wille and J. Vennik. Computational complexity of the ground-state determination of atomic clusters. *Journal of Physics A: Mathematical and General*, 18(8):L419–L422, 1985a.

L. T. Wille and J. Vennik. Electrostatic energy minimisation by simulated annealing. *Journal of Physics A: Mathematical and General*, 18(17):L1113–L1117, 1985b.

D. H. Wolpert and W. G. Macready. No Free Lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, April 1997.

V. A. Zhuravlev and G. D. Petrov. Scattering of radiation by finite volumes of relativistic plasma streams. *Soviet Journal of Plasma Physics*, 5(1):3–5, January 1979.

Colophon

This thesis was typeset using the LaTeX system, occasionally delving into TeX as necessary. The body font is Computer Modern at 11 point on 13.6 point.

Program code extracts are displayed with the relevant line number typeset in italics and placed in the margin. When the file has been taken from the CDROM, the relevant filename is displayed preceding the code. Wherever possible, the code was taken directly from the source file to reduce the possibility of introducing errors.

The graphs were generated using the venerable 'gnuplot' program, except for the contour plots (2.17 and 2.18) which were generated using IDL. The diagrams were generated using xfig, except for figures 2.1, 2.2 and 2.3.

Figures 2.1, 2.2 and 2.3 were taken from the 'Glossary of Genetic Terms' web-pages (`http://www.nhgri.nih.gov/DIR/VIP/Glossary/`), part of the Division of Intermural Research site, and are included with their kind permission.