FOURIER TRANSFORM ION CYCLOTRON RESONANCE MASS SPECTROMETRY FOR PETROLEOMICS

Jennifer M. Hauschild

Oriel College

A thesis submitted for the degree of Doctor of Philosophy

Trinity Term 2011

Physical and Theoretical Chemistry Laboratory

University of Oxford

Fourier Transform Ion Cyclotron Resonance Mass Spectrometry for Petroleomics

Abstract

The past two decades have witnessed tremendous advances in the field of high accuracy, high mass resolution data acquisition of complex samples such as crude oils and the human proteome. With the development of Fourier transform ion cyclotron resonance mass spectrometry, the rapidly growing field of petroleomics has emerged, whose goal is to process and analyse the large volumes of complex and often poorly understood data on crude oils generated by mass spectrometry.

As global oil resources deplete, oil companies are increasingly moving towards the extraction and refining of the still plentiful reserves of heavy, carbon rich and highly contaminated crude oil. It is essential that the oil industry gather the maximum possible amount of information about the crude oil prior to setting up the drilling infrastructure, in order to reduce processing costs.

This project describes how machine learning can be used as a novel way to extract critical information from complex mass spectra which will aid in the processing of crude oils.

The thesis discusses the experimental methods involved in acquiring high accuracy mass spectral data for a large and key industry-standard set of crude oil samples. These data are subsequently analysed to identify possible links between the raw mass spectra and certain physical properties of the oils, such as pour point and sulphur content.

Methods including artificial neural networks and self organising maps are described and the use of spectral clustering and pattern recognition to classify crude oils is investigated. The main focus of the research, the creation of an original simulated annealing genetic algorithm hybrid technique (SAGA), is discussed in detail and the successes of modelling a number of different datasets using all described methods are outlined.

Despite the complexity of the underlying mass spectrometry data, which reflects the considerable chemical diversity of the samples themselves, the results show that physical properties can be modelled with varying degrees of success. When modelling pour point temperatures, the artificial neural network achieved an average prediction error of less than 10% while SAGA predicted the same values with an average accuracy of more than 85%. It did not prove possible to model any of the other properties with such statistical significance; however improvements to feature extraction and pre-processing of the spectral data as well as enhancement of the modelling techniques should yield more consistent and statistically reliable results.

These should in due course lead to a comprehensive model which the oil industry can use to process crude oil data using rapid and cost effective analytical methods.

Table of Contents

CHAPTER 1 – Introduction

CHAPTER 1 briefly discusses the objectives of the project and introduces the techniques and instruments involved.

1.1 Background

Our natural resources of light, fluid and barely contaminated crude oils are steadily depleting. Oil companies are facing the reality of having to focus attention on harnessing petroleum from sources which are harder to reach or from which crude oils are more expensive to extract, such as oil sands and shale found in Canada and Southern America. Light, alkane-rich and sulphur-free crude oils, such as those most commonly found in the Arab Emirates, are easy to extract from the ground and yield large quantities of high-quality gasoline. However, such fuel sources are also commonly found in politically unstable territories and are depleting at a steadily increasing rate.

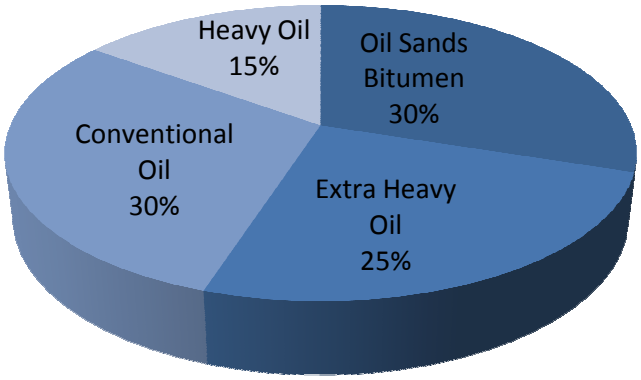


Figure 1.1: Global oil reserves

It is essential for oil companies to locate new resources and start setting up infrastructure in order to harness these new resources, in preparation for the inevitable running dry of existing reservoirs.

There are many known sources of crude oil in politically stable territories. However, crude oil found in areas such as Venezuela or the Canadian oil sands is known to be very

heavy and strongly contaminated with aromatic hydrocarbons, and therefore currently of little to no economic value.

Heavy crude oils are mainly characterised by having high contents of asphaltenes, compounds that lead to deposits which clog up pipelines and can cause costly repairs (Mullins *et al*., 2007). In addition, they contain large amounts of sulphur contaminants which lead to high emissions of the pollutant sulphur dioxide during the refining process. While it is possible for refineries to process such oils, it is not yet financially viable. Problems with processing extra-heavy crude oils begin at the stage of extraction and transportation, where high viscosities lead to low flow rates and thus decreased production rates. Higher levels of nitrogen contaminants lead to higher levels of coke formation which further decrease pumping speeds. Once the oil has been successfully transported to one of the few refineries which are equipped for handling the carbon rich heavy crudes, complex chemical procedures are required to remove excess carbon, or introduce additional hydrogen to the molecules, in order to render the crude oil usable. Furthermore, heavy crude oils have to undergo a process known as fluid catalytic cracking (FCC), which breaks open long-chain molecules characteristic of high-boiling hydrocarbon fractions to create shorter ones.

It is for these reasons that oil companies are reluctant to begin harnessing these easily accessible and plentiful reserves.

By gaining a better understanding of the chemical structures of crude oils and how their chemical composition affects their physical properties, a discipline known as petroleomics, it is thought that the extraction and refining process of these difficult,

undesirable crude oils could be better understood and optimised, making the process easier and cheaper, therefore ensuring future fuel security.

1.2 Motivation and Objectives

The objective of this project in petroleomics is to use intelligent data analysis in the form of machine learning to help model the physical properties of crude oils, using simple, fast and cheap data acquisition. Successfully predicting physical properties of a crude oil will enable oil companies to gain a better understanding of oil wells and help optimise the processes involved in extracting and refining the crude. Much of the infrastructure surrounding an oil well must be carefully matched to the type of crude oil that is being drilled for, and the more information one can gain about an oil prior to setting up the well, the more economically viable the process becomes.

As an example, in 2001 the world's largest commercial oil field, Ghawar, which is located in Saudi Arabia, produced between 10,000 and 40,000 barrels of light crude oil a day at the cost of just US$2 per barrel. Oil production of the heavier, less accessible crude oil available within the Americas on the other hand cost approximately US$10 a barrel, due to the need for far more advanced extraction methods (Campbell and Zagar, 2001). Owing to the increased convenience of local oil production, local production nevertheless accounted for 51% of the total oil consumption of the United States in 2010 (U.S. Energy Information Administration).

It is widely understood that increased knowledge of the physical properties of the crude oil can improve reservoir management, which in turn will help optimise production thereby maximising economic value.

Furthermore, a large number of oil wells are abandoned early due to the high cost of advanced recovery of crude oil from nearly empty reservoirs. Processes such as thermal, miscible and chemical flooding of wells in order to extract the remaining 5-10% of reserves from an otherwise depleted well are costly, leading to a total production cost of up to US$15 per barrel in 2001. Despite its high cost, enhanced oil recovery can extend a well's production by up to 25 years (US Department of Energy) and it is therefore highly advantageous to optimise the processes involved. Advanced modelling of the crude oil reserves of such wells can help tailor the extraction process to a specific well, thereby making the process more economically viable.

Another use for fingerprinting crude oils, which is far removed from the effort of maximising financial gain by the oil industry, is that of identifying sources of oil spills. With vast numbers of tankers travelling the oceans at any one time, oil leaks are inevitable. While the culprits of large scale catastrophes such as the Exxon Valdez oil spill in 1989 or BP's Deepwater Horizon oil spill in 2010 can easily be identified, minor oil contaminations are far more difficult to trace back to any one source. Fingerprinting of oil samples gathered from a contaminated beach or water source can match them to the crude oil cargo on a particular tanker or in a particular well using petroleomics (Wang and Fingas, 1999). It stands to reason that in-depth analysis of the crude oils involved in an oil spill can further aid in the fast and efficient clean up of the crude oil, according to the National Oceanic and Atmospheric Administration (NOAA) and United States Coast Guard Marine Safety Laboratory (USGC MSL).

The applications of the methods to be discussed are not limited to petroleomics; there are many areas from pharmaceuticals to airport security in which it may be necessary

to predict the physical properties of substances using chemical data. This project aims to create a model that links high accuracy mass spectral data with a list of physical properties of a set of 76 industry-standard crude oil samples (American Petroleum Institute). The finished model should ideally be capable of predicting physical behaviour of an unknown oil using only the data extracted from its mass spectrum.

Mass spectral data is chosen for this research as it is information rich, yet very fast and simple to acquire. Conventional crude oil assays can be costly and time consuming and may not always be favourable compared to the cheaper, faster alternative of modelling properties using mass spectral data.

There are a number of other analytical methods that have been used to analyse crude oils, including Fourier transform infrared spectroscopy (FTIR), UV-VIS spectroscopy and gas chromatography (GC), which all fail by comparison to FT-ICR MS.

FTIR, while equally fast and allowing high levels of accuracy, requires database matching in order to identify chemical compounds of samples. As petroleomics is a field that commonly deals with previously unseen samples, or samples that have changed significantly over time due to biodegradation (Kim *et al.*, 2005), it is not practicable to use database matching as a means of sample identification. In addition, FTIR is not well suited for the analysis of high-concentration samples due to the limitations of Beer's law. FTIR works well when analysing dilute samples with low absorbance, however Beer's law begins to break down with increasing sample concentration, which is generally the case with crude oil samples in petroleomics.

GC is a very powerful technique for identifying unknown samples and breaking down samples into their constituents, and is widely used for drugs and hazardous chemicals analyses in both forensics laboratories and in situations where fast in-situ chemical analysis is required to ensure national security, such as airports. However, the main limitation of GC is that the analyte must be volatile. If the molecular weight of the compound to be analysed exceeds 500Da, it becomes immovable in the GC column. Large fractions of heavy oils are made up of materials that are GC immovable (i.e. exceed 500Da) and can therefore not be analysed using GC. GC is also better suited to analysing the non-polar fractions of the crude oil, as they move faster through the column. This research is primarily interested in the polar fractions (nitrogen compounds), as it is those that are of greatest importance to the refining process (Hughey *et al.*, 2002b).

Furthermore, while both FTIR and GC can provide high accuracy data, FT-ICR MS with resolutions of up to 450,000 $m/\Delta m_{50\%}$ allows for the most unambiguous sample identification of any of these methods.

1.3 General Overview of Techniques

1.3.1 Chemometrics

Chemometrics is the term used to describe the mathematical discipline of analysing chemical data in order to describe or model it (Wold, 1995). Its applications range from initial data collection (experimental design, calibration and signal processing) to the eventual analysis of that data (statistical analysis, pattern recognition and modelling) and it is commonly associated with computational methods such as principal

component analysis (PCA) or partial least squares (PLS) clustering and artificial intelligence techniques such as machine learning.

With a large number of improvements being made in terms of experimental techniques and instrumentation, there is a constant need for faster and better methods of data analysis. Machine learning (ML) is now extensively used for the analysis of chemical data, especially in the field of cancer diagnosis, where models help to distinguish biological samples taken from both ill and healthy patients (Zhou *et al.*, 2004; Cruz and Wishart, 2006). However, numerous scientific problems exist in which the current conventional methods of analysis are ineffective or suboptimal. For example, in petroleomics an abundance of high accuracy mass spectral data is generated, and although the data is information-rich, its analysis is challenging because of the complexity of the underlying spectra (Marshall and Rodgers, 2008). Machine learning can be used as a novel and efficient way to extract critical processing information from mass spectral petroleum data, an application of considerable interest to the petrochemical industry.

1.3.2 Petroleomics

Petroleomics, a field largely dominated by the Marshall group at Florida State University, is the chemical analysis of crude oils in order to determine physical properties and predict economic value of petroleum feedstock (Marshall and Rodgers, 2004).

Petroleomics has a host of different applications which are all of financial interest to the oil industry. As an example, vast amounts of crude oil are stored in reservoirs below the

sea floor, underneath 5,000 to 10,000 feet of water. Drilling for oil under such conditions is incredibly costly, and it is essential to know precisely what type of crude oil is being dealt with in order to ensure that the correct drilling infrastructure is set up. Should the crude oil that is being drilled for contain high levels of asphaltenes, this must be noted prior to extraction, as deposits and subsequent blocking of the pipes is not easily fixed that far below the surface of the ocean.

Furthermore, offshore platforms containing the rigging must be prepared to process the particular oil that is being drilled for.

Petroleomics also encompasses the discipline of taking small samples of oil from newly discovered reserves and processing them in order to determine the relative worth of the crude oil and to inform the decision as to whether or not production of the reservoir should be pursued.

In addition, once the crude oil has been extracted, the study of petroleomics can further help optimise the refining process. By identifying the precise chemical fingerprint of the crude oil, oil companies can identify the end-products it best lends itself to and tailor the processing to make it as economical as possible.

1.3.3 Mass Spectrometry

Mass spectrometry is an analytical method in which samples are vapourised and the vapours ionised. By applying an electromagnetic field, ions are separated according to their individual mass-to-charge ratios. A detector records the individual abundances of each of the ions, and mass-to-charge ratios, provided they are known with sufficient

accuracy, can then be used to identify the molecular fragments generated by decomposition of the sample.

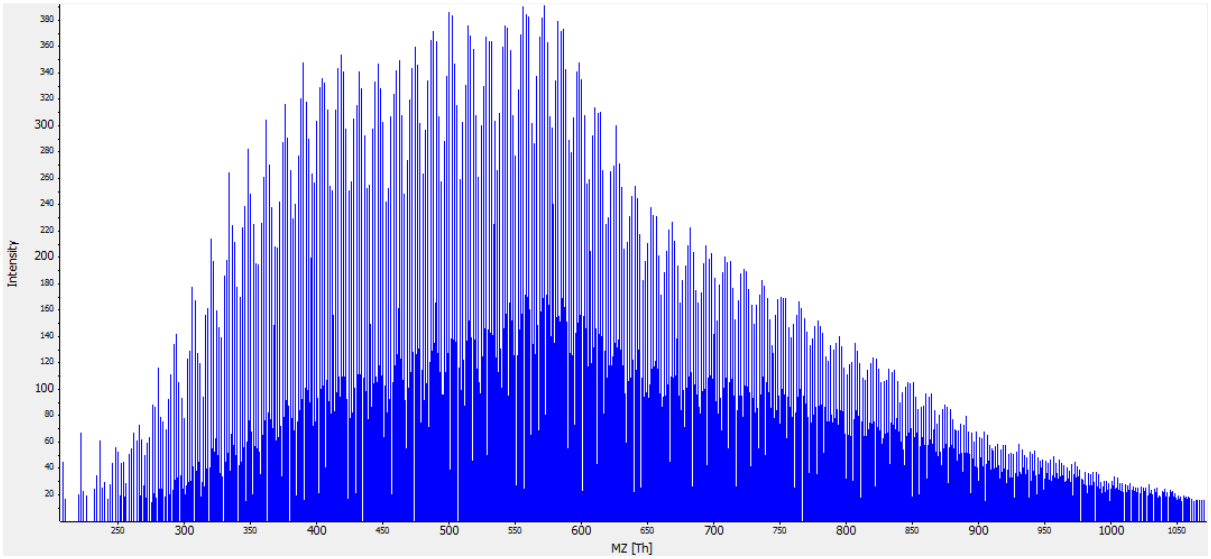


Figure 1.2: Mass spectrum of a 1mg crude oil sample diluted in 0.5ml toluene and 0.5ml methanol, recorded in positive ion ESI mode on a 9.4T FT-ICR MS instrument. (Units are in Thomson [Th] or Da/e ≡ m/z)

Molecular masses as determined by mass spectrometry can be accurate to within 5ppm, depending on the specific instrument, which is generally considered sufficient for positive identification of chemical compounds. An example quoted by de Hoffmann and Stroobant in *Mass Spectrometry: Principles and Applications Second Edition* (Wiley, 2002) lists the 36 possible combinations of $C_{6\text{-}15}$, $H_{0\text{-}24}$, $N_{0\text{-}4}$ and $O_{0\text{-}4}$ which could give rise to a peak at 180m/z. The smallest gap between the exact masses of any of these compositions is 0.0012Da, proving that an accuracy of 6ppm is sufficient to correctly identify the exact composition of the peak.

Figure 1.2 shows a typical mass spectrum of a crude oil. The spectrum clearly shows the complexity of the data and therefore the importance of high resolution analytical methods. A crude oil spectrum will contain in the region of 36,000 peaks of which on

average 5,000 to 30,000 can be positively identified by their chemical formula. The higher the mass resolution of the instrument, the more peaks can be resolved and the more information one can thus gather on the sample.

1.3.3.1 Fourier Transform Ion Cyclotron Resonance Mass Spectrometry

FT-ICR MS is a high accuracy, high-resolution mass analysis method for the identification of chemical compounds, first developed by Comirsarov and Marshall (Comisarov and Marshall, 1974). As with conventional mass spectrometry, samples are first vapourised and the molecules ionised before being introduced into the accelerator or ion trap. The ion trap, known as a Penning trap (Dehmelt, 1990), consists of an electric field that oscillates perpendicularly to a fixed magnetic field. The ions are first caused to oscillate with a frequency dependent on their individual m/z by the static magnetic field. The application of the oscillating electric field further excites the oscillating ions, causing their cyclotron radii to increase to within detectable limits. Once the ions have reached their maximum orbits, the RF excitation source is turned off, allowing the cyclotron motion of the ion packets to gradually slow down as a function of time, resulting in a decaying signal in the time domain. In order for this to be turned into a useful frequency domain signal or spectrum, Fourier transformation is applied. The resultant frequency domain spectrum is given by equation 1.1.

$$E(\upsilon) = E_0 \frac{\sin(2\pi\upsilon_c T_{ex})}{2\pi\upsilon_c} \quad (1.1)$$

where $E(\upsilon)$ is the frequency domain signal, E_0 is the initial signal strength, υ_c is the cyclotron frequency and T_{ex} is the excitation time in seconds (Marshall *et al.*, 1998).

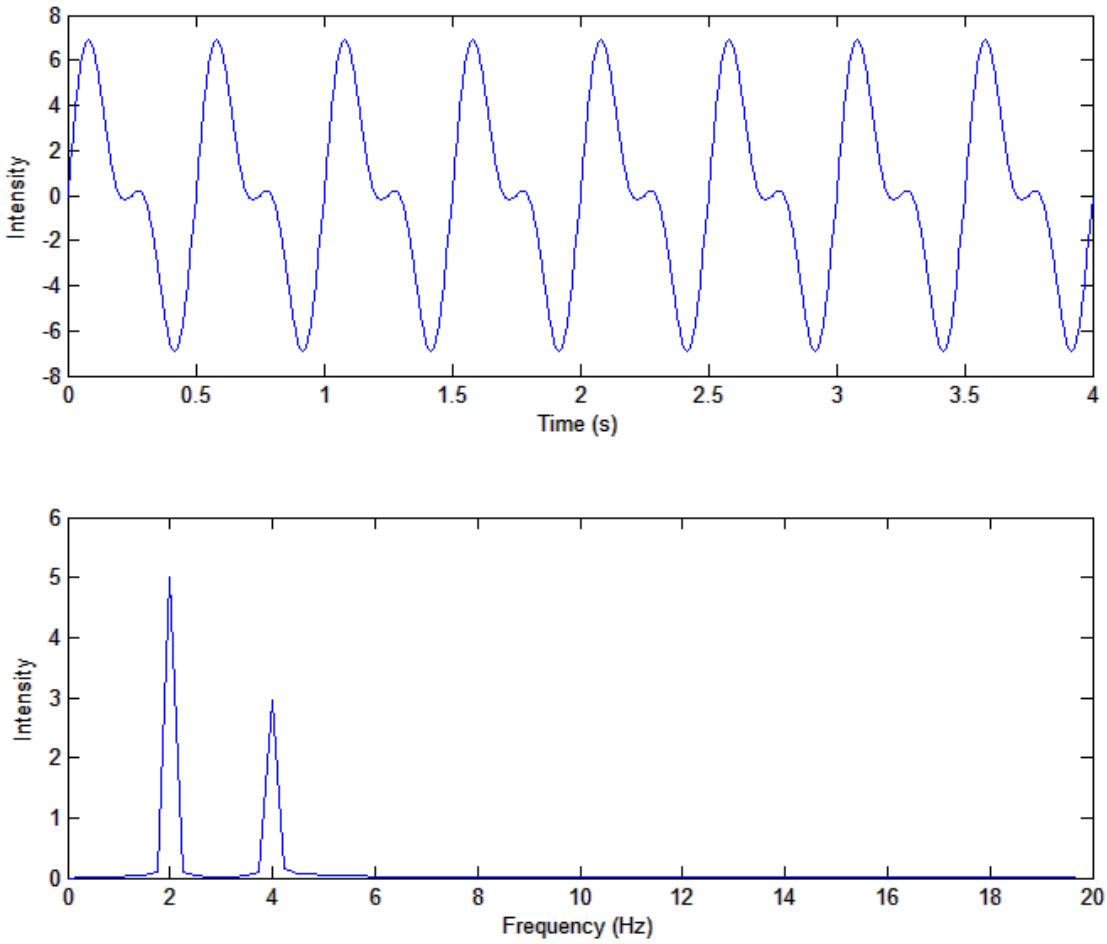


Figure 1.3: The time domain signal (top) and the Fourier transformed frequency domain spectrum (bottom) for an arbitrary signal.

The cyclotron frequency of the ions is directly proportional to the magnetic field and peak resolution of the resultant spectrum improves with increasing frequency. This means that larger superconducting magnets lead to higher accuracy spectra. Conventional FT-ICR MS instruments are fitted with 7T or 9.4T magnets, however researchers at the National High Magnetic Field Laboratory (NHMFL) at Florida State University (FSU) in Tallahassee are working on creating a 21T ICR instrument and currently possess a world-record 14.5T FT-ICR MS research instrument (www.magnet.fsu.edu).

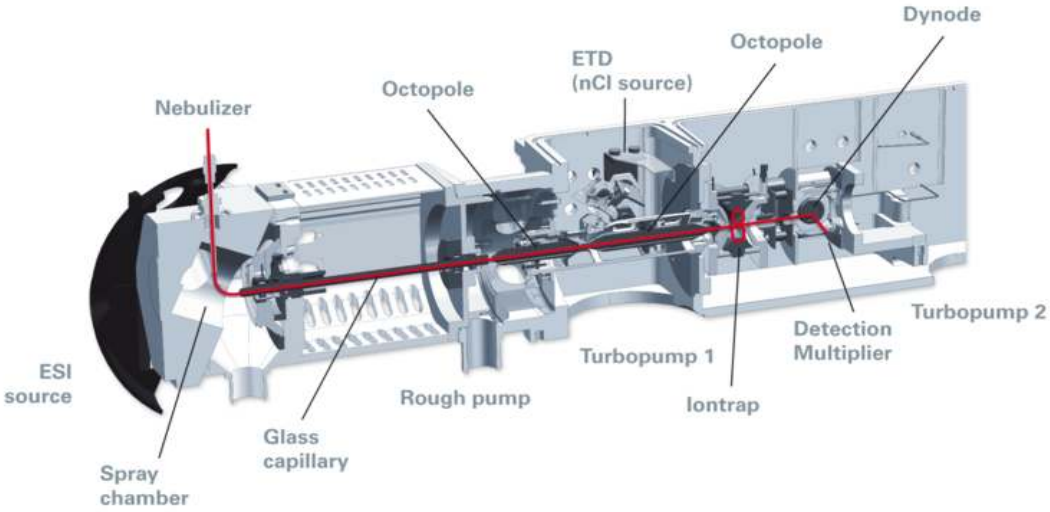


Figure 1.4: The basic layout of a FT-ICR MS instrument fitted with an electrospray ionisation source (Image reproduced with permission).

The relationship between the recorded frequency and the mass-to-charge ratio of each molecule is governed by the following equation:

$$f = \frac{qB}{2\pi m} \tag{1.2}$$

where f is the frequency, q the ionic charge, B the magnetic field strength and m the mass. The relationship between the ion's cyclotron radius and the magnetic field strength of the instrument is given by equation 1.3.

$$r = \frac{1}{qB_0}\sqrt{2mkT} \tag{1.3}$$

where r is the cyclotron radius, q is the charge of the ion, B_0 is the magnetic field strength, m is the mass of the ion, k is Boltzmann's constant and T is the temperature.

Equation 1.3 shows that unlike many other analytical techniques, FT-ICR MS can easily analyse large molecules as even a singly charged ion of 50,000Da is confined to within a 1cm cyclotron radius at room temperature using a 3T magnetic field. Once the ion's

orbital radius is increased to within the detection limit by the excitation RF, it can be shown that the largest detectable singly charged ion at room temperature is still 274,000Da, based on a 7T instrument with a cubic ion trap of dimensions a=2.54cm (Marshal *et al.*, 1998).

Mass resolution in an FT-ICR experiment is high, leading to very high mass accuracy and therefore a higher degree of positive identification of the individual peaks in the spectrum. Mass resolution is defined as the FWHM of the peaks (see figure 1.5), that is the full width of the peak at half the maximum height of that peak in the mass domain, $\Delta m_{50\%}$, which represents the minimum width two peaks of equal amplitude must be apart in order for them to be fully separated. The mass resolving power is $m/\Delta m_{50\%}$.

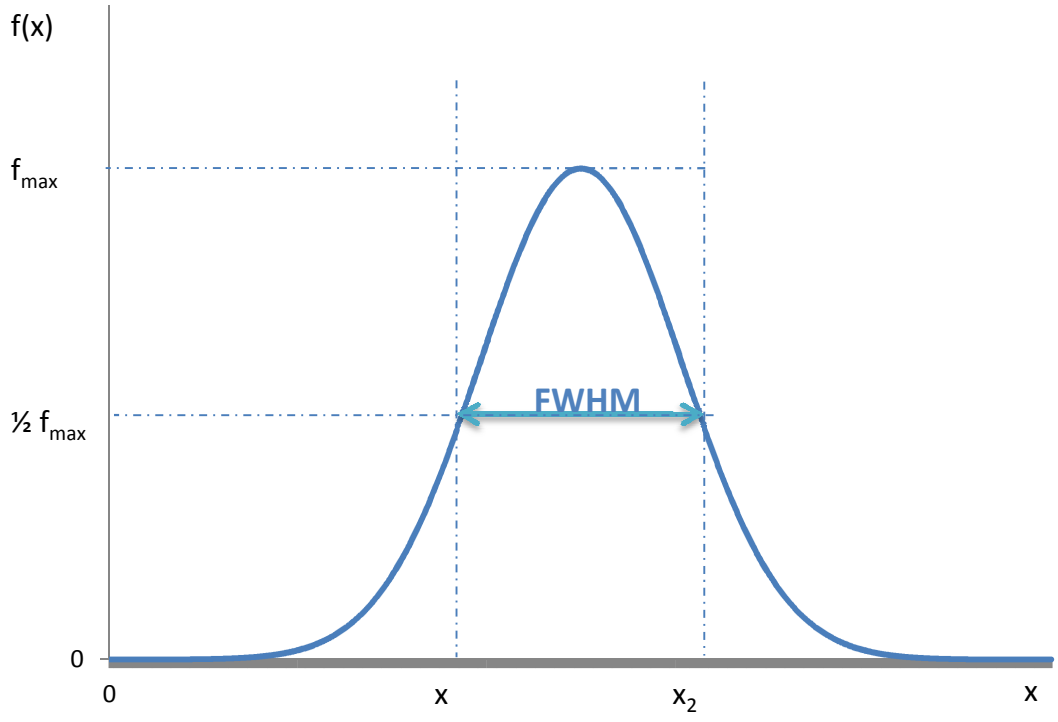


Figure 1.5: Full width at half maximum of a peak.

The relationship between magnetic field strength and mass resolving power is governed by equation 1.4.

$$\frac{m}{\Delta m_{50\%}} = -\frac{qB_0}{m\Delta w_{50\%}} \quad (1.4)$$

where m is the mass of the ion, $\Delta w_{50\%}$ is the FWHM of the peaks in the frequency domain, q is the ionic charge and B_0 is the magnetic field.

Equation 1.4 shows that mass resolving power increases with increasing magnetic field strength and is inversely proportional to m/z, as is shown in figure 1.6.

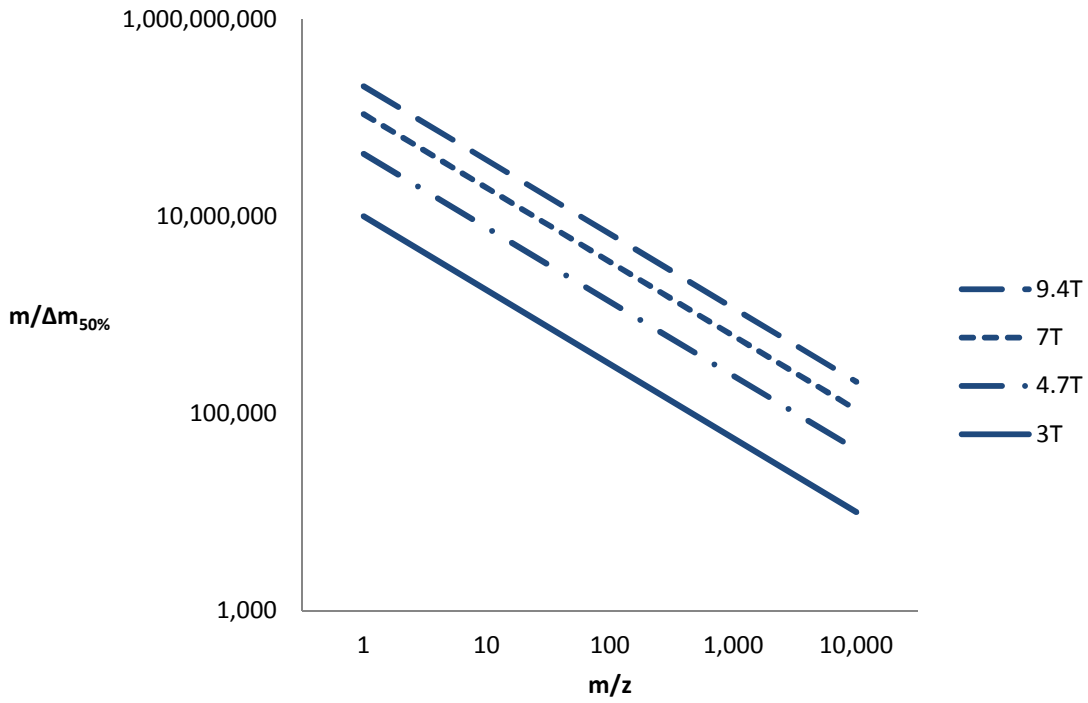


Figure 1.6: Mass resolving power $\frac{m}{\Delta m_{50\%}}$ as a function of m/z for different magnetic field strengths (Marshall et al., 1998).

Another major advantage of FT-ICR MS is that it requires very small amounts of sample. A single drop of crude oil (typically no more than 1mg in weight) in solution provides 1ml of sample. The sample is introduced into the instrument at a rate of 200nL/min. Ion collection times vary depending on the experiment but are typically in the range of 30s (Rodgers *et al.*, 2004), resulting in a rate of sample consumption of merely 0.0001ml per spectrum.

1.3.3.2 Ionisation Techniques

There exist a number of different ionisation techniques, which give rise to very different mass specra. Ionisation techniques are most commonly categorised as internal and external ionisation techniques.

If a sample contains mainly volatile species, it can easily be ionised within the ion trap using internal ionisation. The three examples of internal ionisation are electron ionisation (EI), photo ionisation (PI) and chemical ionisation (CI). EI and CI both use an electron beam to either ionise the sample directly, or to ionise a suitable reagent which will form an ionised analyte with the sample. In PI the sample is ionised using a light source, such as a laser, to induce proton or electron transfer within the sample.

External ionisation, which ionises the sample before it enters the ion trap and transports the ions into the trap using an ion guide, is particularly useful for non-volatile samples and samples that are ionised using very high pressure.

The most common external ionisation methods are electrospray ionisation (ESI) and matrix assisted laser desorption ionisation (MALDI). MALDI (Karas *et al.*, 1985; Karas and Hillenkamp, 1988) is best suited to the ionisation of large unstable molecules such as polymers, peptides and proteins and is most commonly coupled with time of flight (TOF) ion detection. While MALDI has been used to detect hydrocarbons of up to 20,000Da in mass, it is thought that some of these high molecular weight ions are formed as a result of chemical modification of the crude oil during ionisation (Qian *et al.*, 2001). Furthermore, MALDI struggles to ionise samples of large polydispersity (Byrd and McEwen, 2000) making it unsuitable for crude oil analysis.

ESI is used for most conventional analyses of crude oils as it focuses mainly on the polar heteroatom containing compounds N, S and O, which ultimately give rise to NO_X and SO_X emissions and are thus of great interest to the refining industry, and is easily coupled with high resolution ion cyclotron resonance (Marshall *et al.*, 1998).

In an electrospray instrument, the dissolved sample is injected into the ionisation cell via a thin, highly charged metal capillary, which ionises the analyte upon entry into the ion trap. The repulsion between like charged particles then causes the analyte to break down into its constituent molecules in a process known as Coulomb fission.

ESI can be operated in either positive or negative ion mode. Positive ion mode is used for analysing the basic nitrogen compounds and works by protonating the sample (Qian *et al.*, 2001a; Hughey *et al.,* 2002a). Negative ion electrospray ionises the acidic fractions and focuses on the weakly acidic nitrogen compounds.

Another commonly used ionisation mode used in conjunction with petroleomics is atmospheric pressure photo ionisation (APPI), used for analysing hydrocarbon compounds and non-polar sulphur (Purcell *et al.,* 2006). Research has shown that APPI is the most appropriate ionisation technique for crude oil analysis, however the cost of an APPI source prevents most laboratories from owning one and thus ESI is generally favoured.

1.4 Thesis Outline

CHAPTER 2 introduces the experimental procedure and discusses in detail the protocol that was followed in order to ensure that all recorded data are comparable. The chapter

covers the samples and sample preparation, the instrument and its settings as well as the precautions taken to eliminate contaminations and minimise variability in data.

CHAPTER 3 discusses the pre-processing procedures that were applied to all data before modelling. These techniques include calibration, peak identification and dimensionality reduction of data. This chapter also contains a brief discussion of the initial data and comments on the results of the day-to-day and run-to-run variability analysis.

CHAPTER 4 introduces the concept of machine learning and discusses various techniques within the field of artificial intelligence. It includes a brief literature review and also contains a description of the physical properties of the samples.

CHAPTER 5 discusses the use of artificial neural networks to model highly complex crude oil mass spectra, the algorithm used and the success of the method.

CHAPTER 6 describes a hybrid modelling technique of simulated annealing and a genetic algorithm and its results.

CHAPTER 7 contains a brief overview of clustering techniques such as principal component analysis and self organising maps as applied to the spectral data and shows how samples with similar physical properties can be grouped by similar mass spectral features.

CHAPTER 8 concludes this thesis with an in-depth discussion of the results of this research, draws any conclusions to be made and highlights any future work that is recommended.

References

Byrd H. C. M. and McEwen C.N. (2000), The Limitations of MALDI-TOF Mass Spectrometry in the Analysis of Wide Polydisperse Polymers, *Anal. Chem.*, 19, 4568 – 4576.

Campbell C.J. and Zagar J.J. (2001), Newsletter of Colorado School of Mines, Issue 2, *Petroleum Engineering Department*, Hubbart.mines.edu.

Comisarov M.B. and Marshall A.G. (1974), *Chem. Phys. Lett.*, 25, 282.

Cruz J. A. and Wishart D. S. (2006), Applications of Machine Learning in Cancer Prediction and Prognosis, *Cancer Informatics*, 2, 59 – 78.

Dehmelt H. (1990), Ion traps, *Rev. Mod. Phys., 62,* 525 – 530.

Hughey C.A., Rodgers R.P. and Marshall A.G. (2002a), Resolution of 11,000 compositionally distinct components in a single electrospray ionization Fourier transform ion cyclotron resonance mass spectrum of crude oil, *Anal. Chem.,* 74, 4145 – 4149.

Hughey C.A., Rodgers R.P., Marshall A.G., Qian K. and Robbins W.K. (2002b), Identification of acidic NSO compounds in crude oils of different geochemical origins by negative ion electrospray Fourier transform ion cyclotron resonance mass spectrometry, *Organic Geochemistry*, 33, 743 – 759.

Karas M., Bachmann D. and Hillenkamp F. (1985), Influence of the Wavelength in High-Irradiance Ultraviolet Laser Desorption Mass Spectrometry of Organic Molecules, *Anal. Chem.,* 57, 2935 –2939.

Karas M. and Hillenkamp F. (1988), Laser Desorption Ionization of Proteins with Molecular Masses Exceeding 10,000 Daltons, *Anal. Chem.*, 60, 259 – 280.

Kim S., Stanford L.A., Rodgers R.P., Marshall A.G., Walters C.C., Qian K., Wenger L.M. and Mankiewicz P. (2005), Microbial alteration of the acidic and neutral polar NSO compounds revealed by Fourier transform ion cyclotron resonance mass spectrometry, *Organic Geochemistry*, 36, 1117 – 1134.

Marshall A.G., Hendrickson C.L. and Jackson G.S. (1998), Fourier Transform Ion Cyclotron Resonance Mass Spectrometry: a primer, *Mass Spectrom. Rev.*, 17(1), 1 – 35.

Marshall A.G. and Rodgers R.P. (2004), Petroleomics: The next grand challenge for chemical analysis, *Acc. Chem. Res.,* 37(1), 53 – 59.

Marshall A.G. and Rodgers R.P. (2008), Petroleomics: Chemistry of the underworld, *Proceedings of the National Academy of Sciences,* 105(47), 18090 – 18095.

Mullins O., Sheu E., Hammami A. and Marshall A.G. (eds.) (2007), Asphaltenes, Heavy Oils and Petroleomics, *Springer,* New York, NY.

Purcell J.M., Hendrickson C.L., Rodgers R.P. and Marshall A.G. (2006), Atmospheric pressure photoionization Fourier transform ion cyclotron resonance mass spectrometry for complex mixture analysis, *Anal. Chem.,* 78, 5906 – 5912.

Qian K., Robbins W. K., Hughey C. A., Cooper H. J., Rodgers R. P. and Marshall A. G. (2001a), Resolution and identification of 3,000 crude acids in heavy petroleum by negative-ion microelectrospray high-field Fourier transform ion cyclotron resonance mass spectrometry, *Energy and Fuels,* 15, 1505 – 1511.

Qian K., Rodgers R. P., Hendrickson C. L., Emmett M. R. and Marshall A. G. (2001b), Reading chemical fine print: Resolution and identification of 3,000 nitrogen-containing aromatic compounds from a single electrospray ionization fourier transform ion cyclotron resonance mass spectrum of heavy petroleum crude oil, *Energy and Fuels,* 15, 492 – 498.

Rodgers R.P, Klein G.C., Stanford L.A., Kim S. and Marshall A.G. (2004), Characterization of heavy, biodegraded crude oils by high resolution ESI FT-ICR Mass Spectrometry, *Am. Chem. Soc., Fuel Chem.*, 49(2), 483 – 486.

Wang Z. and Fingas M. (1999), Identification of the source(s) of unknown oil spills, *Proceedings of International Oil Spill Conference*, Seattle, Washington, U.S.A.

Wold S. (1995), Chemometrics: what do we mean with it and what do we want from it?, *Chemometrics and Intelligent Lab Systems*, 30(1), 109 – 115.

Zhou X., Liua K. and Wong S. (2004), Cancer classification and prediction using logistic regression with Bayesian gene selection, *J. Biomedical Informatics*, 37, 249 – 259.

CHAPTER 2 – Experiment:

Materials and Methods

CHAPTER 2 introduces the experimental procedure and discusses in detail the protocol that was followed in order to ensure that all recorded data were comparable. This chapter covers the samples and sample preparations, the instrument and its settings as well as the precautions taken to eliminate contaminations and minimise variability in the data.

2.1 Experimental Protocol

2.1.1 Samples

Crude oils are complex mixtures of hydrocarbon compounds and contain heteroatom contaminants such as sulphur, nitrogen and oxygen. They also contain trace elements such as vanadium, nickel and iron. Exact compositions of crude oils vary greatly depending on their geographic origin and thus their geochemical formation as discussed at length in *Origin and Refining of Petroleum*, published by the American Chemical Society in 1971. Composition can also change over time as a result of aging and biodegradation (Rodgers *et al.,* 2004).

Table 2.1 shows a comparison of four crude oils from different geographical origins in terms of relative abundance of different heteroatom groups. Abundance of heteroatom groups is defined as the number of peaks found to be part of any one series as a percentage of the total number of peaks identified in the spectrum.

	Bachaquero (#10) – Venezuela	**Arab Light (#4) – Saudi Arabia**	**Forties (#5) – UK/North Sea**	**Bach Ho (#43) – Malaysia**
Heteroatom Group	Abundance (%)	Abundance (%)	Abundance (%)	Abundance (%)
Positive ESI				
N_1	19	19	24	27
$N_1{}^{13}C_1$	15	15	20	21
$N_1{}^{13}C_2$	5	0	7	5
N_1O_1	14	12	16	21
$N_1O_1{}^{13}C_1$	6	2	7	12
$N_1O_1S_1$	4	4	0	0
$N_1O_1S_1{}^{13}C_1$	0	0	0	0
N_1O_2	3	0	0	0
N_1S_1	13	17	6	0
$N_1S_1{}^{13}C_1$	6	9	0	0

N_1S_2	2	4	0	0
N_2	6	1	4	0
$N_2{}^{13}C_1$	1	0	0	0
O_1	0	1	0	4
O_1S_1	4	12	10	7
$O_1S_1{}^{13}C_1$	1	3	6	3
O_1S_2	1	2	0	0
O_2S_1	0	0	0	0
O_2S_2	0	0	0	0
Total Sulphur	31	51	22	10
Total Oxygen	34	35	39	47
Total Nitrogen	94	83	84	87
Negative ESI				
N_1	9	19	21	17
$N_1{}^{13}C_1$	1	15	16	13
$N_1{}^{13}C_2$	0	2	4	6
N_1O_1	0	6	11	13
$N_1O_1{}^{13}C_1$	0	0	4	7
N_1O_2	10	0	2	9
$N_1O_2{}^{13}C_1$	3	0	0	0
$N_1O_1S_1$	0	4	0	0
N_1S_1	1	14	2	0
$N_1S_1{}^{13}C_1$	0	10	0	0
N_1S_2	0	5	0	0
$N_1S_2{}^{13}C_1$	0	1	0	0
N_2	0	1	2	5
N_2O_1	0	0	0	0
O_1	0	9	16	13
$O_1{}^{13}C_1$	0	1	8	7
O_1S_1	0	7	0	0
$O_1S_1{}^{13}C_1$	0	1	0	0
O_2	18	0	12	8
$O_2{}^{13}C_1$	15	0	1	0
$O_2{}^{13}C_2$	5	0	0	0
O_2S_1	12	0	0	0
$O_2S_1{}^{13}C_1$	7	0	0	0
O_2S_2	2	0	0	0
O_3	4	0	0	0
O_3S_1	4	1	0	0
$O_3S_1{}^{13}C_1$	0	0	0	0
O_4	7	0	0	0
$O_4{}^{13}C_1$	1	0	0	0

O_4S_1	1	2	0	0
Total Sulphur	28	47	2	0
Total Oxygen	90	32	55	58
Total Nitrogen	23	78	62	71

Table 2.1: Relative percentages of different heteroatom groups as identified in four different crude oils of varying geographic origin.

Table 2.1 demonstrates clear differences between crude oils from different sources, most notably in their relative contents of sulphur containing compounds. The crude oils listed in this table represent three light (Forties, Bach Ho and Arab Light) and one heavy crude oil (Bachaquero). Both Forties and Bach Ho have relatively low sulphur contamination, as verified by their individual crude assays (0.25wt% and 0.06wt% respectively) and are thus classed as sweet crudes. Arab Light (1.98wt%) and Bachaquero (2.78wt%) are both examples of sour crudes.

The 76 crude oils analysed in this work are representative of the diversity of the global crude oil stock (excluding North America), ranging from very heavy to very light. The set of 76 crude oils has been analysed extensively by Shell Research laboratories over the past years and a large database of physical data exists for each of the samples.

Figure 2.1 shows a reasonably diverse dataset in terms of geographic origin. Samples from the UK/North Sea region and the Far East are slightly over-represented, while samples from the US/Canada region are not present at all.

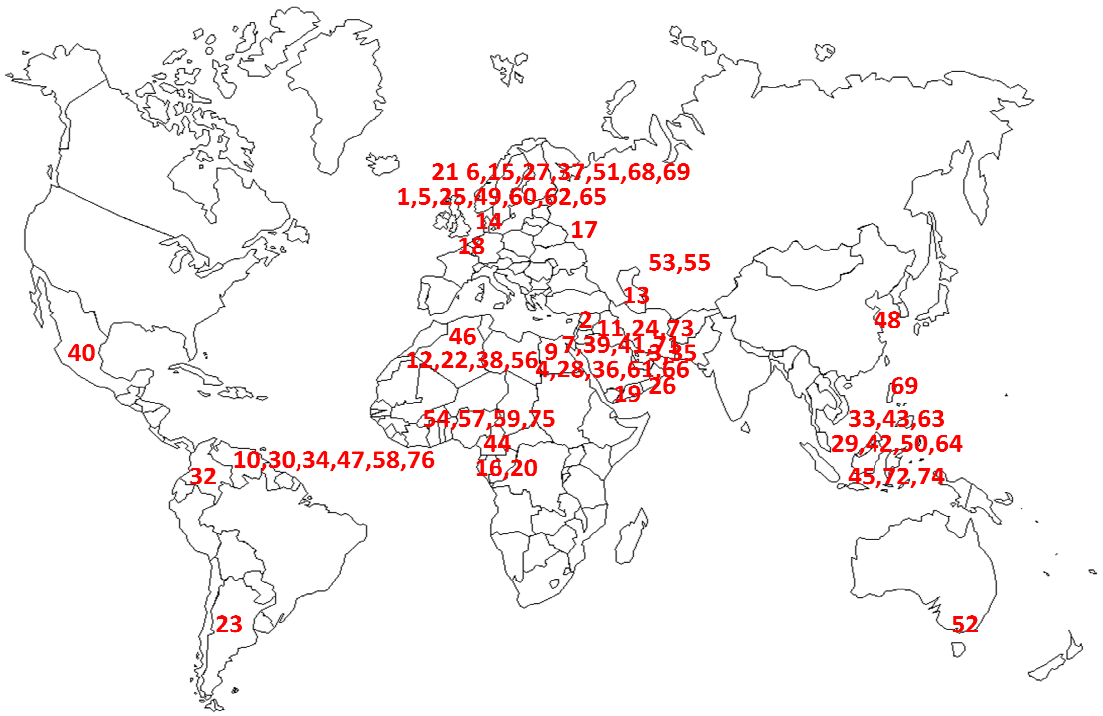


Figure 2.1: The geographical origins of the 76 crude oils analysed in this research. A key linking the names of the crude oils to their sample numbers can be found in appendix B.

The geographical regions represented by the dataset and the number of samples from each region can be seen in table 2.2.

Geographic Region	Number of crude oils in total dataset	Share of total global fuel reserves (%)
UK/North Sea	18	0.8
Middle East	18	54.4
Africa	13	9.5
Far East	13	2.9
South/Central America	9	18.1
Russia/Caspian Sea	4	9
South Pacific	1	0.4
USA/Canada	0	4.5
Other	0	0.6
TOTAL	76	100

Table 2.2: Total number of crude oils from each geographic region in dataset and share of each region of the total global fuel reserves (BP Statistical Review of World Energy June 2011).

2.1.1.1 Crude Oil Classifications

Crude oils are commonly classified as light, medium or heavy depending on their American Petroleum Institute (API) gravity. API gravity is a measure of "heaviness" with respect to water and can be derived from the sample's specific gravity using the following equation:

$$\theta = \frac{141.5}{\rho} - 131.5 \qquad (2.1)$$

where θ is the API gravity and ρ is the specific gravity.

The internationally recognised classification is as follows:

API Gravity	Crude Classification
> 31.1	Light Crude Oil
22.3 - 31.1	Medium Crude Oil
10 - 22.3	Heavy Crude Oil
< 10	Extra Heavy Crude Oil

Table 2.3: Classification of crude oils according to their API gravity.

Light crudes are very fluid and volatile and lend themselves best to the production of fuel (American Petroleum Institute). They consist mainly of short chain hydrocarbons or alkanes (C_nH_{2n+2}) which are highly combustible.

Heavy crudes are mainly aromatics, rich in asphaltenes (deposits) and are very viscous. They usually have very high boiling points and are difficult to refine. Heavy crudes are mainly used for the production of asphalt and bitumen, however with the global stock of light crudes depleting, the oil industry seeks ways to make heavy crude oils more accessible to the production of commercial and domestic fuels.

In an energy report published in 2005, the Canadian National Energy Board (NEB) quoted an average upgrading cost of crude bitumen to synthetic crude oil (SCO) of around US$16 per barrel.

The light-heavy crude price differential, which is defined as the difference in price between a barrel of light crude and heavy crude oil (U.S. Energy Information Administration), fluctuated between US$15 and US$30 per barrel during the fiscal year 2005-2006 according to the same report. These figures show that production of heavy crudes or bitumen is not financially viable unless the upgrading costs can be reduced or the light-heavy differential increases significantly.

Apart from light, medium or heavy, crude oils can be further categorised into sweet or sour crudes, depending on their sulphur content. Sweet crudes have a sulphur content of $< 1wt\%$ while sour crudes have $> 1wt\%$ sulphur.

A complete list of the crude oils' bulk properties and their minimum and maximum values within the dataset as determined by in-depth assay analysis is shown in table 2.4.

The cargo properties, which are those properties that are considered the most crucial to transporting the crude oil, are denoted with asterisks (*). They include the API gravity, sulphur content, pour point temperature and total acid number of the sample.

Property	Units	Min	Max
API Gravity *		**9.28**	**45.77**
Density	g/cm^3	0.7979	1.0045
Barrel Factor		6.271	7.898
Sulphur (Total) *	**wt%**	**0.03**	**4.003**
V50		6.102	37.626
Pour Point *	**°C**	**-48**	**43.33**
TAN *	**mg KOH/g**	**0**	**4.2**
Reid Vapour Pressure	PSI	0	11.4
Methane & Ethane	wt%	0	0.09
Propane	wt%	0	0.96
Isobutane	wt%	0	0.7
n-Butane	wt%	0	1.9
Isopentane	wt%	0	1.89
n-Pentane	wt%	0	2.59
Cyclopentane	wt%	0	0.45
C6 Iso-paraffins	wt%	0.01	2.35
n-Hexane	wt%	0	2.05
Methylcyclopentane	wt%	0	1.14
Benzene	wt%	0	0.81
Cyclohexane	wt%	0	1.35
Yield by Temperature (per cut)	wt%	0.02	82.17
Density (per cut)	g/cm^3	0.6399	1.0784
Sulphur (Total) (per cut)	wt%	0	7.45357
V50 (per cut)		-10.467	65.436
Octane Number (per cut)		19.5	83.5
Naphthenes (per cut)	wt%	0	83
Aromatics (per cut)	wt%	0	46.8
Paraffins (per cut)	wt%	9.2	100
Smoke Point (per cut)	Mm	10.17	49.91
Freeze Point (per cut)	° C	-176.33	13.97
Cloud Point (per cut)	° C	-133.64	72.19
Pour Point (per cut)	° C	-99.02	125.49
Vanadium (per cut)	ppm	0	1401.314
Nickel (per cut)	ppm	0	204.52
Asphaltenes (C7) (per cut)	wt%	0	28.591
Carbon Residue (per cut)	wt%	0	31.45

Nitrogen (Total) (per cut)	ppm	6.8	12454
Nitrogen (Basic) (per cut)	ppm	0.2	4478.7
Wax (COED) (per cut)	wt%	0.46	52.97
Molecular weight (per cut)	g/mol	250	892
Aromatics (1-ring) (per cut)	wt%	0.348	8.492
Aromatics (2-ring) (per cut)	wt%	0.001	6.617
Aromatics (3-ring) (per cut)	wt%	0.05	8.435
Aromatics (4-ring+) (per cut)	wt%	0.064	33.223

Table 2.4: Complete list of assay data available for all 76 crude oils with minimum and maximum values. Asterisks (*) denote the cargo properties. "Per cut" signifies measurements which were taken of individual distillation cuts of a crude oil, instead of the original crude oil.

2.1.2 Sample Preparation

Samples were heated to around 20°C above their pour point temperature while mixing in order to ensure homogeneous sampling. Approximately 10mg of the sample was weighed out and mixed with the appropriate amount of a 1:1 mixture of HPLC grade Sigma Aldrich Chromasolv® Toluene (99.9%) and Sigma Aldrich Chromasolv® Methanol (≥ 99.9%) to create a 1mg/ml solution.

1ml of each sample was sub-sampled and spiked with 10µl of ammonium hydroxide (Sigma Aldrich, 28% in H_2O, ≥ 99.9%) or 5µl of formic acid (Sigma Aldrich GC-MS Chromasolv® Methanol with 0.1% formic acid (≥ 99.5%)) to aid deprotonation and protonation respectively (Marshall and Rodgers, 2004).

It is essential that all solvents that were used in the experiment are HPLC grade (≥ 99.9% purity) as the instrument is very sensitive and will pick up on any trace contaminants in the solvents.

Toluene was used to dissolve the crude oil, while methanol, a volatile organic compound, was used to aid in the evaporation of the solvent during electrospray ionisation.

Sample concentration was a vital aspect of instrument setup. A too dilute sample resulted in a recorded spectrum that was too weak, leading to a poor signal-to-noise ratio. If the sample was too concentrated it caused contamination of the instrument and thus affected the accuracy of subsequent sample recordings.

A concentration of 1mg of sample per ml of solvent was chosen as it provided the best signal-to-noise ratio without the samples being too concentrated and thus causing contamination of the instrument.

2.2 Instrument Setup

The two instruments used for this research were a 9.4T Apex-Qe FT-ICR MS (Bruker Daltonics) located in the Chemistry Research Laboratory at the University of Oxford and a 7T LTQ FT Ultra MS (Thermo Scientific) at Shell's Westhollow Technology Centre in Houston, Texas.

As the project relied heavily on the accuracy and quality of the data a lot of effort was put into optimising the experimental design: Issues such as the repeatability and reproducibility of data had to be evaluated and errors and uncertainties quantified. Discussions in this area were held with statisticians and experimental design experts at Shell Technology Centre Thornton in Chester, and a visit was made to the Westhollow

Technology Centre in Houston to discuss details of the experimental techniques involved.

Effects on the instrument, both environmental and operational, were investigated and logged in order to discount them as possible sources of ambiguity during data analysis. Environmental effects were quickly discounted as both instruments are kept in climate controlled laboratories and were therefore largely unaffected by environmental parameters.

Operational parameters, however, proved crucial to reliability of data, which is why strict protocols were followed at both data collection sites. Further details of operating parameters and their effects on the recorded spectrum are given in section 2.2.2.

As the set of samples varied greatly in terms of chemical composition, the effects of experimental variables and instrument parameters on the spectrum were monitored in order to assure that different samples did not respond differently to changes in the instrument setup. For this purpose, three standards, each one a typical heavy, medium or light crude, were chosen and run at random intervals throughout the experimentation process and monitored in order to quantify run-to-run and day-to-day variability.

2.2.1 Calibration

As well as the three standards chosen to verify repeatability and reproducibility, a separate calibrant solution, namely sodium formate (Sigma Aldrich, $\geq$ 99.0%), was run at the beginning of each experimental session to assess and reset the machine's mass

calibration. The spectrum of sodium formate solution covers the same m/z range as most of the crude oil samples (0 – 1,400m/z) with clearly resolved peaks at 68Da separation (see figure 2.2). The use of an external calibrant is fairly superfluous, but acts as a means of reassurance that the instrument is still performing well. An additional calibrant is not added to each of the samples, as the hydrocarbons themselves act as internal standards.

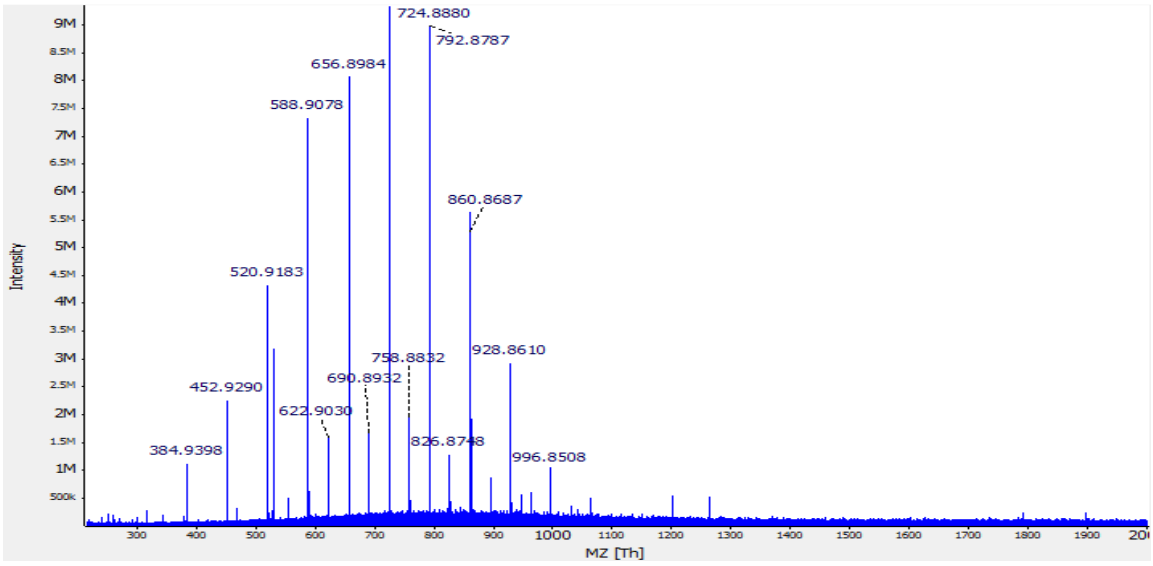


Figure 2.2: Sodium formate calibrant solution recorded in positive ESI mode on a 9.4T FT-ICR MS instrument. The repeat unit of singly charged HCOONa adducts (68Da) can be easily identified. The peak at 622.90m/z and subsequent peaks at 68Da separation are due to doubly charged adducts (www.sigmaaldrich.com).

2.2.2 Tuning of the Instrument

In order to achieve maximum mass accuracy and resolution the instrument needs to be finely tuned. Parameters such as the ion trapping time or the trapping voltage can be varied to boost or suppress the signal over certain mass ranges and can thus help to

enhance certain areas of interest. For example, very light or very heavy crudes will have their most intense peaks at very low and very high masses respectively (see figure 2.3).

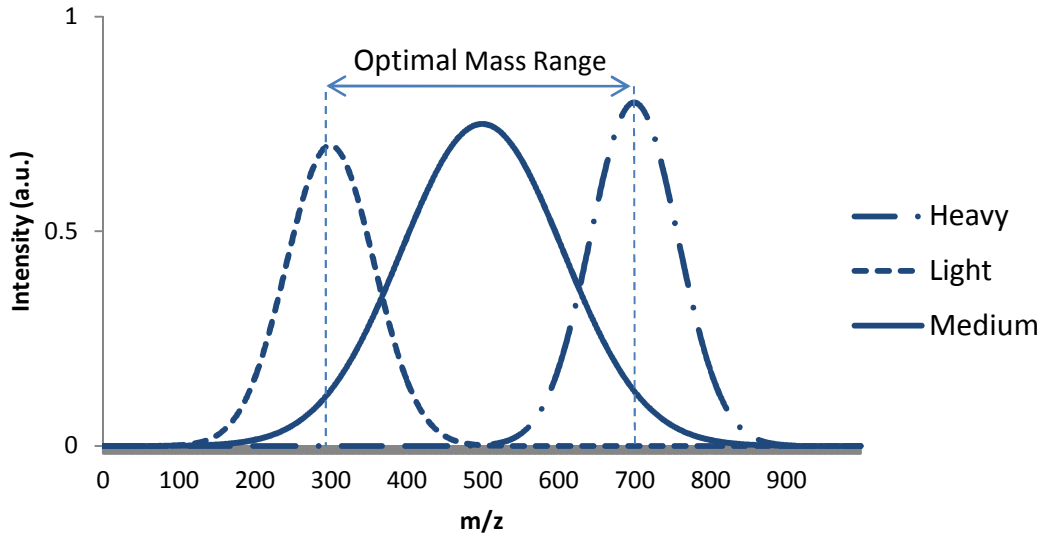


Figure 2.3: The overall peak distribution for medium, light and heavy crudes.

The instrument's optimal scanning range is from 300m/z to 700m/z with lower resolution and decreased mass accuracy above and below that range (table 2.5). Researchers may therefore choose to tune the instrument in such a way as to favour the lower or higher masses when running extremely light or heavy crude samples.

However, as data from all 76 samples was analysed as a group, it was essential for this project that an optimal set of working parameters was chosen and used consistently. This allowed meaningful comparison of different spectra; much time was spent investigating this aspect of the work.

Once the instrument parameters had been chosen, the effect of these parameters on the resulting spectrum was determined. As mentioned previously, due to the diversity of the sample set it was necessary to verify that each parameter affected each sample

in a comparable fashion. This again was achieved by running three chosen standards and comparing the effects on their resultant spectra.

	Experimental m/z	Theoretical m/z	Error (ppm)
Low Masses	536.42507	536.425077	-0.01
	536.42836	536.428448	-0.16
	536.4826	536.482592	0.01
	536.49407	536.494069	0.00
	536.51011	536.510037	0.14
Average			**0.064**
High Masses	969.79556	969.795928	-0.38
	969.80263	969.804034	-1.45
	969.8084	969.809194	-0.82
	969.89758	969.897934	-0.36
	969.99179	969.991834	-0.05
Average			**0.61**

Table 2.5: Decreased mass accuracy becomes evident at higher m/z. Higher masses have up to 10× higher ppm errors.

A list of instrument parameters and their chosen values can be found in table 2.6. The effect of each parameter on the overall mass spectrum is briefly described.

Parameter	Value	Effect
Capillary Voltage	4400V	Absolute voltage. Voltage is positive for negative ion ESI and negative for positive ions.
Spray Shield Voltage	3400V	-
Nebulizer Gas Flow	1L/min	-
Drying Gas Flow	6L/min	-

Drying Gas Temperature	~ 200°C	-
Collision Cell Accumulation Time	1.5s	Increase to boost the intensity of low concentration samples. Causes charge stripping and fragmentation if sample is too concentrated.
Time Of Flight	0.0018s	Must be increased for higher mass ions
Capillary Exit Voltage	80 - 90V	Higher voltage causes fragmentation
Trapping Voltage	4.1V (11V)	-
Hexapole Voltage	1.6V (3.5V)	-
Source Accumulation Time	0.05s	-

Table 2.6: Instrument parameters

Adjustment of instrumental parameters may severely alter the shape of the spectrum, so the accuracy of the resultant peak distribution must be confirmed. Several of the tunable instrument parameters can affect the relative intensity of peaks in certain parts of the spectrum, especially of fragment ions in high molecular mass samples. The relative abundance of the individual chemical components within the sample is very important, it was therefore essential to verify that the peak intensities were a true reflection of the molecular abundances and not biased by suboptimal instrument parameters.

To do this the three standards were separately run on a low resolution instrument and the shape of the envelope governing the peaks was compared with that yielded by the high resolution instrument. Since with the low resolution instrument the spectrum cannot easily be manipulated, the peak distribution acquired was considered to be accurate. It was then used to verify the peak distribution of the high resolution instrument following optimisation of the experimental parameters.

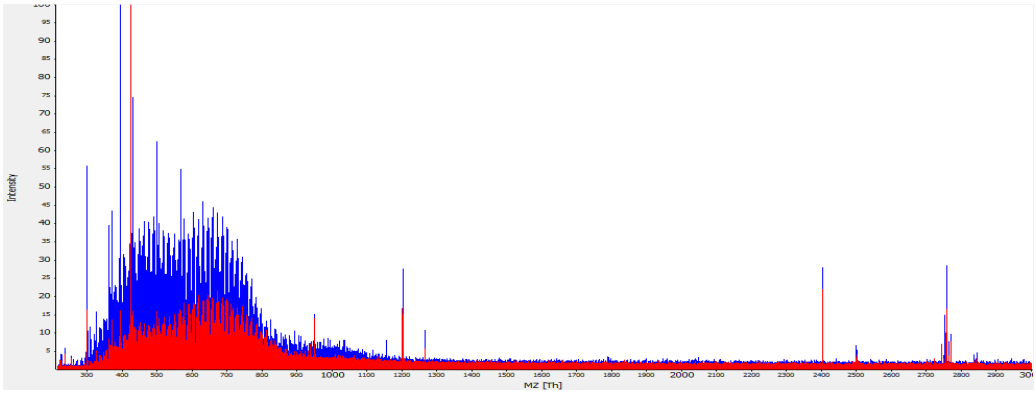


Figure 2.4: The effects of varying collision cell accumulation times and time of flight on a light crude oil spectrum. Increased time of flight (D2=0.002) and high cell accumulation time (D10=2s) increases peak intensity, but also skews the mass spectrum to favour lower masses (blue spectrum), while a lower time of flight (D2=0.0015) and collision cell accumulation time (D10=1.5s) results in a much smoother spectrum (red spectrum), while suffering a slightly higher signal-to-noise ratio.

2.3 Anti-contamination Procedure

Contamination of the sample is a serious issue when using sensitive instrumentation such as the FT-ICR MS. In order to avoid contamination of glassware, all glassware was washed with solvent and baked before use. Borosilicate glass vials with polytetrafluoroethylene (PTFE) faced rubber-lined caps were used in order to avoid sample contamination.

Cross contamination between samples was avoided by rinsing the syringe which injects the solution into the ionisation cell with solvent after every sample run. In order to reduce the influence of left over ions from the previous sample in the cell, each sample was scanned 200 times and the individual recordings were averaged to give the resultant spectrum for each sample.

The solvent mixture of 0.5ml toluene with 0.5ml methanol used to dilute the samples was run separately on the FT-ICR MS in order to record the background spectrum (figure 2.5) and to ensure that the solvents were not contaminated.

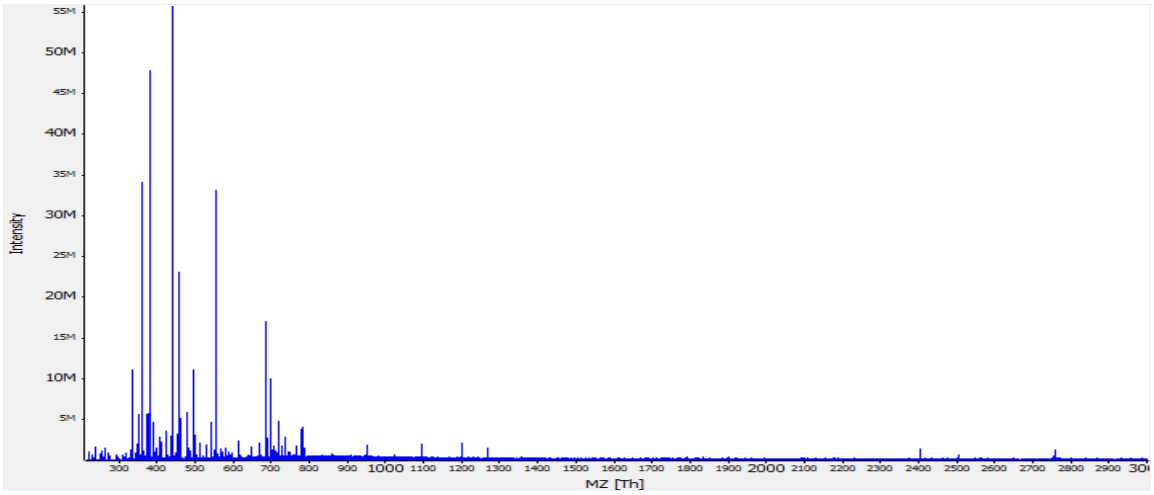


Figure 2.5: The mass spectrum of the solvent (0.5ml toluene/0.5ml methanol) to show whether either solvent is contaminated.

2.4 Experimental Log

A rigorous experimental protocol was required in order to ensure that data remained reliable and no data were lost or swapped accidentally. The run order of all samples and standards, as well as the regular runs of blanks and calibration solution, was logged in order to ensure that scan and sample numbers were able to be matched up during the data analysis stage.

Recalibration using the sodium formate (HCOONa) external calibrant was performed at the beginning of each experimental session and following every sixth sample. While this might be considered to be excessive and analysis of the HCOONa spectrum before calibration showed that the mass accuracy had not deteriorated significantly in

between calibrations, there is no known disadvantage to calibrating the instrument frequently.

The light, medium and heavy standards were introduced into the sample sequence at random intervals, in order to monitor the run-to-run variability of the instrument. Samples were run in alphabetical order according to their names.

2.5 Discussion

The envelope verification technique mentioned in 2.2.2 was performed on the initial set of spectra recorded during the first phase of data acquisition and proved successful. A small number of spectra were recorded using different instrument parameters and again on a lower resolution GC-MS instrument and the envelopes of the spectra for the two methods were compared visually. As the individual instrument parameters for the high resolution FT-ICR MS have such large effects on the shape of the spectrum, it was relatively easy to determine those parameters that most closely matched the resultant spectrum to that gathered on the low resolution GC-MS (Bruker Daltonics). The notable features are the centroid of the spectral envelope (i.e. the m/z that has the highest intensity) and the width of the envelope (i.e. the range of m/z values that the spectrum encompasses). The intensity of the spectrum is relative and cannot be compared between the two analytical techniques.

The quality of the fit between the two envelope functions was initially verified by eye and later proved using the chi square test (equation 2.2)

$$\chi^2 = \sum_{i=1}^{N} \frac{(O_i - E_i)^2}{E_i} \qquad (2.2)$$

where O_i is the intensity of the i^{th} point on the high resolution spectral envelope and E_i is the i^{th} point on the low resolution spectral envelope, where both envelopes are described by a total of N points. The value of χ^2 can then be related to a probability value using the cumulative distribution function (CDF) given by equation 2.3 which determines the likelihood that any point on the envelope of the low resolution spectrum E_i can be better described by the envelope function for the high resolution spectrum, $f(O_i)$.

$$F(x) = \frac{\gamma(\frac{\upsilon}{2},\frac{x}{2})}{\Gamma(\frac{\upsilon}{2})} \tag{2.3}$$

where $\gamma(\frac{\upsilon}{2},\frac{x}{2})$ is the incomplete gamma function (equation 2.4), $\Gamma(\frac{\upsilon}{2})$ is the gamma function (equation 2.5) and υ is the number of degrees of freedom.

$$\gamma\left(\frac{\upsilon}{2},\frac{x}{2}\right) = \int_0^x t^{\frac{\upsilon}{2}-1}e^{-t}dt \tag{2.4}$$

$$\Gamma\left(\frac{\upsilon}{2}\right) = \int_0^\infty t^{\frac{\upsilon}{2}-1}e^{-t}dt \tag{2.5}$$

A probability value of one denotes that the two envelopes are identical, while a probability less than 0.5 implies that the envelopes are a poor fit.

The aim of the chi square test is to determine the parameter set which results in the best spectral representation of the crude oil, as compared with a low resolution spectrum. There exist only a few possible solutions for the spectral envelope with best fit and the instrument parameters have to be accepted regardless of the probability, provided the fit is better than that of any other envelope function. The parameter set

which resulted in the lowest χ^2 was chosen as the optimum parameter set, where $\chi^2 = 0$ if the two envelopes are identical.

As mentioned previously, the comparison of the envelope functions merely ensures that the intensities of the peaks in the final crude oil spectra are representative of the relative abundances of the chemical compounds within the samples. The intensity of different peaks with respect to one another is used in the final model.

Analysis of the solvent spectrum verifies that there are no major contaminants present in the solvents. Initial scans showed heavy contamination of polyethylene glycol (PEG) which was subsequently traced back to a chemical used when washing glassware. As a result, fresh glassware was used for each sample, thus ensuring that none of the glassware could be contaminated.

The results of the day-to-day and run-to-run variability analyses are discussed in chapter 3.

References

BP p.l.c., BP Statistical Review of World Energy, *www.bp.com,* accessed June 2011.

Bruker Daltonik GmbH, *www.bdal.com*, accessed June 2011.

Canadian National Energy Board (NEB), Short-term Outlook for Canadian Crude Oil to 2006, September 2005, *www.neb-one.gc.ca.*

Marshall A.G. and Rodgers R.P. (2004), Petroleomics: The next grand challenge for chemical Analysis, *Acc. Chem. Res.,* 37(1), 53–59.

Rodgers R.P, Klein G.C., Stanford L.A., Kim S. and Marshall A.G. (2004), Characterization of heavy, biodegraded crude oils by high resolution ESI FT-ICR Mass Spectrometry, *Am. Chem. Soc., Fuel Chem.*, 49(2), 483-486.

Sigma Aldrich Co., *www.sigmaaldrich.com*, accessed June 2011.

Thermo Fisher Scientific Inc., *www.thermo.com*, accessed June 2011.

U.S. Energy Information Administration, *www.eia.org*, accessed June2011.

Witt M. (2007), Application Note FTMS-36, Bruker Daltonik GmbH.

CHAPTER 3 – Data Analysis

CHAPTER 3 discusses the pre-processing procedures that were applied to all data before modelling. These techniques include calibration, peak identification and dimensionality reduction of data. This chapter also contains a brief discussion of the initial data and comments on the results of the day-to-day and run-to-run variability analyses.

3.1 Introduction

Gathering of high accuracy data in petroleomics is only a small part of the overall challenge. Vast amounts of research have been carried out in order to ensure increasing reliability and quality of data collected for highly complex crude oil samples. In particular, researchers at the National High Magnetic Field Laboratory, NHMFL, in Tallahassee are focussed on producing increasingly detailed spectra of crude oils with higher resolution and higher mass accuracy using superconducting magnets up to 21T in strength (www.magnet.fsu.edu).

The real challenge lies in analysing this data in order to make the most sense out of these highly complex samples, containing up to 30,000 different elemental compositions (Hsu *et al.,* 2011).

3.2 Pre-processing

There are a number of ways to analyse data, from actual processing of the data in order to find patterns or classifications between samples, to mere graphical representations of simple spectra that can themselves reveal a lot of information about a sample.

In order to ensure that any analysis, be it simple graphical representation or the application of complex modelling algorithms, can be done successfully, it is important to pre-process the data.

Pre-processing can achieve a number of different goals, most notably the removal of both noise and experimental outliers from a dataset. Furthermore, pre-processing can

involve baseline correction, data reduction and dimensionality reduction. It should also include some analysis on the reliability of the data.

3.2.1 Calibration

The most important aspect of data collection is that of calibration. Unless the data is well calibrated, there is no cohesion between samples. Most modelling algorithms rely greatly on the comparability of data from sample to sample.

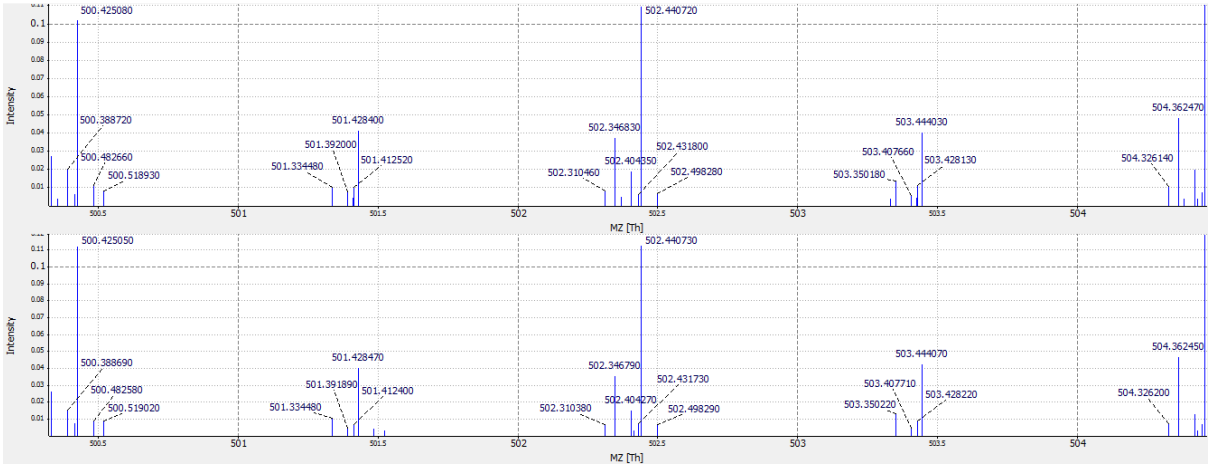


Figure 3.1: Two spectra of the same sample showing minor disagreement between the m/z values. Calibration was used to realign these spectra with one another.

Calibration is most commonly applied while the data are recorded by calibrating the apparatus. As discussed in chapter 2, a lot of effort has gone into calibrating the apparatus on a regular basis in order to ensure that the data remain reliable. However, good algorithms should be able to be used with data from a number of different sources and it is not always possible for a data analyst to verify the provenance of the data. It is therefore advisable to perform some form of calibration during the pre-processing of the data.

The most common method of calibration is to identify a known component present in each of the sample datasets and then to shift the x-axis of each spectrum so that the position of that component coincides in each of the datasets.

Mass spectra are more complex in that the mass accuracy varies as a function of m/z, with high mass accuracy at the centre of the spectrum (at around 500m/z) and decreasing levels of accuracy towards the edges of the spectrum at 300m/z and 700m/z, therefore calibration using only one component in the spectrum does not mean that the entire spectrum will be well calibrated.

For this reason it is advisable to calibrate using a common series of components in each of the samples.

Crude oils are primarily made up of hydrocarbons; it is therefore justifiable to assume that the basic C_nH_{2n+2} series is present in every sample.

By individually identifying members of any homologous series with peaks 14Da apart across the full range of the spectrum and then adjusting the position of those peaks to the exact masses of that series, one can ensure that the spectrum is calibrated over the full range of masses.

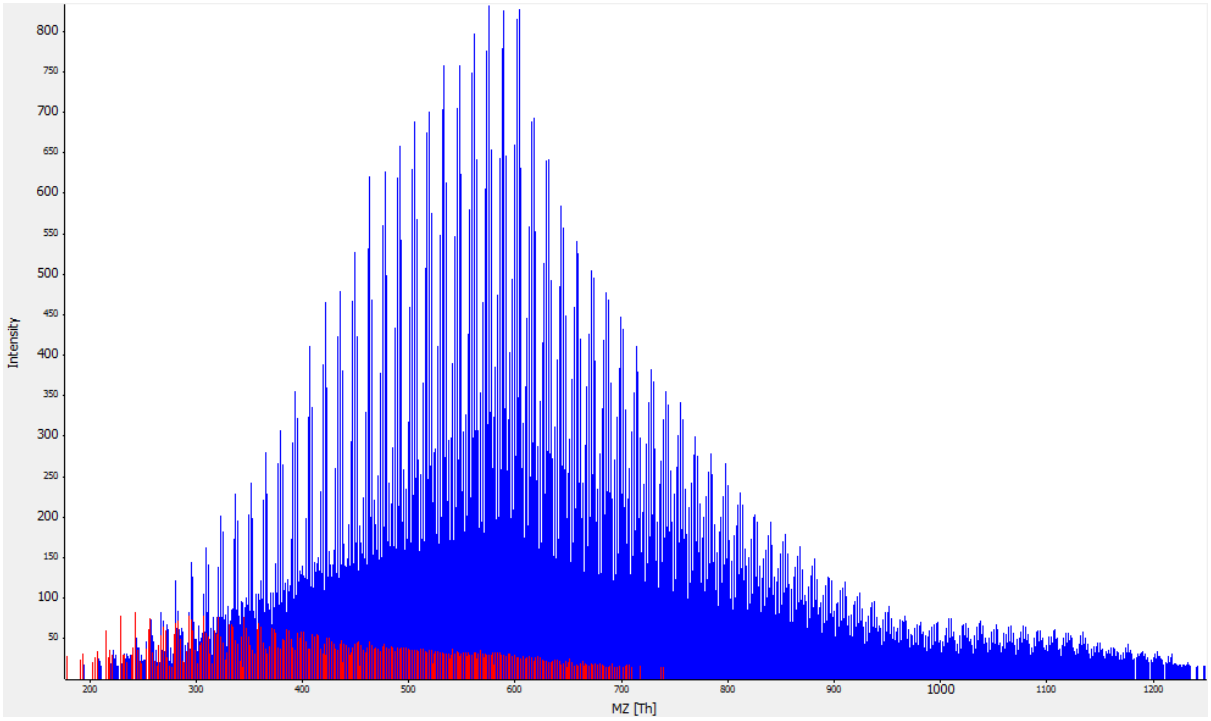


Figure 3.2a: Mass spectrum of Kittewake crude oil gathered in positive ESI mode on a 7T Thermo LTQ Ultra FT MS. Peaks belonging to the homologous series with repeat unit CH_2 are identified in red.

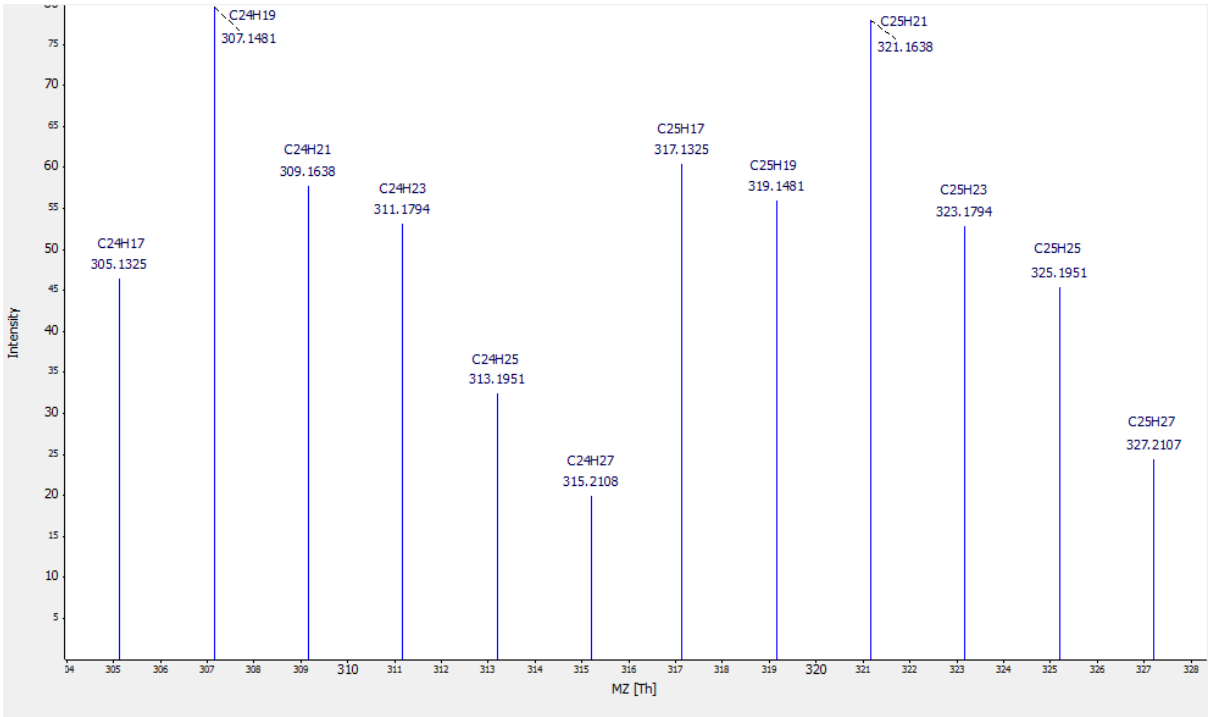


Figure 3.2b: A close up of the CH_2 repeat unit showing the 14Da peak separation between the most intense peaks in each 'packet'. Subsequent peaks are separated by 2Da or two hydrogen atoms.

Despite the spectra being calibrated, there still exists a small level of inaccuracy in the masses of each of the peaks in the spectrum. This is due to a number of reasons, from general limitations within the detector to peak broadening as a result of insufficient time domain bandwidth. These inaccuracies should lie within only a few parts per million but are nevertheless sufficient to impede successful modelling of the sample. It is therefore advisable to identify the formula of each chemical that gives rise to a peak and use the calculated masses for these molecules in subsequent analyses rather than the recorded values.

3.2.2 Peak Identification

Following calibration, peak identification is a major step in reducing noise from a spectrum by isolating only those data points which are genuine peaks. Peak identification algorithms depend on the nature of the spectrum.

Chemical Formula	Measured Mass	Calculated Mass
$C_{32}H_{28}N_1$	426.22177	426.22163
$C_{29}H_{32}N_1S_1$	426.22511	426.22500
$C_{30}H_{36}N_1O_1$	426.27925	426.27914
$C_{29}H_{37}N_2{}^{13}C_1$	426.29854	426.29848
$C_{31}H_{40}N_1$	426.31562	426.31553
$C_{28}H_{44}N_1S_1$	426.31895	426.31890
$C_{30}H_{52}N_1$	426.40949	426.40943

Table 3.1: An example of the molecules found in a crude oil, highlighting the importance of mass accuracy and resolving power of the mass spectrometer. The largest deviation between calculated and measured mass is 0.00014Da, showing a minimum accuracy of 0.3ppm is required in order to resolve these peaks.

If the peaks are discrete, then peak identification can only be done by matching m/z values to a known database of masses of chemical components.

If the spectra are not discrete, it is possible to use a peak fit function such as a Gaussian function to identify peaks which are smooth curves and discard noise signals, which are likely to be uneven in shape. Overlapping peaks in a spectrum that are the result of noise or insufficient resolution can be deconvoluted using peak fitting.

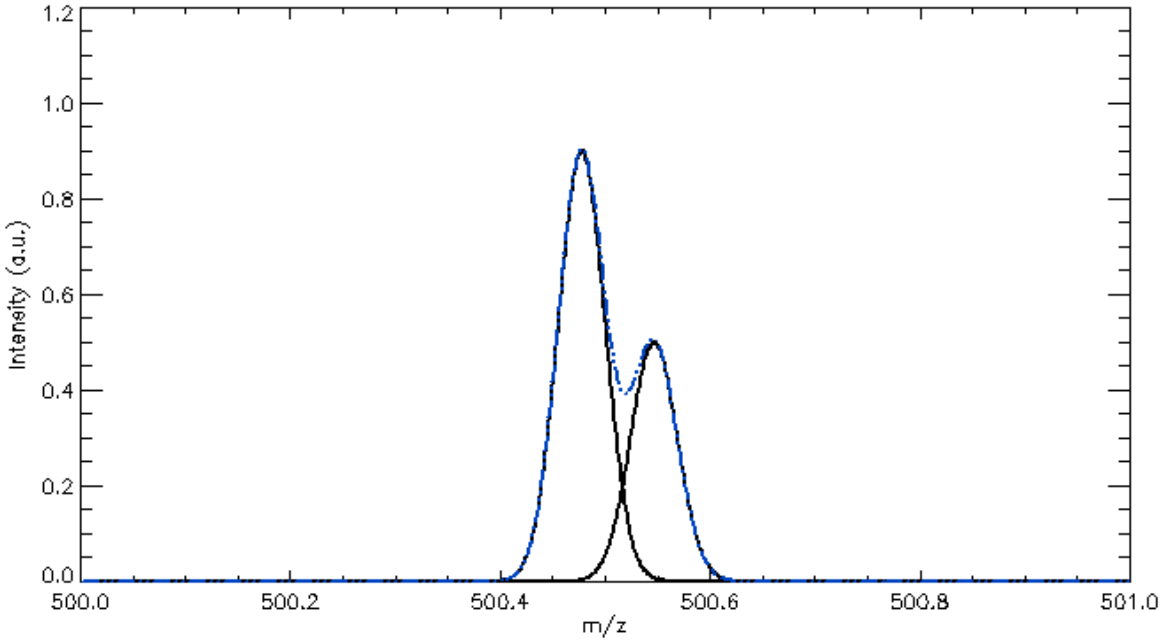


Figure 3.3: Two separate peaks that may result in a convoluted peak (blue line) if the mass spectrum does not have sufficient resolution.

A commonly used peak fitting function in mass spectrometry is the Gaussian distribution, as the peaks in a standard mass spectrum should be described by only three parameters which typically give a good approximation of the function given by equation 3.1; the position of the peak x_0, the height of the peak A, and the full width at half maximum (FWHM) of the peak or 2.355σ , where σ is the standard deviation of the peak (figure 3.4).

$$G(x) = \frac{A}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x_i - x_0)^2}{2\sigma^2}} \qquad (3.1)$$

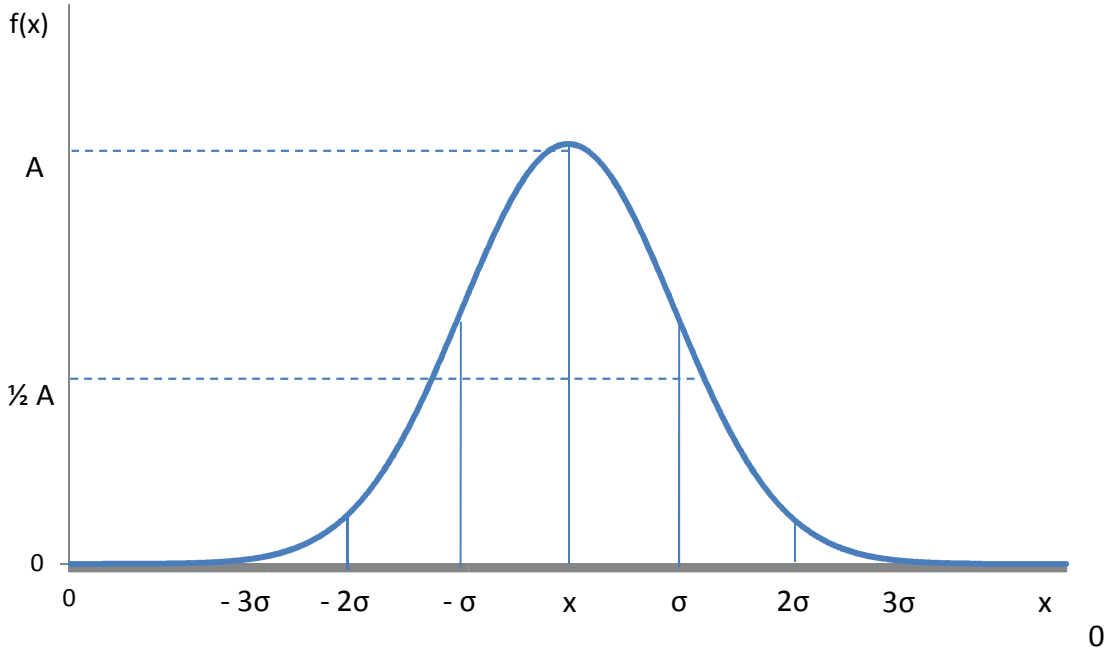


Figure 3.4: In a Gaussian distribution, 68.3% of the area under the curve falls within one standard deviation σ of the mean x_0. 95.4% lies within 2σ and 99.7% within 3σ. The full width at half maximum (FWHM) of the peak is equal to 2.355σ.

The calibration, peak identification and noise reduction processes are all performed using specifically designed software called COMPOSER, created by Sierra Analytics in collaboration with researchers at the National High Magnetic Field Laboratory, FSU, Tallahassee (www.massspec.com). The methodology employed by COMPOSER is described in the following section.

3.2.2.1 COMPOSER

COMPOSER was created in collaboration with internationally recognised experts in the field of petroleomics (www.massspec.com) and special licenses were issued to

researchers at Shell Global Solutions to aid in the development of the user interface of the software.

Most instrument manufacturers provide software which is capable of interpreting the spectra recorded using their instruments, most notably Bruker Daltonics' Compass (http://www.bdal.com) and Thermo Scientific's Xcalibur (http://www.thermoscientific.com). Both these peak identification tools are recognised as being capable of successfully identifying chemical composition of peaks using generic atomic search criteria for samples with elemental composition up to $C_{200}H_{200}N_{200}O_{200}Br_3Cl_3F_3S_4$ with high-level accuracy (Vahlenkamp *et al.*, 2007) and are supplied as standard with the instrument's operation software. However, instrument software is limited to analysing only data which has been collected on that particular instrument and can seldom be used on external data files. As the data for this project were gathered on both a Bruker Daltonics Apex-Qe FT-ICR MS and a Thermo Scientific LTQ FT Ultra MS instrument, it was essential that all spectra were analysed using the same software to make peak identification comparable. Furthermore, researchers at Sierra Analytics, with the help of scientists at NHMFL, have ensured that the atomic search criteria for peak identification in COMPOSER are limited to those elements known to be present in crude oils ($C_cH_hN_nO_oS_s$ with $c \leq 200$, $h \leq 200$, $n \leq 200$, $o \leq 200$ and $s \leq 4$), therefore reducing the number of possible candidates in a database search and increasing the likelihood of positive identification of peaks. Another inbuilt feature of the COMPOSER software is the creation of Kendrick plots (Kendrick, 1963) and Van Krevelen diagrams (Van Krevelen, 1950), both are commonly used graphical representation methods in the field of petroleomics (Wu *et al.*, 2004).

The software scans through the continuous raw mass spectrum searching for peaks with a predefined peak width at FWHM. In the work reported here, this value was set to 0.002Da, as the mass resolution of the data was very high; $\frac{m}{\Delta m_{50\%}} \geq 350{,}000$ on average.

The intensity of the centroided peaks was determined using the total height of the identified peaks, although the software also allows selection of the area under the peak.

Once the continuous spectrum was converted into centroided peaks, the spectrum was calibrated using a known homologous series. This research used the N_1 series with the base chemical formula of $C_{14}H_{11}N$ and the repeat unit CH_2.

Formula	Calculated Mass
C_9HN	124.0182
$C_{10}H_3N$	138.0338
$C_{11}H_5N$	152.0495
$C_{12}H_7N$	166.0651
$C_{13}H_9N$	180.0808
$C_{14}H_{11}N$	194.0964
$C_{15}H_{13}N$	208.1121
$C_{16}H_{15}N$	222.1277
$C_{17}H_{17}N$	236.1434
$C_{18}H_{19}N$	250.1590
$C_{19}H_{21}N$	264.1747

Table 3.2: The homologous N_1 series used to calibrate the crude oil spectra.

The tolerance for matching of peaks to elemental compositions was set to 5ppm with intensity threshold 5%.

The peak fit function was automated, allowing the software to choose a polynomial which best described the distribution of target peaks and used the coefficients to reposition the peaks in the recorded spectrum along the m/z axis. The polynomial is chosen in the range of 1st to 5th order (equations 3.2a and 3.2b) depending on the lowest standard deviation between target and recalibrated peak.

$$y = a_0 + a_1 x \quad (3.2a)$$

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 \quad (3.2b)$$

The peaks in the calibrated spectrum were compared with those in a database that contains only those series known to be common in crude oils in order to increase the likelihood of positive identification. Series commonly searched for include C_nH_{2n+2}, $C_nH_{2n+2}S$, $C_nH_{2n+2}O$ and $C_nH_{2n+2}N$.

The assignment algorithm first converts the m/z values to the Kendrick mass scale by dividing every peak position by the Kendrick factor (Wu *et al.*, 2004). The Kendrick factor is 1.0012, or the monoisotopic mass of the repeat unit CH_2 divided by its exact mass (14Da / 14.01565Da).

The Kendrick mass defect is determined as the remainder of the subtraction of the nominal Kendrick mass from the Kendrick mass. All members of the same homologous series have the same Kendrick mass defect and are grouped together (see figure 3.5). The elemental compositions that best match the isolated peaks are assigned to the spectrum.

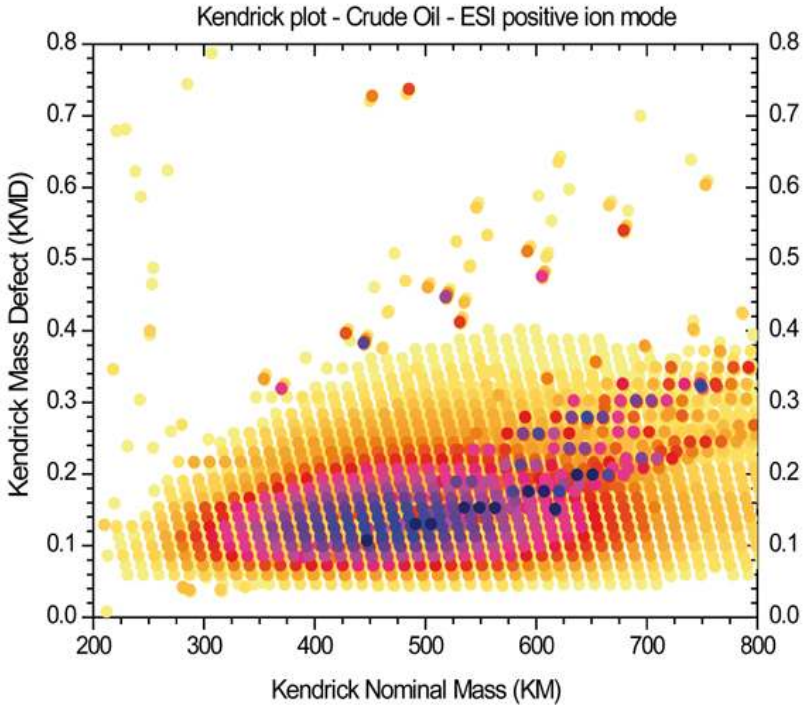


Figure 3.5: Kendrick plot of a crude oil. Members of the same 'class' (same numbers of N, S and O atoms) are grouped together along the y-axis, while members of the same CH_2 group are grouped along the x-axis (Wu *et al.*, 2004).

Once identifications have been made, the experimental m/z values are set to the calculated masses. All unidentified peaks are discarded (COMPOSER Software User Manual, Version 1.0, Sierra Analytics).

3.2.3 Dimensionality Reduction

Once the entire dataset has been calibrated and the individual peaks have been identified, the spectra are still of very high dimensionality to the order of approximately $2 \times 5{,}000$. As different crude oils vary slightly in terms of elemental composition, not all identified components are present in each of the samples, thus data for each crude oil sample requires specification of mass and intensity value for each peak.

Most algorithms will struggle with the dimensionality of the data, which will significantly increase computational expense. In addition, the bulk of the data for each sample is common for all samples and will thus hold very little informative value for the modelling process. Crucial features which hold the most information about the variability between samples are likely to be overshadowed by trivial information.

In order to avoid this one must find ways of filtering the data to only analyse that which holds vital information about each sample.

3.2.3.1 Peak Isolation

The simplest way of reducing the amount of data that is used for the modelling process is to isolate those peaks within each spectrum which are most likely to contain useful information about the samples. This task is far from trivial as there exists little information as to what aspects of the crude oil's mass spectrum are most closely correlated to the sample's physical properties.

For the majority of the research dataset A was used, which consists of the 712 peaks that were found to be present in every one of the set of 76 crude oil spectra. This dataset was chosen as it retains the majority of diversity in peaks found in the spectra while significantly reducing the dimensionality from the raw data. The raw spectra contain more than 10,000 data points while the peak detected spectra are still to the scale of 4,000 to 6,000 data points. The 712 common peaks encompass a range of different heteroatom classes and span the full m/z range of the original spectra.

Dataset B consists of the set of peaks that have the most variability from sample to sample over the full set of 76 crude oils. The way these peaks were selected was by determining the mean intensity for each of the 712 common peaks over the 76 spectra. The standard deviation of each of the crudes from this mean was then determined. The peaks with the largest standard deviation were saved into a new data array, dataset B. The dimensions of dataset B varied depending on the value of standard deviation chosen as the threshold and thus the number of peaks retained. For this project, dataset B was of the scale 76×50.

As the physical properties of the samples vary greatly from one another, dataset B aims to isolate those peaks which describe the differences between the samples, in order to identify changes in mass spectrum that may be correlated to changes in physical properties. Standard deviation of the intensity of a peak is chosen as a measure of deviation as it takes into account the behaviour of each peak across all of the 76 spectra. Unlike maximum or minimum deviation of any one peak from the mean, which may be influenced by experimental outliers, the standard deviation highlights those peaks which consistently change in intensity from spectrum to spectrum.

One could have chosen to focus on those peaks which, while remaining fairly constant in intensity across the full set of spectra, have exceptionally high or low intensities in only a few spectra. While this method would help isolate 'unusual' samples, it is also more sensitive to noise and experimental outliers than the method discussed previously. As mentioned in section 2.1.2, sample dilution has a large effect on intensities of spectra, therefore focussing on peaks which are much stronger or much weaker in certain spectra may merely be focussing on peaks which are dominant in very

light or very heavy crudes. Assuming that the intensities of all peaks within the spectra are affected comparably by sample dilution, the standard deviation of each peak across all samples should remain unaffected by intensity fluctuations caused by sample dilution.

Dataset C is the result of principal component analysis, which in contrast to dataset B enhances the similarities, or principal components, between the 76 spectra. Principal component analysis (PCA) identifies the components within the dataset which account for the largest variance between datasets. The total number of components created by PCA is equal to the total number of variables in the original dataset; however in favourable cases the majority of the variance of the dataset is described by only a handful of those components, known as principal components.

The method of PCA is discussed in more detail in the following section. The dimensions of dataset C were 76 × 50, where the first fifty principal components for each crude oil are chosen for further analysis.

3.2.3.2 Principal Component Analysis

A productive way of identifying those features of a large dataset that account for the greatest variability is using principal component analysis (Pearson, 1901; Hotelling, 1933; Jolliffe, 2002).

Principal component analysis reduces a large dataset by computing a set of artificial variables (principal components) that represent the largest amount of variance in the dataset. These components can then replace the original data in subsequent analyses

with minimal loss of crucial information, depending on the number of components chosen to be retained.

The entire dataset is linearly transformed into a new co-ordinate system. For a p-dimensional dataset there will be p new components. Each of these p components represents the maximum variance within the dataset that is not already described by another of the components and is entirely uncorrelated to any other component. The majority of the variance of the dataset is then represented by a total number of q components, the principal components, that is less than the p original variables. The difference between q and p is highly dependent on the nature of the data, but generally a case where $q \ll p$ is favourable.

Principal components are a weighted linear combination of the observed variables, where each variable's contribution to each component is

$$c_1 = b_{11}(x_1) + b_{12}(x_2) + \ldots + b_{1p}(x_p) \quad (3.3)$$

where c_1 is the first principal component, b_{1p} is the weight contribution of the p^{th} variable on the first principal component, also known as the PC loading, and x_p is the subject's score on the p^{th} variable.

Before computing the principal components of the data it is necessary to standardise the dataset by subtracting the mean of the data when analysing a covariance matrix, or dividing by the standard deviation when analysing a correlation matrix. Standardisation is essential in cases where the data points are variables which are quoted in different units, and thus variables with a large range of values may overshadow those with

smaller ranges. Subtracting the mean of the data centres the dataset around the origin of the original co-ordinate system.

Once standardised, the first principal component is chosen to be a straight line which passes through the centroid of the data $(0, 0, 0)$ and minimises the squared distance of each data point to that line. It also passes through or close to the two most extreme points in the co-ordinate system, thereby encompassing the greatest variance in the dataset.

The second principal component is orthogonal to the first principal component.

The axes of the co-ordinate system that encompasses the dataset are then rotated to make the first principal component the x-axis and the second the y-axis.

Given a two-dimensional dataset $Z(x, y)$, the covariance of the data is given by equation 3.4.

$$cov(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)} \tag{3.4}$$

where x_i is the i^{th} x-value of data matrix Z and $\bar{x}$ is the mean of x.

The covariance matrix for the two-dimensional case is then

$$cov = \begin{matrix} cov(x, x) & cov(x, y) \\ cov(y, x) & cov(y, y) \end{matrix} \tag{3.5}$$

and for the three-dimensional case the covariance matrix is given by

$$cov = \begin{matrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{matrix} \quad (3.6)$$

Next we compute the eigenvectors X_n and eigenvalues λ_n of the $n \times n$ covariance matrix C, so that

$$CX_n = \lambda_n X_n \quad (3.7)$$

Each eigenvalue represents the amount of variance described by the component it is matched with.

The eigenvectors are then rearranged in order of decreasing eigenvalues. The eigenvectors with the largest eigenvalues represent the principal components of the data. Typically, more than 80% of the total variance will be described by the first ten components, though this is strongly dependent on the type of dataset and its dimensionality.

PCA works by assigning noise and features which hold little information about the sample, known as minor factors (a term derived from the related field of factor analysis), to lower score components, which are likely to be discarded when picking PCs for further analysis. As noise is common to all samples, even if the signal-to-noise ratio is very high, it will account for little variance between samples and therefore will not be transformed onto a principal axis. The choice of how many principal components to retain becomes very important.

In this thesis, a 76 × 712 matrix was fed into the PCA. This resulted in 712 eigenvalues and a 712 × 712 eigenvector matrix, of which the first four accounted for more than 90% of the total variance (table 3.3).

Component	Eigenvalue	Percent Variance (%)
1	7.86974×10^6	51.6900
2	3.16465×10^6	20.7861
3	1.71406×10^6	11.2583
4	999848	6.56721
5	412477	2.70923
6	239539	1.57334
7	196326	1.28951
8	117003	0.768499
9	88542.2	0.581564
10	71172.4	0.467475

Table 3.3: Eigenvalues and percent variance for the first ten principal components calculated for the crude oil dataset.

Once the eigenvectors were rearranged in order of their eigenvalues, the variables' contributions to each principal component, i.e. their loadings, were computed.

The final output of the PCA is the factor scores, which show the amount of influence each component has on each sample. This new dataset, which takes the dimensions of $n \times q$ where n is the number of samples and q is the number of principal components, can be used as predictor variables in subsequent analyses. PCA results are listed in appendix B.

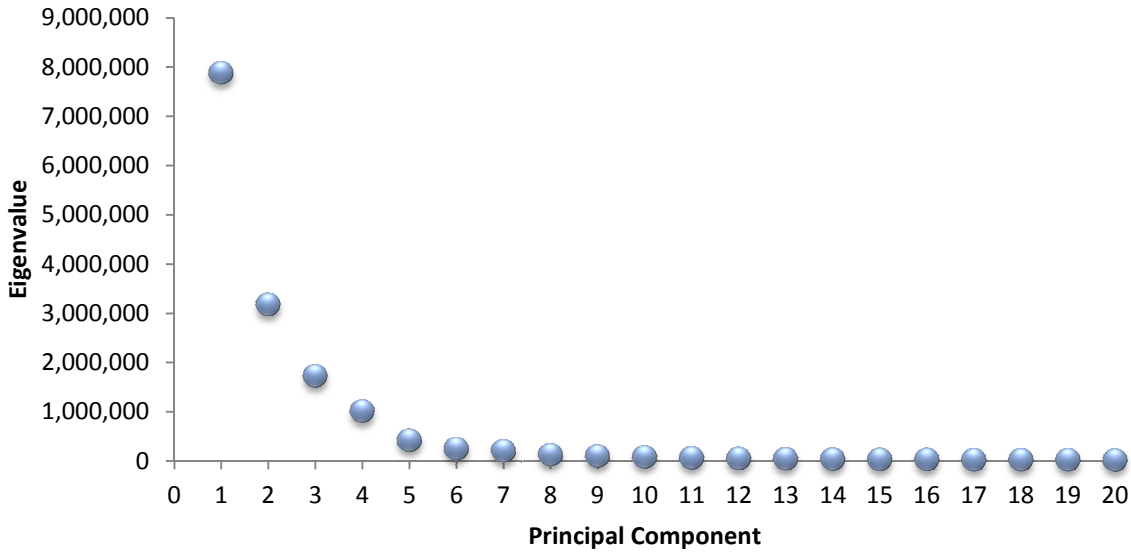


Figure 3.6: Scree plot showing that the first five principal components describe the vast majority of variance in this dataset.

One area of difficulty in interpreting PCA results is choosing a suitable selection process for principal components. Some researchers choose an eigenvalue threshold and discard all components with eigenvalues below that value. This method is known as the Kaiser criterion (Kaiser, 1960) or Kaiser-Guttman rule (Kaiser, 1970) and is generally not applicable in cases where there are a large number of variables, there exists a large q/p ratio (where q is the number of principal components that contain crucial information and p is the total number of variables) or the data are highly complex (Hakstian *et al.*, 1982). In this project, all three of these cases apply. Another commonly used method is to create a scree plot (Cattell, 1966). In a scree diagram, one plots the value of each eigenvalue against the number of the principal component it represents. The scree diagram should show a clear break point between the significant principal components and those that do not account for a lot of variance. One prerequisite for a successful scree test is that the data can be accurately described by only a few principal components and that the smallest principal component that is due to variance in the

data (a major factor) is significantly larger than the largest principal component that describes only noise (a minor factor) (Hakstian *et al.*, 1982). This is not always the case and most certainly is not true in this project (see figure 3.6). While lower score components may not describe large amounts of variance across the whole dataset they may still hold information about features that are of importance in only a few of the samples.

The PC loadings, which describe the amount that each component contributes to any one sample, are of particular importance with datasets such as this, where it is possible that a low score component which does not describe the majority of the data has high loadings for only a handful of samples.

The application of PCA to the crude oil data is described in more detail in chapter 7.

3.2.3.3 Factor Analysis

Factor analysis is a method used to determine quantitatively the individual factors such as peaks which contribute to a measured global effect such as a physical property of a sample. Similar to principal component analysis, it investigates the variance of a dataset, however it particularly focuses on the effect of communality, the variance caused by factors that are common to all variables.

The technique assumes that a linear combination of a number of factors gives rise to the final property.

$$x_{ij} = \sum_{k=1}^{p} a_{ik} f_{kj} \qquad \begin{matrix} i = 1, m \\ j = 1, n \end{matrix} \tag{3.8}$$

where x_{ij} is the i^{th} property of the j^{th} sample, a_{ik} is the concentration of the p individual elements of the i^{th} type and f_{kj} is the mass contribution per unit volume of the k^{th} element in the j^{th} sample.

Like PCA it also serves to decrease dimensionality of data by reducing the number of m recorded variables (in this case peaks) to the p elements which contribute to the physical properties of interest.

As can be seen from equation 3.8, the relationship between the individual elements and the physical property they give rise to is assumed to be linear.

Starting with matrix X, containing information on the m physical properties of n samples, and matrix F which contains the contributions of the p factors in each of the n samples, matrix A, the values of each of the p causative factors for the m properties, is determined.

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ \vdots & \vdots & & \vdots \\ a_{p1} & a_{p2} & \cdots & a_{pm} \end{pmatrix} \begin{pmatrix} f_{11} & \cdots & f_{1p} \\ f_{21} & \cdots & f_{2p} \\ \vdots & \ddots & \vdots \\ f_{n1} & \cdots & f_{np} \end{pmatrix} \tag{3.9}$$

As with principal component analysis, one starts by determining the correlation or covariance matrix of X and then performing diagonalisation to obtain the eigenvalues λ_n and eigenvectors X_n as described by equation (3.7).

The next step is to determine the number of factors p with which to perform subsequent analysis. This is once again done by inspection of the eigenvalues. As with

PCA, one can create a scree plot or apply the Kaiser-Guttman rule in order to identify those factors which can be deemed significant.

3.3 Data

3.3.1 Data Reliability

In order to verify the validity and reliability of the data, repeat measurements were taken of sample 16 (Rabi Export Blend) at four different times over the space of four months. The recordings were taken for the same stock sample and the instrument parameters were kept constant for each of the recordings. Instrument parameters chosen for the statistical analysis were the same as for all sample analyses and are as quoted in section 2.2.2.

The variance for each of the peaks across the four duplicates was determined and compared to the variance for each of those peaks across all samples. The average standard deviation of peaks from run to run is 9.5% of the mean intensity, while the standard deviation of peaks from sample to sample has an average of 31.3% of the mean intensity.

Plots of the normalised intensities of the duplicate recordings against one another show fairly high correlation (figure 3.7). By comparison, a plot of the intensities of one sample against those of another sample, chosen at random from the complete sample pool of 76 crude oils, shows much lower correlation between peak intensities (figure 3.8).

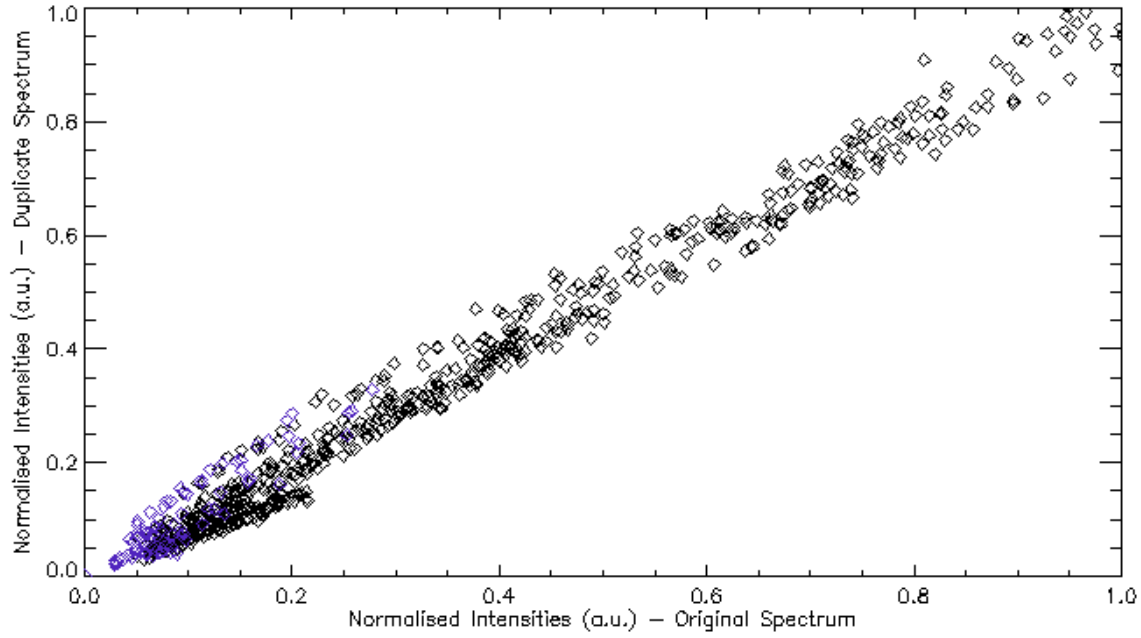


Figure 3.7: Plot of the normalised intensities of a duplicate spectrum vs. the intensities of the original spectrum. Points highlighted in blue denote peaks of masses below 350m/z.

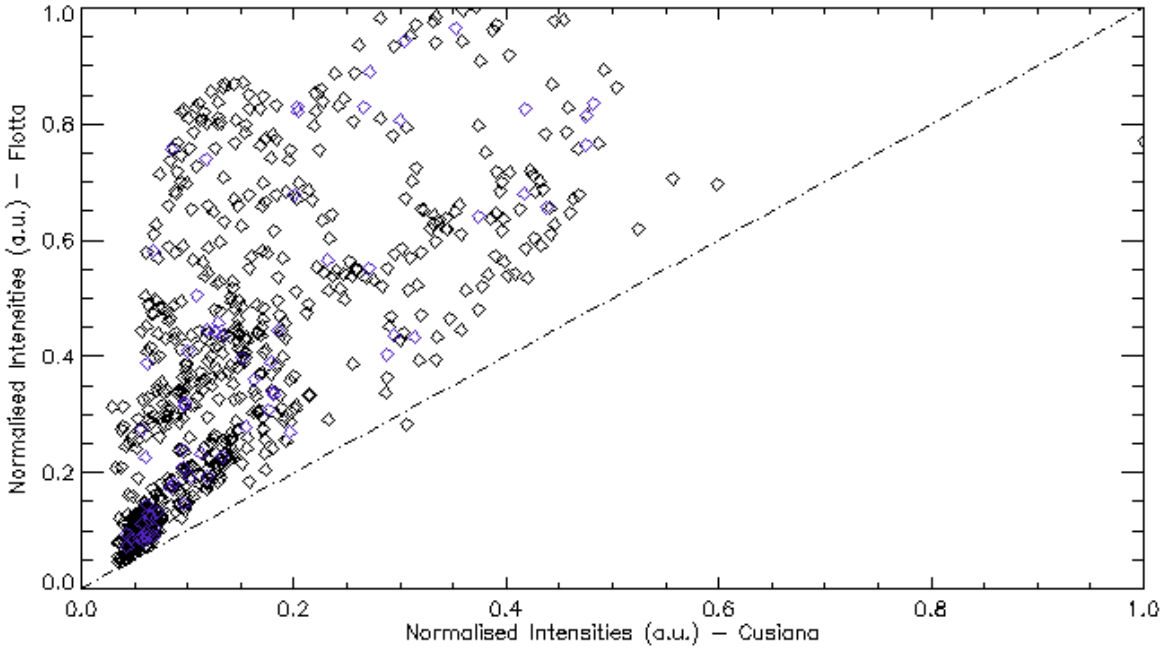


Figure 3.8: Normalised intensities of samples 13 (Cheleken) and 45 (Duri) plotted against each other. Blue points denote all peaks of m/z below 350.

The two samples plotted against one another in figure 3.8 show a distinct distribution that falls above the line of direct proportionality. The two crude oils being compared are Cheleken, a light crude oil of API gravity 35.58, and Duri, a heavy oil with API gravity 20.47. As is typical for heavy crude oils, the sample has a high abundance of high molecular weight species and therefore has high intensity peaks ranging up to 1,000m/z, while the light crude oil has very few high intensity peaks at masses above 700m/z. This 'spreading out' of the spectrum over a larger m/z range (see figure 3.9) causes the distribution observed in figure 3.8, which favours the left-hand side of the norm.

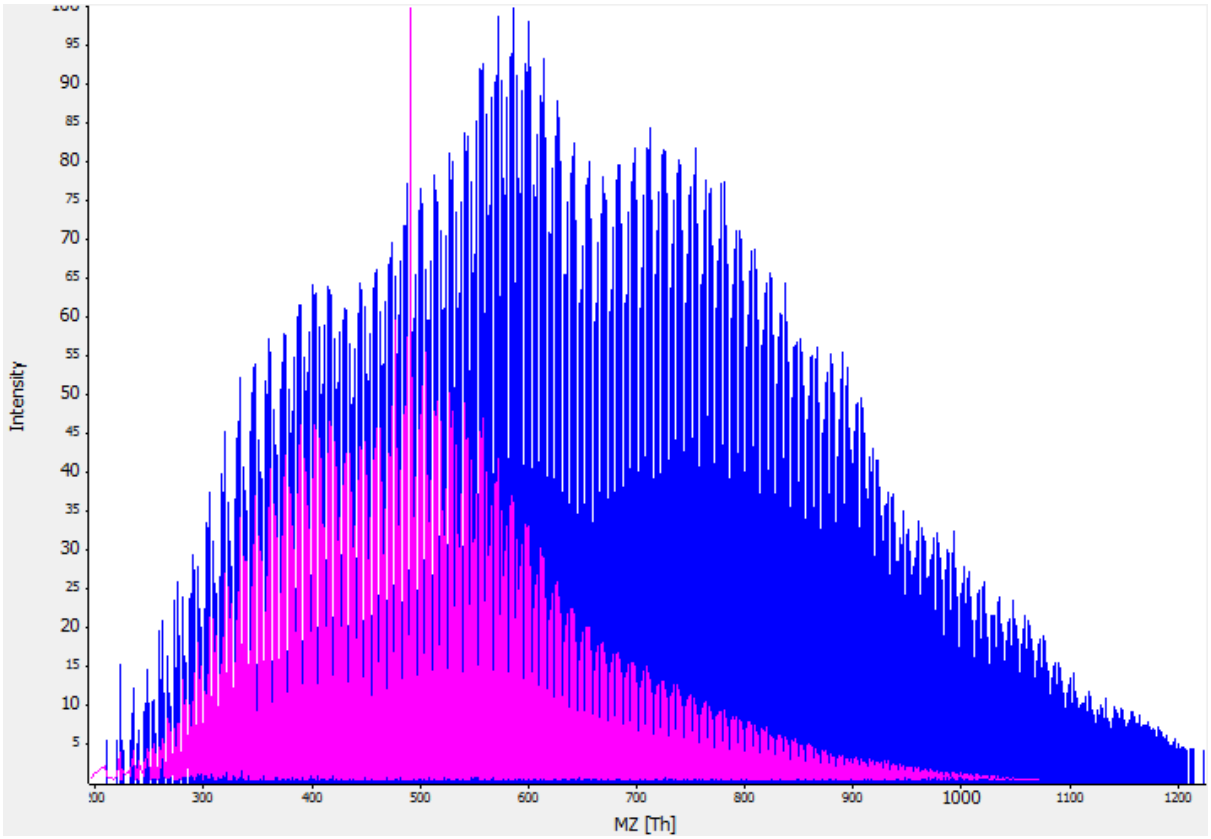


Figure 3.9: The normalised mass spectra of Cheleken (Pink) and Duri (Blue) overlaid. The difference in intensities and peak distribution between light and heavy crudes seen in figure 3.8 becomes evident.

References

Cattell R. B. (1966), The Scree Test for the Number of Factors, *Multivar. Behav. Res.,* 1(2), 245 – 276.

Hakstian A.R., Rogers W.T. and Cattell R.B. (1982), The behaviour of number-of-factors rules with simulated data, *Multivar. Behav. Res.*, 17, 193 – 219.

Hotelling H. (1933), Analysis of a Complex of Statistical Variables with Principal Components, *Journal of Educational Psychology,* 24, 417 – 441 and 498 – 520.

Hsu C. S., Hendrickson C.L., Rodgers R.P., Kenna A.M. and Marshall A.G. (2011), Petroleomics: advanced molecular probe for petroleum heavy ends, *J. Mass. Spectrom.*, 46, 337 – 343.

Jolliffe I.T. (2002), Principal Component Analysis, Second edition, *Springer-Verlag,* New York, NY.

Kaiser H.F. (1960), The application of electronic computers to factor analysis, *Educational and Psychological Measurement*, 20, 141 – 151.

Kaiser H.F. (1970), A second generation Little Jiffy, *Psychometrika*, 35, 401 – 415.

Kendrick E. (1963), A mass scale based on CH_2 = 14.0000 for high resolution mass spectrometry of organic compounds, *Anal. Chem.*, 35, 2146 – 2154.

Pearson K. (1901), On lines and planes of closest fit to systems of points in space, *Philosophical Magazine*, Series 6, 2(11), 559 – 572.

Thermo Fisher Scientific Inc., *www.thermo.com,* accessed June 2011.

Vahlencamp M., Zey T., Klomburg K. and Baessmann C. (2007), De Novo Formula Generation with "sub-ppm" Confidence using Compass OpenAccess and the micrOTOF, *Bruker Daltonik*, www.bdal.com.

Van Krevelen D. W. (1950), *Fuel*, 29, 269 – 284.

Wu Z., Rodgers R.P. and Marshall A.G. (2004), Two- and Three-Dimensional van Krevelen Diagrams: A Graphical Analysis Complementary to the Kendrick Mass Plot for Sorting Elemental Compositions of Complex Organic Mixtures Based on Ultrahigh-Resolution Broadband Fourier Transform Ion Cyclotron Resonance Mass Measurements, *Anal. Chem.,* 76(9), 2511 – 2516.

CHAPTER 4 – Introduction to Machine Learning

CHAPTER 4 introduces the concept of machine learning and discusses various techniques within the field of artificial intelligence. It includes a brief literature review and contains a description of the physical properties of the samples.

4.1 Introduction

Machine learning is the computational discipline of constructing algorithms which will identify patterns in complex datasets and using such algorithms to predict certain properties of previously unknown data (Michalski, Carbonell and Mitchell, 1983).

Execution of a learning algorithm is an iterative process, in which the modelling parameters are amended after each of a large number of cycles with the hope that the model will converge towards an ideal solution. Once the algorithm is deemed to have learned to recognise patterns in a training dataset, it can be used on a previously unknown dataset to predict its properties.

Machine learning can be used for classification or regression problems (Mitchell, 1997). In classification, the learning algorithm is merely required to determine whether a sample belongs to a given group by comparing the predicted output value to a target value and using this comparison as a means to classify the sample.

In a regression problem, the user aims to predict a physical quantity, making the learning task generally more complex in nature. Section 4.3 discusses a number of experiments, which represent both classification and regression problems.

4.2 Supervised and Unsupervised Learning

Learning algorithms are most commonly classified as supervised or unsupervised (Cartwright, 2008). In supervised learning, the training phase of the algorithm is monitored by comparison of a predicted value to a target value, determining the quality

of the fit. The algorithm requires both an input and an 'ideal solution' output value. The error between the desired and predicted output is then used to amend the training parameters in the next training cycle, aiming to converge toward an ideal solution. Support vector machines (Cortes and Vapnik, 1995) and back-propagating neural networks (Rumelhart *et al.*, 1986b) are examples of learning techniques that require supervision.

Unsupervised learning requires no prior knowledge of the data to be learned and uses the input data as a means to identify patterns within a dataset. Unsupervised learning techniques include self organising maps, in which an algorithm is used to compare a number of different sample datasets and group samples with similar characteristics together. The aim of the algorithm is to minimise the 'distance' between samples within a group and to maximise the 'distance' between samples in different groups (Kohonen, 1982), where 'distance' can be both a measure of dissimilarity or similarity between two samples in n-dimensional space and the physical or Euclidian distance between those samples' co-ordinates in a two-dimensional sample landscape. Such an algorithm generally involves mapping samples onto a different co-ordinate system, invariably of much lower dimension, in order to separate the groups spatially. The modelling phase then introduces a previously unseen sample and classifies it into one of the previously determined groups.

4.3 Machine Learning Techniques

4.3.1 Artificial Neural Networks

Artificial neural networks are based on an analogy with the processing of information by biological neurons in the human central nervous system (Rosenblatt, 1962). They are generally made up of a set of input neurons or nodes equal to the number of data points that receive the input information i and a layer of output nodes o, that provide the predicted output.

Processing in the neural network takes place in a number of hidden layers j, k, l,.., each containing a number of hidden nodes that represent the non-linear relationship between the input and output data.

In a standard multi-layer feed-forward network the input data y_i is first multiplied by randomly chosen weights w_{ij} and summed over all input neurons to create the input for the first hidden layer nodes x_j.

$$x_j = \sum_{i=1}^{n} y_i \, w_{ij} \tag{4.1}$$

The node input is processed using an activation function $F(x)$, such as a sigmoid function, to give the node output y_j.

$$y_j = \frac{1}{1+e^{-x_j}} \tag{4.2}$$

The output of each of the first hidden layer nodes is again multiplied by weights w_{jk} and summed, before being processed by another activation function in hidden layer 2. This

process is repeated for each of the hidden layers and the output layer. The output from the activation function in the final layer, y_o, is then compared to the desired output d_o to determine the network error $E_o = d_o - y_o$.

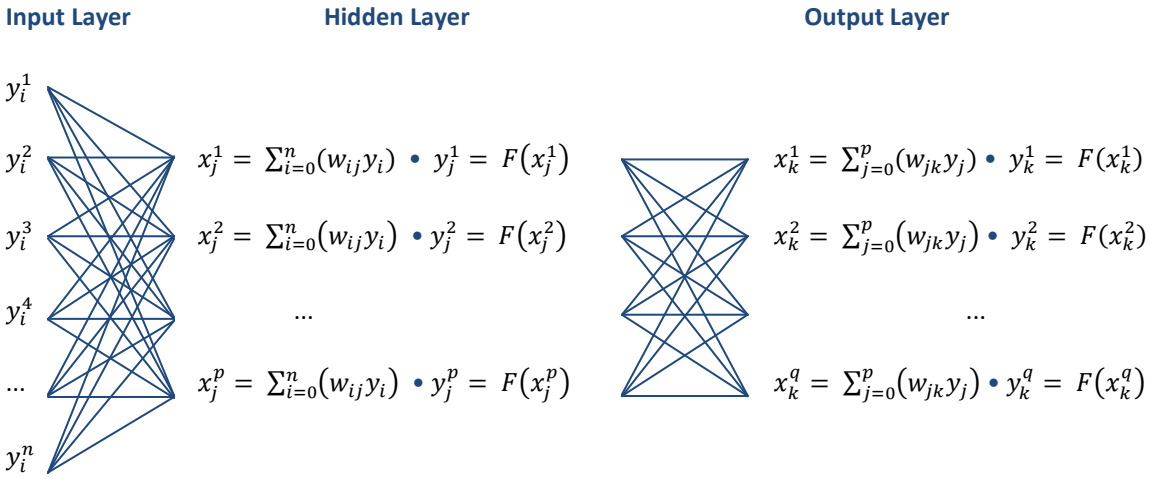


Figure 4.1: Schematic showing the layout of a basic feed-forward neural network with n input nodes, q output nodes and one hidden layer with p hidden nodes.

The algorithm learns by using back-propagation to amend the weights following each iteration of the training phase in order to minimise the error E_o (Rumelhart *et al.*, 1986a).

After each sample from the training set has been processed by the network (or in some algorithms after a single iteration through the complete training set, called an epoch), the error function of the output layer

$$\delta_o = y_o(1 - y_o)(d_o - y_o) \tag{4.3}$$

and each of the hidden layers is determined:

$$\delta_k = y_k(1 - y_k)\sum_{o=1}^{q}\delta_o w_{ko} \tag{4.4}$$

This error function is used to adjust the weights for the next cycle of the training algorithm.

$$w_{ij} = w_{ij} + \alpha y_i \delta_j \quad (4.5)$$

where α is the learning rate which should be set to decay with increasing numbers of iterations. The learning rate determines the amount by which the weights are adjusted in each training cycle. If a fixed learning rate is chosen, there is a risk that the learning rate is either too small, meaning that the network will take a very large number of cycles to converge or may not converge at all, or too big, leading to good convergence initially, but not allowing the network to settle into a global minimum, as the weights are adjusted too harshly, even if the network output is approaching the target value. An ideal learning rate starts off large and decays toward zero as a function of the number of iterations the network has performed.

Once the weights have been adjusted, the next loop begins and the training set samples are once again processed by the algorithm. As training proceeds, the deviation between the predicted output y_o and the desired output d_o should tend to zero, although limitations due to noise and non-ideality of samples prevent the error from ever reaching zero in real-world applications. At the end of each training epoch, the network is introduced to the validation set, a set of previously unseen data, which are modelled using the current network parameters. The validation set is used to verify that the network is still learning sufficiently and has not started to learn individual samples, known as overfitting, rather than trends in the overall dataset.

Overfitting has occurred if the output error E_o for the training set is converging rapidly towards zero, while the output error for the validation set is remaining constant or increasing after every epoch. A well performing network has output errors for both the training and the validation set that are approximately equal and steadily decreasing towards zero.

The validation set is also used to test whether learning can be halted should an acceptable level of fit have been achieved. An appropriate value of E_o to signify successful modelling is dependent on the data and should be significantly smaller than the random fluctuations between samples. It should also fall within 10% of the starting error ($E_o(t_{final}) \leq 0.1\ E_o(t_0)$)) or should typically be less than 10% of the desired output and is highly dependent on the level of noise in the data.

A break clause is incorporated in the validation section of the algorithm, allowing the program to exit the training cycle should the network be failing to converge or have converged sufficiently before the maximum number of iterations has been completed.

The final step in a neural network is the test step. Another set of previously unseen data are introduced to the network and the output compared to the desired output in order to verify that the network has been trained successfully. Again, the performance of the network is determined by the size of the error between the actual output and the desired output. In an ideal case, this should be equal to or approximately equal to zero. Depending on the complexity of the data, values that are larger than zero may be considered acceptable, provided that the value of E_0 following training is significantly

lower than it was at the beginning of the training cycle (figure 4.2) and is significantly lower than the sample-to-sample variability of the data.

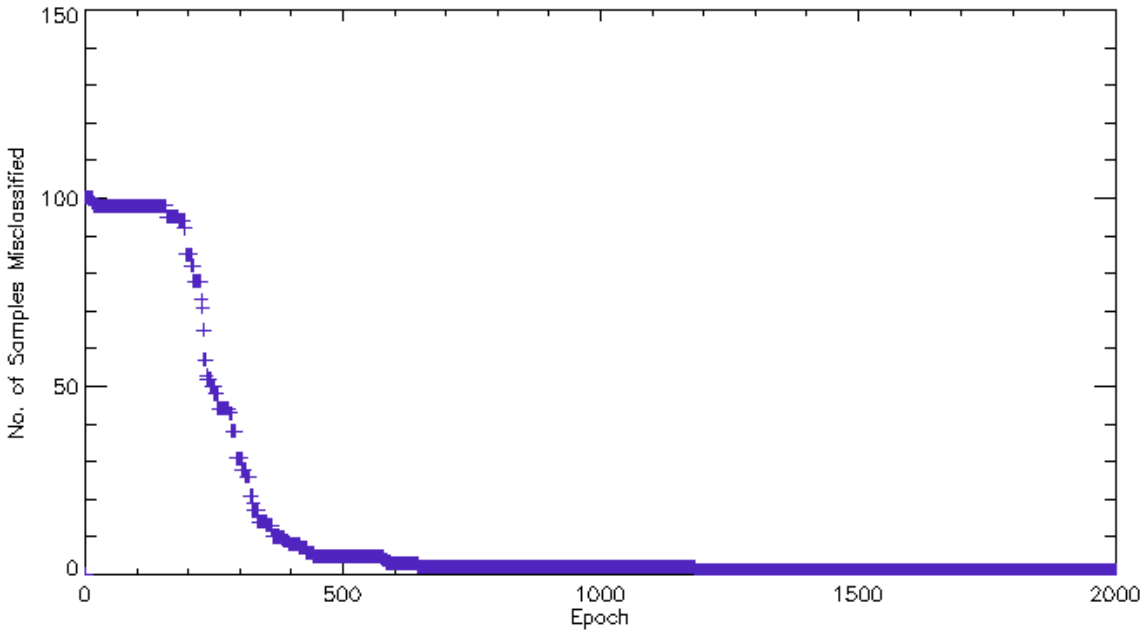


Figure 4.2: The rate of misclassification of samples in the iris dataset as a function of the number of iterations t.

4.3.2 Self Organising Maps

Self organising maps are an example of unsupervised learning techniques mainly used for classification problems, first proposed by Teuvo Kohonen (Kohonen, 1982), and are thus commonly referred to as Kohonen networks. They work by rearranging a multidimensional surface of samples into a one- or two-dimensional feature space, in which regions with the most similarities are placed adjacent to one another. The resulting map is a surface that represents the clustering of like samples, with those that have the largest differences separated by the largest amount of space on the map (i.e. diagonal corners).

The untrained self organising map consists of a $n \times n$ matrix of nodes with map co-ordinates (x_i, y_j) ... (x_n, y_n) where i = 1, 2 ... n and j = 1, 2 ... n and respective weight vector w_1, w_2 ... w_m.

Each new input into the network is represented by a vector v_1, v_2, v_3... v_m.

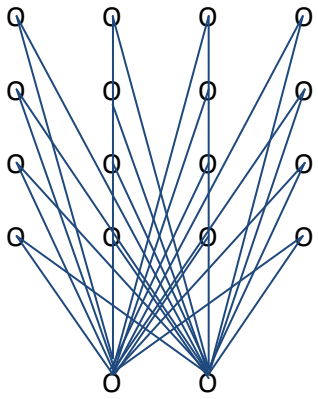


Figure 4.3: A schematic of a 4 × 4 node Kohonen map with 2 input nodes.

The input vector for each sample is compared with the weights vector at each node in turn and the Euclidean distance between the input vector and the respective weights vector is determined. The sample is then assigned to the node with the smallest Euclidean distance d.

$$d = \sqrt{\sum_{i=0}^{m} (v_i - w_i)^2} \tag{4.6}$$

Once an input vector has been assigned to a node, called the best matching unit or BMU, the members of the local neighbourhood for that node are determined. The members of the local neighbourhood are those nodes which are located in close proximity of the BMU as determined by the neighbourhood function. There exist a number of different methods for determining the size of the neighbourhood, including

Gaussian, linear and Mexican hat decay functions (see figure 4.4). A fourth commonly used function is exponential decay:

$$\sigma_t = \sigma_0 \, e^{-\frac{t}{\lambda_n}} \qquad t = 0, 1, 2, 3, 4... \tag{4.7}$$

where σ_t and σ_0 are the widths of the neighbourhood in iteration t and 0 respectively, λ_n is a time constant and t is the current iteration in the training cycle.

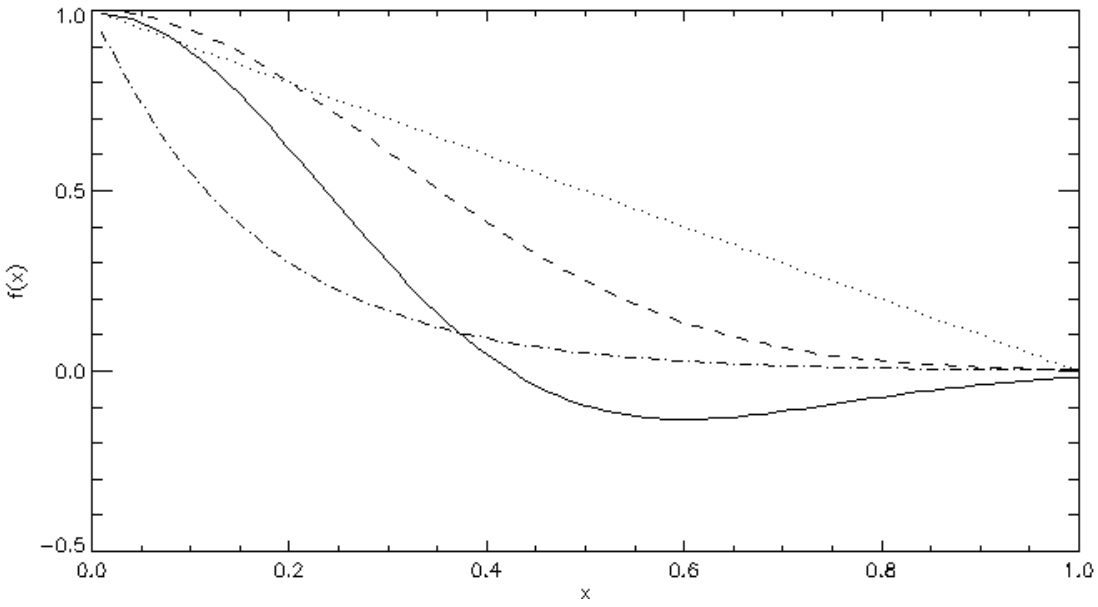


Figure 4.4: Four different decay functions commonly used to reduce the size of the neighbourhood in a SOM as a function of distance or adjust the weights of a learning algorithm as a function of time. The four functions are a) Linear (....), b) Gaussian (-----), c) Exponential (-.-.-.-) and d) Mexican hat (____).

During every loop of the training cycle all nodes within the best matching unit's neighbourhood have their weights adjusted according to equation 4.8.

$$w_{t+1} = w_t + \Theta_t \gamma_t (v_t - w_t) \tag{4.8}$$

where w_{t+1} is the weight for any given node in the next iteration, w_t is the weight in the current iteration, v_t is the input vector for the current iteration and γ_t is the

learning rate of the network in the current iteration, which decays with time according to equation 4.9.

$$\gamma_t = \gamma_0 \, e^{-\frac{t}{\lambda_l}} \tag{4.9}$$

Θ_t is an adjustment factor which describes the amount by which a node's weight vector is adjusted and decays as a function of the distance between that node and the BMU. This ensures that nodes near the edge of the neighbourhood are affected less than those near the centre. As with the size of the neighbourhood, the weights adjustment factor can take the form of any decay function (figure 4.4) but is most commonly defined by the two-dimensional Gaussian distribution function (see figure 4.5):

$$F(x,y) = Ae^{(-\frac{(x-\bar{x})^2}{2\sigma x^2}+\frac{(y-\bar{y})^2}{2\sigma y^2})} \tag{4.10}$$

where A is the amplitude, $\bar{x}$ is the mean of the vector x, $\bar{y}$ is the mean of the vector y and σ_x and σ_y are the standard deviations of x and y respectively.

The application of the two-dimensional Gaussian function (equation 4.10) to the adjustment factor leads to equation 4.11:

$$\Theta_t = e^{(-\frac{d^2}{2\sigma_t{}^2})} \tag{4.11}$$

where d is the distance between a given node and the BMU in cycle t and σ_t is the width of the neighbourhood in the present cycle as defined by equation 4.7.

Once the weights of each of the nodes in the neighbourhood have been adjusted, the next training cycle begins and a new set of input vectors is introduced.

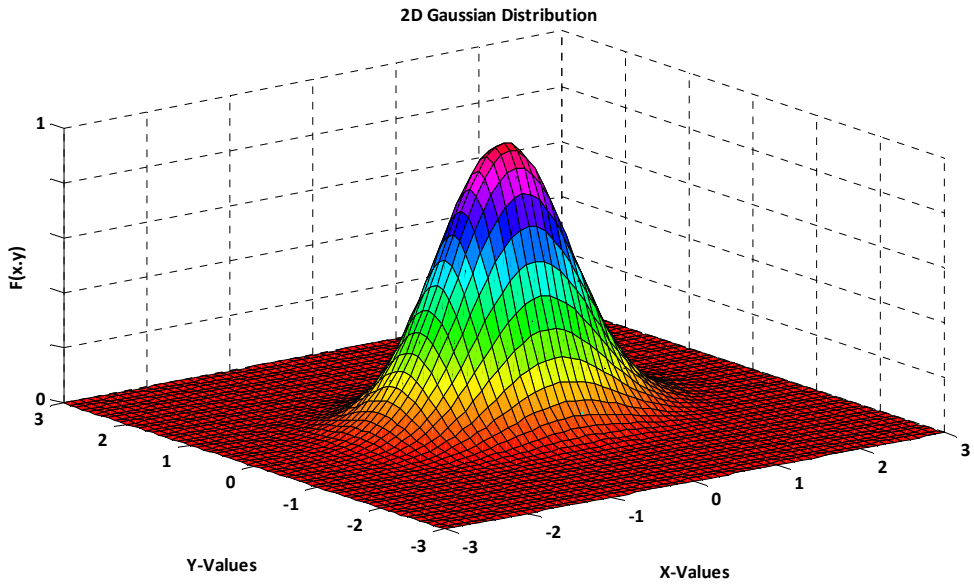


Figure 4.5: Two-dimensional Gaussian distribution.

The training continues until a given number of iterations has been completed and both the learning rate and the size of the neighbourhood radius have decayed to a value close to zero. The training phase is then followed by the mapping phase in which an unknown sample is shown to the network and classified using the trained weights.

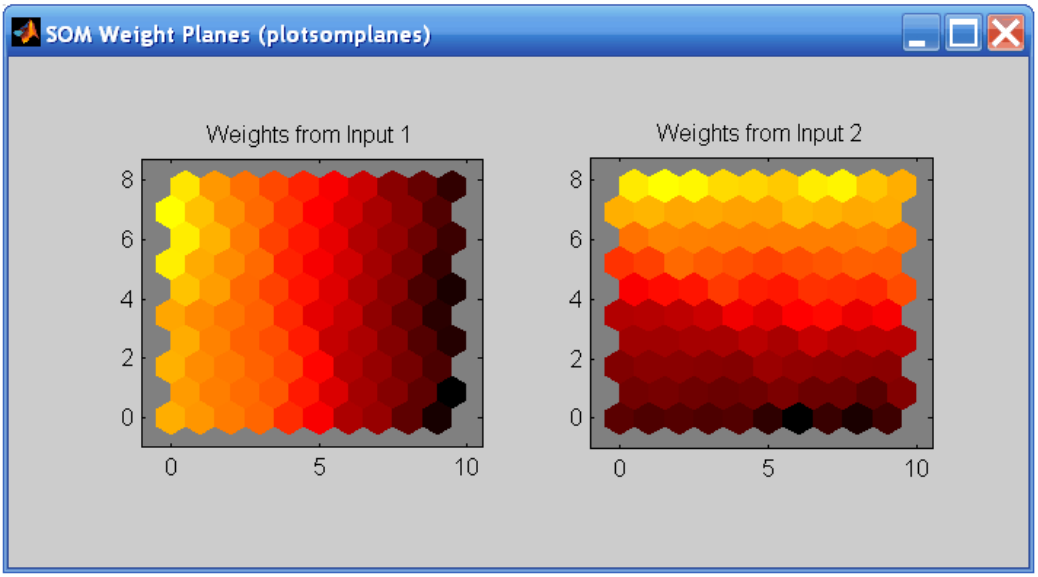


Figure 4.6: A typical graphical representation of a self organising map in the form of a unified distance or U-matrix (www.mathworks.com).

4.3.3 Simulated Annealing

Simulated annealing, a method first proposed by Kirkpatrick, Gelatt and Vecchi in 1983, is a technique for finding a good approximation to the global optimum of a problem in as short a time as possible. Such techniques are particularly useful in cases where the search space is large and varied, presenting a large number of local minima. The name simulated annealing is derived from the field of metallurgy (Kirkpatrick, 1983) where the process of annealing, like the computational analogue, involves the temporary worsening of the solution in order to help settle into a new, better solution, i.e. the heating of a metal to break the atoms free from a local minimum in internal energy to allow them to settle into a lower energetic state.

Given a search space s, the algorithm chooses a neighbouring state s' with a probability which decreases as a function of time t, as shown in equation 4.12.

$$P(s, s', t) = \mathrm{e}^{(\frac{s-s'}{t})} \tag{4.12}$$

Further theory as well as the application of simulated annealing in this project is discussed in section 6.2.

4.3.4 Genetic Algorithms

Genetic algorithms are machine learning algorithms that are based on the survival of the fittest principle and were first discussed in papers on computational modelling of genetic evolution by Barricelli (1957) and Fraser (1957).

Genetic algorithms are an example of supervised learning in which an optimum solution is chosen from a large number of possible solutions by means of evolution.

In each of a number of cycles, several candidates or strings from the starting population are chosen to create a new population by means of mutation or crossover with another string. There exist a number of different selection processes for choosing the strings that are made to reproduce; a common algorithm is stochastic remainder.

In stochastic remainder, the fitness of each member of the population is determined, and the standard fitness for that population is scaled to an average of one. Fitness functions can vary depending on the particular problem and the choice of function is largely arbitrary, as there is no such thing as a perfect function. The most basic fitness function is

$$F(y) = \frac{1}{\sqrt{1+(d_o - y_o)^2}} \tag{4.12}$$

where d_o is the desired outcome and y_o is the value predicted by the algorithm. Using this formula, an ideal solution has a fitness of one.

Each string is then copied into the new population a number of times equal to the integer part of its fitness value. This ensures that the fittest strings are strongly represented in the reproduction subset. The remainder of those fitnesses, as well as those strings with below average fitness are entered into a random selection process, which chooses the remaining strings needed to fill up the reproduction population or mating pool. This ensures that even substandard strings have a non-zero chance of contributing towards the next generation.

The algorithm continues for either a set number of cycles or until a satisfactory level of fitness has been met.

In a population of very similar individuals, crossover reproduction can very quickly lead to saturation of the population and will halt any progress (Koza, 1992). This is because two near identical parents will create a child that is not much different genetically from its parents. In such situations one must favour mutation over crossover reproduction, so the balance between crossover and mutation often changes as the algorithm proceeds.

4.4 Data

The choice of data used for machine learning is as important as the learning algorithm itself.

Machine learning is most principally a method of modelling and understanding complex data. Whether the data need to be classified or analysed, the importance is that the data are both accurate and representative of the problem in question.

A trend cannot be learned or modelled if it is not evident within the chosen dataset. Furthermore, if a bias exists within the dataset, then the resultant model is likely to be biased as well. Take for example a dataset that contains biological samples used to diagnose a certain type of cancer, in which the rate of illness of commonly screened patients is high (e.g. breast cancer in female patients over 30). If the training set contains samples which do not follow this natural distribution (for example if the dataset contains samples of patients that are not usually at risk, such as men or young

women) then training will be performed on a non-representative dataset. If the dataset contains a significantly higher percentage of 'good' data over 'bad' data, as it would in the example mentioned here, then in an effort to reduce the overall error, there exists a bias toward 'good' classifications and thus an increase in false negatives (Bishop, 1995). Subsequent diagnoses of patients who may be genuinely at risk, would be biased towards a healthy diagnosis and thus increase the likelihood of misdiagnosis, which could prove detrimental.

As the complete dataset is subdivided into training, test and validation sets, there needs to be both a sufficient amount of data to populate each of the three subsets and the data must be subdivided homogeneously, so that no biases exist in any of the three datasets. Taking the iris data – described in detail in the following section – for example, which consists of 50 samples each of three categories, if the subdivision were to be performed so that the training set consists of 50 samples which are each type A, the validation set contains a further 50 samples which are each type B and the test set consists of the remaining 50 type C samples, then the algorithm would find it impossible to learn an otherwise easy trend between the experimental data and the iris classification.

As the size of the dataset is most crucial for the training phase, the training set is typically comprised of at least 90% of the total dataset, with the remaining 10% being allocated to the test and validation sets. Algorithms are typically applied a number of times to varying combinations of these subsets and the resulting values for quality of fit are averaged, in order to ensure unbiased sampling.

If supervised learning is being performed, then the choice of output data is equally important, as there must exist an actual correlation between input and output data, as otherwise it is impossible to create a mathematical relationship between the two. Imagine an input dataset which contains information regarding the day of the week and that the output data contains the local weather corresponding to those days. These two datasets are completely uncorrelated and a trend between the two would be impossible to detect and thus learn.

4.4.1 Iris Data

Iris data, as mentioned previously, is data that lends itself to basic analysis using machine learning algorithms, due to its simplicity and therefore relative ease of being learnt. The iris dataset (Fisher, 1936) used in this research comprised 150 samples, which were equally divided into three classes describing the species of iris flower: *Setosa*, *Virginica* and *Versicolor*. There were four input variables per sample: sepal width and length and petal width and length.

The output was classified as one of three classes using binary notation: [0,0,1], [0,1,0] and [1,0,0] respectively.

Iris data is ideally used for classification problems, as discussed in the introduction to this chapter.

The complete set of data can be found in appendix B.I.

4.4.2 Crude Oil Data

The sample set for this research consists of 76 industry standard crude oils. They span a range of different physical properties (see chapter 2.1.1) and represent the full diversity of known crude oils (American Petroleum Institute).

Each of the set of 76 crude oils has been intensively analysed in a laboratory and a complete crude assay has been carried out. The data gathered in these assays was used as the desired output that the mathematical model was designed to predict. For the majority of this research the modelled output consisted only of the four 'cargo properties' of each of the crude oils, which includes the API gravity, pour point temperature in Kelvin, sulphur content and total acid number (TAN) of the crude oils. The complete cargo property data can be found in appendix B.II.ii.

The input data for the modelling process is gathered using high magnetic field Fourier transform ion cyclotron resonance mass spectrometry. NMR and near infrared spectroscopy datasets of the same samples are also available.

Mass spectrometry data is in the format mass/charge by intensity in arbitrary units. Raw mass spectra are highly dimensional with in excess of 10,000 data points in each raw spectrum.

These data were further processed using peak identification software which used a peak picking algorithm to determine the individual peaks in each spectrum and then used a database of chemical components commonly found in crude oils to identify the

peaks, reducing the dimensionality of the data to approximately 2 × 5,000. The software is described in more detail in section 3.2.2.1.

The data was further reduced in size using a number of different methods.

The three principal datasets used for the modelling process were those of a peak list that was found to be common between each of the 76 samples of dimensions 76 × 712, the 50 most variable peaks within that dataset and its first 50 principal components, further discussed in chapter 3.2.3.1.

4.5 State of the Art

Machine learning has been widely used for problems of both regression and classification of scientific processes in research as well as many day-to-day applications in a host of different industries, from commercial and financial to security and health sectors.

The most common commercial applications of machine learning are in the area of consumer technologies and electronics such as smartphones, robotics, interactive gaming and online search engines (Joachims and Radlinski, 2007).

Common financial applications include stock market predictions and oil price forecasts using neural networks or genetic algorithms applied to historical data to identify trends in market fluctuations (Azoff, 1994; Lo and Mackinlay, 2002).

National security relies heavily on the application of machine learning in areas such as CCTV monitoring, where face recognition algorithms help identify criminal suspects

(Darker *et al.*, 2009) and telephone and electronic mail correspondence monitoring (Hwang and Lee, 2004), where artificial intelligence is used for speech and context recognition. Furthermore, electronic nose technology which is commonly used at airports and other areas of high risk to national security requires high-throughput intelligent computational methods to analyse a host of chemical data collected. Machine learning is commonly applied in cases where mere database matching of chemical samples proves insufficient (Dutta and Dutta, 2006).

In the health industry, machine learning is used to model drug responses prior to carrying out clinical trials and can be applied to drug development (Murphy, 2011), a field known as metabonomics (Nicholson *et al.*, 1999), as well as biosurveillance (Zeng *et al.*, 2011) and disease control, monitoring and diagnosis (Sajda, 2006). Machine learning can further help to model chemosensitivity of individual patients and thus customise treatment, thereby ensuring the highest possible success rates (Ma *et al.*, 2006). The discipline of analysing such biological samples with the main aim of disease control and diagnostics is known as proteomics (Kahn, 1995) and is a popular area of bioinformatics.

Weather forecasts and by extension flight planning and air traffic management, which depend heavily on the prediction of weather systems that may pose risks to aviation, commonly use machine learning to help identify patterns and elucidate vital predictor variables from a host of data gathered from satellites, weather stations and historical case studies (Gasiewski *et al.*, 1996; Williams and Abernethy, 2008; Wolfson *et al.* 2008).

Migration patterns of birds and general animal behaviour are commonly used as indicators for changes in global climate and are thus closely monitored. A vast amount of geolocator and microchip tracking data is analysed and modelled using machine learning techniques (Guildford *et al.*, 2009) in order to better understand ocean health and other ecological systems.

Regression algorithms are commonly applied in the fields of engineering and material science, where algorithms are used to predict properties of complex alloys in metallurgy (Bhadeshia, 1999).

A host of other applications of machine learning exist that affect our everyday lives, from using handwriting recognition to sort our mail to intelligently adjusting a vehicle's suspension according to changes in terrain (Cao *et al.*, 2008).

While all of the above disciplines aid the development of ever faster more efficient learning algorithms, the complexity of data used in the field of petroleomics separates this application from most others. The related field of proteomics however deals with similarly complex data and computational analysis has been researched to a great extent. It stands to reason that computational methods that work well in the field of proteomics are likely to be successful for petroleomics applications. Both fields commonly deal with mass spectrometry data (Marchiori *et al.*, 2006), and while the field of proteomics is more commonly associated with classification rather than regression algorithms, techniques are nevertheless likely to be transferable.

Research by Barla *et al*. (2007) details the application of machine learning techniques to a set of mass spectral proteomics data, discussing in detail the data preparation, pre-

processing, model selection and performance evaluation of machine learning techniques such as support vector machines.

In the field of petroleomics, crude oils have successfully been classified using gas chromatographic data by Clark and Jurs (1979) and more recently, support vector machines and artificial neural networks have been applied to near infrared spectroscopy data of gasoline products and crude oils by Balabin and Lomakina (2011).

4.6 Discussion

The state of the art suggests that machine learning techniques should be successful in modelling the complex data of crude oils gathered by mass spectrometry as they have enjoyed significant success in many other areas of complex data analysis.

A number of machine learning techniques have been discussed in this chapter, each of which is likely to be useful for the analysis of the crude oils' mass spectral data. While self organising maps are best used for classification problems and could be used for spatially separating crude oils into different categories, the other three methods are likely to be successful at modelling the crude oils in terms of their physical properties.

References

Azoff E.M. (1994), Neural Network Time Series Forecasting of Financial Markets, *John Wiley and Sons Ltd*, New York, NY.

Balabin R.M. and Lomakina E.I. (2011), Support vector machine regression (SVR/LS-SVM) – an alternative to neural networks (ANN) for analytical chemistry? Comparison of nonlinear methods on near infrared (NIR) spectroscopy data, *Analyst*, 136, 1703 – 1712.

Barla A., Jurman G., Riccadonna S., Merler S., Chierici M and Furlanello C. (2008), Machine learning methods for predictive proteomics, *Briefings in Bioinformatics*, 9(2), 119 – 128.

Barricelli N.A. (1957), Symbiogenetic evolution processes realized by artificial methods, *Methodos*, 143 – 182.

Bhadeshia, H.K.D.H. (1999), Neural Networks in Materials Science, *ISIJ International*, 39(10), 966 – 979.

Bishop C. (1995), Neural Networks for Pattern Recognition, *Oxford University Press,* Oxford, UK.

Cao J., Liu H., Li P. and Brown D.J. (2008), State of the Art in Vehicle Active Suspension Adaptive Control Systems Based on Intelligent Methodologies, *IEEE Transactions on Intelligent Transportation Systems*, 9(3), 392 – 405.

Cartwright H.M. (2008), Using Artificial Intelligence in Chemistry and Biology: A Practical Guide, *CRC Press*, London, UK.

Clark H.A. and Jurs P.C. (1979), Classification of Crude Oil Gas Chromatograms by Pattern Recognition Techniques, *Anal. Chem.*, 51(6), 616 – 623.

Cortes C. and Vapnik V. (1995), Support-Vector Networks, *Machine Learning*, 20, 273 – 297.

Darker I.T., Kuo P., Yang M.Y., Blechko A., Grecos C., Makris D., Nebel J.-C. and Gale A.G. (2009), Automation of the CCTV-mediated detection of individuals illegally carrying firearms: combining psychological and technological approaches, *SPIE Defense Security and Sensing Conference*, Orlando, FL.

Dutta R. and Dutta R. (2006), Intelligent Bayes Classifier (IBC) for ENT infection classification in hospital environment, *Biomedical Engineering Online,* 5, 65.

Fisher R.A. (1936), The Use of Multiple Measurements in Taxonomic Problems, *Annals of Eugenics,* 7, 179 – 188.

Fraser A. (1957), Simulation of genetic systems by automatic digital computers. I. Introduction, *Aust. J. Biol. Sci.,* 10, 484 – 491.

Gasiewski A.J., Showman G.A., and Skofronick G.M. (1996), Application of Neural Nets to Rain Rate Retrieval from Simulated Multichannel Passive Microwave Imagery, *Proceedings of Geoscience and Remote Sensing Symposium, Remote Sensing for a Sustainable Future, IGARSS '96*, 3, 1688 – 1691.

Guildford T., Meade J., Willis J., Phillips R.A., Boyle D., Collett M., Freeman R. and Perrins C.M. (2008), Migration and stopover in a small pelagic seabird, the Manx shearwater Puffinus puffinus: insights from machine learning, *Proc. R. Soc. B.*, 1, 1577.

Hwang B. and Lee B. (2004), An efficient e-mail monitoring system for detecting proprietary information outflow using broad concept learning, *Lecture Notes in Computer Science*, 3002, 72 – 78.

Joachims T. and Radlinski F. (2007), Search Engines that Learn from Implicit Feedback, *IEEE Computer*, 40(8), 34 – 40.

Kahn P. (1995), From genome to proteome: looking at a cell's proteins, *Science*, 279, 396-370.

Kirkpatrick S., Gelatt C. D. and Vecchi M. P. (1983), Optimization by simulated annealing, *Science*, 220, 671 – 680.

Kohonen T. (1982), Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, 43, 59 – 69.

Koza J. R. (1992), Genetic Programming: On the Programming of Computers by Means of Natural Selection, *MIT Press*, Cambridge, MA.

Lo A.W. and Mackinlay A.C. (2002), A Non-Random Walk Down Wall Street, 5th Ed., *Princeton University Press*, Princeton, NJ.

Ma Y., Ding Z., Qian Y., Shi X., Castranova V., Harner E.J. and Guo L. (2006), Predicting Cancer Drug Response by Proteomic Profiling, *Clin. Cancer Res.,* 12, 4583 – 4589.

Marchiori E., Jimenez C., West-Nielsen M. *et al.* (2006), Robust SVM-based biomarker selection with noisy mass spectrometric proteomic data, *Applications of Evolutionary Computing, EvoBIO: Evolutionary Computation and Machine Learning in Bioinformatics*, LNCS, 3907, 79 – 90, *Springer*, Berlin, Germany.

Michalski R.S., Carbonell J.G. and Mitchell T.M. (1983), Machine Learning: An Artificial Intelligence Approach, *Tioga Publishing Company*, Wellsboro, PA.

Mitchell T.M. (1997), Machine Learning, *McGraw Hill*, New York, NY.

Murphy R.F. (2011), An active role for machine learning in drug development, *Nature Chemical Biology*, 7, 327 – 330.

Nicholson J.K., Lindon J.C. and Holmes E. (1999), Metabonomics: understanding the metabolic responses of living systems to pathophysiological stimuli via multivariate statistical analysis of biological NMR spectroscopic data, *Xenobiotica,* 29(11), 1181–1189.

Rosenblatt F. (1962), The Principles of Neurodynamics, *Spartan Books*, New York, NY.

Rumelhart D.E., Hinton G.E. and Williams R.J. (1986a), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations, 318 – 362, *MIT Press*, Cambridge, MA.

Rumelhart D.E., Hinton G.E. and Williams R.J. (1986b), Learning representations by back-propagating errors, *Nature*, 323, 533 – 536.

Sajda P. (2006), Machine Learning for Detection and Diagnosis of Disease, *Annu. Rev. Biomed. Eng.*, 8, 1 – 29.

Williams J. K. and Abernethy J. (2008), Using random forests and fuzzy logic for automated storm type identification, *AMS Sixth Conference on Artificial Intelligence Applications to Environmental Science*, New Orleans, LA, 2.2.

Wolfson M.M., Rasmussen R. M. and Benjamin S. G. (2008), Consolidated Storm Prediction for Aviation (CoSPA), *AMS 13th Conference on Aviation, Range, and Aerospace Meteorology,* New Orleans, LA, J6.5.

Zeng D., Chen H., Castillo-Chavez C., Lober W.B. and Thurmond M. (eds.) (2011), Infectious Disease Informatics and Biosurveillance, 1st Edition, *Springer*, New York, NY.

CHAPTER 5 – Artificial Neural Networks

CHAPTER 5 discusses the use of artificial neural networks to model highly complex crude oil mass spectra, the algorithms used and the success of the method.

5.1 Introduction

Artificial neural networks are commonly used for modelling complex data. ANNs are capable of identifying mathematical relationships between datasets that may not be immediately apparent to the researcher.

They are very robust and are highly tolerant of noise in the input data, which may prove to be an advantage in this project. Once trained successfully, neural networks require little additional input from the researcher and are therefore very user-friendly. While they operate as black-boxes, and the exact relationship between the input and output variables is not known, a well-trained network can be used repeatedly without any requirement for changing parameters or manipulating the network.

Neural networks, like many other methods of machine learning are commonly used in day-to-day applications, but particularly lend themselves to classification tasks such as image processing in computer vision and are thus commonly used in medical diagnosis, CCTV monitoring, quality control in manufacturing and target recognition in military applications (Morris, 2003). They are also commonly used for pattern recognition tasks in speech recognition software, intelligent search engines, credit rating and stock market predictions and fraud detection (Bolton and Hand, 2002).

Unlike other machine learning algorithms, neural networks are fairly flexible with regards to input and output dataset dimensions, and by simply changing the network geometry by adding or removing hidden layers and nodes they are easily scalable to handle even highly dimensional and complex data (Cartwright, 2008).

They are however highly dependent on having a sufficiently large sample pool for successful training of the model.

5.2 The Algorithm

The algorithm is implemented in IDL, an interactive data language, whose syntax is largely based on FORTRAN 77.

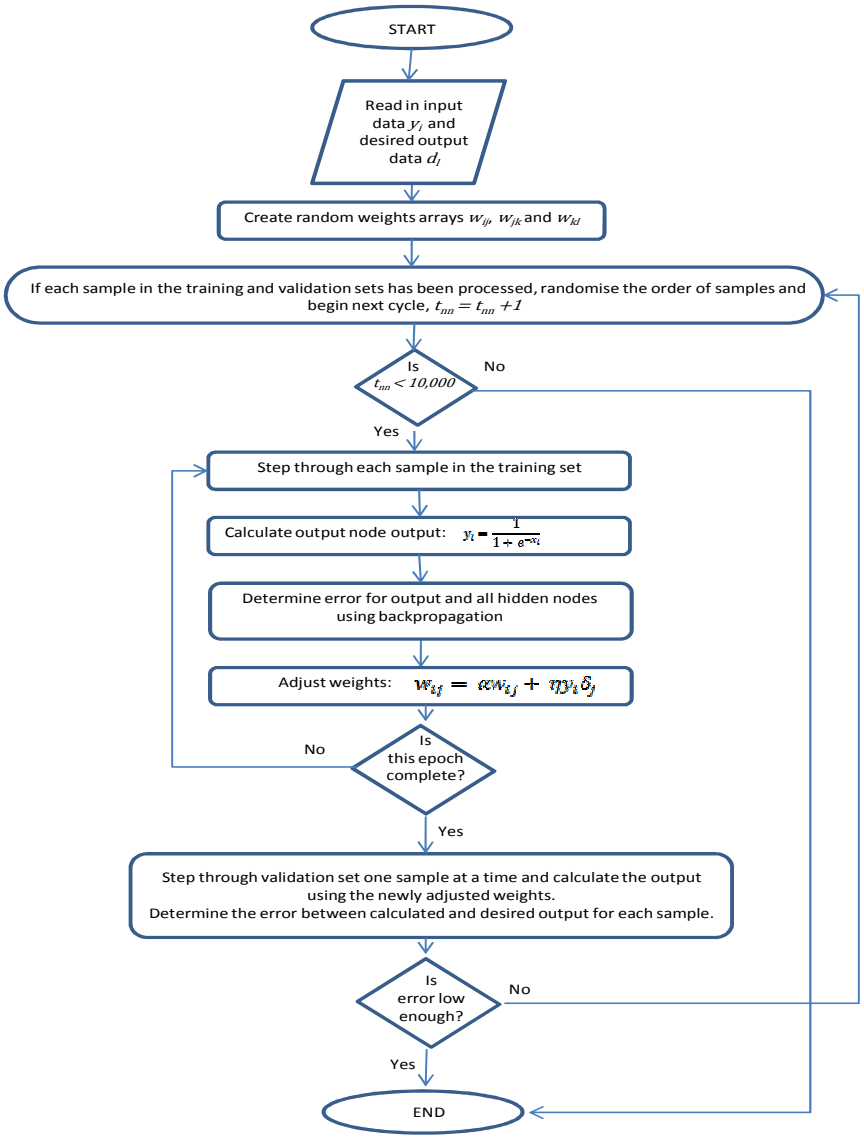


Figure 5.1: Flowchart of the artificial neural network program.

Figure 5.1 shows the basic layout of the algorithm, consisting of only two separate steps, the training and the validation phase.

A brief summary of the artificial neural network program used in this project is as follows:

1. Load input and desired output data, y_i and d_l.

2. Generate random weights arrays with the appropriate dimensions for the number of hidden layers and nodes chosen for the dataset.

3. Divide the available data into training and validation sets.

4. Iterate through each sample in the training set and determine the network output, then adjust the weights using back-propagation.

5. Following each training cycle, verify the performance of the network using the validation set.

6. Randomise the training and validation sets and begin the next cycle.

7. Continue for 10,000 cycles or until the percentage error achieved between the calculated output for samples in the validation set and their desired output drops below a certain threshold.

The source code can be found in appendix A.I.

The program accepts input in the form of data matrices, which must be saved in *.sav* format from the original ASCII file prior to execution of the program. The algorithm

expects m output variables and n input variables and the input and output data must be contained in a $p \times n$ and $p \times m$ matrix respectively, where p is the number of samples for which observations exist.

The output variables d_l consist of the dataset that is to be modelled.

In an artificial neural network the input into the first hidden layer's nodes is a summation of all the input variables of each sample

$$x_j = \sum_{i=1}^{n} y_i \, w_{ij} \tag{5.1}$$

where x_j is the input vector in the j^{th} layer, y_i is the output vector for the i^{th} layer and w_{ij} is the corresponding weight matrix.

The nodes' outputs are then determined using a sigmoid function (see figure 5.2):

$$y_j = \frac{1}{1+ e^{-x_j}} \tag{5.2}$$

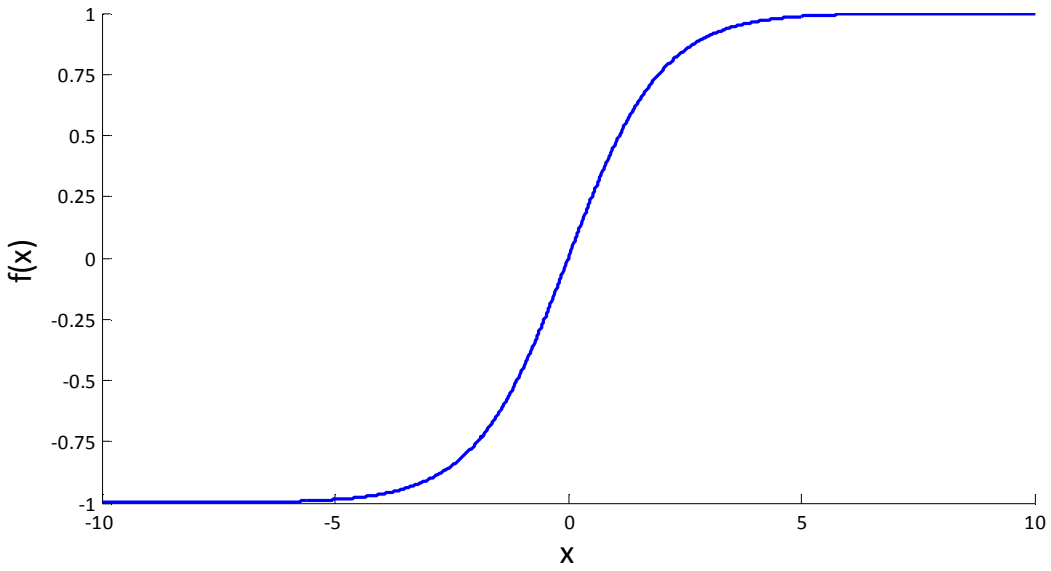


Figure 5.2: A sigmoid function showing that as $x \rightarrow \infty$, $f(x) \rightarrow 1$.

If the input dataset has high dimensions or the variables are large, the summation of the input variables is likely to result in a very large number. Equation 5.2 shows that if $x >> 1$, then $F(x) \rightarrow 1$, which would lead to a uniform output from each node if the input is not normalised.

By determining for which of the p samples the sum of the n variables is the maximum and then dividing each variable by this maximum, input y_i is normalised so that the summation of the n variables is always less than or equal to one (equation 5.3a).

$$(y_{np})_{norm} = \frac{y_{np}}{\sum_{i=1}^{n} y_{iq}} \tag{5.3a}$$

where $(y_{np})_{norm}$ is the normalised n^{th} input variable of the p^{th} sample, y_{np} is the original value for that input and q denotes the sample for which $\sum_{i=1}^{n} y_i$ is a maximum.

The choice of transfer function applied to each node imposes strict limits on the input and output data of the network. It has been previously discussed how the sigmoid function given by equation 5.2 requires the input data to be normalised so that $\sum_{i=1}^{p} y_i \leq 1$. As the output of each node in the network is generated by the sigmoid function, the final output will always be less than one. In most real-world applications, the data to be modelled using a regression algorithm will not fall within these bounds imposed by the transfer function, instead $d_l \in \mathbb{R}$. In order to ensure that the network is able to successfully model the data, the output too is normalised to fall within the range of possible node output (i.e. $-1 \leq (d_l)_{norm} \leq 1$). More specifically, figure 5.2 shows that the sigmoid function's optimal operating range is where $f(x)$ falls between -0.8 and +0.8, as very small changes in x result in fairly large changes in $f(x)$ and therefore the function is most sensitive if $-0.8 \leq (d_l)_{norm} \leq 0.8$.

For that reason it is important to normalise the desired output by dividing the output dataset by the maximum value in that dataset and then multiplying by a factor of 0.8 as given by equation 5.3b.

$$(d_{mp})_{norm} = \frac{d_{mp}}{\max\,(d_{lp})} \times 0.8 \qquad (5.3b)$$

where $(d_{mp})_{norm}$ is the normalised output for the m^{th} variable of the p^{th} sample and $\max\,(d_{lp})$ is the maximum value across the l outputs of the p^{th} sample.

The next step in the program is to define the training, validation and test set parameters. A subset of 90% is selected for the training phase with 8% of all samples used for validation, while the remaining 2% of the dataset is used for testing the network.

A random order in which the samples are to be presented to the network is generated. The first 90% of samples of this random order represent the training set.

The network is a standard scalable network with two hidden layers, with $h1$ and $h2$ hidden nodes and one bias node in each layer (figure 5.3). The bias node has uniform value of one, which remains unchanged throughout the program.

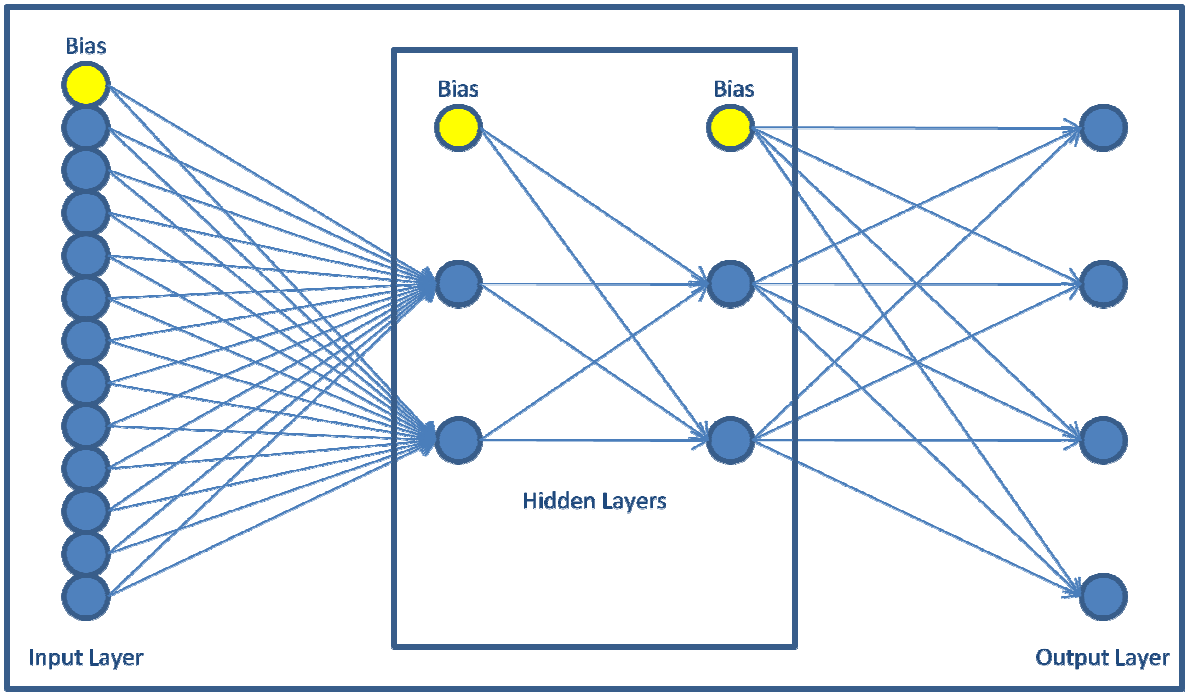


Figure 5.3: A schematic of the neural network layout used in this algorithm.

The algorithm generates random weights arrays for each of the input variables to the nodes in the network.

$$w_{ij} \quad \begin{matrix} i = 1, 2, 3, \ \ldots, \ n \\ j = 1, 2, 3, \ldots, \ h1 + 1 \end{matrix}$$

$$w_{jk} \quad \begin{matrix} j = 1, 2, 3, \ldots, \ h1 + 1 \\ k = 1, 2, 3, \ldots, \ h2 + 1 \end{matrix}$$

$$w_{kl} \quad \begin{matrix} k = 1,2, 3, \ldots, \ h2 + 1 \\ l = 1,2, 3, \ldots, \ m \end{matrix}$$

where there are n input variables, m output variables and $h1$ and $h2$ nodes plus bias in hidden layers 1 and 2 respectively.

The weights are initially chosen to have uniform distribution and to fall in the range $0 < w_{ij} < 1$ and are amended at the end of each training cycle according to

$$w_{ij} = \alpha w_{ij} + \eta y_i \delta_j \quad (5.4)$$

where α = 1 and η is the learning rate and is set to 0.99 for the first 2,500 of 5,000 cycles, decreases to 0.6 for the next 2,000 cycles and finally reduces to 0.1 for the remaining 500 cycles and δ_j is the error in the output from the j^{th} layer. The learning rate determines the speed with which the network converges toward a solution. If the learning rate is too small, the network will not converge; if the learning rate is too large, the network is likely to fail to converge. Learning rates are usually chosen to start with a value close to one and tend to zero with increasing numbers of training cycles (Cartwright and Sztandera, 2003), i.e. $\eta \to 0$ as $t_{nn} \to \infty$.

In each of 5,000 training cycles the network output y_l is computed from the input node output y_i of each sample in the training set according to equation 5.2.

Each node has one more input variable than the previous layer has output variables. This is to account for the bias node, which has a contribution of one to every node, weighted by w.

Once y_l has been calculated, the error function δ for each node is determined according to

$$\delta_l = y_l(1 - y_l)(d_l - y_l) \quad (5.5)$$

and

$$\delta_k = y_k(1 - y_k)\sum_{l=1}^{m} \delta_l w_{kl} \quad (5.6a)$$

and

$$\delta_j = y_j(1 - y_j)\sum_{k=1}^{n} \delta_k w_{jk} \quad (5.6b)$$

which is then used to adjust the weights as described by equation 5.4.

Following the adjustment of the weights for one sample, the network is presented with the next randomly selected sample in the training set. Each sample must be processed by the network before the next training cycle begins and the order in which the training set is introduced to the network is randomised again. Once the network has been presented with each member of the training set, it is introduced to the validation set, which is used to verify that the network is learning as expected. The validation set can also be used to end the learning phase early if a sufficiently good fit is achieved.

On completion of the total number of training cycles or once the validation set has been modelled with sufficient success, the network is introduced to the test set. The test set consists of the remaining samples that are as yet unknown to the network.

The test samples are fed into the network one at a time and processed with the weights that have been optimised in the training step.

5.3 Results

5.3.1 Iris Data

The iris dataset is described in detail in chapter 4.4.1. It is a dataset containing four observations for each of 150 samples. The samples are evenly divided into three classes, class A denoted by [1, 0, 0], B [0, 1, 0] and C [0, 0, 1]. Due to the limitations of the sigmoid function described in the previous section, the output data was normalised

to 0.8. The input data did not need to be normalised. The network used to model this data had two hidden layers with two hidden nodes per layer, as well as four input nodes and three output nodes.

The observations for each of the classes are different enough for the network to easily distinguish between the classes, as can be seen from the data in table 5.1

The output from the test set y_l, together with the desired output d_l and the percentage errors between the two values

$$\Delta y_l = \frac{d_l - y_l}{d_l} \times 100 \qquad (5.7)$$

are shown in table 5.1.

Sample	Desired d_l	Predicted y_l	Error Δy_l (%)	Desired d_l	Predicted y_l	Error Δy_l (%)	Desired d_l	Predicted y_l	Error Δy_l (%)
1	0.8	0.782091	2%	0	0.068554	-	0	0.000868	-
2	0.8	0.826939	3%	0	0.049863	-	0	0.000972	-
5	0.8	0.719427	10%	0	0.006609	-	0	0.00565	-
10	0.8	0.774621	3%	0	0.010249	-	0	0.003608	-
11	0.8	0.85295	7%	0	0.029478	-	0	0.001318	-
13	0.8	0.847404	6%	0	0.038973	-	0	0.001092	-
14	0.8	0.385928	52%	0	0.287704	-	0	0.000562	-
16	0.8	0.836062	5%	0	0.019878	-	0	0.001866	-
18	0.8	0.786575	2%	0	0.066798	-	0	0.000874	-
21	0.8	0.785803	2%	0	0.06711	-	0	0.000873	-
23	0.8	0.692586	13%	0	0.005472	-	0	0.006857	-
27	0.8	0.852465	7%	0	0.028348	-	0	0.001359	-
29	0.8	0.826489	3%	0	0.050076	-	0	0.00097	-
34	0.8	0.851387	6%	0	0.027082	-	0	0.001411	-
36	0.8	0.760539	5%	0	0.077296	-	0	0.000835	-
37	0.8	0.851156	6%	0	0.03606	-	0	0.001142	-

38	0.8	0.84757	6%	0	0.038874	-	0	0.001094	-
39	0.8	0.835826	4%	0	0.019831	-	0	0.00187	-
43	0.8	0.546201	32%	0	0.002231	-	0	0.017206	-
46	0	0.036609	-	0.8	0.84972	6%	0	0.000301	-
62	0	0.097649	-	0.8	0.679293	15%	0	0.000379	-
63	0	0.052068	-	0.8	0.800104	0%	0	0.000326	-
70	0	0.031614	-	0.8	0.867009	8%	0	0.000291	-
71	0	0.059116	-	0.8	0.779177	3%	0	0.000336	-
73	0	0.034688	-	0.8	0.856288	7%	0	0.000297	-
80	0	0.033802	-	0.8	0.859352	7%	0	0.000295	-
82	0	0.03982	-	0.8	0.838957	5%	0	0.000307	-
83	0	0.116192	-	0.8	0.63823	20%	0	0.000396	-
88	0	0.114151	-	0.8	0.642574	20%	0	0.000394	-
94	0	0.03704	-	0.8	0.848262	6%	0	0.000302	-
99	0	0.026141	-	0	0.000011	-	0.8	0.816466	2%
102	0	0.027953	-	0	1.21E-05	-	0.8	0.801002	0%
104	0	0.025901	-	0	1.09E-05	-	0.8	0.81852	2%
108	0	0.0279	-	0	1.21E-05	-	0.8	0.801458	0%
117	0	0.028074	-	0	1.22E-05	-	0.8	0.79997	0%
123	0	0.029356	-	0	0.000013	-	0.8	0.789081	1%
124	0	0.029261	-	0	1.29E-05	-	0.8	0.78988	1%
125	0	0.027166	-	0	1.16E-05	-	0.8	0.807713	1%
127	0	0.027054	-	0	1.16E-05	-	0.8	0.808668	1%
129	0	0.027344	-	0	1.17E-05	-	0.8	0.80619	1%
131	0	0.027527	-	0	1.19E-05	-	0.8	0.804633	1%
133	0	0.027664	-	0	1.19E-05	-	0.8	0.803466	0%
138	0	0.027601	-	0	1.19E-05	-	0.8	0.804003	1%
142	0	0.028413	-	0	1.24E-05	-	0.8	0.797083	0%
144	0	0.027851	-	0	1.21E-05	-	0.8	0.801872	0%
Average			9%			9%			1%

Table 5.1: Output from a neural network trained on 105 samples of the iris dataset over 1,000 iterations, and then used to predict the class of 45 previously unseen samples.

Table 5.1 shows that data class C [0, 0, 0.8] is predicted the most successfully, with higher percentage errors for data classes A [0.8, 0, 0] and B [0, 0.8, 0]. For this particular

test run, the rate of misclassification was equal to zero, with no samples misclassified. A percentage error of < 10% is considered sufficient for successfully modelling this dataset. This experiment proves that the neural network is performing well and can subsequently be applied to the more complex crude oil data.

5.3.2 Crude Oil Data

The input parameters for the crude oil data, as mentioned previously, are datasets A, B and C.

For each of these datasets, the network discussed in 5.2 had to be rescaled in order to account for the more complex, higher dimensionality data. The network consisted of two hidden layers with five nodes each, 712 input nodes for dataset A and 50 each for datasets B and C, and four output nodes.

Depending on the input dataset, there were 713×6 or 51×6 input weights, 6×6 hidden layer 1 to hidden layer 2 weights and 6×4 hidden layer 2 to output weights.

The desired output for each of the three datasets was a 76×4 physical properties matrix.

Physical property C is the pour point temperature in degree Celsius. The values of pour point temperature for the dataset, as quoted in table 2.4, range from -48°C to 43.33°C. The nature of the algorithm may cause the network to run into difficulty when trying to predict negative values. Furthermore, the range of values for pour point temperature is significantly larger than that of the remaining output variables. The success of the network relies heavily on the comparability of the variables and variables which are

very different in size and range are unlikely to be modelled reliably. In order to eliminate these negative values, the data was standardised by converting all temperatures into Kelvin.

The conversion reduces the fractional differences between the extreme values of the dataset, which may marginally reduce the precision of the network. However, ensuring that the overall output dataset is more homogenous was considered of greater importance and thus the conversion from degree Celsius to Kelvin was performed on all temperature values.

60% of the sample data were selected for training with another 10% of samples forming the validation set. The remaining 30% were selected as test set. The test set was chosen to be reasonably large in order to increase the results database used for analysis.

Tables 5.2a), b) and c) list the results of one run of the artificial neural network algorithm for each of the three crude oil datasets A, B and C respectively.

Pour point temperature and API gravity were modelled significantly more successfully, with an average of 7.48% modelling error for pour point temperature and 21.01% for API gravity. Sulphur content and TAN could not be modelled with reliable accuracy.

The data shows that the output from the principal component analysis provided a better basis for modelling than the original spectra, as modelling errors were significantly reduced for dataset C. Reducing the number of properties modelled in any one training cycle only improved the model's performance by a near negligible amount.

	Pour Point Actual	Pour Point Modelled	Error (%)	API Actual	API Modelled	Error (%)	TAN Actual	TAN Modelled	Error (%)	Sulphur Actual	Sulphur Modelled	Error (%)
1	297	272.9944	8.08	30	31.34141	4.47	2.02	0.988443	51.07	0.2	0.448934	124.47
2	261	273.2224	4.68	29.3	28.69868	2.05	1.54	0.593676	61.45	0.05	0.157829	215.66
9	246	215.5771	12.37	21	18.15981	13.52	0.52	9.684684	1762.44	0.95	2.533098	166.64
10	246	261.4633	6.29	45.9	42.78166	6.79	0.08	0.499902	524.88	0.05	0.070618	41.24
20	249	276.0585	10.87	24.4	16.71925	31.48	3.98	48.82623	1126.79	0.1	0.844463	744.46
21	264	290.9116	10.19	32.9	41.46198	26.02	0.16	0.041278	74.20	0.35	2.168307	519.52
23	282	270.4474	4.10	30.6	38.87178	27.03	0.56	1.054615	88.32	0.05	0.035146	29.71
25	297	305.4786	2.85	36.4	35.34658	2.89	0.14	0.831434	493.88	0.05	0.045508	8.98
26	285	255.8089	10.24	43.1	40.39303	6.28	0.15	2.059843	1273.23	0.05	0.117505	135.01
31	261	253.5065	2.87	40.5	34.82603	14.01	0.26	0.356015	36.93	0.1	0.554623	454.62
33	237	262.746	10.86	35.8	36.80065	2.80	0.24	0.090224	62.41	0.25	2.10659	742.64
39	249	242.7208	2.52	30.3	26.49136	12.57	0.17	0.444278	161.34	0.35	1.231017	251.72
44	306	269.9598	11.78	37.5	32.72629	12.73	0.06	0.701325	1068.88	0.1	0.377874	277.87
49	285	254.2358	10.79	45.6	38.12432	16.39	0.03	0.493873	1546.24	0.15	0.669366	346.24
51	294	282.4395	3.93	32.7	27.35263	16.35	0.64	11.95977	1768.71	0.05	0.137346	174.69
60	282	275.7805	2.21	38	30.13568	20.70	0.4	1.067774	166.94	0.05	0.851135	1602.27
61	252	269.9084	7.11	28.4	32.61158	14.83	2.71	0.909527	66.44	0.2	0.216384	8.19
71	264	279.5624	5.89	22.6	30.66235	35.67	0.41	0.09377	77.13	0.1	0.148555	48.55
72	246	250.0869	1.66	27.7	24.37076	12.02	0.28	0.182479	34.83	1.1	6.977717	534.34
73	255	259.3478	1.71	24	21.57479	10.11	3.99	3.523097	11.70	0.15	1.955299	1203.53
74	270	243.3449	9.87	34.1	33.19892	2.64	1.17	8.981178	667.62	0.05	0.040374	19.25
75	300	336.5254	12.18	33.3	26.52856	20.33	0.12	0.528078	340.07	0.3	5.418133	1706.04
76	237	273.8902	15.57	19.9	30.96154	55.59	0.43	1.012941	135.57	2.1	0.552988	73.67
AVERAGE			7.33			15.97			504.39			409.97

5.2 a) Dataset A – 712 common peaks across all spectra.

	Pour Point Actual	Pour Point Modelled	Error (%)	API Actual	API Modelled	Error (%)	TAN Actual	TAN Modelled	Error (%)	Sulphur Actual	Sulphur Modelled	Error (%)
1	297	272.728	8.17	30	31.95241	6.51	2.02	0.959398	52.51	0.2	0.289413	44.71
5	261	300.5327	15.15	29.3	38.49641	31.39	1.54	0.908436	41.01	0.05	0.100395	100.79
9	246	216.0945	12.16	21	18.0732	13.94	0.52	10.28673	1878.22	0.95	2.061241	116.97
14	246	245.5934	0.17	45.9	122.3586	166.58	0.08	0.026798	66.50	0.05	0.01119	77.62
15	249	238.7534	4.12	24.4	20.9898	13.98	3.98	15.72336	295.06	0.1	0.667084	567.08
18	264	295.7111	12.01	32.9	46.98663	42.82	0.16	0.284083	77.55	0.35	0.344475	1.58
24	282	255.4128	9.43	30.6	64.38904	110.42	0.56	0.175507	68.66	0.05	0.134901	169.80
26	297	268.7917	9.50	36.4	33.86426	6.97	0.14	1.859679	1228.34	0.05	0.146087	192.17
27	285	298.2448	4.65	43.1	34.4158	20.15	0.15	0.34797	131.98	0.05	0.321353	542.71
31	261	254.3089	2.56	40.5	35.14109	13.23	0.26	0.32376	24.52	0.1	0.537844	437.84
33	237	261.0086	10.13	35.8	37.6128	5.06	0.24	0.092066	61.64	0.25	0.971902	288.76
52	249	266.4248	7.00	30.3	28.98479	4.34	0.17	0.084942	50.03	0.35	0.302611	13.54
53	306	292.1842	4.51	37.5	28.63984	23.63	0.06	0.688976	1048.29	0.1	1.177876	1077.88
54	285	307.9592	8.06	45.6	58.57606	28.46	0.03	0.008653	71.16	0.15	0.438272	192.18
58	294	274.888	6.50	32.7	31.18421	4.64	0.64	0.919563	43.68	0.05	0.558366	1016.73
61	282	301.6952	6.98	38	43.05977	13.32	0.4	0.152703	61.82	0.05	0.03777	24.46
64	252	218.3699	13.35	28.4	21.69548	23.61	2.71	68.78379	2438.15	0.2	0.349838	74.92
68	264	268.2482	1.61	22.6	20.69022	8.45	0.41	0.306699	25.20	0.1	0.455385	355.38
70	246	232.8736	5.34	27.7	23.37559	15.61	0.28	0.623575	122.71	1.1	24.86395	2160.36
72	255	260.187	2.03	24	20.87176	13.03	3.99	2.320103	41.85	0.15	1.23367	722.45
74	270	247.0241	8.51	34.1	32.13224	5.77	1.17	9.032733	672.03	0.05	0.08085	61.70
75	300	331.1237	10.37	33.3	26.17373	21.40	0.12	0.432336	260.28	0.3	4.605984	1435.33
76	237	274.5448	15.84	19.9	31.39966	57.79	0.43	0.924284	114.95	2.1	0.487567	76.78
AVERAGE			7.75			28.31			385.92			423.99

5.2 b) Dataset B – 50 peaks with greatest variance across all samples.

	Pour Point Actual	Pour Point Modelled	Error (%)	API Actual	API Modelled	Error (%)	TAN Actual	TAN Modelled	Error (%)	Sulphur Actual	Sulphur Modelled	Error (%)
1	297	268.3717	9.64	30	29.1006	3.00	2.02	1.685259	16.57	0.2	9E-06	100.00
3	261	268.0892	2.72	29.3	28.77092	1.81	1.54	1.793377	16.45	0.05	8.4E-06	99.98
18	246	267.5299	8.75	21	20.99146	0.04	0.52	0.72758	39.92	0.95	0.075101	92.09
20	246	283.0276	15.05	45.9	42.81677	6.72	0.08	0.020631	74.21	0.05	0	100.00
21	249	267.2103	7.31	24.4	25.15318	3.09	3.98	3.945056	0.88	0.1	0.184975	84.97
25	264	273.994	3.79	32.9	32.64916	0.76	0.16	0.071118	55.55	0.35	0.00023	99.93
34	282	273.8185	2.90	30.6	30.48866	0.36	0.56	0.378231	32.46	0.05	1.8E-06	100.00
36	297	277.4542	6.58	36.4	39.55819	8.68	0.14	0.109753	21.60	0.05	1.62E-05	99.97
38	285	278.2787	2.36	43.1	36.88277	14.43	0.15	0.124091	17.27	0.05	0	100.00
41	261	275.8196	5.68	40.5	36.25789	10.47	0.26	0.075424	70.99	0.1	5.4E-05	99.95
42	237	274.7963	15.95	35.8	36.09664	0.83	0.24	0.279001	16.25	0.25	1.68E-05	99.99
45	249	274.6127	10.29	30.3	31.77443	4.87	0.17	0.442215	160.13	0.35	0	100.00
49	306	275.82	9.86	37.5	33.48345	10.71	0.06	0.291797	386.33	0.1	0	100.00
50	285	275.4511	3.35	45.6	37.24061	18.33	0.03	0.216293	620.98	0.15	0.000015	99.99
58	294	275.1558	6.41	32.7	35.21187	7.68	0.64	0.091789	85.66	0.05	7.2E-05	99.86
60	282	278.8937	1.10	38	37.70554	0.77	0.4	0.099357	75.16	0.05	0	100.00
61	252	261.4039	3.73	28.4	13.98528	50.76	2.71	2.857074	5.43	0.2	0.216087	8.04
63	264	277.1935	5.00	22.6	35.40186	56.65	0.41	0.182638	55.45	0.1	0	100.00
67	246	279.7654	13.73	27.7	38.83974	40.22	0.28	0.070838	74.70	1.1	0	100.00
71	255	275.2332	7.93	24	37.34605	55.61	3.99	0.253542	93.65	0.15	4.2E-06	100.00
73	270	263.4884	2.41	34.1	23.45839	31.21	1.17	3.593047	207.10	0.05	6.6E-06	99.99
74	300	269.5437	10.15	33.3	21.41474	35.69	0.12	0.047591	60.34	0.3	0	100.00
76	237	272.295	14.89	19.9	33.5237	68.46	0.43	0.524862	22.06	2.1	1.32E-05	100.00
AVERAGE			7.37			18.74			96.05			94.99

5.2 c) Dataset C – 50 first principal components.

5.4 Discussion

As mentioned in section 5.1, neural networks have the advantage of being easily scalable and can therefore be adjusted to handle both the low-dimensionality iris and the high-dimensionality crude oil datasets with relative ease.

The results from both the iris dataset and the crude oil dataset show that the network is performing well on some parts of the data. While the iris dataset is significantly easier to learn, the network does show that some of the crude oils' physical properties can be modelled reasonably successfully, as errors of below 10% as evident with pour point temperature can be considered conclusive.

5.5 Conclusion

Artificial neural networks have been applied to the problem of modelling crude oils with only limited success. While in all three dataset analyses it has been shown that some of the properties can be modelled with slightly more success than the others, the reliability of the prediction is not enough to conclusively state that the data can be modelled successfully.

References

Bolton R.J. and Hand D.J. (2002), Statistical Fraud Detection: A Review, *Statistical Science*, 17(3), 235 – 255.

Cartwright H.M. and Sztandera L.M. (eds.) (2003), Soft Computing Approaches in Chemistry, Studies in Fuzziness and Soft Computing, *Springer*, Berlin, Germany.

Cartwright H.M. (2008), Using Artificial Intelligence in Chemistry and Biology: A Practical Guide, *CRC Press*, London, UK.

Morris T. (2004), Computer Vision and Image Processing, *Palgrave Macmillan*, Basingstoke, UK.

CHAPTER 6 – Simulated Annealing Genetic Algorithm

CHAPTER 6 discusses the success of a simulated annealing and genetic algorithm hybrid technique and its results.

6.1 Introduction

Intelligent data analysis, as described in chapter 2, is widely used to model the physical properties of unknown chemicals using algorithms that intelligently train to recognise patterns in known data (Michalski *et al.*, 1983). There exist a host of different machine learning techniques, some of which have been discussed in chapter 4, which make different demands on the type of *a priori* information one has on the samples (Russell and Norvig, 1995).

While each has particular strengths that lend themselves to certain types of problems, it is the combination of several different techniques that may be particularly successful in modelling highly complex problems. Using two techniques sequentially, such as PCA followed by a neural network, may significantly improve the performance of the model by first eliminating those data which are of no value to the model and thereby streamlining the neural network.

Used in combination, two machine learning methods can create a hybrid algorithm with fewer limitations than the individual techniques (Sun and Bookman, 1995). For example, artificial neural networks have been combined with decision trees to create a hybrid model for stock market predictions (Tsai and Wang, 2009). While the ANN is very capable of making accurate predictions, the underlying decisions which lead to the correct predictions are unknown. In combination with decision tree algorithms, the rules which govern successful prediction, the cause and effect relation, can be elucidated.

This chapter discusses the development of a new hybrid algorithm, which uses two different techniques in conjunction with one another to model the highly complex crude oil mass spectra and outlines its success.

6.2 Simulated Annealing Genetic Algorithm

Simulated annealing, discussed in detail in chapter 3, is a heuristic method that seeks to find the closest practical solution to a global minimum. It was first developed by Metropolis *et al.* (1953) and builds on the Monte Carlo method devised by Neuman and Ulam in the late 1940s (Eckhart R., 1987; Metropolis and Ulam, 1949). Unlike other experience-based optimisers such as hill climbing algorithms, it is able to temporarily accept a solution that is worse than the previous one, in order to escape local minima and continue moving toward the optimal solution.

The simulated annealing algorithm starts in a random location of the solution landscape and is asked to take a small step. If the score or fitness of the solution after that step is improved, the step is performed. In most optimisers, such as hill climbing, the step is not performed if the score is not improved (Russell and Norvig, 1995). In a simulated annealing algorithm, the step can be performed irrespective of the change to the score, however there is a probability attached to any step.

The probability of performing a 'worsening' step is dependent on the temperature and decreases as a function of time.

$$P(e, e', T) = \mathrm{e}^{\left(\frac{e-e'}{T}\right)} \tag{6.1}$$

where $P(e, e', T)$ is the acceptance probability function, known as the Metropolis criterion (Metropolis *et al.*, 1953), e and e' are the scores of the current solution and the closest approximation of the global minimum found to date respectively and T is the temperature. $T \rightarrow 0$ as $t \rightarrow \infty$.

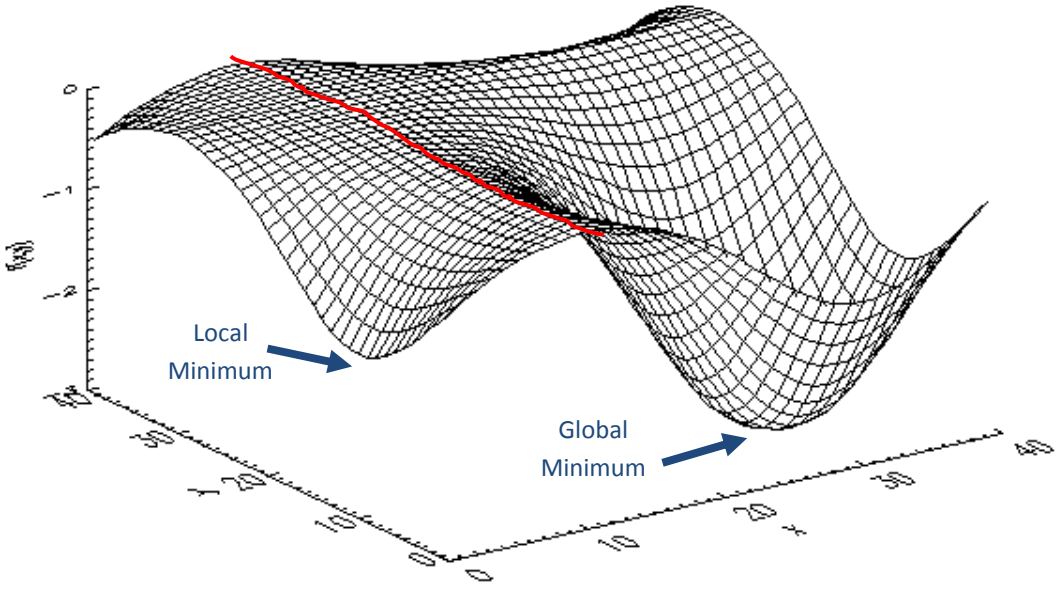


Figure 6.1: The solution landscape in a simulated annealing process. In a typical hill climbing algorithm, any search that starts to the left-hand side of the watershed (red line) would fall into the local minimum and end there. The simulated annealing algorithm however, can temporarily accept a poorer solution in order to escape from the local minimum and reach a global minimum.

The steps are analogous to the random thermal fluctuations of atoms in a lattice, which decrease as a function of temperature and can be observed in the physical process of cooling heated metals. As the heated metal is cooled, the atoms settle into an energy state which is lower than the starting energy. This process is known as annealing.

Another mathematical optimisation technique commonly used is the genetic algorithm. As with simulated annealing, it aims to find the optimal solution in a solution landscape.

A subset of possible solutions chosen as the mating pool in any one epoch is known as the population. The score of any solution is known as its fitness. The population can evolve via a process of selection, followed by mating and mutation.

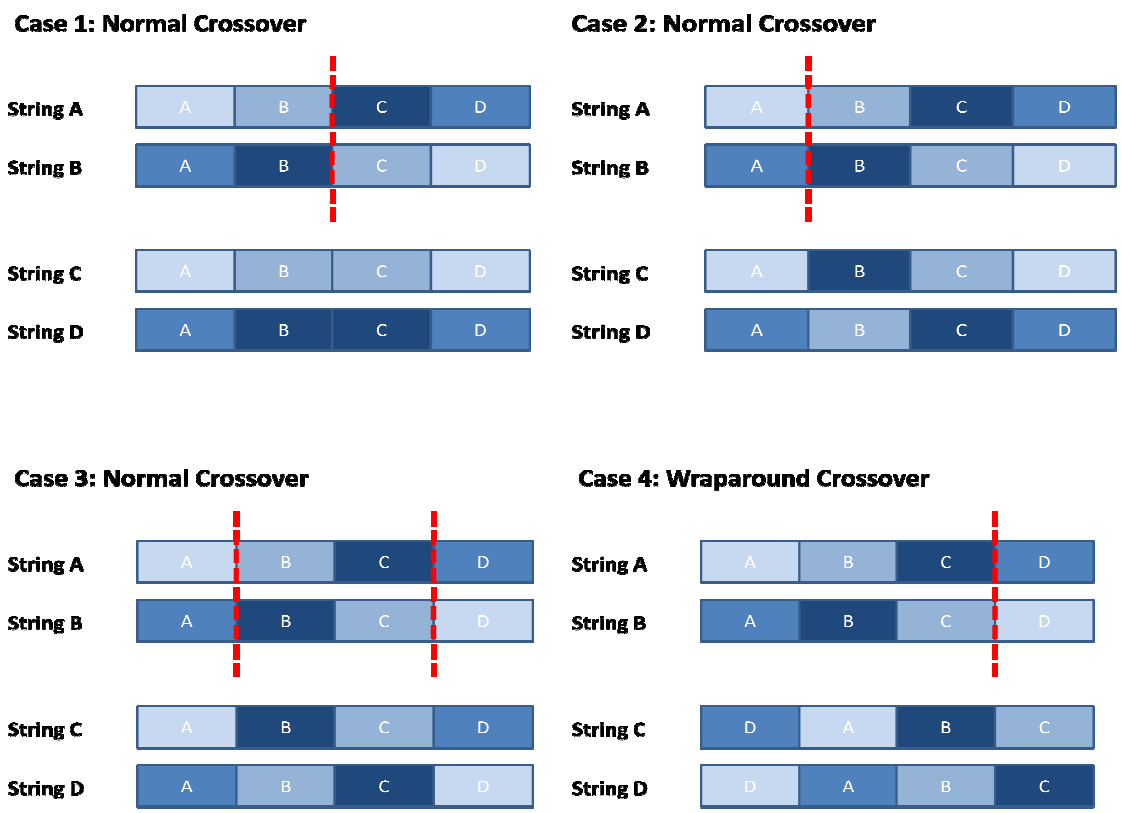


Figure 6.2: Figure showing the creation of new strings using crossover. In this case each string has four parameters, which can be crossed in one of four ways.

In mating, two members of the population (candidates) are chosen and their individual parameters known as chromosomes or strings are crossed in a process known as crossover to create two new solutions. The crossover point along the string is chosen at random and the two halves of the strings for mother and father solution are swapped.

In mutation a random parameter of a randomly selected solution is changed. Once the new string has been created using crossover or mutation, the fitness of that offspring is

determined. Selection involves the scoring of an entire population in terms of their fitness and selecting the fittest stochastically to undergo mutation or mating.

Stochastic remainder selection works by normalising the fitnesses of an entire population so that the average fitness is equal to one.

$$F(i) = F(i) \times \frac{n}{\sum_{i=1}^{n} F_i} \tag{6.2}$$

where n is the total number of strings and $F(i)$ is the fitness of the i^{th} string. Each string with fitness greater than or equal to one is copied into the new population a number of times equal to the integer part of its fitness. The remainder of the new population is filled with the remaining strings selected stochastically. If $F_{res}(i) > random$ then the string is selected for the new population, where $F_{res}(i)$ is the residual fitness of the i^{th} string (with i chosen at random) and $random$ is a random number in the range 0-1. Following selection $F_{res}(i)$ is set to a negative value to prevent any string from being selected more than once.

The algorithm continues until the overall fitness of the population has reached a target value or a given number of cycles have been performed. The fittest member of the population is chosen as the optimal solution.

The combination of simulated annealing and a genetic algorithm, dubbed SAGA, is a powerful tool in modelling complex crude oil data. The algorithm used is described in more detail in the following section.

6.2.1 The Algorithm

The program was written in FORTRAN 95 and later translated into IDL and was executed on a standard personal computer (Intel Core i3 2.13 GHz, 4GB RAM). The method is computationally expensive and can take several hours to complete. A flowchart outlining the algorithm is shown in figure 6.4.

The algorithm is designed to generate a set of random spectra, known as masks, and a set of property spectra, which, once weighted optimally, will combine linearly to recreate the mass spectrum of a crude oil. The starting mask and property spectra, as well as the mask weights, are chosen at random and optimised using simulated annealing. The success of the method is discussed in section 6.2.2.

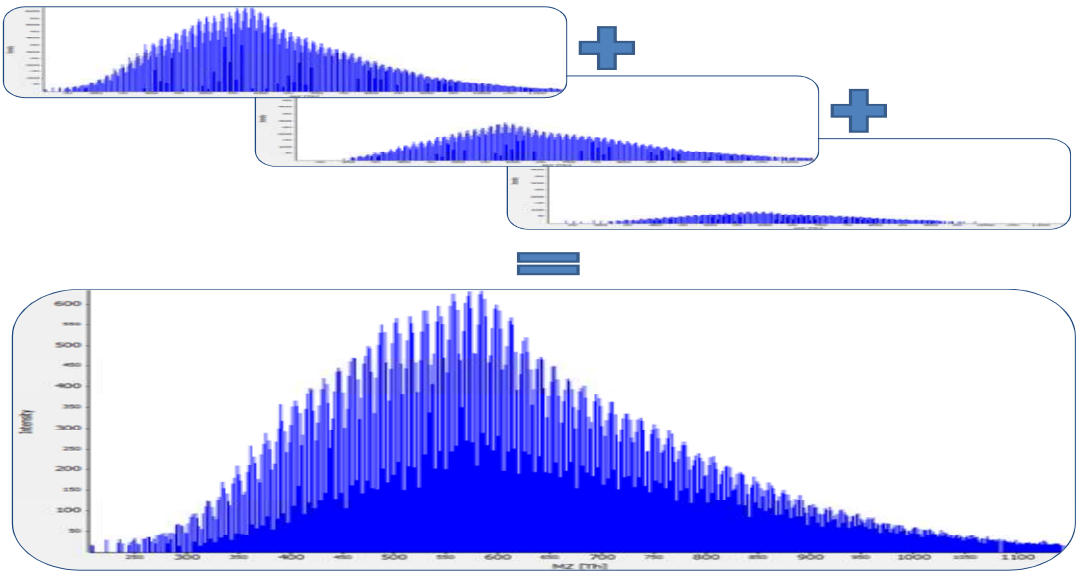


Figure 6.3: Linear combination of generic mask spectra recreating the complete mass spectrum.

6.2.1.1 Input Data

The program accepts ASCII files containing the crude oil spectra in the form m/z value by intensity in arbitrary units and a list of properties for each of the samples as input.

The program also requires a set of instructions, which must include the number of properties which are listed in the data file, a property selector which dictates the properties that are to be used for the analysis, the number of mask spectra to create, the number of cycles in the SA routine, a parameter which determines the cooling schedule and the starting temperature for the simulated annealing process.

In this case the simulated annealing process performed 10,000 cycles and created six mask spectra to model the crude oils.

Immediately following data entry the raw data should be normalised by taking the logarithm of the intensities of each peak. In a typical crude oil mass spectrum the intensities vary greatly along the m/z range and stronger peaks are likely to dominate the analysis. The log-normalisation of the spectrum results in a more evenly distributed intensity scale and reduces the risk that the calculation will be biased by attempting to fit a small number of the most intense peaks.

6.2.1.2 Creating the Mask and Property Spectra

The initial mask and property spectra are generated with the same dimensions as the input spectra, each peak having an m/z equal to the mean value of m/z (equation 6.3a) and intensity equal to the mean intensity (equation 6.3b) of its corresponding peak across the full set of input spectra.

$$x'_p = \frac{\sum_{i=1}^{n} x_{ip}}{n} \quad (6.3a)$$

where x'_p is the m/z value for the p^{th} peak in the initial mask and property spectra, x_{ip} is the m/z for the p^{th} peak in the i^{th} sample spectrum and n is the total number of samples.

$$y'_p = \frac{\sum_{i=1}^{n} y_{ip}}{n} \quad (6.3b)$$

where y'_p is the intensity value for the p^{th} peak in the initial mask and property spectra, y_{ip} is the intensity for the p^{th} peak in the i^{th} sample spectrum and n is the total number of samples.

The property weights are chosen to be equal to the property values for each of the 76 samples and the mask weights are generated at random according to the Mersenne Twister (Matsumoto and Nishimura, 1998), which is the default random number generator in a number of programming language libraries including MATLAB, Python and IDL.

The result is a set of four property spectra containing 712 data points each, six mask spectra containing 712 data points each and two matrices of dimensions 4×76 and 6×76 containing the property weights and mask weights respectively.

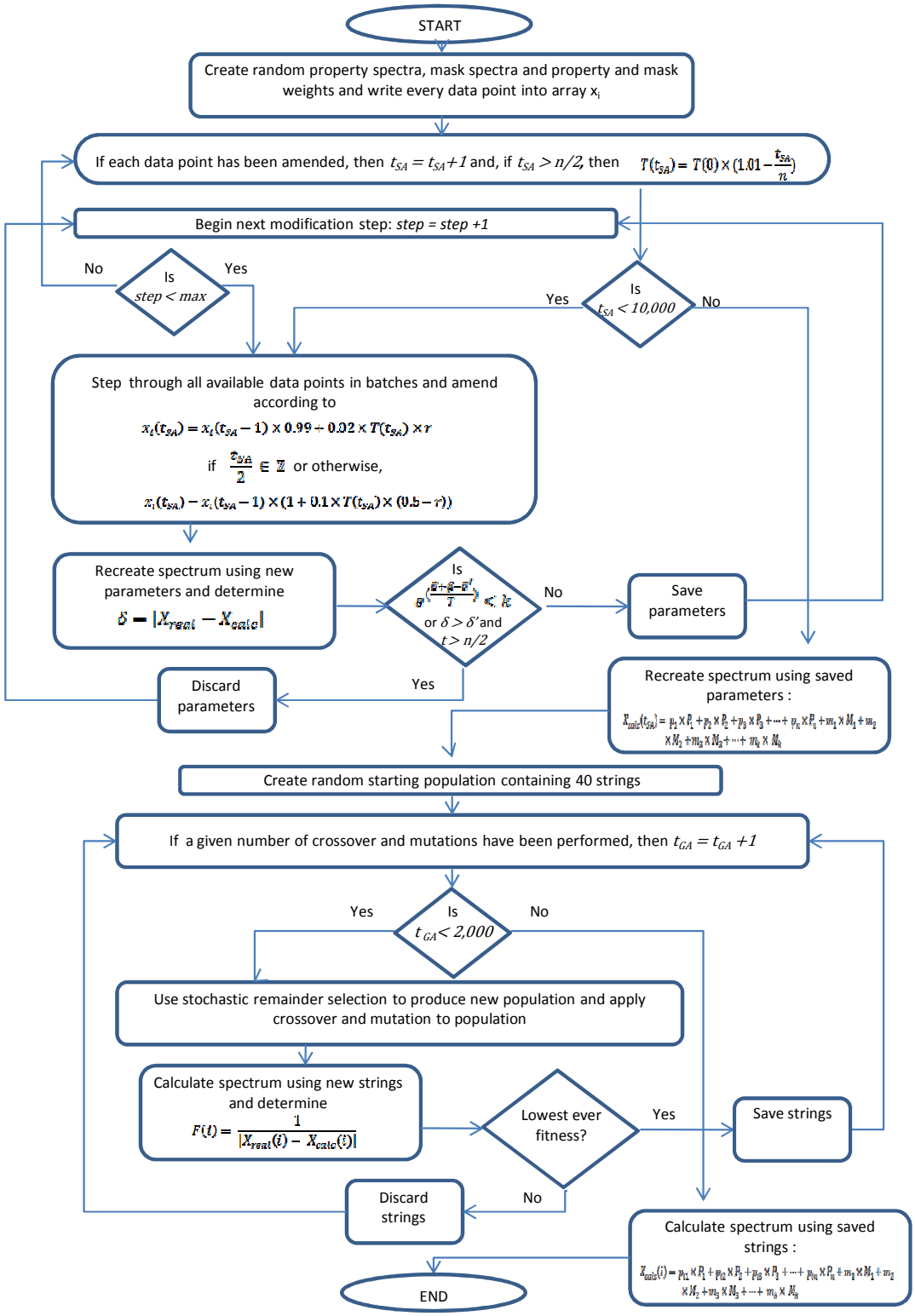


Figure 6.4: Flowchart of the simulated annealing genetic algorithm (SAGA) algorithm.

6.2.1.3 Modification

In order to identify the optimum combination of mask and property spectra and their respective weights, every point in each of the arrays is adjusted in a series of modification steps and the resultant spectrum after each step is compared to the original spectrum in order to monitor the success of the modifications.

As there are a very large number of data points which must be modified, the algorithm adjusts a "batch" of points in any one modification step in order to speed up processing time. The number of points which are open for modification is governed by the parameter BATCHSIZE, which is equal to eight for the work described here.

There exists an optimum value for BATCHSIZE which is likely to be smaller than that chosen for this project. The relationship between BATCHSIZE and quality of solution is complex and cannot be expressed analytically. Empirical determination of that relationship would be computationally too demanding, therefore trial and error is used to find a value which proves effective.

A randomised vector of every adjustable data point is created, from which the batches are then selected. This vector contains 7,576 ($712 \times 4 + 712 \times 6 + 6 \times 76$) data points, thus the algorithm undergoes 947 modification steps in every cycle of the SA algorithm.

The modification performed on any point in the vector varies depending on whether the algorithm is in an even numbered cycle ($\frac{t_{SA}}{2} \in \mathbb{Z}$) or an odd numbered cycle ($\frac{t_{SA}}{2} \notin \mathbb{Z}$) and is given by equations 6.4a and 6.4b respectively.

$$x_i(t_{SA}) = x_i(t_{SA} - 1) \times 0.99 + 0.02 \times T(t_{SA}) \times r \qquad (6.4a)$$

where x_i is the point to be modified, $T(t_{SA})$ is the temperature in the current simulated annealing cycle t_{SA} and $T \rightarrow 0$ as the number of cycles increases and r is a random number between zero and one.

$$x_i(t_{SA}) = x_i(t_{SA} - 1) \times (1 + 0.1 \times T(t_{SA}) \times (0.5 - r)) \quad (6.4b)$$

Once a batch of eight points has been modified, the composite spectrum is calculated and compared with the original crude oil mass spectrum.

The composite spectrum is a linear addition of the weighted mask and property spectra

$$X_{calc} = p_1 \times P_1 + p_2 \times P_2 + p_3 \times P_3 + \cdots + p_n \times P_n + m_1 \times M_1 + m_2 \times M_2 + m_3 \times M_3 + \cdots + m_k \times M_k \quad (6.5)$$

where X_{calc} is the calculated spectrum, p_n is the n^{th} property weight and P_n is the n^{th} property spectrum, m_k is the k^{th} mask weight and M_k is the k^{th} mask spectrum.

The calculated spectrum is then compared to the original spectrum and the deviation determined.

$$\delta = |X_{real} - X_{calc}| \quad (6.6)$$

where δ is the deviation, X_{real} is the original mass spectrum and X_{calc} is the calculated spectrum.

If the deviation in the current cycle is better than any deviation calculated previously, it is saved as the parameter DEVBEST or δ'.

The probability of accepting a solution that does not improve the overall deviation, ($\delta > \delta'$) where δ' is the lowest observed deviation, is determined according to the cooling schedule (equation 6.8) of the simulated annealing method.

$$P(e, e', T) = \mathrm{e}^{(\frac{e-e'}{T})} \quad \text{[See equation 6.1]} \quad (6.7)$$

where $P(e, e', T)$ is the acceptance probability function, known as the Metropolis criterion (Metropolis *et al.*, 1953), and e and e' are the scores of the current and best solutions respectively. If $e < e'$ then $P(e, e', T) = 1$, otherwise, as $T \to 0$, $P(e, e', T) \to 0$ if $e > e'$ according to Kirkpatrick *et al.* (1983).

The SAGA algorithm further introduces a bias to the probability function, which is set by the BIAS parameter. BIAS is a variable which contributes to the probability of accepting a new solution that does not improve the deviation. If BIAS is set to zero, any new solution that results in an unchanged deviation will be accepted. If BIAS is set to be greater than zero, any new solution must significantly improve the previous best deviation in order to be accepted. BIAS is set to 2.5.

The algorithm dictates that if $e^{(\frac{\delta+\beta-\delta'}{T})} < k$, where k is a random number between zero and one and β is the bias, or $\delta > \delta'$ and the algorithm is more than midway through its total number of SA cycles, then the new parameters x_i are discarded. Otherwise, the new parameters are accepted.

The program must make one complete iteration through every modifiable data point during each SA loop.

6.2.1.4 The Cooling Schedule

The cooling schedule is called upon to adjust the temperature of the system, $T(t_{SA})$, before the next SA cycle can commence. The cooling schedule governs the change in temperature over the course of the program. Temperature is a parameter which determines the probability of performing a step, irrespective of its effect on the score of the solution. As time progresses, the probability of performing a step decreases as $T \rightarrow 0$. If temperature reaches zero, the annealing process stops and no more steps can be taken. The decreasing temperature is important as it ensures that the solution cannot escape from a global minimum once it has found it.

In this algorithm, the temperature remains constant for the first $\frac{n}{2}$ cycles and is then linearly reduced to within 1% of the starting temperature over the remaining $\frac{n}{2}$ cycles.

$$T(t_{SA}) = T(0) \times (1.01 - \frac{t_{SA}}{n}) \quad (6.8)$$

where $T(t_{SA})$ is the temperature in the current cycle, $T(0)$ is the starting temperature, t_{SA} is the current cycle and n is the total number of cycles. The starting value for temperature is one.

Once the temperature has been adjusted, the next SA cycle begins and the data point vector is once again randomised and then stepped through in batches of eight for modification.

6.2.1.5 Genetic Algorithm String Initialisation

Following the completion of the SA cycles, the genetic algorithm section of the method is implemented. It uses the optimised mask and property spectra created using SA to determine the property weights which best recreate the original spectra.

The GA first sets the number of strings in each population and the fraction thereof to be used for crossover and mutation, which are 40, ½ and ¾ respectively.

The values for the strings forming the starting population are chosen at random, bound by the maximum and minimum values for each property.

6.2.1.6 Determination of Fitness

In order to evaluate the "quality" of a new string, its fitness is determined by calculating the resultant spectrum for each of the strings as described by equation 6.5 and determining the deviation of the calculated spectrum from the original spectrum (equation 6.6).

The fitness function for the i^{th} string is given by

$$F(i) = \frac{1}{|X_{real}(i) - X_{calc}(i)|} \quad (6.9)$$

where

$$X_{calc}(i) = p_{i1} \times P_1 + p_{i2} \times P_2 + p_{i3} \times P_3 + \cdots + p_{in} \times P_n + m_{i1} \times M_1 + m_{i2} \times M_2 + m_{i3} \times M_3 + \cdots + m_{ik} \times M_k \quad (6.10)$$

where $X_{calc}(i)$ is the calculated spectrum for the i^{th} string and p_{in} is the weighting for the n^{th} property spectrum as calculated for the i^{th} string. P_n, M_k and m_{ik} are the property spectra, mask spectra and mask weights as determined in the SA procedure.

The string of highest fitness in every population is saved, along with its fitness.

6.2.1.7 Creation of the Next Generation – Survival of the Fittest

Once the fitness of each member of the population has been determined, the fitnesses are scaled

$$F(i) = 1 + 9 \times \frac{F(i) - \min(F(i))}{\max(F(i)) - min(F(i))} \quad (6.11)$$

To conclude the first evolutionary cycle, the members of the population that will form part of the next population are determined. This is done by stochastic remainder selection, which is one of the selection methods described by Baker *et al.* (1985) and is outlined in the previous section.

6.2.1.8 Crossover and Mutation

Next the algorithm applies crossover to the new population by choosing two strings at random and then randomly selecting crossover points within those strings. Crossover is applied ten times, while ensuring that no string is chosen more than once.

The algorithm then performs a user-defined number of cycles of mutation (in this case 30) on randomly selected strings. Once a string has been picked, a parameter within the string is chosen at random and mutated according to:

$$x_{in} = 0.9 \times \min(x_n) + 1.2k \times (\max(x_n) - \min(x_n)) \quad (6.12)$$

where x_{in} is the n^{th} point of the i^{th} string, $\min(x_n)$ is the minimum value, $\max(x_n)$ is the maximum value for the n^{th} property in the sample set and k is a random number between zero and one.

Once crossover and mutation have been performed on the new population, the composite spectrum is calculated for each string and the deviation from the original spectrum and thus the string's fitness is once again calculated before a new population is created. The genetic algorithm performs a total of 2,000 iterations, a number that has been determined experimentally to be sufficient for the algorithm to converge. The string that produces the best fitness throughout the whole procedure is presented as the output of the program.

A summary of the algorithm is as follows:

1. Create random spectra for each of the properties and a number of random mask spectra and mask weights.

2. Step through all available data points in BATCHSIZE batches and adjust them in accordance with the SA cooling schedule.

3. Once the maximum number of batch modifications has been performed, calculate the composite spectrum using the new arrays and determine the deviation from the original crude oil spectrum.

4. Accept the new parameters according to the acceptance probability function.

5. Repeat the SA routine for 10,000 cycles.

6. Create 40 random strings of four parameters each representing the four crude oil properties.

7. For each of 2,000 cycles of the GA determine the fitness of each string by recreating the crude oil spectrum using the optimised mask and property spectra and mask weights from the SA routine and the property parameters saved in the strings and calculating the deviation from the original spectrum. The new population for the next GA cycle is determined based on the fitness of the strings. In each cycle, adjust the parameters in the strings via crossover and mutation and re-evaluate their fitness.

8. The fittest string found after 2,000 cycles represents the modelled properties of the crude oil.

6.2.2 Results

Table 6.1 shows the percentage error between actual physical properties and those calculated during the genetic algorithm routine of the SAGA program. The method appears to be working well when modelling no more than two properties at any one time, with decreasing success as more properties are modelled. As the properties are entirely uncorrelated, it stands to reason that the modelling process becomes increasingly difficult the more independent variables one aims to predict. The prediction error for property A (pour point temperature in Kelvin) is 11.76% averaged over four separate runs of the algorithm. Properties B (API gravity), C (TAN) and D

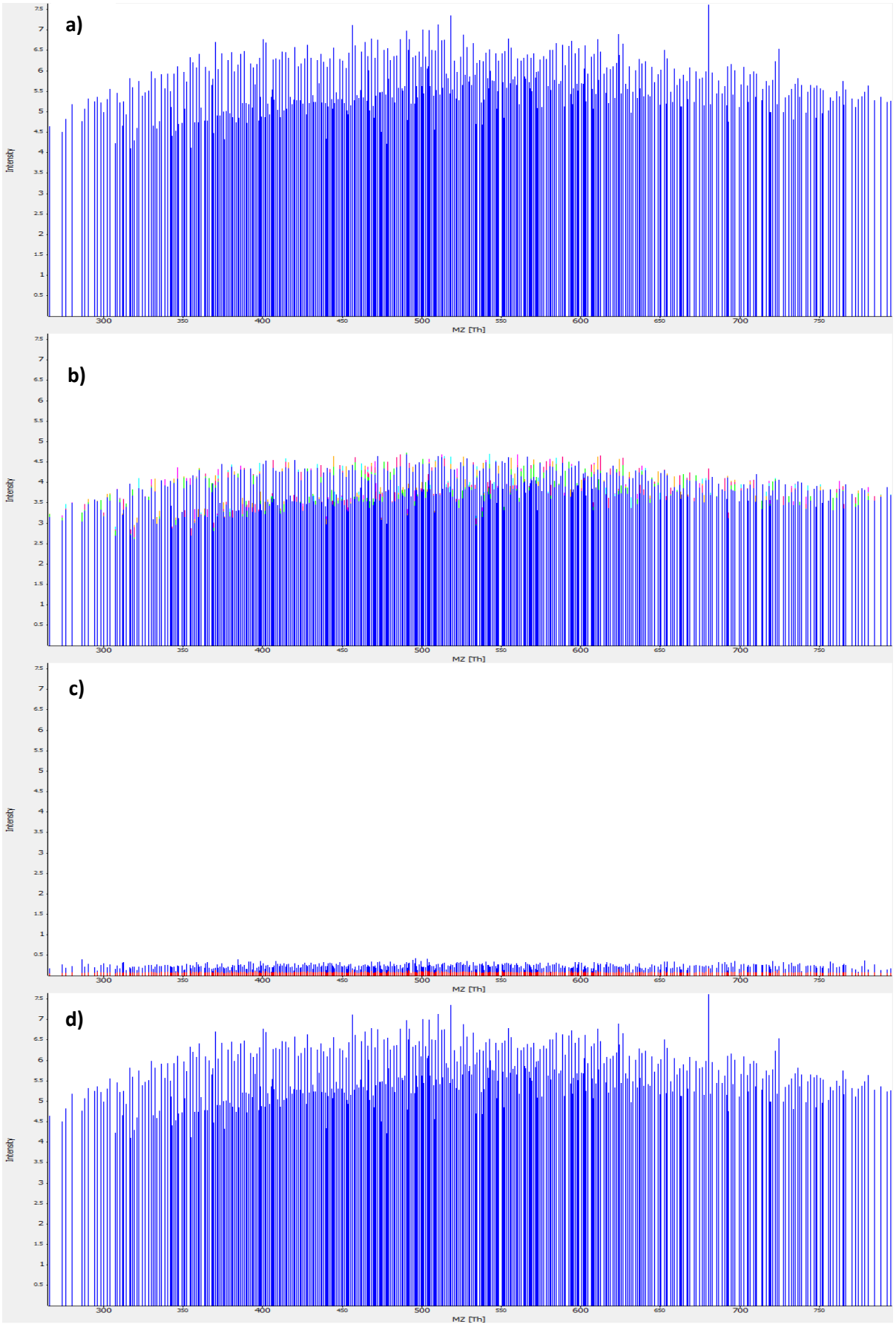
(sulphur content) have significantly higher errors of 69.87%, 102.96% and 73.06% respectively.

Analysis Run No.	Pour Point Temp. Error (%)	API Gravity Error (%)	TAN Error (%)	Sulphur Content Error (%)
1	-	-	139.24	64.58
2	13.23	124.76	-	-
3	12.23	-	82.98	-
4	11.03	-	-	64.82
5	-	54.60	77.89	
6	-	48.56	-	61.83
7	10.55	51.55	111.73	101.00
Average	**11.76**	**69.87**	**102.96**	**73.06**

Table 6.1: The percentage errors between predicted and actual crude oil properties as determined for seven separate runs. Runs 1 to 6 only attempted to model two of the properties, while all four were modelled simultaneously in run number 7.

The creation of mask and property spectra in the simulated annealing section of the algorithm proves slightly more successful. Figure 6.5 shows the original spectrum for sample 13, the calculated mask and property spectra for two physical properties and the composite spectrum. In this case, the average error between real and calculated intensities is 4%, with the largest difference between any one calculated peak and the original peak equalling 23%. The individual mask and property spectra hold little meaning by themselves and must be weighted and combined with one another in order to create a meaningful spectrum.

A closer look at the average reproducibility of each of the samples shows clearly that some samples are much easier to model than others.


a)
b)
c)
d)
Intensity
MZ [Th]

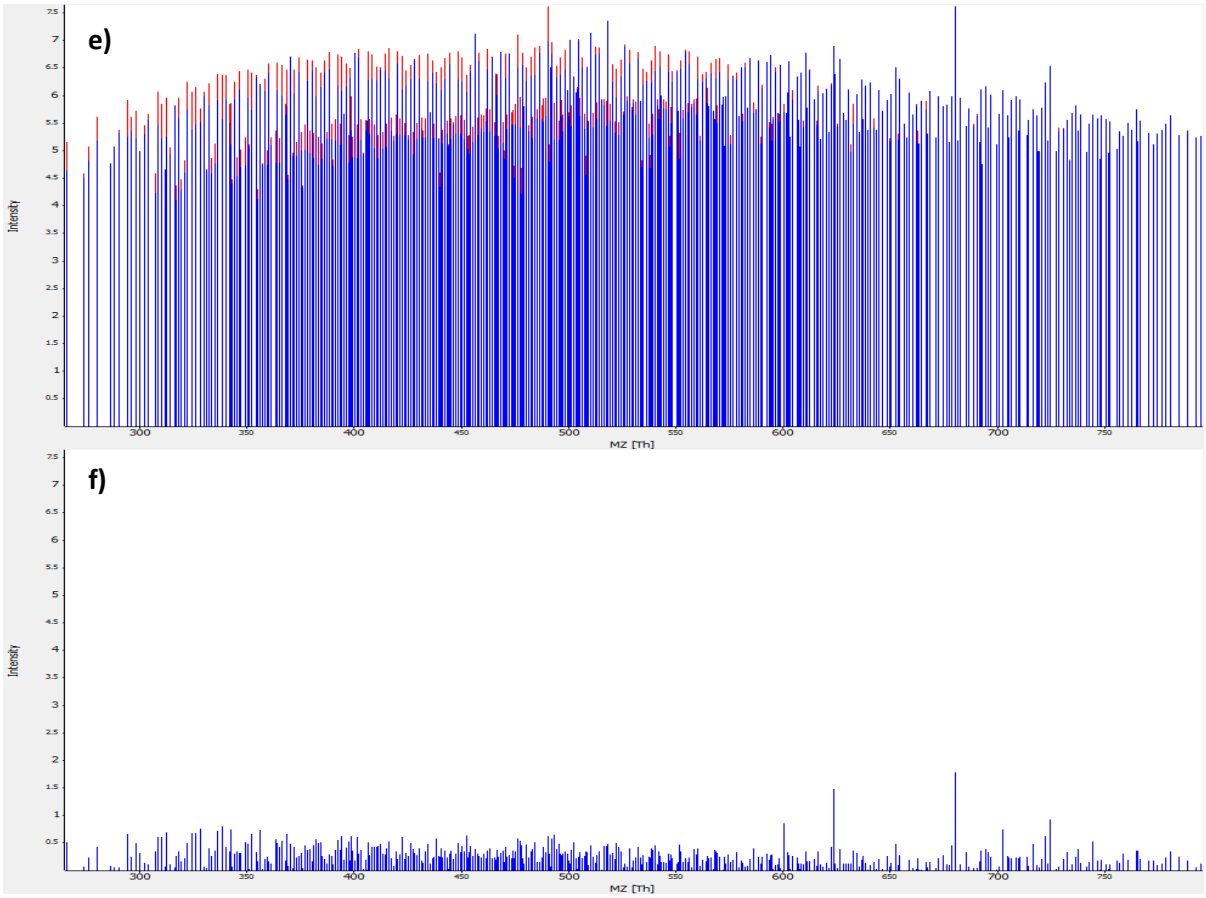


Figure 6.5: (a) A log-normalised crude oil spectrum of sample 13. (b) The mask spectra calculated using the SA algorithm. (c) The property spectra for pour point temperature (red) and sulphur content (blue). (d) The calculated spectrum for sample 13. (e) The original (blue) and calculated (red) spectra overlaid. (f) The absolute error between the original and calculated spectra for sample 13.

6.3 Discussion

Table 6.1 shows that there is a significant difference in the success of modelling the four crude oil properties. While property A appears to be modelled fairly successfully with an average error of 11.76%, property B is only modelled very poorly (69.87%), while properties C and D can only be modelled in a very small number of samples, but cannot be modelled in general.

The difference in success of modelling between the different properties is directly linked to the coefficient of variation (CV) for the properties and could suggest that the model is not working at all and is merely guessing solutions. The much lower CV for property A implies that a random guess at predicting a value for pour point temperature is more likely to be successful than a random guess at property D, whose CV is significantly larger. The CVs or standard deviations of the datasets as a percentage of the means of those data for each property are shown in table 6.2.

	Property A (Pour Point)	Property B (API Gravity)	Property C (TAN)	Property D (Sulphur)
Mean	272.76	31.81	0.44	0.97
Standard Deviation	22.19	8.35	0.96	1.13
Coefficient of Variation	8.13%	26.23%	216.09%	116.77%

Table 6.2: The mean, standard deviation and coefficient of variation for the crude oils' property data.

6.4 Conclusion

The results of the simulated annealing genetic algorithm hybrid technique show that the crude oil data is very difficult to model successfully. While the simulated annealing algorithm has been successful in creating artificial spectra that are capable of recreating a crude oil spectrum once optimally weighted and combined linearly with one another, the genetic algorithm failed to predict the physical properties of an unknown crude oil using these spectra as "building blocks". It has been shown that not all samples can be modelled with the same level of success and that some samples can in fact have their physical properties predicted with some level of accuracy.

References

Baker J.E. (1985), Adaptive Selection Methods for Genetic Algorithms, *Proceedings of the 1st ICGA*, Pittsburgh, PA, 101 – 111.

Eckhart R. (1987), Stan Ulam, John Von Neumann and the Monte Carlo Method, *Los Alamos Science Special Issue*, 131 – 137.

Kirkpatrick S., Gelatt C. D. and Vecchi M. P. (1983), Optimization by simulated annealing, *Science*, 220, 671 – 680.

Matsumoto M. and Nishimura T. (1998), Mersenne twister: a 62three-dimensionally equidistributed uniform pseudo-random number generator, *ACM Transactions on Modelling and Computer Simulation,* 8(1), 3 – 30.

Metropolis N. and Ulam S. (1949), The Monte Carlo Method, *J. Amer. Stat. Assoc.,* 44(247), 335 – 341.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, A. (1953), Equation of state calculations by fast computing machines, *J. Chem. Phys.*, 21, 1087 – 1092.

Michalski R.S., Carbonell J.G. and Mitchell T.M. (eds.) (1983), Machine Learning: An Artificial Intelligence Approach, *Morgan Kaufmann*, San Mateo, CA.

Russell S.J. and Norvig P. (1995), Chapter 18: Learning from Observations, Artificial Intelligence: A Modern Approach, *Prentice Hall*, Upper Saddle River, NJ.

Sun R. and Bookman L.A. (1995), Computational Architectures Integrating Neural and Symbolic Processes, *Kluwer Academic Publishers*, Boston, MA.

Tsai C.F. and Wang S.P. (2009), Stock Price Forecasting by Hybrid Machine Learning Techniques, *Proceedings of the International MultiConference of Engineers and Computer Scientists, IMECS*, Hong Kong, China.

CHAPTER 7 – Spectral Clustering

CHAPTER 7 contains a brief overview of clustering techniques, including principal component analysis and self organising maps as applied to the spectral data, and shows how samples with similar physical properties can be grouped by similar mass spectral features.

7.1 Spectral Clustering

It is not always possible to successfully model data as it may be too complex, too noisy or there may be insufficient correlation between the known data and the variables that are to be modelled. In such cases it may be beneficial to use clustering techniques in order to identify and group together similar samples. Clustering can be used either as a pre-processing step in order to identify features of the data that vary with changes in physical properties or as a final analytical tool. Within the scope of this project it may well suffice to be able to class a new unknown crude oil into a category with other similar crudes (e.g. heavy, light or medium) based on its mass spectrum.

A vast amount of research has been conducted into the field of spectral clustering, mainly because in areas such as cancer prognosis it is not necessary to model samples, simply to classify new data into two categories, ill and healthy; an area of research in which clustering techniques are commonly used (Patel and Sinha, 2010).

Cluster analysis, first discussed by Tryon (Tryon, 1939) is the discipline of grouping similar objects into categories or classes.

There are a number of different methods of clustering commonly used for statistical data analysis and machine learning, which can be broadly subcategorised into hierarchical (or tree) clustering and k-means clustering (Hartigan, 1975).

In hierarchical clustering, every data point is initially treated as a single class. The two most similar classes as determined by a distance or dissimilarity measure between them are then combined to form a new class and the process is repeated until a preset

criterion is reached. The determination of separate clusters depends on the definition of the maximum linkage distance, above which two separate clusters are no longer merged. This distance can be defined in a number of ways, the most common of which is the Euclidian distance given by equation 7.1.

$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{7.1}$$

Methods which start with a large number of independent clusters and then merge them to form larger clusters are known as bottom-up or agglomerative methods. Methods in which all data points are step-wise allocated into new clusters are known as top-down or divisive (Tan, Steinbach and Kumar, 2006). Hierarchical clustering is most commonly visualised using dendrograms (figure 7.1).

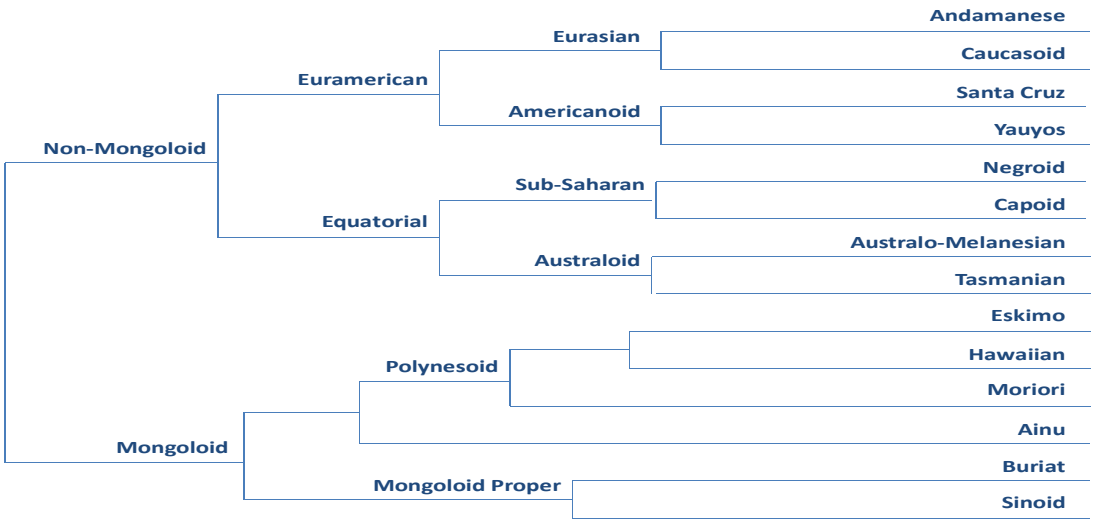


Figure 7.1: A dendrogram showing hierarchical clustering of craniometric variations in *homo sapiens* (Pontikos, 2004).

The second of the two clustering methods, k-means clustering, works by first defining the total number of clusters, k, that the data are to be separated into. The k clusters

are then selected at random and data points are allocated to each of the clusters based on their Euclidian distance from each cluster centroid. Once each data point has been assigned to a cluster, the cluster centroids are moved to a new position equal to the mean of the data points within the cluster. K-means is the most commonly used clustering technique (Hastie, Tibshirani and Friedman, 2009).

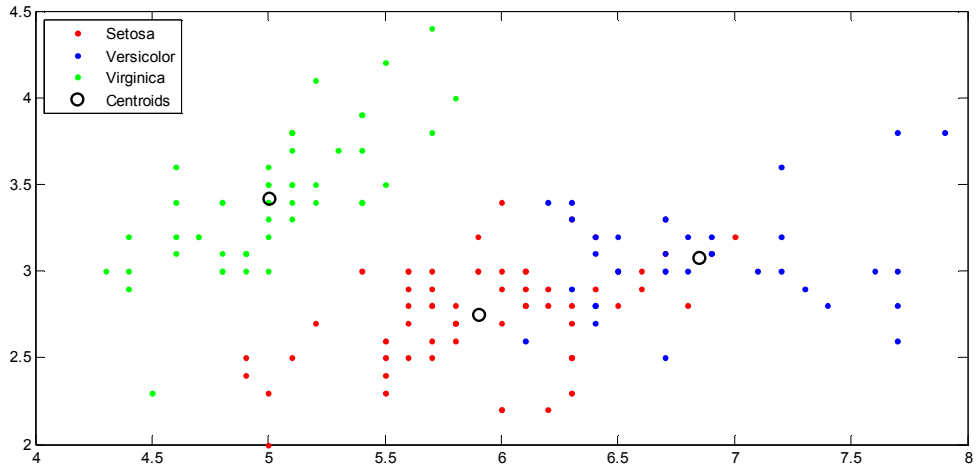


Figure 7.2: K-means clustering of the iris dataset.

While classification and subsequent clustering of data is used primarily for visualisation, it can be further applied for predicting class affinity of unknown data, known as regression.

There are a number of methods used for regression, including linear or partial least squares (PLS) and k-nearest neighbour (kNN) as described in Hastie, Tibshirani and Friedman (2009).

The principal difference between these types of predictive classifiers is the method by which class or cluster boundaries are determined. In a linear classifier, a straight line threshold is determined, samples on either side of which fall into different categories.

An example of a linear predictor, PLS (Wold *et al.*, 2001) determines a linear relationship between the predictor variables X and the response variables Y (equation 7.2).

$$Y = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \cdots + b_pX_p \quad (7.2)$$

K-Nearest Neighbour methods define a neighbourhood surrounding any one data point and determine the most common class for members of that neighbourhood. The data point is then assigned to the class which represents the majority of the members of the neighbourhood. A further example of a neighbourhood based classifier is kernel regression, which makes predictions based on a kernel function which decreases exponentially as a function of distance away from the data point. A common example of kernel regression is the support vector machine (SVM) (Cortes and Vapnik, 1995). SVMs work by mapping data into a higher-dimensional space in which classes become more easily linearly separable and determine a hyperplane boundary in multidimensional space which separates data of different classes.

Finally, there exist probabilistic classifiers, such as the Bayes' classifier, which utilise any of the above threshold or boundary rules and then apply a probability function $P(G|X)$, governing the likelihood that data point X belongs to class G.

While it is possible to use machine learning techniques such as neural networks for classification tasks, it is of interest to test purpose built classifiers for this problem, some of which are described below.

7.2 Principal Component Analysis

Principal component analysis (PCA), as described in chapter 3 is a method of identifying those features within a dataset which have the greatest variance from sample to sample. The method has been described at length in a previous chapter and the description will not be repeated here.

While PCA is primarily used as a means of reducing dimensionality of data, it lends itself to visualisation of complex data and is thus commonly used in cluster analysis.

The principal components of a dataset are weighted linear combinations of the original variables, so that they are rotated into a new co-ordinate system in which each dimension describes a source of variability in the data that is not described by any other component (equation 7.3).

$$c_1 = b_{11}(x_1) + b_{12}(x_2) + \ldots + b_{1p}(x_p) \quad (7.3)$$

where c_1 is the first principal component, b_{qp} is the coefficient for the q^{th} principal component on the p^{th} variable and x_p is the p^{th} variable.

The result is a rotated dataset of q variables or components that describes the majority of the original p-dimensional dataset, even though $q \ll p$.

Since it is difficult to graphically represent a dataset that spans more than three dimensions, the dimensionality reduction achieved by PCA is well suited to plotting a multidimensional dataset into just two or three dimensions.

As each of the principal components is likely to hold crucial information about the samples, the choice of principal components to utilise for the graphical representation is not trivial. A plot of one principal component against another shows the projection of the variance within the dataset onto a two-dimensional grid. The first and second principal components contain the greatest variance within the dataset and it is for that reason that they are most suitable for cluster analysis, as the formation of distinct clusters requires certain levels of variability within the dataset.

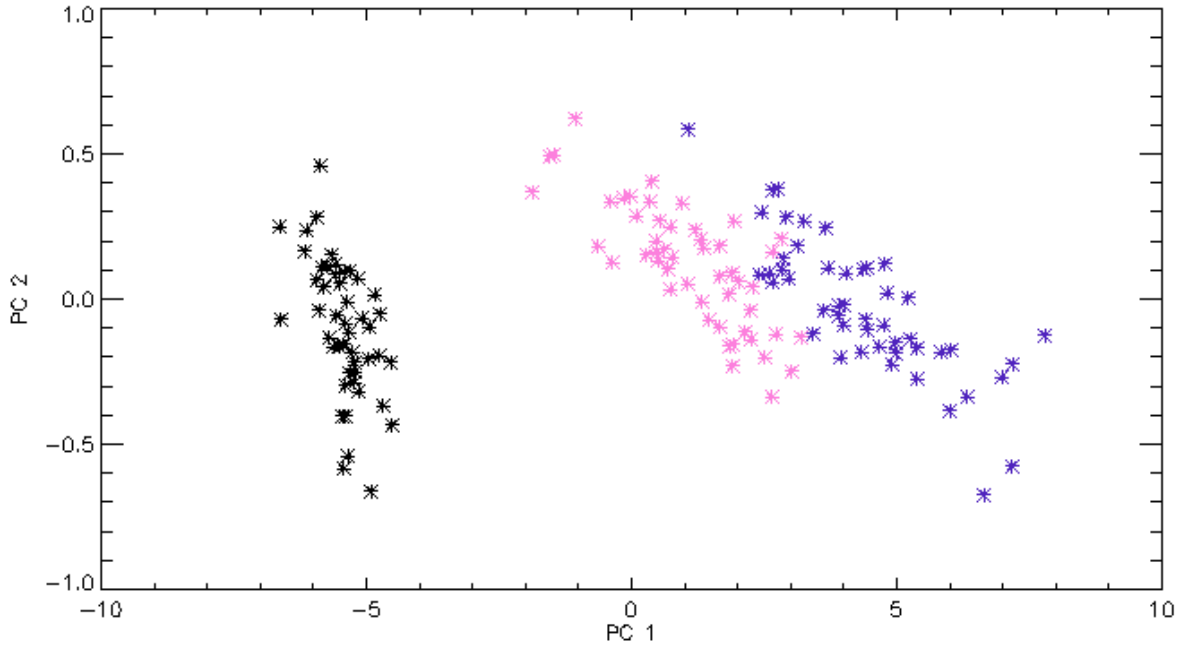


Figure 7.3: Clustering of three types of iris flower data according to their first and second principal components. Type A (Setosa) shown in black is clearly separated from types B (Versicolor) in pink and C (Virginica) in blue, which show low levels of overlap.

A scatter plot of higher rank PCs is less likely to show distinct clusters but may be successful in isolating experimental outliers or samples which are notably different from the rest.

7.2.1 Clustering of Crude Oil Spectra

In terms of classification, samples can be grouped by similar factor loadings on particular components.

By plotting principal components against each other one can create a scatter plot of the samples and try to form clusters of like crudes.

Figure 7.4 shows a scatter plot of the first and second principal components, highlighting a number of different crude oils which can be grouped together as determined by their physical properties.

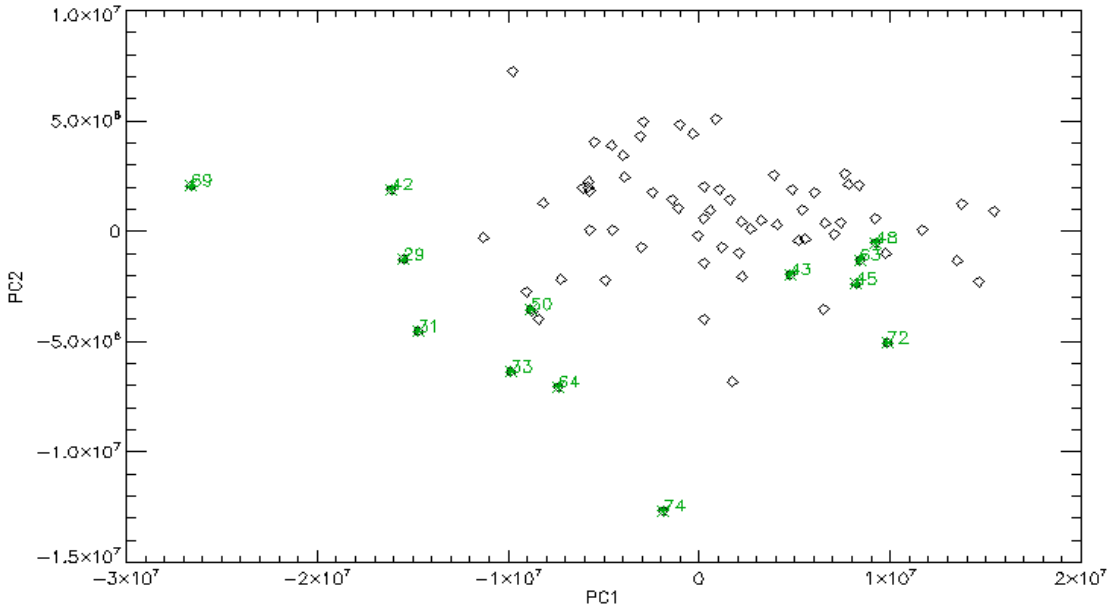


Figure 7.4: Scatter plot of the first and second principal components of the crude oil dataset.

Highlighted samples all originate from the Far East and are characterised as being light and low in sulphur. Samples to the left of $x = 0$ have low kinematic viscosity at 50°C, while those to the right have higher viscosities.

7.3 Self Organising Maps

Self organising maps, as described in detail in chapter 4, are a very effective method of visualising high dimensionality complex data by mapping them onto a two-dimensional topological map.

SOMs work by taking an input vector and mapping it onto a two-dimensional feature space by calculating the Euclidean distance between the input vector v and any given node (x_i, y_j) in the feature space, represented by a weight matrix w_{ij}. Once the node with the least distance between it and the input vector, the best matching unit, has been identified, the weights corresponding to that node and any node that falls within a given neighbourhood are changed to become incrementally more like the input vector. The SOM is trained by iterating through a training set of samples for a number of training epochs, while continuously adjusting the weights. Once the network has been trained, the mapping phase begins, in which the network is presented with a previously unknown sample and using the trained weights identifies the BMU and maps the sample onto the two-dimensional surface.

Samples which are similar will have their input vectors mapped onto the same or adjacent nodes on the map.

The final topography of the map shows clustering of like and unlike samples.

While in the simplest case each input sample would be represented by only one set of input co-ordinates (x_i, y_i), the crude oil data are significantly more highly dimensional.

This presents problems with the mapping, as the data must be scaled down from 712-dimensional to only two-dimensional. Since each of the 712 original variables will be weighted equally, the SOM is unlikely to perform effectively given only the raw data as input, as it has already been discovered that not all of the 712 peaks identified in every crude oil are of importance in determining the differences between the physical properties of the crude oils.

Owing to the fact that a SOM is primarily a data visualisation tool, the graphical representation of the output is crucial in analysing the results of the SOM. The most common way of data representation is the U-matrix (unified distance matrix), in which the distances between neighbouring nodes are represented by differently coloured hexagonal cells. Each node is linked to its six neighbouring nodes by a hexagonal tile whose colour index is by convention chosen to be directly proportional to the distance between the two nodes. Light coloured tiles represent small distances between nodes and thus areas of lighter colour indicate clusters, while dark colours indicate large distances and thus separation between clusters.

Another method of data visualisation is that of hit histograms (Iivarinen *et al.*, 1994), which indicate the number of samples which have been allocated to any one node in the network. Nodes with a high hit rate indicate clusters of similar samples, while nodes that have only single hits suggest that those samples are unlike any other sample analysed by the network. These hit rates, also known as relative signal frequencies, are an important aspect of network geometry in growing cell structures and are discussed in more detail in section 7.4.2.

7.3.1 The Algorithm

The self organising map algorithm used in this project, a summary of which is given by figure 7.5, is written in IDL and executable on any standard desktop computer.

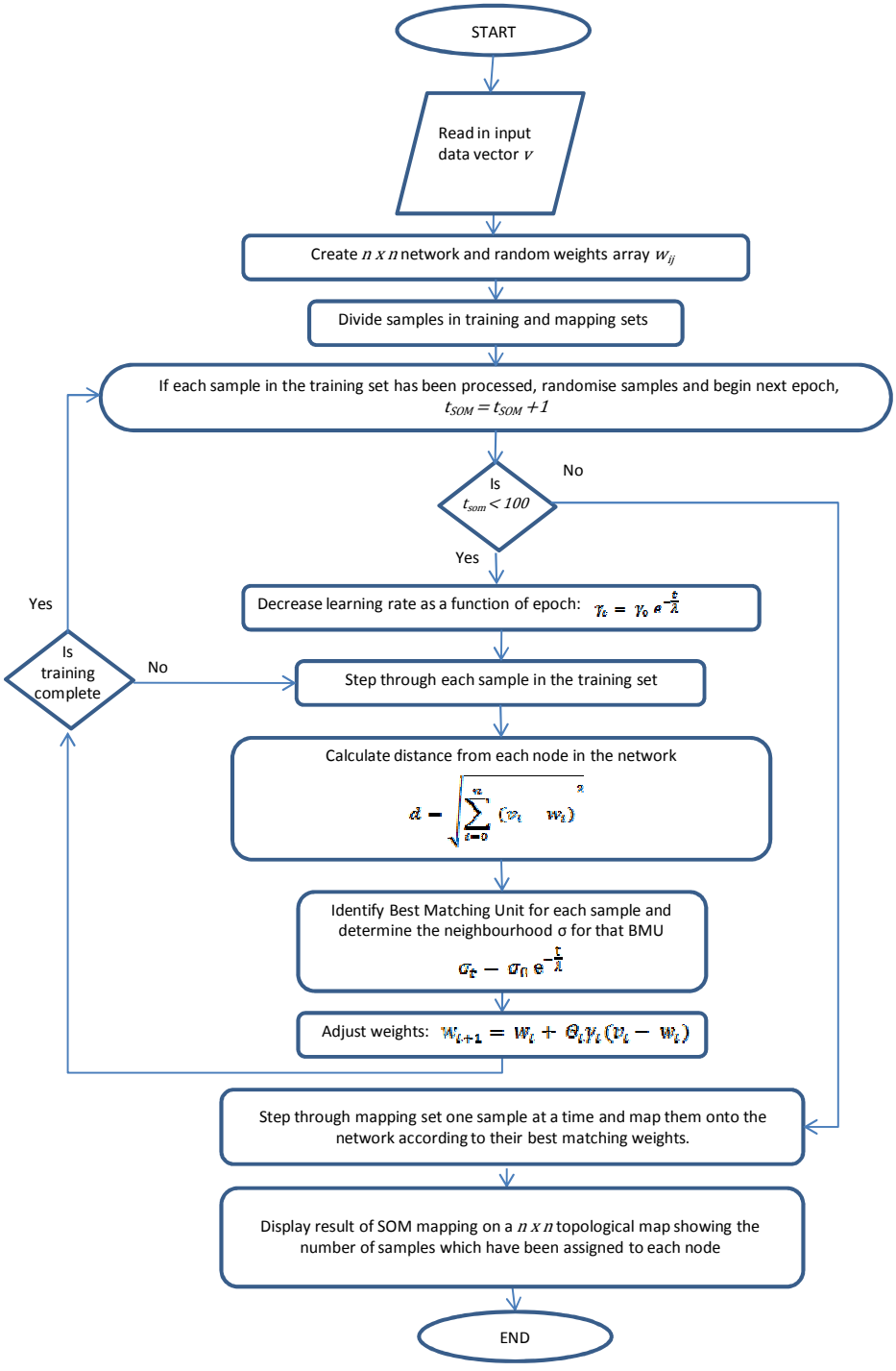


Figure 7.5: Flowchart of the SOM algorithm.

The input data is dataset A, the 712 common peaks present in all 76 crude oil spectra. As the input data consists of the intensities of the peaks in the crude oil spectra, the range in values between individual data points in the input vectors is too large for successful mapping to take place as strong peaks would significantly overpower weaker ones. For this reason the dataset is normalised so that the variance of peak intensities from sample to sample is equal to one (Vesanto *et al.*, 1999), according to equation 7.4.

$$x_i' = \frac{x_i - \bar{x}}{\sigma} \tag{7.4}$$

where x_i' is the statistically normalised data, x_i is the i^{th} input variable, $\bar{x}$ is the mean of the data and σ is the standard deviation of the data for a particular peak position. The normalised data has a mean of zero and variance of one.

The Kohonen network onto which the input data is mapped is of the dimensions 7×7. The adjustable weights w_{ij} for each node are created at random to take values of the range $0 \leq w_{ij} \leq 1$. The weights matrix has dimensions $7 \times 7 \times 712$.

The algorithm performs 1,000 training epochs. In each epoch, a sample is selected at random and the Euclidean distance between the sample's input variables x_i and the weights of each node in the network, w_{ij}, is determined.

Once the node with the least distance to the input vector has been determined, the BMU, the node's weights and those of each node in the neighbourhood are modified to move closer to the input vector x. The size of the neighbourhood σ_t and the learning rate γ_t decrease as a function of time t according to equations 7.5 and 7.6 respectively.

$$\sigma_t = \sigma_0 e^{-\frac{t}{\lambda_n}} \tag{7.5}$$

$$\gamma_t = \gamma_0 e^{-\frac{t}{\lambda_l}} \tag{7.6}$$

where λ_n and λ_l are time constants for the neighbourhood function and learning rate function respectively.

Finally, the adjustment factor Θ_t, which determines the amount by which any one node's weights are modified in each iteration as a function of its proximity to the BMU, is exponentially decreased in accordance with equation 7.7.

$$\Theta_t = e^{-\frac{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}{{\sigma_t}^2}} \tag{7.7}$$

Once each sample of the training set has been allocated to a BMU and the weights have been adjusted, the next training epoch begins; the epoch-dependant parameters are adjusted before the process restarts by being presented with the samples in a new random order. At the end of a completed training phase, the trained map is introduced to the mapping sample set, a set of data previously unseen by the network. Each sample is allocated to a node in the map according to the smallest Euclidean distance between the input vector and the node's weights.

Once each sample of the mapping set has been mapped onto the network, the data is visualised by showing the number of samples allocated to each node.

7.3.2 Results

Table 7.1 shows the results of the SOM algorithm displayed in terms of samples allocated to each node. Some clustering of samples is apparent along the edges of the map. This suggests that the differences between samples are much more significant than the similarities between them; therefore the network more easily achieves separation between unlike samples than clustering of like samples.

Repeated analysis of the sample set failed to consistently reproduce the same clusters; however a small number of samples are reliably singled out as "different" from the rest.

	1	2	3	4	5	6	7
1	4,7,28,39,61,67	52,62,74	32,64,71		29,42		69
2	9,17,23,36,40	24	10,51		3		31,33
3	43,45	11,25		46	1,2,5		50
4	27,59	46,65		37,49	76	58,68	73
5	26	35,41	15	34		6,19	13,44
6	56,65	38	53	12	21	8	54
7	20,55	16	18	48,60,63,72	30	14,22,57,70	75

Table 7.1: Node assignment for each of the 76 samples on a 7 × 7 map after 1,000 training cycles.

There appear to be two distinct classes, one in the top left corner and another in the bottom right. The fact that these two clusters have formed at opposite ends of the map suggests that there are great dissimilarities between the samples. Inspection of the physical properties which correspond to the samples show that the crude oils which

have formed clusters in the bottom right are predominantly light, sweet crudes (14, 22, 57 and 70 and 48, 60, 63 and 72 respectively), while the clusters at the opposite region of the map are mostly medium crude oils which all contain large amounts of sulphur (4, 7, 28, 39, 61 and 67 and 9, 17, 23, 36 and 40).

None of the physical properties result in conclusive cluster formation on the self organising map, however it is interesting to note that samples which are successfully separated by the map are in fact dissimilar in terms of their physical properties.

7.3.3 Discussion

A summary of the algorithm:

1. Statistically normalise input data to set variance of each variable to one.

2. Initiate $n \times n$ network and create random weights.

3. In each of 1,000 epochs take one sample from the training set at a time and calculate the Euclidean distance between that sample and each of the $n \times n$ nodes' weights. Determine BMU and neighbourhood for each sample.

4. Update weights.

5. Reduce learning rate and neighbourhood function and start new epoch.

6. Once all epochs have been completed, take each sample from the mapping set and map onto the trained network.

7. Create a U-matrix or similar graphical representation to show clustering of samples.

As with all learning algorithms, further manipulation of the learning parameters is likely to help improve the performance of the SOM. A decrease in the size of the neighbourhood function, coupled with an increase in the number of training epochs is likely to improve the classification of crude oils; however this improvement would come at great computational expense.

7.4 Other Clustering Techniques

There exist a number of other clustering techniques which will be discussed in the following section; however these have not been applied to the data due to time constraints. The clustering techniques mentioned below are thought to be less effective in cases such as this one and are thus not likely to provide results that are any better than those obtained from both PCA and SOM.

7.4.1 Sammon Projection

A data analysis method commonly used in place of PCA or SOM is that of Sammon projection, first described by J.W. Sammon in 1969. Sammon projection, while similar to both SOM and PCA in reducing multidimensional data by mapping it onto a lower dimensional surface, is a non-linear mapping algorithm and therefore does not lend itself easily to visual cluster analysis.

In Sammon projection, any two points $({x^*}_i,\ {y^*}_i)$ and $({x^*}_j,\ {y^*}_j)$ in multidimensional space are separated by distance ${d^*}_{ij}$ and their respective projections $(x_i,\ y_i)$ and $(x_j,\ y_j)$ in lower dimensional space are separated by distance d_{ij}. The error function

between the original dataset and its low-dimensional projection is given by equation 7.8, which is reduced in an iterative process.

$$E = \frac{1}{\sum_{i<j}^{N} d^*_{ij}} \sum_{i<j}^{N} \frac{(d^*_{ij} - d_{ij})^2}{d^*_{ij}} \quad (7.8)$$

Sammon projection can however be used as a visualisation method of the resulting nodes of a SOM, and it is therefore commonly used in conjunction with a self organising map algorithm (Vesanto, 1997) in order to visualise the overall shape of the map.

7.4.2 Growing Cell Structures

Another commonly used clustering method is growing cell structures (Fritzke, 1993), which is largely based on SOM.

Growing cell structures (GCS), unlike SOM, can automatically adapt to create the ideal network size to model the data. While SOMs have a predefined number of nodes that are used to map the data, GCS can add or delete nodes during the iteration process in order to adapt to the size and structure of the data.

Furthermore, GCS does not apply a time-dependant adjustment factor nor a decaying neighbourhood function. Only the best matching unit and its immediate topological neighbours are adjusted and thus the need for a cooling schedule is eliminated (Fritzke, 1993).

Each existing node in the network carries an independent hit counter τ, recording the number of samples it has been identified as best matching unit for. The counter is used to determine the relative signal frequency h for each node according to equation 7.9.

$$h_i = \frac{\tau_i}{\sum_{j=1}^{n} \tau_j} \tag{7.9}$$

where τ_i is the total number of hits for the i^{th} node and n is the total number of nodes.

The ultimate aim of GCS is to ensure that all nodes have a similar relative signal frequency. After each training epoch the node with the highest relative signal frequency is identified and a new node is inserted between this node and its furthest immediate neighbour. Insertion of nodes helps to distribute the load of BMU hits more evenly across the network. Similarly, if a node has zero or very low relative signal frequency and therefore the probability of that node being identified as BMU for any input vector is small, the node can be removed.

The act of introducing new nodes in areas of high probability density and removing those in low probability density regions allows the GCS to adapt to the size and structure of the data in ways which are superior to a SOM. Nevertheless, the increased difficulty in visual representation of GCS makes this method less favourable than SOMs if the aim is to graphically represent the clustering of data (Fritzke, 1993).

7.5 Discussion

Both the PCA and SOM show similar cluster patterns. The crude oils are highly complex and it is as such not possible to clearly separate all the samples into distinct classes; however we can see that some samples are consistently clustered together. The SOM shows that samples have a tendency of aligning themselves along the edges of the map, suggesting that the differences between samples are more distinct than the similarities between them.

It has been further noted that there exists significant overlap between some clusters which may lead to ambiguity when using the clustering technique as a class predictor on unknown data.

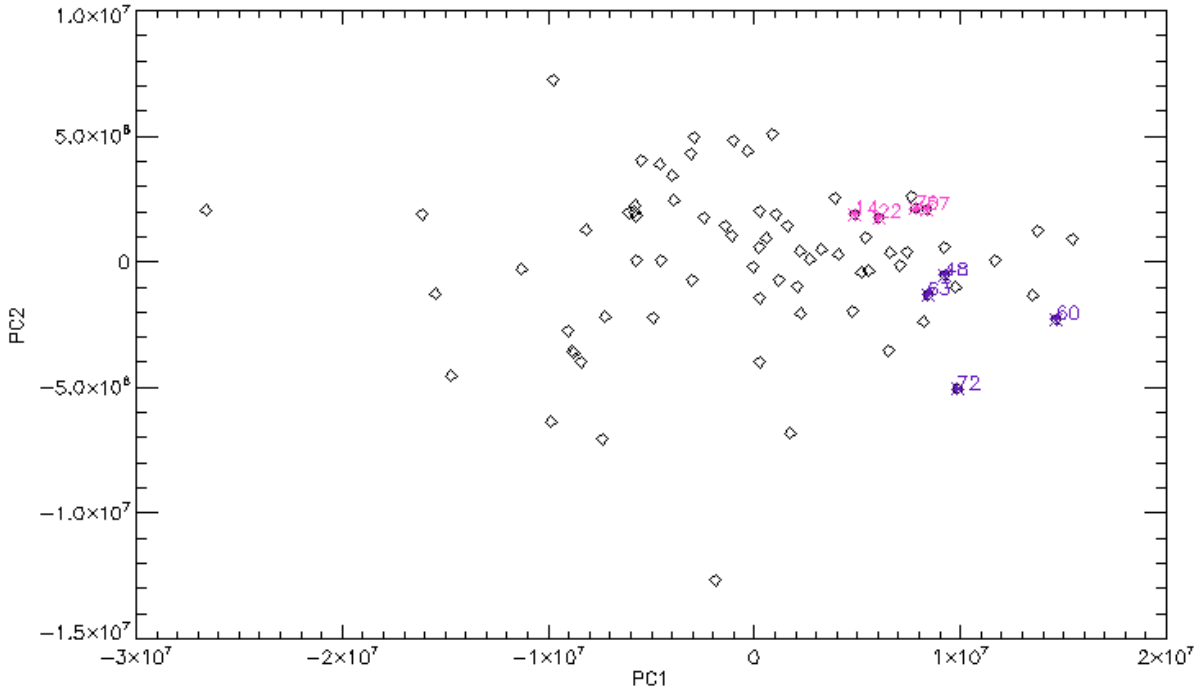


Figure 7.6: Identification of some of the groups of crude oils that were assigned to the same node in the self organising map in the PCA scatter plot. Samples 14, 22, 57 and 70 show distinct clustering in both methods.

Figure 7.6 highlights the set of samples which formed two of the clusters in the self organising map on a PCA scatter plot. Samples 14, 22, 57 and 70 clearly show cluster formation in both methods, confirming that they must share some similarities. Inspection of the physical properties of the samples show that they all have high API gravities, very low sulphur contents and low TAN, confirming the trend observed in both SOM and PCA analyses.

References

Cortes C. and Vapnik V. (1995), Support-Vector Networks, *Machine Learning*, 20, 273 – 297.

Fritzke B. (1993)**,** Growing Cell Structures: A Self-Organizing Network for Unsupervised and Supervised Learning, *Technical Report, TR-93-026, Int'l Computer Science Inst.*, Berkeley, CA.

Hartigan J.A. (1975), Clustering Algorithms: Probability & Mathematical Statistics, *John Wiley & Sons Inc.*, New York, NY.

Hastie T., Tibshirani R. and Friedman J. (eds.) (2009), The Elements of Statistical Learning: Data Mining, Inference and Prediction, Second Edition, *Springer*, New York, NY.

Iivarinen J., Kohonen T., Kangas J. and Kaski S. (1994), Visualizing the clusters on the self-organizing map, *Proc. Conf. on Artificial Intelligence Res.*, Helsinki, Finland, 12, 122 – 126.

Patel B.C. and Sinha G.R. (2010), An Adaptive K-means Clustering Algorithm for Breast Image Segmentation, *J. Computer Applications*, 10(4), 35 – 38.

Pontikos D. (2004), Model-Based Clustering of World Craniometric Variation, *unpublished*.

Sammon J.W. (1969), A Nonlinear Mapping for Data Structure Analysis, *IEEE Transactions on Computers*, 18(5), 401 – 409.

Tan P., Steinbach M. and Kumar V. (eds.) (2006), Introduction to Data Mining, *Addison-Wesley*, Boston, MA.

Tryon R. C. (1939), Cluster Analysis, *Edwards Brothers*, Ann Arbor, MI.

Vesanto J., Himberg J. Alhoniemi E. And Parhankangas J. (1999), Self-organizing map in Matlab: the SOM Toolbox, *Proceedings of the Matlab DSP Conference 1999*, Espoo, Finland, 35 – 40.

Vesanto J. (1999), SOM-based data visualization methods, *Intelligent Data Analysis*, 3, 111 – 126.

Wold S., Sjöström M. and Eriksson L. (2001), PLS-regression: a basic tool of chemometrics, *Chemometrics and Intelligent Laboratory Systems*, 58, 109 – 130.

CHAPTER 8 – Conclusion

CHAPTER 8 concludes this thesis with an in-depth discussion of the results of this research, draws any conclusions to be made and highlights any future work that is recommended.

8.1 Results

This project encompassed a vast range of data manipulation and analysis techniques, ranging from simple processing of the data to in-depth analysis thereof. The results of the different techniques applied throughout the project are summarised in the following.

8.1.1 Data Reliability

Analysis of the day-to-day and run-to-run variability of the data show that the method chosen for data acquisition, while well suited for the analysis required, lacks in reliability.

Investigation of the change in average intensity of peaks of any one sample spectrum across a number of repeat measurements shows fluctuations in standard deviation of an average of 9.5% of the mean intensities of the spectra, which are significant, considering that deviation from the mean intensity across the whole set of samples can be as low as 9.8% and is on average only 31.3%.

Furthermore, despite careful calibration of the instrument and the output data, mass accuracy is on average in the 1ppm range, which is not always considered sufficient, especially when unambiguously identifying components in the high mass ranges (500-1,000Da) (de Hoffmann and Stroobant, 2002).

The validity of all data analyses in this project relies heavily on the reliability of the data and the reproducibility of the spectra that are being analysed. If a model is to be rigid in

predicting physical properties of a crude oil with high levels of precision, even if those properties vary only by small amounts from sample to sample, one must be sure that the data acquisition techniques are robust and accurate.

8.1.2 Artificial Neural Network Modelling

The neural network algorithm described in this research has enjoyed varying levels of success when analysing the highly complex data. The network geometry struggles to model the unprocessed raw data due to the extremely high dimensionality and even the reduced dimensionality of the 712 common peaks dataset (dataset A) still proves very highly dimensional.

Due to the summation of the data points, any highly dimensional dataset stands to suffer a loss of precision and detail, as the fine points which may be responsible for the most valuable information are overshadowed by the bulk of the data which may be irrelevant or redundant. The modelling errors for properties pour point, API, sulphur content and TAN when modelling dataset A were 7.33%, 15.97%, 409.97% and 504.39% respectively.

It can be observed that the properties pour point and API have higher success rates at being modelled, which is likely due to the decreased variability of the output datasets and therefore a random prediction of modelled output has a higher probability of being "guessed" correctly. The standard deviations of properties API and pour point as a percentage of the mean of the data are 26.23% and 8.13% respectively, compared with 216.09% and 116.77% for properties TAN and sulphur content.

The figures suggest that the network is currently unable to learn anything other than rough estimates of each of the properties.

Prior processing of the data in order to eliminate the high-dimensionality and help to extract those features which are most likely to elucidate meaningful correlations between input and output data lead to slightly higher prediction rates, albeit not significant enough to declare the neural network as a reliable modelling technique.

Analysis of dataset C, consisting of the PCA data of the spectra, improved the prediction error of sulphur content and TAN significantly to 94.99% and 96.05%, while fractionally decreasing the prediction accuracy for pour point and API to 7.37% and 18.74% respectively. One can still observe the previously mentioned trend, where physical properties with lower variance across the dataset are more easily modelled, mainly due to the fact that random prediction is more likely to "guess" a correct value.

The average prediction error for each analysis is larger than the overall variance of the datasets, suggesting that the method is currently inconclusive.

8.1.3 Simulated Annealing Genetic Algorithm Modelling

The greatest amount of modelling success has been observed with the hybrid algorithm, SAGA.

While the final prediction of physical properties A, B, C and D has not yielded statistically significant results at prediction errors of 11.76%, 69.87%, 102.96% and 73.06% respectively, the reproduction of a crude oil spectrum using the linear combination of mask and property spectra has lead to an average error between

spectra of only 4%, which is significantly lower than the average sample to sample variability and can therefore be treated as significant.

8.1.4 Principal Component Analysis Clustering

Clustering of samples has shown relative levels of success, as samples belonging to different classes can be shown to form clusters in a number of PC scatter plots. Samples with very high and very low sulphur contents are particularly successfully at being separated in the two-dimensional plane according to their principal components.

Despite this successful clustering of crude oils according to their physical properties, the clusters are not spatially defined to a point where they could be used for prediction of class affinity of unknown samples, as too much overlap exists between different clusters.

Nevertheless, it is interesting to note that samples which have been placed near one another on the SOM can also be clustered in the PCA.

8.1.5 Self Organising Map Clustering

SOM clustering proves slightly more successful than PCA due to the improved accuracy of the technique. The SOM shows very clear cluster formation, which can be shown to correspond to samples sharing similar physical properties, most notably very high or very low sulphur contents.

While much of the cluster formation cannot be directly identified, it is interesting to note that some samples (#36 and #74) are consistently placed near one another on the

topological map, while others (#30) are consistently placed apart from the bulk of samples, suggesting that they are unlike any of the other samples.

8.2 Discussion

The results quoted in the previous section confirm that while there are low levels of success to some of the modelling algorithms, this project has failed to consistently predict the physical properties of complex crude oils using their mass spectra and is therefore currently unsuitable for the purposes outlined in chapter 1.

While the theory suggests such a task to be feasible, limitations of the instrument used to gather the data and the algorithms used for modelling, as well as suboptimal data selection mean that the overall method is unlikely to prove conclusive at present.

Effective prediction of physical properties was only achieved for values of pour point temperature with average prediction errors of less than 15% for SAGA and less than 10% for the artificial neural network. While none of the other physical properties were able to be predicted with statistical significance, the SAGA algorithm demonstrated very strong capabilities of recreating the mass spectrum of a crude oil using only its assay data, with an average error between calculated and recorded peak intensity of just 4%.

The methods can be further used to successfully verify previous findings of similarity between samples and, given certain improvements, are likely to be able to provide sufficient evidence to confidently classify previously unknown crude oils and should eventually be able to predict a host of different physical properties.

It has been recognised that the limited success of the project is largely owed to the suboptimal choice of datasets to be modelled. The removal of the bulk of the data by extracting only the 712 peaks common to all spectra has discarded large amounts of valuable information, most notably the majority of each sample's sulphur and oxygen contents, and therefore severely hindered the prediction of some of the physical properties. It is believed that improvements in the process of feature extraction from the raw spectra as well as enhancement of the analytical methods used to gather the raw data will significantly improve the outcome of the project.

8.3 Future Work

While the success of this project was limited, there is nevertheless sufficient evidence to suggest that improvements of the methods involved may yield more consistent and statistically important results in future.

With increasing accuracy of the FT-ICR MS instrument, achievable with increasing magnet sizes, data reliability can be significantly improved and sample-to-sample variance can thus be maximised, while also ensuring sufficient levels of reproducibility of data (Marshall and Rodgers, 2004).

Furthermore, there are a number of different experimental techniques that are yet to be explored, such as atmospheric pressure photo ionisation (APPI) data acquisition (Purcell *et al.*, 2006). As any modelling technique improves with increased sizes of training datasets, the more training data can be gathered the more the model can be optimised (Cartwright, 2008).

Improvements in the computational techniques and further enhancement of the algorithms are certain to lead to increased levels of success in modelling the data. While research time has been evenly split between refining each of the five algorithms, further focus on any one of these methods may help improve the algorithm and thus increase its analytical power.

The neural network for example may be further fine-tuned by increasing or decreasing the number of hidden layers and nodes as well as adjusting the weights, biases and transfer functions used (Mitchell, 1997).

Similarly, the network geometry of the self organising map as well as the relatively small number of training epochs were chosen due to time constraints and could easily be improved, thereby giving the network a greater chance of successfully modelling the data.

More work should also be done in analysing the datasets used for modelling. Both input and output datasets were chosen fairly arbitrarily for lack of sufficient information as to where possible correlations between input and output data may lie. Further analysis of the individual datasets using more advanced methods of feature extraction such as genetic algorithms (Kudo and Sklansky, 2000) may help select better suited input and output datasets.

While this particular project cannot report conclusive results, successes in similar disciplines as well as the relative success of the SAGA method suggest that effective prediction of crude oil properties according to their mass spectra is still plausible and work on this particular project should most certainly continue.

References

Cartwright H.M. (2008), Using Artificial Intelligence in Chemistry and Biology: A Practical Guide, *CRC Press*, London, UK.

De Hoffmann E. and Stroobant V. (2002), Mass Spectrometry: Principles and Applications, *Wiley and Sons Ltd*, Chichester, UK.

Kudo M. and Sklansky J. (2000), Comparison of Algorithms that Select Features for Pattern Classifiers, *Pattern Recognition*, 33(1), 25 – 41.

Marshall A.G. and Rodgers R.P. (2004), Petroleomics: The Next Grand Challenge for Chemical Analysis, *Acc. Chem. Res.*, 37, 53 – 59.

Mitchell T.M. (1997), Machine Learning, *McGraw Hill*, New York, NY.

Purcell J.M., Hendrickson C.L., Rodgers R.P. and Marshall A.G. (2006), Atmospheric Pressure Photoionisation Fourier Transform Ion Cyclotron Resonance Mass Spectrometry for Complex Mixture Analysis, *Anal. Chem.*, 78, 5906 – 5912.

Acknowledgements

I would like to thank Dr Hugh Cartwright for his continued support and patience with my research.

Special thanks must also be given to Shell researchers Martin Selby at Thornton Technology Centre and Jerry Purcell at Westhollow Technology Centre, as well as Steve Hammond, Bessy Fan, Wim Genuit, Jean Nowlin, as well as Phil Jonathan, Murray Brown and Malcolm Salisbury for their knowledge and expertise, as well as their hospitality.

Furthermore, thanks to Shell Global Solutions for the funding of this project by means of a CASE award, as well as their financial support for conference attendances and several trips to Shell research centres.

Lastly, I would like to thank my friends and family for their love, support and (though sometimes limited) patience during this 'ordeal' and helping me juggle my countless other commitments. Thanks also to the Aspire girls and Oriel and Accies crews for suffering through moments of my sleep-deprived lunacy. Thank you to Samantha, Elizabeth, Tori, Hannah and Gillian.

Thank you, Mama, Papa and Vivi, for giving me the drive and self-belief to pursue my studies, and to Dr Milan Dlabal for making Physics 'cool'.

Thank you James, for making life more interesting.

APPENDIX A: Source Codes

A.I Artificial Neural Network

CrudeNeurNet.pro

```
;Program performs a Neural Network on one of three Crude Oil Datasets
;Network geometry is detailed below.
;Please specify input dataset 'A', 'B' or 'C' (A = 712 common peaks, B=50 most variable peaks, C=50 fist PCs

;
; Neural Net layout:

; Input:            A:      712x76      (Common peaks matrix)
;                   B:      50x76       (50 peaks with greatest variance across all spectra)
;                   C:      50x76       (First 50 principal components from pca_on_crudes.pro)
; Output: 4x76                          (1: Pour Point Temp, 2: API Gravity, 3: TAN, 4: Sulphur Content)
; Hidden layers: 2
; Nodes per layer: 5
; Bias: 1 per hidden layer
;
; If network is used for prediction, increase the training set and decrease the
; test set (e.g. 90%, 9%, 1%). For analysing the performance of the network, keep size of validation set large.
; Training set: 60%
; Validation set: 10%
; Test set: 30%
;

;  O
;  O
;  O
;  O   O  \      O   \                    O
;           O         (wij1) -   O          (wjk1) - O   (wkl1) \ O
;           O -       (wij2) -   O         - (wjk2) - O  - (wkl2) - O
;           O -     - O       - O
;           O -       (wij3) -   O         - (wjk3) - O  - (wkl3) - O
;           O         (wij4) -   O          (wjk4) - O   (wkl4) / O
;  O   O  /      O     /                    O
;  O
;  O
;  O

; Created:          13/02/2008 by Jennifer Hauschild (Cartwright Group, PTCL, Oxford University)
;
; History: 20/02/2008: amended to feature node-input arrays instead of vectors - JMH
;          27/02/2008: changed back to its initially and then expanded to include a validation section - JMH
;          18/05/2009: Program amended to read new crude data (gathered at Shell Westhollow, Houston) - JMH
;                              ...
;                              ...
;
;          14/03/2011: amended to make sure that training and validation section are completely independent - JMH
;          04/04/2011: amended to include third dataset input - JMH


pro CrudeNeurNet

;Load graphics settings
device, decomposed=0
loadct, 10

DAT=['A','B','C']

;Specify dataset
```

```
dataset='C'

noutputs=4                    ;number of output nodes
nl1=5                         ;nodes in layer 1
nl2=5                         ;nodes in layer 2
cyc=0
random2=RANDOMU(seed, 3, 5000)

;IF USING A BREAK CLAUSE IN THE VALIDATION SECTION, PLEASE SPECIFY MINIMUM AND MAXIMUM
;ACCEPTABLE ERRORS
;ERROR IS CALCULATED AS THE MEAN OF EL = ABS(YL - DL) ACROSS ALL MEMEBRS OF THE VALIDATION SET
;BREAK CLAUSE IS NOT IMPLEMENTED IN THIS VERSION OF THE CODE
MIN_ERR=1
MAX_ERR=100

;Load input data
if dataset eq DAT(0) then begin
        restore, 'C:\Users\Jennifer\Desktop\Crudes\Pos_dat\peaks_matrix.sav'
        ninputs=712
endif

if dataset eq DAT(1) then begin
        restore, 'C:\Users\Jennifer\Desktop\Crudes\Pos_dat\DatasetB.sav'
        ninputs=50
        new_mat=data
endif

if dataset eq DAT(2) then begin
        restore, 'C:\Users\Jennifer\Desktop\Crudes\Pos_dat\PCA.sav'
        ninputs = 50
        ;if using PCA data, must convert array names!!
        new_mat=fltarr(2,76,ninputs)
        new_mat(1,*,*)=transpose(tmp(0:ninputs-1,*))
endif

;Load property data
restore, 'C:\Users\Jennifer\Desktop\Crudes\Pos_dat\crude_props.sav'        ;Cargo Properties Dataset

nsamples = n_elements(A(0,*))

data = fltarr(ninputs+1,nsamples)
dl = fltarr(noutputs,nsamples)

;Convert Deg Celsius into Kelvin
A(0,*)=A(0,*)+273.

data(1:ninputs,*) = transpose(new_mat(1,*,*))
dl(0:noutputs-1,*) = A(0:noutputs-1,*)


;xxxxxxxxxxxxxxxxxxxxxx
;define node parameters
;xxxxxxxxxxxxxxxxxxxxxx
yl = fltarr(noutputs)
xl = fltarr(noutputs)
yi = fltarr(ninputs+1)
xj = fltarr(nl1+1)
yj = fltarr(nl1+1)
xk = fltarr(nl2+1)
yk = fltarr(nl2+1)

el = fltarr(noutputs)
ek = fltarr(nl2+1)
ej = fltarr(nl1+1)

;monitor performance by plotting error as a function of training cycle
```

```
;oplot command is not used in this version of the code
plot, [0,0], [0,0], xrange=[0,5000], yrange=[0,100], background=255, color=0


        ;scale input data so that the sum of each set of input variables is equal to or less than 1
        maxtot=0
                for n=0,n_elements(data(0,*))-1 do begin
                        newmax = TOTAL(data(0:ninputs-1,n))
                        if newmax gt maxtot then maxtot = newmax
                endfor ;k

        data(0:ninputs-1,*) = data(0:ninputs-1,*)/maxtot

        ;normalise output data -> max=1 then scale to 0.8
                for m=0,n_elements(A(*,0))-1 do begin
                        A(m,*) = A(m,*)/max(A(m,*))*0.8
                endfor ;m


;define training parameters
training_fraction = 0.60
test_fraction = 0.30
validation_fraction = 1.0-training_fraction-test_fraction
ntraining = FIX(nsamples*training_fraction)
nvalidation = FIX(nsamples*validation_fraction)
ntest = nsamples-ntraining-nvalidation

;Construct a Boolean to determine the order in which samples are picked. Sample_chosen (binary array) is set to 0 to
;start with. Once a sample has been chosen, the array entry is set to 1

sample_chosen=intarr(nsamples)
sample_chosen(*)=0

random = randomu(seed,nsamples)
training_ids = intarr(ntraining)
validation_ids = intarr(nvalidation)
test_ids = intarr(ntest)

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;RANDOMLY SELECT A PERCENTAGE OF THE SAMPLES TO FORM THE TRAINING, VALIDATION SETS
;AND TEST SETS
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

train=0
while train lt ntraining do begin
        for n=0,nsamples-1 do begin
                if (sample_chosen(n) eq 0) AND (train lt ntraining) then begin
                        if random(n) lt training_fraction then begin
                                sample_chosen(n)=1
                                training_ids(train)=n
                                train=train+1
                        endif
                endif
        endfor ;n
endwhile ;train

test=0
while test lt ntest do begin
        for n=0,nsamples-1 do begin
                if (sample_chosen(n) eq 0) AND (test lt ntest) then begin
                        if random(n) lt test_fraction then begin
                                sample_chosen(n)=1
                                test_ids(test)=n
                                test=test+1
                        endif
                endif
```

```
        endfor ;n
endwhile ;test

validation_ids(*)=where(sample_chosen ne 1)

;xxxxxxxxxxxxxx
;DEFINE WEIGHTS
;xxxxxxxxxxxxxx
wij = RANDOMU(seed, nl1+1, ninputs+1)
wkl = RANDOMU(seed, noutputs, nl2+1)
wjk = RANDOMU(seed, nl2+1, nl1+1)

;xxxxxxxxxxxxxxxxxxxxx
;STRART TRAINING CYCLE
;xxxxxxxxxxxxxxxxxxxxx
cyc=0

while cyc lt 10000 do begin
cyc=cyc+1

;xxxxxxxxxxxxxxxxxxxxxxxx
;ADJUST LEARNING RATE ETA
;xxxxxxxxxxxxxxxxxxxxxxxx
if cyc lt 7000 then eta=0.99
if cyc gt 7000 and cyc lt 9500 then eta=0.6
if cyc gt 9500 then eta=0.1


rand = RANDOMU(seed, ntraining)
order = fltarr(ntraining)

;RANDOMISE THE ORDER OF THE TRAINING SET
for n=0,ntraining-1 do begin
        biggest = rand(0)
        biggest_location = 0
        for m=0,ntraining-1 do begin
                if (rand(m) gt biggest) then begin
                        biggest = rand(m)
                        biggest_location = m
                endif
        endfor ;m
        order(n)=biggest_location
        rand(biggest_location)=-1.0
endfor ;n


for n=0,ntraining-1 do begin
        next_sample=order(n)

        ;xxxINPUT LAYERxxx
        for i=1,ninputs do begin
                yi(i)=patterns(i-1, next_sample)
        endfor ;i
        yi(0)=1.0

                ;xxxLAYER1xxx
                xj=0.0
                for i=0,ninputs do begin
                        for j=0,nl1 do begin
                                xj(j) = xj(j)+yi(i)*wij(j,i)
                        endfor ;j
                endfor ;i
                for j=0,nl1 do begin
                        yj(j) = 1.0/(1.0+exp(-xj(j)))
                endfor ;j
                yj(0)=1.0
```

```
        ;xxxLAYER2xxx
        xk=0.0
        for k=0,nl2 do begin
                for j=0,nl1 do begin
                        xk(k)=xk(k)+yj(j)*wjk(k, j)
                endfor ;j
        endfor ;k
        for k=0,nl2 do begin
                yk(k)=1.0/(1.0+exp(-xk(k)))
        endfor ;k
        yk(0)=1.0

;xxxOUTPUT LAYERxxx
xl=0.0
for k=0,nl2 do begin
        for l=0,noutputs-1 do begin
                xl(l)=xl(l)+yk(k)*wkl(l,k)
        endfor ;l
endfor ;k
for l=0,noutputs-1 do begin
        yl(l)=1.0/(1.0+exp(-xl(l)))
endfor ;l

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;;XXX START BACKPROPAGATION XX
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

for l=0,noutputs-1 do begin
        el(l) = (patterns(ninputs+l, next_sample)-yl(l))*yl(l)*(1.0-yl(l))
endfor ;l

ek(*) = 0.0

for k=0,nl2 do begin
        for l=0,noutputs-1 do begin
                ek(k) = ek(k) + yk(k)*(1.0-yk(k))*wkl(l, k)*el(l)
        endfor ;l
endfor ;k

ej(*) = 0.0

for j=0,nl1 do begin
        for k=0,nl2 do begin
                ej(j) = ej(j) + yj(j)*(1.0-yj(j))*ek(k)*wjk(k,j)
        endfor ;k
endfor ;j

;xxxxxxxxxxxxxxxxxxxxxx
;;XXX ADJUST WEIGHTS XX
;xxxxxxxxxxxxxxxxxxxxxx

for l=0,noutputs-1 do begin
        for k=0,nl2 do begin
                wkl(l, k) = wkl(l, k)+eta*el(l)*yk(k)
        endfor ;k
endfor ;l

for j=0,nl1 do begin
        for k=0,nl2 do begin
                wjk(k, j) = wjk(k, j)+eta*ek(k)*yj(j)
        endfor ;k
endfor ;j

for i=0,ninputs do begin
```

```
                for j=0,nl1 do begin
                        wij(j,i) = wij(j,i)+eta*ej(j)*yi(i)
                endfor ;j
        endfor ;i


endfor ;n

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;ONE TRAINING LOOP FINISHED - START VALIDATION
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

MIN_EO=0
MAX_EO=0
EO=fltarr(4)

for n=0,nvalidation-1 do begin
        next_sample=order(n)

        ;xxxINPUT LAYERxxx
        for i=1,ninputs do begin
                yi(i)=patterns(i-1, next_sample)
        endfor ;i
        yi(0)=1.0

                ;xxxLAYER1xxx
                xj=0.0
                for i=0,ninputs do begin
                        for j=0,nl1 do begin
                                xj(j) = xj(j)+yi(i)*wij(j,i)
                        endfor ;j
                endfor ;i
                for j=0,nl1 do begin
                        yj(j) = 1.0/(1.0+exp(-xj(j)))
                endfor ;j
                yj(0)=1.0


                ;xxxLAYER2xxx
                xk=0.0
                for k=0,nl2 do begin
                        for j=0,nl1 do begin
                                xk(k)=xk(k)+yj(j)*wjk(k, j)
                        endfor ;j
                endfor ;k
                for k=0,nl2 do begin
                        yk(k)=1.0/(1.0+exp(-xk(k)))
                endfor ;k
                yk(0)=1.0

        ;xxxOUTPUT LAYERxxx
        xl=0.0
        for k=0,nl2 do begin
                for l=0,noutputs-1 do begin
                        xl(l)=xl(l)+yk(k)*wkl(l,k)
                endfor ;l
        endfor ;k
        for l=0,noutputs-1 do begin
                yl(l)=1.0/(1.0+exp(-xl(l)))
        endfor ;l

        ;xxxxxxxxxxxxxxxxxxxxxxx
        ;COMPUTE VALIDATION ERROR
        ;xxxxxxxxxxxxxxxxxxxxxxx

                EO = ABS(dl-yl)
```

```
                    if EO lt MIN_EO then MIN_EO=EO
                    if EO gt MAX_EO then MAX_EO=EO

          endfor ;n

          if MIN_EO lt MIN_ERR then begin
                    print, "Network has converged successfully!"
                    stop
          endif else begin
          if MAX_ERR gt MAX_ERR then begin
                    print, "Network has failed to converge!"
                    stop
          endif
          endelse

endwhile ;cyc

print, "Training Complete!"
print, "Loading Test data..."

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;XXXX START TEST SECTION XXXXX
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

;save predicted outputs for all members of the test set
outputs = fltarr(noutputs,ntest)

for n=0,nvalidation-1 do begin
          next_sample = validation_ids(n)
          next_sample=order(n)

          ;xxxINPUT LAYERxxx
          for i=1,ninputs do begin
                    yi(i)=patterns(i-1, next_sample)
          endfor ;i
          yi(0)=1.0

                    ;xxxLAYER1xxx
                    xj=0.0
                    for i=0,ninputs do begin
                              for j=0,nl1 do begin
                                        xj(j) = xj(j)+yi(i)*wij(j,i)
                              endfor ;j
                    endfor ;i
                    for j=0,nl1 do begin
                              yj(j) = 1.0/(1.0+exp(-xj(j)))
                    endfor ;j
                    yj(0)=1.0


                    ;xxxLAYER2xxx
                    xk=0.0
                    for k=0,nl2 do begin
                              for j=0,nl1 do begin
                                        xk(k)=xk(k)+yj(j)*wjk(k, j)
                              endfor ;j
                    endfor ;k
                    for k=0,nl2 do begin
                              yk(k)=1.0/(1.0+exp(-xk(k)))
                    endfor ;k
                    yk(0)=1.0

          ;xxxOUTPUT LAYERxxx
          xl=0.0
          for k=0,nl2 do begin
```

```
                for l=0,noutputs-1 do begin
                        xl(l)=xl(l)+yk(k)*wkl(l,k)
                endfor ;l
        endfor ;k
        for l=0,noutputs-1 do begin
                yl(l)=1.0/(1.0+exp(-xl(l)))
        endfor ;l

;collate the predicted output for each member of the test set into one variable
outputs(*,n) = yl(*)

endfor ;n

;xxxxxxxxx
;SAVE DATA
;xxxxxxxxx

;Save data with a label of today's date
DAY=STRMID(SYSTIME(0),8,2)
MONTH=STRMID(SYSTIME(0),4,3)
Months=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
id=where(MONTH eq months)
YEAR=STRMID(SYSTIME(0),20,4)
HR=STRMID(SYSTIME(0),11,2)
MINS=STRMID(SYSTIME(0),14,2)


;Save data as IDL file
save, filename='C:\Users\Jennifer\Desktop\Data\Crude_NN_'+strcompress(DAY, /remove_all)+ $
        strcompress(id+1, /remove_all)+strcompress(YEAR, /remove_all)+'_'+strcompress(HR, /remove_all)+ $
        strcompress(MINS, /remove_all)+'_'+strcompress(dataset, /remove_all)+'.sav', outputs, test_ids, wij, wjk,
wkl

;Save ASCII files
openw, lun, 'C:\Users\Jennifer\Desktop\Data\Crude_NN_fit_'+strcompress(DAY, /remove_all)+ $
        strcompress(id+1, /remove_all)+strcompress(YEAR, /remove_all)+'_'+strcompress(HR, /remove_all)+ $
        strcompress(MINS, /remove_all)+'_'+strcompress(dataset, /remove_all)+'.dat', /get_lun
printf, lun, outputs, format='(4F15)'
free_lun, lun

openw, lun, 'C:\Users\Jennifer\Desktop\Data\Crude_NN_actual_'+strcompress(DAY, /remove_all)+ $
        strcompress(id+1, /remove_all)+strcompress(YEAR, /remove_all)+'_'+strcompress(HR, /remove_all)+ $
        strcompress(MINS, /remove_all)+'_'+strcompress(dataset, /remove_all)+'.dat', /get_lun
printf, lun, patterns(ninputs:ninputs+noutputs-1,test_ids), format='(4F15)'
free_lun, lun

openw, lun, 'C:\Users\Jennifer\Desktop\Data\Crude_NN_order_'+strcompress(DAY, /remove_all)+ $
        strcompress(id+1, /remove_all)+strcompress(YEAR, /remove_all)+'_'+strcompress(HR, /remove_all)+ $
        strcompress(MINS, /remove_all)+'.dat', /get_lun
printf, lun, test_ids+1, format='(1I4)'
free_lun, lun

openw, lun, 'C:\Users\Jennifer\Desktop\Data\Crude_NN_input_weights_'+strcompress(DAY, /remove_all)+ $
        strcompress(id+1, /remove_all)+strcompress(YEAR, /remove_all)+'_'+strcompress(HR, /remove_all)+ $
        strcompress(MINS, /remove_all)+'.dat', /get_lun
printf, lun, wij, format='(6I15)'
free_lun, lun

openw, lun, 'C:\Users\Jennifer\Desktop\Data\Crude_NN_h1_weights_'+strcompress(DAY, /remove_all)+ $
        strcompress(id+1, /remove_all)+strcompress(YEAR, /remove_all)+'_'+strcompress(HR, /remove_all)+ $
        strcompress(MINS, /remove_all)+'.dat', /get_lun
printf, lun, wjk, format='(6I15)'
free_lun, lun

openw, lun, 'C:\Users\Jennifer\Desktop\Data\Crude_NN_h2_weights_'+strcompress(DAY, /remove_all)+ $
        strcompress(id+1, /remove_all)+strcompress(YEAR, /remove_all)+'_'+strcompress(HR, /remove_all)+ $
```

```
        strcompress(MINS, /remove_all)+'.dat', /get_lun
printf, lun, wkl, format='(4I15)'
free_lun, lun

;xxxx
;END
;xxxx
End
```

A.II Simulated Annealing Genetic Algorithm

SAGA.pro

```
; Simulated Annealing Genetic Algorithm program for modelling crude oil mass spectra
;
; Program uses SA to derive a number of mask and property spectra to model the crude oil spectra
; Mask and Property Spectra are saved to MSPEC and PSPEC
;
; Weights associated with mask and property spectra are saved in MWTS and PWTS
;
; Genetic Algorithm is applied to find properties of an unknown sample.
;
; Parameters:
; NPROPS = 4 (number of property spectra)
; NSPEC = 76 (number of samples)
; NMASK = 6  (number of mask spectra)
;
; Required input:
; Text file containing instructions in the format:
;           No. of properties
;           Property A selected? 'T' or 'F'
;           Property B selected? 'T' or 'F'
;           Property C selected? 'T' or 'F'
;           Property D selected? 'T' or 'F'
;           No. of mask spectra
;           Starting temperature for the cooling schedule
;           ...

; Based on FORTRAN program sagaf07_01.f written by HMC, 2010
;
; Written: 10/02/2011 by Jennifer Hauschild, Cartwright Group, PTCL, Oxford University
;
; History: 24/07/2011 - amended to save PHOLD into a NPROPSxNSPECS array - JMH


pro SaGa

;PRINT START TIME
PRINT, 'Start:'
PRINT, FORMAT='(C(CHI2.2,":",CMI2.2,":",CSF05.2))',SYSTIME(/JULIAN)

;xxxxxxxxxxxxxxxxxxxx
;VARIABLE DECLARATION
;xxxxxxxxxxxxxxxxxxxx

NPROPS= 4
NSPEC= 76
NPOINTS= 712
NMASKS= 6

oilspec=fltarr(NSPEC,NPOINTS)
calcspec=fltarr(NSPEC,NPOINTS)
pwts=fltarr(NSPEC,NPROPS)
mwts=fltarr(NSPEC,NMASKS)
pspec=fltarr(NPROPS,NPOINTS)
mspec=fltarr(NMASKS,NPOINTS)
pwtshold=fltarr(NSPEC,NPROPS)
mwtshold=fltarr(NSPEC,NMASKS)
pspechold=fltarr(NPROPS,NPOINTS)
```

```
mspechold=fltarr(NMASKS,NPOINTS)

deviation=fltarr(50000)

order=intarr(20000)
mwtsorder=intarr(NSPEC,NPOINTS)
mspecorder=intarr(NMASKS,NPOINTS)
pspecorder=intarr(NPROPS,NPOINTS)

;xxxxxxxxxxxxxxxxxxx
;GA SOLVER VARIABLES
;xxxxxxxxxxxxxxxxxxx

P=fltarr(NSPEC,NPROPS)
PNEW=fltarr(NSPEC,NPROPS)
FITNESS=fltarr(NSPEC)
PWTSTRIAL=fltarr(NSPEC,NPROPS)
MSCALNEW=fltarr(NSPEC,NPOINTS)
DEVARRAY=fltarr(NSPEC,NPOINTS)
PHOLD=fltarr(NPROPS)
RESTORES=fltarr(NPROPS)

properties=fltarr(NPROPS,NSPEC)                    ;xxxxxxxxxxAMENDMENT: 24/07/2011


;INTEGERS
PROPNO=intarr(10000)
PROPPOINT=intarr(10000)
MASKSPECTRUM=intarr(10000)

;BOOLEANS
CROSSED=intarr(100)
pdefine=intarr(NSPEC,NPROPS)
propertysel=intarr(NPROPS)

print, "SA program for assessing oil mass spectra"

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;READ PARAMETERS FOR THE CURRENT RUN
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

pathx='C:\Documents and Settings\Jennifer\Desktop\IDL\SAGA\inst.txt'
nlines1=FILE_LINES(pathx)          ;determine the number of datapoints in instruction file
Ax=STRARR(nlines1)                 ;create string array to contain the data from the text file

OPENR, 1, pathx                    ;open text file
READF, 1, Ax                       ;read text file into variable
close, 1                           ;close channel 1

nprops=FIX(Ax(0))
for iprop=1,nprops do begin
        if Ax(iprop) eq 'T' then propertysel(iprop-1)=1 else propertysel(iprop-1)=0
endfor ;iprop
nmasks=FIX(Ax(5))
print, 'NMASKS is '+strcompress(nmasks, /remove_all)
ncycles=FIX(Ax(6))
print, 'NCYCLES is '+strcompress(ncycles, /remove_all)
icool=FIX(Ax(7))
tinit=FIX(Ax(8))
print, 'TINIT is '+strcompress(tinit, /remove_all)

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;LOAD RAW SPECTRA FROM FILE
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

restore, 'C:\Users\Jennifer\Desktop\Crudes\Pos_dat\peaks_matrix.sav'
```

```
restore, 'C:\Users\Jennifer\Desktop\Crudes\Pos_dat\crude_props.sav'

;convert degree Celsius into Kelvin
A(0,*)=A(0,*)+273.


;Log-Normalise the input data
for is=0,nspec-1 do begin
        for ip=0,npoints-1 do begin
                oilspec(is,ip)=ALOG(new_mat(1,is,ip))
        endfor ;ip
endfor ;is

for iprop=0,nprops-1 do begin
        for is=0,nspec-1 do begin
                pwts(is,iprop)=A(iprop,is)
                pdefine(is, iprop)=1                    ;Boolean, i.e. set to "True"
        endfor ;is
endfor ;iprop

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;ADJUST PWTS, WEIGHTS OF PROPERTY SPECTRA
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

for iprop=0,nprops-1 do begin
        for is=0,nspec-1 do begin
                pwtshold(is,iprop)=pwts(is,iprop)
        endfor ;is
endfor ;iprop

;Select those pwts chosen in propertysel
pwts(*,*)=0
id=where(propertysel(*) eq 1)
if id(0) eq -1 then begin
        print, 'STOP! No properties have been selected. Please choose some properties and then continue!'
        stop
endif

NPROPS= n_elements(id)
pwts=fltarr(NSPEC,NPROPS)

for nsel=0,n_elements(id)-1 do begin
        for iprop=0,nprops-1 do begin
                for is=0,nspec-1 do begin
                        pwts(is,nsel)=pwtshold(is,id(nsel))
                endfor ;is
        endfor ;iprop
endfor ;nsel

pwtshold=fltarr(NSPEC,NPROPS)
pspec=fltarr(NPROPS,NPOINTS)
pspechold=fltarr(NPROPS,NPOINTS)
propmax=fltarr(NPROPS)
propmin=fltarr(NPROPS)
propav=fltarr(NPROPS)
pspecorder=intarr(NPROPS,NPOINTS)

ncycles=4
devsq=0

;Choose a batchsize to step through the ORDER array. This determines what fraction of the PSPEC,
;MSPEC and MWTS arrays is open to adjustment in each cycle
batchsize=8

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;DETERMINE THE EXTREME VALUES OF THE PROPERTY DATA
```

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

;The maximum and minimum values of each property are required as input into the Genetic Algorithm

for iprop=0,nprops-1 do begin
        propmax(iprop)=pwts(0,iprop)
        propmin(iprop)=pwts(0,iprop)
        propav(iprop)=ABS(pwts(0,iprop))

        for ispec=0,nspec-1 do begin
                if(propmax(iprop) lt pwts(ispec,iprop)) then begin
                        propmax(iprop)=pwts(ispec,iprop)
                endif
                if(propmin(iprop) gt pwts(ispec,iprop)) then begin
                        propmin(iprop)=pwts(ispec,iprop)
                endif
                propav(iprop)=propav(iprop)+ABS(pwts(ispec,iprop))
        endfor    ;ispec
        propav(iprop)=propav(iprop)/nprops
endfor ;iprop

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;CREATE THE INITIAL MASK SPECTRA, PROPERTY SPECTRA AND MASK WEIGHTS
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

;Create random mask weights

for imask=0,nmasks-1 do begin
        for ispec=0,nspec-1 do begin
                mwts(ispec,imask)=randomu(T,1)
        endfor ;ispec
endfor ;imask

;Create semi-random mask and property spectra, centred around the mean m/e value for each point

for ip=0,npoints-1 do begin
        smean=0.0
        for ispec=0,nspec-1 do begin
                smean=smean+oilspec(ispec,ip)
        ;smean(ispec,ip)=smean(ispec,ip)+oilspec(ispec,ip)
        endfor ;ispec
        smean=smean/FLOAT(nspec)

                for iprop=0,nprops-1 do begin
                        pspec(iprop,ip)=(smean+1.0+iprop)/propav(iprop)
                endfor; iprop

        for imask=0,nmasks-1 do begin
                mspec(imask,ip)=smean
        endfor ;imask
endfor ;ip

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;SET ANY REMAINING PARAMETERS BEFORE STARTING SIMULATED ANNEALING
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

;DEVBEST is the best deviation found in the SA loop so far
devbest=1.0E25

;BIAS is one of the factors that determines what weight should be given to new parameter data in the SA procedure.
;If BIAS=0, any new parameter set that gives an unchanged deviation will be accepted, i.e. BIAS>0
bias=2.5

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;STORE THE CURRENT WEIGHTS AND TRIAL SPECTRA IN HOLDING ARRAYS
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
;IMPROVED is a boolean that is set to true (=1) if the calculation during the current loop has improved the deviation
IMPROVED=0

for iprop=0,nprops-1 do begin
        for ip=0,npoints-1 do begin
                pspechold(iprop,ip)=pspec(iprop,ip)
        endfor ;ip
endfor ;iprop

for imask=0,nmasks-1 do begin
        for ip=0,npoints-1 do begin
                mspechold(imask,ip)=mspec(imask,ip)
        endfor ;ip
endfor ;imask

for imask=0,nmasks-1 do begin
        for ispec=0,nspec-1 do begin
                mwtshold(ispec,imask)=mwts(ispec,imask)
        endfor ;ispec
endfor ;imask

;use the current masks and spectra to generate calculated spectra

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

;SUBROUTINE CALCSPECTRA

;start by clearing the calculated spectra
for ispec=0,nspec-1 do begin
        for ip=0,npoints-1 do begin
                calcspec(ispec,ip)=0.0
        endfor ;ip
endfor ;ispec

;include the contribution to each point from the properties spectra
for ispec=0,nspec-1 do begin
        for ip=0,npoints-1 do begin
                for iprop=0,nprops-1 do begin
                        calcspec(ispec,ip)=calcspec(ispec,ip)+pspec(iprop,ip)*pwts(ispec,iprop)
                endfor ;iprop
                ;add the contribution from the mask spectra
                for imask=0,nmasks-1 do begin
                        calcspec(ispec,ip)=calcspec(ispec,ip)+mspec(imask,ip)*mwts(ispec,imask)
                endfor ;imask
        endfor ;ip
endfor ;ispec

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

;SUBROUTINE CALCULATEDEV

dev=0.0

;if devsq= FALSE then the deviation is the absolute difference between the calculated and experimental spectra
;otherwise it is the squared difference

if devsq ne 1 then begin

for ispec=0,nspec-1 do begin
        for ip=0,npoints-1 do begin
                diff=ABS(oilspec(ispec,ip)-calcspec(ispec,ip))
```

```
                dev=dev+diff
        endfor ;ip
endfor ;ispec

endif else begin

for ispec=0,nspec-1 do begin
        for ip=0,npoints-1 do begin
                diff=oilspec(ispec,ip)-calcspec(ispec,ip)               ;diff(ispec,ip)
                dev=dev+diff*diff
        ;dev(ispec,ip)=dev(ispec,ip)+diff(ispec,ip)^2
        endfor ;ip
endfor ;ispec
endelse

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;START SIMULATED ANNEALING LOOP
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

for iloop=0,ncycles-1 do begin

;Every point in the mask and property spectra and also each of the mask weights is considered for updating every loop
;To ensure that every point is dealt with, generate a list of all points in these arrays, taking them in random order.
;These points will then be taken in batches when updating the arrays

;Set the total number of points to be modified
 IPTOT=LONG((nprops*npoints)+(nmasks*npoints)+(nmasks*nspec))

NP=IPTOT
;generate a group of random numbers of a size equal to IPTOT

;xxxxxxxxxxxxxxxxxxxxxxxxxxxx
;SUBROUTINE RANDOMSORT
Xxxxxxxxxxxxxxxxxxxxxxxxxxxx

;Create an array of the numbers from 1 to NP in random order. This routine is a rapidmethod for randomly ordering a
set
;of integers in a time scaling roughly linearly with NP

order=intarr(NP)
done=intarr(NP)

for i=0L,NP-1 do begin
        done(i)=0
endfor ;i

for i=0L,NP-1 do begin
        ;select a random number in the range of 1 to NP
        IPOS=FIX(1+(NP*randomu(S,1)))

        ;if the number has not been selected before, update the order array and set the appropriate flag to 'done'
        if done(IPOS) ne 1 then begin
                done(IPOS)=1
                order(i)=IPOS
        endif else begin
        Jump1:  IPOS=IPOS+1
                        if(IPOS gt NP) then begin
                                IPOS=0
                        endif
                        if done(IPOS) ne 1 then begin
                                done(IPOS)=1
                                order(i)=IPOS
```

```
                    endif else begin
        GOTO, Jump1
        endelse
        endelse
endfor ;i

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;ENDNEXTLOOP is set to TRUE (=1) if the complete set of entries in the order array will have been read and updated
;by the end of the current loop

ENDNEXTLOOP=0

;Before modifying the the spectra and the weights set the starting point for the count through the order arrays
ICOUNTSTART=1-BATCHSIZE
ICOUNTEND=0

;Update the start and end batch numbers

Jump2: ICOUNTSTART=ICOUNTSTART+BATCHSIZE
ICOUNTEND=ICOUNTEND+BATCHSIZE

        if (ICOUNTEND eq IPTOT) then begin
                ENDNEXTLOOP=1
        endif

        if (ICOUNTEND gt IPTOT) then begin
                ICOUNTEND=IPTOT
                ENDNEXTLOOP=1
        endif

;xxxxxxxxxxxxxxxxx
;MODIFICATION LOOP
;xxxxxxxxxxxxxxxxx

;work through the next batchsize elements in the order array
        for istep=icountstart-1,icountend-1 do begin
                inext=order(istep)+1

                    ;Modify PSPEC array
                    if (inext le nprops*npoints) then begin
                        jprop=1+((inext-1)/npoints)
                        jpoint=inext-(npoints*(jprop-1))

                        if (2*(iloop/2) eq iloop) then begin
                                pspec(jprop-1,jpoint-1)=pspec(jprop-1,jpoint-
1)*0.99+0.02*TEMPERATURE*randomu(V,1)
                        endif else begin
                                pspec(jprop-1,jpoint-1)=pspec(jprop-1,jpoint-
1)*1.0+0.1*TEMPERATURE*(0.5-randomu(V,1))
                        endelse
                    endif

                    ;Modify MSPEC array
                    if (inext gt nprops*npoints AND inext le (nprops*npoints + nmasks*npoints)) then begin
                        inextmod=inext-(nprops*npoints)
                        jmask=1+((inextmod-1)/npoints)
                        jpoint=inextmod-npoints*(jmask-1)

                        if (2*(iloop/2) eq iloop) then begin
                                mspec(jmask-1,jpoint-1)=mspec(jmask-1,jpoint-
1)*0.99+0.02*TEMPERATURE*randomu(V,1)
                        endif else begin
                                mspec(jmask-1,jpoint-1)=mspec(jmask-1,jpoint-
1)*1.0+0.1*TEMPERATURE*(0.5-randomu(V,1))
                        endelse
                    endif
```

```
                        ;Modify MWTS array
                        if (inext gt (nprops*npoints + nmasks*npoints) AND inext le IPTOT) then begin
                                inextmod=inext-(nprops*npoints)-(NMASKS*NPOINTS)
                                jmask=1+((inextmod-1)/nspec)
                                jspec=inextmod-nspec*(jmask-1)

                                if (2*(iloop/2) eq iloop) then begin
                                        mwts(jspec-1,jmask-1)=mwts(jspec-1,jmask-
1)*0.99+0.01*TEMPERATURE*randomu(V,1)
                                endif else begin
                                        mwts(jspec-1,jmask-1)=mwts(jspec-1,jmask-
1)*1.0+0.1*TEMPERATURE*(0.5-randomu(V,1))
                                endelse

                        endif

        endfor ;istep

;At this point a max number of batchsize modifications have been made to the adjustable arrays. Now we use the
freshly modified arrays to generate
;new calculated spectra

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

;SUBROUTINE CALCSPECTRA

;start by clearing the calculated spectra
for ispec=0,nspec-1 do begin
        for ip=0,npoints-1 do begin
                calcspec(ispec,ip)=0.0
        endfor ;ip
endfor ;ispec

;include the contribution to each point from the properties spectra
for ispec=0,nspec-1 do begin
        for ip=0,npoints-1 do begin
                for iprop=0,nprops-1 do begin
                        calcspec(ispec,ip)=calcspec(ispec,ip)+pspec(iprop,ip)*pwts(ispec,iprop)
                endfor ;iprop
                ;add the contribution from the mask spectra
                for imask=0,nmasks-1 do begin
                        calcspec(ispec,ip)=calcspec(ispec,ip)+mspec(imask,ip)*mwts(ispec,imask)
                endfor ;imask
        endfor ;ip
endfor ;ispec


;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

;SUBROUTINE CALCULATEDEV

dev=0.0

;if devsq= FALSE then the deviation is the absolute difference between the calculated and experimental spectra
;otherwise it is the squared difference

if devsq ne 1 then begin

for ispec=0,nspec-1 do begin
        for ip=0,npoints-1 do begin
                diff=ABS(oilspec(ispec,ip)-calcspec(ispec,ip))
```

```
                dev=dev+diff
        endfor ;ip
endfor ;ispec

endif else begin

for ispec=0,nspec-1 do begin
        for ip=0,npoints-1 do begin
                diff=oilspec(ispec,ip)-calcspec(ispec,ip)
                dev=dev+diff*diff
        endfor ;ip
endfor ;ispec
endelse

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

;Are the calculated spectra an improvement? Accept the new parameters with a probability that depends on the temperature. T starts
;at 1 and gradually decreases as specified in the ICOOL subroutine

accept=(dev+BIAS-devbest)/temperature

;Test whether the current solution is a global minimum, in which case save the arrays

if(((iloop lt ncycles/2) AND (exp(-accept) lt randomu(V,1))) OR ((iloop ge ncycles/2) AND (dev gt devbest))) then begin
        IMPROVED = 0

        for iprop=0,nprops-1 do begin
                for ip=0,npoints-1 do begin
                        pspec(iprop,ip)=pspechold(iprop,ip)
                endfor
        endfor

        for imask=0,nmasks-1 do begin
                for ip=0,npoints-1 do begin
                        mspec(imask,ip)=mspechold(imask,ip)
                endfor
        endfor

        for imask=0,nmasks-1 do begin
                for ispec=0,nspec-1 do begin
                        mwts(ispec,imask)=mwtshold(ispec,imask)
                endfor
        endfor

;The SA factor is greater than a randomly chosen number in the range 0-1, so update DEVBEST
;and then copy the modified arrays over the holding arrays

endif else begin
        IMPROVED = 1
        devbest=dev

        for iprop=0,nprops-1 do begin
                for ip=0,npoints-1 do begin
                        pspechold(iprop,ip)=pspec(iprop,ip)
                endfor    ;ip
        endfor ;iprop

        for imask=0,nmasks-1 do begin
                for ip=0,npoints-1 do begin
                        mspechold(imask,ip)=mspec(imask,ip)
                endfor ;ip
        endfor ;imask
```

```
        for imask=0,nmasks-1 do begin
                for ispec=0,nspec-1 do begin
                        mwtshold(ispec,imask)=mwts(ispec,imask)
                endfor ;ispec
        endfor ;imask

endelse

;Check to see whether the last entries in the ORDER array have been read.
;If they have, we are at the end of a cycle

        if(endnextloop eq 0) then GOTO, Jump2
                ENDNEXTLOOP=0

;Implement the cooling schedule


;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

;COOLIT SUBROUTINE

;This implements the simulated annealing cooling schedule.
;If ICOOL is 1, the temperature is kept at its initial valuefor the first NCYCLES/2 cycles,
;then lowered linearly to a value of 1% of the initial temperature over the next NCYCLES/2

        if((ICOOL eq 1) AND (ILOOP gt NCYCLES/2)) then begin
                temperature = TINIT*(1.01-FLOAT(ILOOP-NCYCLES/2)/FLOAT(NCYCLES/2))
        endif

;If ICOOL is 2 the temperature is lowered linearly to a value of 1% of the initial temperature over ncycles

        if(ICOOL eq 2) then begin
                temperature = TINIT*(1.01-(FLOAT(ILOOP)/FLOAT(NCYCLES)))
        endif

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

DEVIATION(ILOOP)=DEV

endfor ;iloop

;Simulated Annealing is at an end. Now use a Genetic Algorithm to find the property weights

for is=0,nspec-1 do begin

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

;SUBROUTINE GASOLVER

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;This routine uses a Genetic Algorithm to find the set of property values that best
;fit the model for spectrum IS

;The array P contains a population of NSTRINGS potential property solutions

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Set GA Parameters
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
NSTRINGS=40
NCROSS=10
NMUTATE=30

BESTEVERFITNESS=0.0

;Copy PWTS into PWTSTRIAL. Subsequently we overwrite one part of this array with the
;varying property weights from the strings
        for ispec=0,nspec-1 do begin
                for iprop=0,nprops-1 do begin
                        PWTSTRIAL(ISPEC,IPROP)=PWTS(ISPEC,IPROP)
                endfor
        endfor

;Create the starting population, choosing values at random within the current minimum and
;maximum values for each parameter
for ISTRING=0,NSTRINGS-1 do begin
        for IPROP=0,NPROPS-1 do begin
                P(ISTRING,IPROP)=PROPMIN(IPROP)+randomu(W,1)*(PROPMAX(IPROP)-PROPMIN(IPROP))
        endfor
endfor

;Save the first string in PHOLD so that there is no chance a string might be inadvertently overwritten with zeros

        for IPROP=0,NPROPS-1 do begin
                PHOLD(IPROP)=P(0,IPROP)
        endfor

        for ISTRING=0,NSTRINGS-1 do begin
                for IPROP=0,NPROPS-1 do begin
                        PNEW(ISTRING,IPROP)=P(ISTRING,IPROP)
                endfor
        endfor

;Start the Genetic Algorithm loop. At present a defined number of cycles is used, though one
;could test for convergence of the calculation

for ICYCLE=0,19 do begin
;For each string in turn, generate the calculated spectrum, then compute the deviation from the real spectra and
;use this deviation as a first measure of fitness

        for ISTRING=0,NSTRINGS-1 do begin
                for IPROP=0,NPROPS-1 do begin
                        PWTSTRIAL(IS,IPROP)=P(ISTRING,IPROP)
                        CROSSED(ISTRING)=0
                endfor ;IPROP

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;SUBROUTINE CALC1spectrum

;Start by clearing the calculated spectra array

        for IP=0,NPOINTS-1 do begin
                CALCSPEC(IS,IP)=0.0
        endfor ;IP

;Include the contribution to each point from the properties spectra

        for IP=0,NPOINTS-1 do begin
                for IPROP=0,NPROPS-1 do begin
                        CALCSPEC(IS,IP)=CALCSPEC(IS,IP)+PSPEC(IPROP,IP)*PWTSTRIAL(IS,IPROP)
                endfor ;IPROP

;Add the contribution from the mask spectra
```

```
                for IM=0,NMASKS-1 do begin
                        CALCSPEC(IS,IP)=CALCSPEC(IS,IP)+MSPEC(IM,IP)*MWTS(IS,IM)
                endfor ;IM
        endfor ;IP

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

;SUBROUTINE CALC1DEV

DEV=0.0
        if(DEVSQ eq 0) then begin
;The deviation is the absolute difference between the calculated and experimental spectra

                for IP=0,NPOINTS-1 do begin
                        DIFF=ABS(OILSPEC(IS,IP)-CALCSPEC(IS,IP))
                        DEV=DEV+DIFF
                endfor ;IP

;or the squared deviation
        endif else begin

                for IP=0,NPOINTS-1 do begin
                        DIFF=ABS(OILSPEC(IS,IP)-CALCSPEC(IS,IP))
                        DEV=DEV+DIFF*DIFF
                endfor ;IP

        endelse

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

FITNESS(ISTRING)=1.0/DEV

        endfor ;ISTRING

;Determine the highest and lowest fitness amongst the set of strings

LEASTFIT=FITNESS(0)
MOSTFIT=FITNESS(0)
IBEST=1

        for ISTRING=0,NSTRINGS-1 do begin
                if(LEASTFIT gt FITNESS(ISTRING)) then begin
                        LEASTFIT=FITNESS(ISTRING)
                endif
                if(MOSTFIT lt FITNESS(ISTRING)) then begin
                        IBEST=ISTRING
                        MOSTFIT=FITNESS(ISTRING)
                endif
        endfor ;ISTRING

;Does any string have a fitness that exceeds the best seen so far. If such a string exists, save its fitness and
;the strings itself

        if(besteverfitness lt mostfit) then begin
                besteverfitness=mostfit
                        for IPROP=0,NPROPS-1 do begin
                                PHOLD(IPROP)=P(IBEST,IPROP)
                        endfor ;IPROP
        endif

;If all the strings have the same fitness, which is unlikely in a large population, but much
```

```
;more likely if the population is small, we need to take action, since the next step would lead
;to overflow. The simplest way to proceed is to avoid the selection and the crossover steps this
;cycle and go straight to mutation.

        if(mostfit eq leastfit) then GOTO, Jump3

;Scale the string fitnesses to a common range

        for ISTRING=0,NSTRINGS-1 do begin
                FITNESS(ISTRING)=1.0+9.0*(FITNESS(ISTRING)-LEASTFIT)/(MOSTFIT-LEASTFIT)
        endfor ;ISTRING

;Stochastic Remainder
;Use stochastic remainder to identify members of the next population
;The first step in stochastic remainder is to determine the total fitness ;of all strings. We then normalise the fitness of
;each string so that the average fitness is one

TOTALF=FITNESS(0)
        for ISTRING=1,NSTRINGS-1 do begin
                TOTALF=TOTALF+FITNESS(ISTRING)
        endfor ;ISTRING

        for ISTRING=0,NSTRINGS-1 do begin
                FITNESS(ISTRING)=FITNESS(ISTRING)*FLOAT(NSTRINGS)/TOTALF
        endfor ;ISTRING

;Pass through all strings, making a  number of copies of those that have above average fitness
;equal to the integer part of their fitness

;In making the new population, first make room for the best string

        for IPROP=0,NPROPS-1 do begin
                PNEW(0,IPROP)=PHOLD(IPROP)
        endfor ;IPROP
                INEW=1

        for ISTRING=0,NSTRINGS-1 do begin
                if(FITNESS(ISTRING) ge 1.00) then begin
                        NNEWSTRINGS=FITNESS(ISTRING)
                        FITNESS(ISTRING)=FITNESS(ISTRING)-NNEWSTRINGS
                                for JNEW=0,NNEWSTRINGS-1 do begin
                                        INEW=INEW+1
                                        for IPROP=0,NPROPS-1 do begin
                                                PNEW(INEW,IPROP)=P(ISTRING,IPROP)
                                        endfor ;IPROP
                                endfor ;JNEW
                endif
        endfor ;ISTRING

;We are left now with the remainder of each fitness. We will try using ;a simple approach to select remaining strings and
;amend it if it appears to operate too slowly

;Pick a random string
Jump4: ISTRING=1+NSTRINGS*randomu(x,1)
;Pick a random number
ARAND=randomu(y,1)

;If the random number is above the residual fitness of string ISTRING, try again.
;Otherwise make a copy of the string, set its fitness to a negative value to prevent ;it being selected again, and to continue
;if the population is not yet complete

        if(ARAND(0) lt FITNESS(ISTRING(0))) then begin
                INEW=INEW+1
                for IPROP=0,NPROPS-1 do begin
                        PNEW(INEW,IPROP)=P(ISTRING(0),IPROP)
                endfor ;IPROP
```

```
                FITNESS(ISTRING(0))=-1.0
        endif

        if(INEW lt NSTRINGS) then GOTO, Jump4

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Crossover:
;Having selected strings for the new population, start on crossover
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

;Select a pair of strings, but do not pick the first one, since it's protected from modification

        for ICROSS=0,NCROSS-1 do begin
                Jump5: ISTRING1=2+FIX(randomu(y,1)*(NSTRINGS-1))
        if(CROSSED(ISTRING1(0)) eq 1) then GOTO, Jump5
                Jump6: ISTRING2=2+FIX(randomu(y,1)*(NSTRINGS-1))
                if((CROSSED(ISTRING2(0)) eq 1) OR (ISTRING1(0) eq ISTRING2(0))) then GOTO, Jump6

                CROSSED(ISTRING1(0))=1
                CROSSED(ISTRING2(0))=1

;Choose crossing points. Because each element of a string relates to a different property,
;only use direct one-for-one swap across the strings

                IX1=1+fix(NPROPS*randomu(x,1))
        Jump7: IX2=1+fix(NPROPS*randomu(x,1))
                if(IX1(0) eq IX2(0)) then GOTO, Jump7

;If IX1<IX2 normal crossover applies; if not, apply wraparound crossover

        if(IX1(0) lt IX2(0)) then begin
                for I=IX1(0)-1,IX2(0)-1 do begin
                        A=PNEW(ISTRING1(0),I)
                        PNEW(ISTRING1(0),I)=PNEW(ISTRING2(0),I)
                        PNEW(ISTRING2(0),I)=A
                endfor ;I
        endif else begin
                for I=0,IX2(0)-1 do begin
                        A=PNEW(ISTRING1(0),I)
                        PNEW(ISTRING1(0),I)=PNEW(ISTRING2(0),I)
                        PNEW(ISTRING2(0),I)=A
                endfor ;I
                for I=IX1(0)-1,NPROPS-1 do begin
                        A=PNEW(ISTRING1(0),I)
                        PNEW(ISTRING1(0),I)=PNEW(ISTRING2(0),I)
                        PNEW(ISTRING2(0),I)=A
                endfor ;I
        endelse

        endfor ;ICROSS

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;MUTATION
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;Again the first string cannot be modified, so ISTRING must be at least 2

Jump3:  for IMUT=0,NMUTATE-1 do begin
                ISTRING=2+FIX(randomu(z,1)*(NSTRINGS-1))
                IMUTPOINT=1+FIX(randomu(z,1)*FLOAT(NPROPS-1))
;Create a new value within the permitted property range

        PNEW(ISTRING(0),IMUTPOINT(0))=0.9*PROPMIN(IMUTPOINT(0))+1.2*randomu(z,1)*(PROPMAX(IMUTP
OINT(0))-PROPMIN(IMUTPOINT(0)))
                endfor ;IMUT

;End of string modification. Copy the new strings over the old
```

```
        for ISTRING=0,NSTRINGS-1 do begin
                for IPROP=0,NPROPS-1 do begin
                        P(ISTRING,IPROP)=PNEW(ISTRING,IPROP)
                endfor ;IPROP
        endfor ;ISTRING

endfor ;ICYCLE

properties(0:NPROPS-1, is)=PHOLD(0:NPROPS-1)                    ;xxxxxxAMENDMENT: 24/07/2011

endfor ;is

;PRINT TIME AT FINISH!
PRINT, 'Finish:'
PRINT,  FORMAT='(C(CHI2.2,":",CMI2.2,":",CSF05.2))',SYSTIME(/JULIAN)

DAY=STRMID(SYSTIME(0),8,2)
MONTH=STRMID(SYSTIME(0),4,3)
Months=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
id=where(MONTH eq months)
YEAR=STRMID(SYSTIME(0),20,4)

save, filename='C:\Users\Jennifer\Desktop\Data\saga_'+strcompress(DAY, /remove_all)+ $
        strcompress(id+1,       /remove_all)+strcompress(YEAR,       /remove_all)+'_'+strcompress(propertysel(0),
/remove_all)+ $
        strcompress(propertysel(1),                                  /remove_all)+strcompress(propertysel(2),
/remove_all)+strcompress(propertysel(3), /remove_all)+ $
        '.sav', PWTSTRIAL, p, pnew, pspec, phold, pspechold, pwts, pwtshold, mspec, mspechold, mwts, mwtshold,
calcspec, oilspec, deviation, fitness, properties

;xxx
END
;xxx
end
```

A.III Self Organising Map

CrudeSOM.pro

```
; Program is a self organising map created to analyse crude oil data.
; The algorithm creates a set of random weight vectors linked to each of the nodes on a 7x7 topological map.
; Each new sample introduced to the network is matched to a node according to the distance between the input
; vector and the weight vector.
;
; Training Set:
; Mapping Set:
;
; Network geometry is 7x7
; NCYCLES=1000
;
; Input = 712x76 Dataset A
;
; Created: 26/07/2010 by JMH, Cartwright Group, PTCL, Oxford University
;
; History:


pro crudeSOM


;restore common peaks matrix, 76x712
restore, 'C:\Users\Jennifer\Desktop\Crudes\Pos_dat\peaks_matrix.sav'

x=fltarr(n_elements(new_mat(0,*,0)), n_elements(new_mat(0,0,*)))          ;76x712

;normalize data
for k=0,n_elements(new_mat(0,0,*))-1 do begin
        x(*,k)=(new_mat(1,*,k)-mean(new_mat(1,*,k)))/stdev(new_mat(1,*,k))
endfor ;k

;xxxxxxxxxxxxxxxx
;DEFINE PARAMETERS
;xxxxxxxxxxxxxxxx
loops = 1000                                    ;number of training epochs
nn = 7                                          ;number of nodes. Map will be 7x7 dimensional
alpha = 0.8                                     ;learning rate
nu = 0.01                                       ;momentum parameter
nsamples = n_elements(new_mat(0,*,0))
npoints = n_elements(new_mat(0,0,*))
sig0 = nn                                       ;starting value for neighbourhood size


;create random weights array
wij=randomu(seed, nn, nn, npoints)

;define distance matrix
d=fltarr(nn,nn)

each_dist=fltarr(npoints)

;create subset of samples to use for training and mapping and then create randomised data arrays
training_fraction = 0.90
mapping_fraction = 0.10
ntraining = FIX(nsamples*training_fraction)
nmapping = nsamples-ntraining
```

```
;construction of a boolean structure. Sample_chosen (binary array) is set to 0 to start with. Once a
;sample has been chosen, the array entry is set to 1

sample_chosen=intarr(nsamples)
sample_chosen(*)=0          ;new intarr automatically is filled with zeros, but this line makes doubly sure

random = randomu(seed,nsamples)
training_ids = intarr(ntraining)
mapping_ids = intarr(nmapping)

;step through training epochs
for l=0,loops-1 do begin

;implement cooling
If (l lt 600) then alpha=0.8 else (if l ge 600) then alpha=0.2

sample_chosen(*)=0          ;new int arr automatically is filled with zeros, but this line makes doubly sure

nt=0
while nt lt ntraining do begin
        for is=0,nsamples-1 do begin
                if (sample_chosen(is) eq 0) AND (nt lt ntraining) then begin
                        if random(is) lt training_fraction then begin
                                sample_chosen(is)=1
                                training_ids(nt)=is
                                nt=nt+1
                        endif
                endif
        endfor ;is
endwhile ;nt

mapping_ids(*)=where(sample_chosen ne 1)

;iterate through training set
for k=0,ntraining-1 do begin
;calculate Euclidian distance between input vector (new_mat(0,k,*) i.e. 1x712 vector) and weight
;co-ordinates wij for node xi,yj

leastdist = 200000.
d(*,*)=0

                for i=0,nn-1 do begin
                        for j=0,nn-1 do begin
                                for m=0,n_elements(new_mat(0,0,*))-1 do begin

                                d(i,j)=d(i,j)+sqrt((x(training_ids(k),m) - wij(i,j,m))^2)

                                endfor ;m (peaks loop)

                                if (d(i,j) lt leastdist) then begin

                                        leastdist = d(i,j)
                                        xi=i
                                        yj=j

                                endif


                        endfor; j (node loop 1)
                endfor ;i (node loop 2)

;modify all epoch dependent parameters
sig = fix(sig0*exp(-l/15))
;size of neighbourhood (lattice width must be an integer as it represents the number of nodes)
;update weights for every single point using the neighbourhood function to determine the amount
;that each weight is changed
```

```
        for i=0,nn-1 do begin
                        for j=0,nn-1 do begin

        ;determine neighbourhood function.. if i=xi and j=yj then the node is the BMU
        ;each other node is adjusted according to its distance from the BMU, i.e. (i-xi/j-xj)
                        if (i eq xi) and (j eq yj) then begin
                                nbh=1
                        endif else begin
                                nbh=exp(-((j-yj)^2+(i-xi)^2)/(2*sig^2))        ;neighbourhood function
                        endelse

                        ;ADJUST WEIGHTS
                        wij(i,j,*)=wij(i,j,*) + nbh*nu*(x(training_ids(k),*) - wij(i,j,*))

                        endfor; j (node loop 1)
                endfor ;i (node loop 2)


endfor ;k (sample loop)

endfor ;l (cycles loop)

;xxxxxxxxxxxxxxxxx
;TRAINING COMPLETE
;xxxxxxxxxxxxxxxxx

;load graphics devices and settings
device, decomposed=0
loadct,10

plot, [0], [0], background=255, color=0, xrange=[0,nn], yrange=[0,nn]

;iterate through mapping set
for k=0,nmapping-1 do begin

;calculate euclidian distance between input vector (new_mat(0,k,*) i.e. 1x712 vector) and weight co-ordinates wij for
;node xi,yj
leastdist = 200000.
d(*,*)=0

                for i=0,nn-1 do begin
                        for j=0,nn-1 do begin
                                for m=0,n_elements(new_mat(0,0,*))-1 do begin

                                d(i,j)=d(i,j)+sqrt((x(mapping_ids(k),m) - wij(i,j,m))^2)

                                endfor ;m (peaks loop)

                                if (d(i,j) lt leastdist) then begin
                                        leastdist = d(i,j)
                                        xi = i
                                        yj = j

                                endif

                        endfor; j (node loop 1)
                endfor ;i (node loop 2)


oplot, [xi], [yj], psym=4, color=0
xyouts, [xi], [yi], strcompress(mapping_ids(k)+1, /remove_all), color=0


endfor ;k (sample loop)

;xxxxxxxxxxxx
```

;SAVE RESULTS
;xxxxxxxxxxxx

save, filename='C:\Users\Jennifer\Desktop\Data\som.sav', wij, mapping_ids, training_ids, sig

;xxx
;END
;xxx
end

APPENDIX B: Data

B.I Iris Data

Petal Length	Petal Width	Sepal Length	Sepal Width	Class Descriptor		
				Versicolor	**Virginica**	**Setosa**
6.1	2.9	4.7	1.4	1	0	0
5	2	3.5	1	1	0	0
5.5	2.3	4	1.3	1	0	0
5.8	2.7	4.1	1	1	0	0
5	2.3	3.3	1	1	0	0
6	2.7	5.1	1.6	1	0	0
5.6	2.9	3.6	1.3	1	0	0
5.7	2.8	4.1	1.3	1	0	0
5.6	3	4.5	1.5	1	0	0
6.4	3.2	4.5	1.5	1	0	0
6.2	2.2	4.5	1.5	1	0	0
5.5	2.4	3.8	1.1	1	0	0
5.7	2.9	4.2	1.3	1	0	0
5.5	2.5	4	1.3	1	0	0
6.3	2.3	4.4	1.3	1	0	0
5.5	2.6	4.4	1.2	1	0	0
5.8	2.6	4	1.2	1	0	0
6.3	2.5	4.9	1.5	1	0	0
6.8	2.8	4.8	1.4	1	0	0
7	3.2	4.7	1.4	1	0	0
5.7	2.6	3.5	1	1	0	0
6.7	3.1	4.7	1.5	1	0	0
4.9	2.4	3.3	1	1	0	0
6	2.9	4.5	1.5	1	0	0
5.4	3	4.5	1.5	1	0	0
6.3	3.3	4.7	1.6	1	0	0
6.1	3	4.6	1.4	1	0	0
5.6	3	4.1	1.3	1	0	0
6.2	2.9	4.3	1.3	1	0	0
5.6	2.5	3.9	1.1	1	0	0
6.9	3.1	4.9	1.5	1	0	0
6	3.4	4.5	1.6	1	0	0
5.5	2.4	3.7	1	1	0	0
5.2	2.7	3.9	1.4	1	0	0
6.1	2.8	4	1.3	1	0	0
6.4	2.9	4.3	1.3	1	0	0
5.7	2.8	4.5	1.3	1	0	0
6.1	2.8	4.7	1.2	1	0	0
6.6	3	4.4	1.4	1	0	0
6	2.2	4	1	1	0	0

6.7	3.1	4.4	1.4	1	0	0
5.8	2.7	3.9	1.2	1	0	0
5.9	3	4.2	1.5	1	0	0
6.5	2.8	4.6	1.5	1	0	0
6.6	2.9	4.6	1.3	1	0	0
5.6	2.7	4.2	1.3	1	0	0
5.7	3	4.2	1.2	1	0	0
5.9	3.2	4.8	1.8	1	0	0
5.1	2.5	3	1.1	1	0	0
6.7	3	5	1.7	1	0	0
6.3	3.4	5.6	2.4	0	1	0
6	3	4.8	1.8	0	1	0
5.8	2.7	5.1	1.9	0	1	0
6.5	3.2	5.1	2	0	1	0
6.9	3.1	5.4	2.1	0	1	0
6.5	3	5.8	2.2	0	1	0
6.3	2.9	5.6	1.8	0	1	0
6.3	2.5	5	1.9	0	1	0
7.2	3	5.8	1.6	0	1	0
7.3	2.9	6.3	1.8	0	1	0
5.8	2.7	5.1	1.9	0	1	0
6.5	3	5.2	2	0	1	0
6.4	2.8	5.6	2.1	0	1	0
7.1	3	5.9	2.1	0	1	0
7.7	2.6	6.9	2.3	0	1	0
6.3	2.8	5.1	1.5	0	1	0
7.6	3	6.6	2.1	0	1	0
6.7	3	5.2	2.3	0	1	0
5.6	2.8	4.9	2	0	1	0
7.9	3.8	6.4	2	0	1	0
7.7	3	6.1	2.3	0	1	0
6.9	3.1	5.1	2.3	0	1	0
6.4	2.7	5.3	1.9	0	1	0
7.2	3.2	6	1.8	0	1	0
5.7	2.5	5	2	0	1	0
6.4	2.8	5.6	2.2	0	1	0
6.8	3.2	5.9	2.3	0	1	0
6.7	3.3	5.7	2.5	0	1	0
6.9	3.2	5.7	2.3	0	1	0
7.4	2.8	6.1	1.9	0	1	0
6.1	2.6	5.6	1.4	0	1	0
6.7	3.1	5.6	2.4	0	1	0
7.7	2.8	6.7	2	0	1	0
6.3	3.3	6	2.5	0	1	0
6.7	2.5	5.8	1.8	0	1	0
5.8	2.8	5.1	2.4	0	1	0

4.9	2.5	4.5	1.7	0	1	0
6.8	3	5.5	2.1	0	1	0
6.4	3.2	5.3	2.3	0	1	0
6.4	3.1	5.5	1.8	0	1	0
6.2	3.4	5.4	2.3	0	1	0
7.2	3.6	6.1	2.5	0	1	0
5.9	3	5.1	1.8	0	1	0
6.2	2.8	4.8	1.8	0	1	0
6.3	2.7	4.9	1.8	0	1	0
6.7	3.3	5.7	2.1	0	1	0
7.7	3.8	6.7	2.2	0	1	0
6.1	3	4.9	1.8	0	1	0
6.5	3	5.5	1.8	0	1	0
6	2.2	5	1.5	0	1	0
5	3.6	1.4	0.2	0	0	1
5.2	3.5	1.5	0.2	0	0	1
5.7	4.4	1.5	0.4	0	0	1
4.8	3	1.4	0.1	0	0	1
5.2	4.1	1.5	0.1	0	0	1
4.7	3.2	1.3	0.2	0	0	1
5.5	4.2	1.4	0.2	0	0	1
4.9	3.1	1.5	0.1	0	0	1
4.8	3.4	1.6	0.2	0	0	1
5.4	3.7	1.5	0.2	0	0	1
5.4	3.9	1.3	0.4	0	0	1
4.8	3	1.4	0.3	0	0	1
5	3.4	1.6	0.4	0	0	1
4.9	3	1.4	0.2	0	0	1
4.9	3.1	1.5	0.1	0	0	1
5.3	3.7	1.5	0.2	0	0	1
4.3	3	1.1	0.1	0	0	1
4.8	3.4	1.9	0.2	0	0	1
5.1	3.8	1.6	0.2	0	0	1
5.1	3.3	1.7	0.5	0	0	1
5.7	3.8	1.7	0.3	0	0	1
5.1	3.8	1.9	0.4	0	0	1
5.1	3.5	1.4	0.2	0	0	1
4.9	3.1	1.5	0.1	0	0	1
5	3.5	1.3	0.3	0	0	1
5.1	3.5	1.4	0.3	0	0	1
4.4	2.9	1.4	0.2	0	0	1
5	3.5	1.6	0.6	0	0	1
4.7	3.2	1.6	0.2	0	0	1
5.8	4	1.2	0.2	0	0	1
5.4	3.9	1.7	0.4	0	0	1
4.6	3.2	1.4	0.2	0	0	1

5.4	3.4	1.7	0.2	0	0	1
4.5	2.3	1.3	0.3	0	0	1
4.4	3	1.3	0.2	0	0	1
4.6	3.4	1.4	0.3	0	0	1
5.2	3.4	1.4	0.2	0	0	1
5.5	3.5	1.3	0.2	0	0	1
5.1	3.8	1.5	0.3	0	0	1
5	3.4	1.5	0.2	0	0	1
4.6	3.6	1	0.2	0	0	1
5.4	3.4	1.5	0.4	0	0	1
5.1	3.7	1.5	0.4	0	0	1
5	3.3	1.4	0.2	0	0	1
5.1	3.4	1.5	0.2	0	0	1
4.8	3.1	1.6	0.2	0	0	1
5	3	1.6	0.2	0	0	1
4.6	3.1	1.5	0.2	0	0	1
4.4	3.2	1.3	0.2	0	0	1
5	3.2	1.2	0.2	0	0	1

Table B.1: Iris dataset showing the four distinguishing properties of three different types of iris flower, *Setosa*, *Versicolor* and *Virginica*.

B.II Crude Oil Data

B.II.i Principal Component Analysis Data

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14
1	-5.82E+06	2.00E+06	1.62E+05	-4.60E+05	1.63E+05	8.32E+04	-8.47E+04	3.37E+04	-1.68E+04	2.71E+04	7732	1.15E+04	-1.18E+04	-4.06E+04
2	-3.97E+06	3.44E+06	-4.80E+05	-1.70E+04	-2.32E+05	-4.69E+04	-2.07E+05	-2.07E+04	8.72E+04	-6.42E+04	1.36E+04	-1.38E+04	-3.01E+04	2.85E+04
3	-5.73E+06	1.84E+06	1.86E+05	8.12E+05	8.60E+05	5.90E+05	4.92E+04	-1.16E+05	5.16E+04	-2.26E+04	-6495	2.81E+04	3.36E+04	2.76E+04
4	-7.25E+06	-2.20E+06	1.64E+06	9.03E+04	5.95E+05	8.14E+04	-3.49E+04	1.86E+05	-4.84E+04	-1.17E+04	5.20E+04	6.48E+04	-1.39E+04	-6.84E+04
5	-5.78E+06	2.26E+06	-5.66E+04	-9.82E+05	-2.26E+04	-2.83E+05	-4.48E+04	4.26E+04	8.25E+04	2.12E+04	-1.78E+04	-2.93E+04	-3.67E+04	1.40E+04
6	-3.31E+05	4.39E+06	-1.06E+06	2.86E+05	-1.24E+05	1.30E+04	-1.91E+05	-4.72E+04	6.65E+04	-5256	-6731	-2.88E+04	-5.73E+04	1.44E+04
7	-1.13E+07	-2.86E+05	1.22E+06	7.51E+05	2.61E+05	6.19E+04	-4.69E+04	-8.14E+04	1.90E+05	-1.51E+05	6.62E+04	1.26E+04	2.35E+04	1.88E+04
8	8.47E+05	5.08E+06	-1.24E+06	-2.25E+05	-5.71E+04	2.58E+05	-2.67E+05	-2390	3.45E+04	-2.39E+04	-4.42E+04	7296	-1.13E+04	2.31E+04
9	-2.92E+06	4.97E+06	-5.49E+05	-3.99E+05	-6.19E+05	3.33E+05	-3.70E+05	3.76E+04	1.78E+05	9.65E+04	-6.43E+04	1.04E+04	-2760	-1297
10	-3.03E+06	-7.13E+05	6.05E+05	1.56E+06	4.45E+05	-1.60E+05	3.58E+04	6.60E+04	8.09E+04	3.20E+04	4.40E+04	-1.48E+04	1.04E+04	2.34E+04
11	-6.10E+06	1.96E+06	4.89E+04	-1.16E+06	4.17E+05	-1.60E+05	-4.66E+04	1.39E+05	-3.04E+04	-5.37E+04	1.04E+04	6.97E+04	1.57E+04	-2.29E+04
12	1.63E+06	1.43E+06	-2.51E+05	-7.89E+05	7.06E+05	-9.08E+04	1.98E+05	-5.22E+04	-5.37E+04	-6.26E+04	3958	1.62E+04	-2.14E+04	-3749
13	-5.50E+06	4.05E+06	-8.56E+05	8.17E+05	-6.86E+05	1.30E+05	4.23E+05	1.30E+05	4.50E+04	-2.39E+04	2.14E+04	-3.57E+04	-1.78E+04	1.05E+04
14	4.88E+06	1.90E+06	-7.70E+05	9.50E+05	5.81E+05	-4.03E+04	-3405	1.38E+04	3.84E+04	-1.28E+05	-2.55E+04	1.48E+04	-3.27E+04	9409
15	4.10E+06	3.04E+05	-1.24E+05	1.55E+06	2.71E+05	-8.00E+04	-3.67E+04	1.60E+04	5.98E+04	-8.87E+04	-1.85E+04	3.25E+04	-3.37E+04	7481
16	1.35E+07	-1.35E+06	3.90E+05	-1.27E+06	-2.94E+05	1.58E+04	1.59E+05	-1.06E+05	8369	-1.24E+05	1.42E+05	-2.65E+04	-1.09E+04	-4.98E+04
17	-4.57E+06	3.87E+06	-8.38E+05	1.79E+05	-1.79E+05	-3.69E+05	-1.55E+05	5.19E+04	1.16E+05	-221.4	1.47E+04	-2.63E+04	-6751	8522
18	1.54E+07	8.95E+05	-1.01E+06	1.78E+05	5.56E+05	-1.17E+05	2.77E+04	2.66E+04	-9.80E+04	-7.72E+04	-4.86E+04	-5.73E+04	-4387	8070
19	1.05E+06	1.88E+06	-6.93E+05	1.86E+06	8.08E+05	1.19E+04	6.37E+04	-3.74E+04	1.02E+05	-1.49E+05	-2.57E+04	-9644	-1.96E+04	2.01E+04
20	1.37E+07	1.24E+06	-4.33E+05	-1.72E+06	-4.64E+05	1.19E+05	-2943	-9.04E+04	8.45E+04	-2.94E+04	5.20E+04	-1.10E+04	1.11E+04	-5952
21	3.93E+06	2.55E+06	-6.56E+05	2.11E+05	4.25E+04	6119	-2.01E+04	-4.82E+04	4.97E+04	9.28E+04	-7932	-2.24E+04	-2.07E+04	2160

22	6.02E+06	1.77E+06	-5.32E+05	-2.41E+05	6.18E+05	2.46E+05	1.26E+05	-1.03E+05	1765	-6.20E+04	1.97E+04	-644.2	-1.98E+04	-1.50E+04
23	-2.42E+06	1.78E+06	7.16E+04	-1.59E+06	1.56E+05	-1.26E+05	-1.19E+05	2.24E+05	-1.77E+04	3.84E+04	5926	5.69E+04	-5782	-3.27E+04
24	-4.55E+06	6.44E+04	7.54E+05	2.24E+05	2.76E+04	1.34E+04	-1.69E+05	9.14E+04	-2.65E+04	-6.36E+04	5.17E+04	3.09E+04	4908	-7.40E+04
25	5.87E+05	9.34E+05	1.86E+05	-4.73E+05	-2.04E+05	-1.61E+05	-1.01E+05	-1.52E+04	-3.01E+04	7.45E+04	4120	3241	-3.63E+04	-2485
26	7.05E+06	-1.35E+05	1.36E+05	-8.62E+05	-1.55E+05	-8.38E+04	-8.65E+04	-1.01E+05	-9.84E+04	-3.18E+04	6.39E+04	-2.68E+04	7646	-413.4
27	5.55E+06	-3.74E+05	2.24E+05	-1.66E+05	-1.58E+05	-2.34E+05	-1.18E+04	-1.01E+05	-9.18E+04	3.64E+04	8998	-2.75E+04	-1.68E+04	2057
28	1.19E+06	-7.18E+05	5.57E+05	1.70E+05	1.23E+05	-1.69E+05	-5.08E+04	-3.82E+04	-1.04E+05	-1.52E+04	3.20E+04	1.19E+04	3.07E+04	8551
29	-1.55E+07	-1.29E+06	-1.28E+06	-1.61E+06	-7.71E+05	-7.13E+05	-1004	-2.93E+05	-8.28E+04	-2.22E+05	-1.64E+05	-1.74E+04	8.23E+04	-5270
30	7.62E+06	2.59E+06	-9.50E+05	4.14E+05	2.03E+05	2.41E+05	-2.60E+05	2.16E+04	-1.08E+05	3.36E+04	-5.89E+04	1.63E+04	2.00E+04	-1.08E+04
31	-1.47E+07	-4.55E+06	-4.93E+06	-1.04E+05	-1.76E+05	3.02E+05	2.33E+05	-2.48E+05	1.93E+04	6.18E+04	3.52E+04	-2349	8.24E+04	-2096
32	-8.76E+06	-3.63E+06	2.31E+06	2.08E+05	-8.75E+04	4.48E+04	-1.10E+05	1.10E+05	-2.70E+04	8.46E+04	-2392	-2.76E+04	1.09E+04	1.64E+04
33	-9.85E+06	-6.38E+06	2.69E+06	-8.96E+05	-1.28E+05	3.94E+04	-8.73E+04	1.38E+05	-2.35E+04	-5.50E+04	-2.81E+04	-5.30E+04	-644.1	5.54E+04
34	2.08E+06	-9.96E+05	4.63E+05	9.27E+05	8970	-2.45E+05	-5.10E+04	3.19E+04	-6.46E+04	9.31E+04	4.17E+04	-4.07E+04	2.93E+04	2.03E+04
35	5.19E+06	-4.41E+05	1.37E+05	1.12E+06	4.87E+05	5.95E+04	3.01E+04	5.12E+04	2.02E+04	4.10E+04	-5.70E+04	2.48E+04	4.49E+04	1.87E+04
36	-9.75E+06	7.25E+06	-1.54E+06	-1.06E+05	-4.41E+05	-6.66E+04	-2.47E+05	4.45E+04	1.86E+05	-1.77E+04	7.56E+04	-4.89E+04	5.32E+04	-1.24E+04
37	-3.89E+04	-1.86E+05	5.89E+05	-1.38E+05	-3.41E+05	-2.34E+05	-1.11E+04	-1.34E+05	-9222	1.38E+05	4405	-2.70E+04	-4.15E+04	-5609
38	6.52E+06	-3.53E+06	1.60E+06	7.43E+05	-1.90E+05	9.32E+04	-3.35E+04	-1.34E+05	6.85E+04	-9.46E+04	1.57E+04	3.85E+04	3639	1.21E+04
39	-8.17E+06	1.27E+06	4.90E+05	-1.13E+06	-5.36E+04	-3.87E+05	-1.58E+05	2.10E+05	-1.58E+04	-9754	2520	8.14E+04	1.06E+04	263.7
40	-5.74E+06	4.78E+04	5.86E+05	-7.09E+05	4.23E+05	-3.73E+05	1.91E+04	2.35E+05	4.04E+04	-5.18E+04	-7120	1.21E+04	5933	1.18E+04
41	2.70E+05	2.03E+06	-3.19E+05	-1.30E+06	1.35E+05	-3.18E+05	3.87E+04	2.57E+04	2.79E+04	-6707	-3614	7574	-2.62E+04	2.21E+04
42	-1.61E+07	1.88E+06	-9.94E+05	1.04E+06	2.26E+05	-2.80E+05	7.59E+05	-2.65E+05	6.11E+04	1.70E+05	1.35E+04	9.55E+04	4286	-3.49E+04
43	4.75E+06	-1.97E+06	9.47E+05	-1.09E+06	-4.26E+04	-8.49E+04	1.62E+05	-1.56E+04	1.48E+05	-3.55E+04	4.74E+04	9341	-579.9	1.68E+04
44	-9.92E+05	4.84E+06	-1.66E+06	1.48E+06	-2.20E+04	-3.61E+05	7.16E+04	4.52E+04	6.03E+04	1.01E+05	1.57E+04	-1.96E+04	-1.38E+04	-1.60E+04
45	8.20E+06	-2.40E+06	9.23E+05	-2.16E+06	6.19E+05	-5.54E+04	4.17E+05	1.16E+05	1.43E+05	2.30E+04	1.19E+05	-3.43E+04	3.22E+04	1.51E+04
46	2.78E+05	-3.99E+06	1.80E+06	1.80E+06	-2.70E+05	-1.92E+05	-2.45E+04	-1.73E+05	1.86E+04	-5.99E+04	2977	-1.76E+04	-6.42E+04	-2.10E+04
47	2.20E+06	4.25E+05	-7.31E+04	6.63E+05	-4.45E+04	-2.81E+05	-1.17E+05	1.38E+04	-1.20E+05	9.33E+04	2.28E+04	-3.54E+04	1.10E+04	1.05E+04
48	9.24E+06	-5.39E+05	3.39E+05	-9.73E+05	-3.75E+05	1.21E+05	7980	1.01E+05	1035	3.18E+04	1.60E+04	-6380	6.14E+04	2.29E+04
49	2.26E+06	-2.07E+06	9.25E+05	1.16E+06	4.72E+04	-8286	-1.50E+04	-8.52E+04	-7016	1.43E+04	-3.27E+04	-1256	-1.92E+04	5554

50	-8.82E+06	-3.55E+06	9.83E+05	-9.42E+04	-3.23E+05	-1.59E+05	5.83E+04	-1.76E+04	5.31E+04	2.28E+04	-1.14E+05	1.57E+04	7195	6521
51	-4.90E+06	-2.21E+06	1.68E+06	-9.18E+04	8.80E+04	2.13E+05	-4.34E+04	-7.97E+04	-1.06E+05	6.62E+04	1.57E+04	1.95E+04	-2.47E+04	2.89E+04
52	-9.04E+06	-2.76E+06	2.17E+06	-8.79E+04	-1.70E+04	5.62E+05	-6.20E+04	-3.05E+04	1.06E+05	-1.02E+04	-7.74E+04	-1.58E+05	-2.53E+04	-1.05E+05
53	9.20E+06	5.85E+05	-6.46E+05	5.69E+05	2.66E+05	-8.54E+04	-4.99E+04	-5.95E+04	-1.38E+05	-7.09E+04	2503	-3.09E+04	2.85E+04	-1.26E+04
54	-1.41E+06	1.44E+06	-2.40E+05	8.02E+05	-4.55E+05	8.81E+04	3.32E+05	1.03E+05	-4.08E+04	4.46E+04	-2.82E+04	-1.12E+04	1357	-1.46E+04
55	1.17E+07	6.71E+04	1.14E+05	-1.83E+06	-2.10E+05	2.62E+05	7.42E+04	-5.64E+04	-2.62E+04	-3.48E+04	-4.59E+04	3.82E+04	1.52E+04	-1.24E+04
56	9.76E+06	-1.02E+06	5.65E+05	-1.49E+06	-7.17E+04	-4.25E+04	1.63E+05	-1.05E+05	6.17E+04	5.91E+04	2.79E+04	550.9	-8965	1.91E+04
57	8.39E+06	2.08E+06	-1.17E+06	-7.91E+04	5.33E+05	-1.04E+05	4.27E+04	4.39E+04	-1.23E+05	7.70E+04	5152	-7.87E+04	3.87E+04	-3361
58	2.33E+05	6.00E+05	2.93E+05	-3.21E+04	1.15E+05	2.03E+05	-1.76E+05	7.33E+04	-6.07E+04	8.06E+04	-1.62E+04	1.07E+04	2.38E+04	1.15E+04
59	5.40E+06	9.63E+05	-3.28E+05	1.08E+06	-1.41E+06	-1.47E+05	2.73E+05	1.16E+05	-1.81E+05	-9.22E+04	1.28E+04	5.32E+04	-6.34E+04	-2.01E+04
60	1.47E+07	-2.27E+06	1.12E+06	-8.27E+05	-6.14E+05	3.00E+05	-4.97E+04	-1.01E+05	1.63E+05	4.80E+04	-9.14E+04	9.08E+04	-4.15E+04	1.66E+04
61	2.68E+06	1.08E+05	1.81E+05	8.31E+05	1.50E+05	8.49E+04	-1.48E+05	-1.96E+04	-8.34E+04	-1.77E+04	-4.17E+04	4.10E+04	2.41E+04	-9912
62	1.74E+06	-6.83E+06	2.88E+06	1.29E+06	-4.88E+05	-1.85E+05	-1.40E+05	8.54E+04	-4.11E+04	6000	8.58E+04	-3587	519.4	2.32E+04
63	8.41E+06	-1.32E+06	6.70E+05	-1.04E+06	-1.10E+05	2.78E+05	9.17E+04	-9136	5.19E+04	2.93E+04	-3.44E+04	1.11E+04	3.98E+04	-2.05E+04
64	-7.38E+06	-7.06E+06	-7.31E+05	-9.13E+05	5.72E+05	-3.80E+04	3.66E+05	2.85E+05	2.11E+04	1.54E+04	-1.43E+05	-5.12E+04	-1.96E+04	-5115
65	7.42E+06	3.93E+05	-4.73E+04	-4.56E+05	8.36E+04	7.65E+04	5.79E+04	-1.26E+05	-4.77E+04	8.53E+04	-3.69E+04	2.54E+04	-7670	1.11E+04
66	3.24E+06	4.71E+05	5.46E+04	9.35E+05	1.31E+04	6.62E+04	-1.92E+05	1.19E+04	-8.66E+04	-1.56E+04	-2.34E+04	3.01E+04	2.96E+04	-2.68E+04
67	6.58E+06	3.41E+05	-3.73E+05	8.89E+05	5.84E+05	-8999	6.88E+04	-7.91E+04	-8.48E+04	-5523	-5.49E+04	-1.55E+04	-9034	1776
68	2.50E+05	-1.45E+06	7.65E+05	9.38E+05	-7.44E+04	-2.88E+05	6.53E+04	-6.58E+04	2.12E+04	9.97E+04	-3470	-2.68E+04	-1.44E+04	-5617
69	-2.66E+07	2.05E+06	1.54E+06	-2.48E+06	4.04E+05	5.08E+05	-2.52E+04	-2.00E+05	-2.30E+05	-4632	4.72E+04	-3652	-9.02E+04	4.36E+04
70	7.83E+06	2.11E+06	-8.64E+05	-5.04E+05	1.52E+05	-1.39E+05	1.47E+04	-7.52E+04	-8.51E+04	5.25E+04	-1.23E+04	-1.95E+04	486.7	7122
71	-8.40E+06	-3.99E+06	2.65E+06	1.61E+06	-3.25E+05	1.47E+05	-2.15E+05	-9.96E+04	-1.63E+04	-6859	1.00E+05	1.54E+04	4.08E+04	4327
72	9.84E+06	-5.05E+06	1.62E+06	7.56E+05	-1.88E+05	6.60E+04	1.64E+04	8.43E+04	1.53E+05	3.88E+04	-7.57E+04	3.46E+04	1.63E+04	-3675
73	-3.92E+06	2.45E+06	-1.03E+05	9.09E+05	-1.33E+05	2.38E+05	-2.64E+05	-1.82E+04	-6.80E+04	7931	2.02E+04	7011	4.89E+04	4105
74	-1.86E+06	-1.27E+07	-1.07E+07	1.08E+04	-4.95E+04	1.45E+05	-3.37E+05	1.06E+05	1.30E+04	-1294	6.25E+04	1.78E+04	-5.41E+04	3331
75	-3.09E+06	4.30E+06	-1.02E+06	1.11E+06	-1.00E+06	6.74E+05	6.50E+05	3.05E+05	-1.35E+05	-1.01E+05	4200	-1.75E+04	1.54E+04	3.74E+04
76	-1.09E+06	1.05E+06	3.24E+05	-4.44E+05	-1.30E+05	1.43E+05	-2.27E+05	3.35E+04	-8.64E+04	6.40E+04	7261	1.12E+04	1.58E+04	-7681

Table B.2: First 14 principal components for the full set of crude oils.

B.II.ii Physical Property Data

Crude No.	Crude Name	Sample No.	Pour Point (°C)	CSI Pour Point (°C)	TAN (mgKOH/g)	CSI TAN (mgKOH/g)	API gravity	CSI API	Sulphur (wt%)	CSI Sulphur (wt%)
1	Amna Crude	56	24	15	0.1	0.1	37.1	37.3	0.09	0.08
2	Arabian Heavy Crude	61	-21	-45	0.20	0.26	28.4	28.1	2.71	2.92
3	Arabian Light Crude	4	-36	-15	0.05	0.08	32.4	31.8	1.99	1.99
4	Arabian Medium Crude	28	-27	-33	0.20	0.14	30.7	30.5	2.49	2.45
5	Ardjuna Crude	74	27	24	0.30	0.4	33.3	35.5	0.12	0.10
6	Asgard Crude	51	-9	-21	0.05	0.01	39.8	40.5	0.20	0.20
7	Bach Ho Crude	43	39	39	0.05	0.03	40.0	40.6	0.04	0.06
8	Bachaqero Crude	10	3	15.56	4	4.15	11.3	10.8	2.75	2.78
9	Basrah Light Crude	11	-12	-17.78	0.15	0.15	31.9	32.0	2.58	2.60
10	Beatrice Crude	60	9	23.89	0.1	0	38.5	39.4	0.05	0.05
11	Bintulu Crude	64	-36	-9	0.25	0.28	32.2	32.4	0.11	0.11
12	Bonny Light Crude	54	-18	6	0.20	0.18	34.5	33.8	0.15	0.14
13	Brega Crude	12	6	0	0.15	0.06	38.7	39.9	0.23	0.20
14	Brent Crude	25	9	-6.11	0.05	0.02	38.0	37.6	0.40	0.41
15	Bu Attifel Crude	38	39	43.33	0.1	0.2	42.1	43.5	0.04	0.04
16	Burgan Crude	66	-18	-12	0.30	0.21	23.5	23.2	3.19	3.05
17	Champion Crude	42	-36	-30	0.40	0.56	22.0	23.7	0.14	0.13
18	Cheleken Crude	13	-9	6	0.35	0.45	32.9	35.6	0.16	0.14
19	Curlew Crude	49	12	-20.7	0.05	0.01	43.1	42.8	0.15	0.16
20	Cusiana Crude	32	15	6	0.7	0	41.6	42.3	0.17	0.14
21	DNSCO	70	-36	-36	0.25	0.3	35.8	35.6	0.24	0.24
22	Draugen Crude	68	-27	-15	0.10	0.09	40.3	40.2	0.16	0.15
23	Dulang Crude	29	33	33	0.10	0.42	37.5	37.7	0.06	0.05
24	Duri Crude	45	21	15	1.30	1.3	20.9	20.5	0.20	0.22
25	El Ward Crude	41	18	-	0.05	-	34.8	-	0.42	-
26	Flotta Crude	1	-3	-23.33	0.1	0.05	34.4	34.4	1.24	1.19
27	Foinaven Crude	65	0	9	0.15	0.15	26.3	26.3	0.38	0.36

28	Forcados Crude	75	-24	-15	0.35	0.38	30.3	30.1	0.17	0.15
29	Forties Crude	5	-12	-18	0.1	0.12	40.5	41.5	0.26	0.25
30	Gemsa Crude	9	24	27	0.20	0.28	30.0	30.5	2.02	2.09
31	Gippsland Crude	52	18	1.67	0.15	0.01	44.8	44.1	0.10	0.10
32	Gulfaks Crude	27	-24	-18	0.1	0.12	35.7	36.8	0.27	0.25
33	Kalamkas Crude	53	-36	-9	0.15	0.8	24.3	24.3	1.62	1.50
34	Kekwa Crude	33	36	36	0.1	0.06	37.3	37.0	0.05	0.04
35	Kirkuk Crude	73	-24	-30	0.10	0.11	33.6	33.5	2.24	2.26
36	Kittewake Crude	62	-12	-6	0.05	0.04	39.9	39.1	0.41	0.42
37	Kumkol Crude	55	15	24	0.05	0.17	40.7	40.8	0.08	0.08
38	Kuwait Crude	24	-27	-21	0.05	0.15	30.6	30.3	2.61	2.60
39	Lagocinco Crude	76	-3	-35	0.05	0.14	34.1	34.5	1.17	1.30
40	Lagotreco Crude	58	-36	-24	0.15	0.41	33.1	30.9	1.08	1.10
41	Laguna Crude	34	6	9	4.6	3.7	11.1	11.1	2.87	2.71
42	Lavan Blend	3	24	-	0.05	-	15.3	-	3.35	-
43	Lokele Crude	44	-36	-30	2.1	2.84	19.9	20.0	0.43	0.41
44	Malampaya Crude	69	21	6	0.05	0.08	32.7	30.0	0.64	0.71
45	Masila Crude	19	9	1.67	0.05	0.03	30.6	30.9	0.56	0.57
46	Maya Light Crude	40	-12	-30.01	0.05	0.06	29.3	30.4	1.54	1.55
47	Minas Crude	72	39	36	0.2	0.06	34.8	34.2	0.09	0.09
48	Norne Crude	37	24	21	0.05	0.01	32.2	32.7	0.18	0.19
49	Oman Export Crude	26	-21	-27	0.45	0.59	32.7	32.9	1.07	1.07
50	Qatar Crude	71	-6	-9	0.05	0.04	40.7	41.2	1.18	1.22
51	Qatar Marine Crude	7	-3	-12.22	0.10	0.11	34.9	35.3	1.58	1.57
52	Rabi Export Blend	16	27	24	0.15	0.1	32.9	33.8	0.09	0.06
53	Rabi-blend Crude	20	30	9	0.15	0.09	33.9	33.3	0.07	0.07
54	Ratawi Crude	36	-18	-33.89	0.15	0.18	24.0	24.2	3.99	4.00
55	Rio Negro Crude	23	-18	-12	0.05	0.07	27.1	26.8	2.05	1.99
56	Ruby Crude	63	30	27	0.30	0.16	35.6	37.1	0.08	0.07
57	Saharan Blend	46	-27	-15	0.05	0.04	45.9	45.6	0.08	0.07
58	Sarir Crude	22	24	21	0.05	0.05	36.4	37.7	0.14	0.16
59	Schiehallion Crude	21	9	6	0.35	0.39	25.5	26.3	0.49	0.45
60	Seme-2 Crude	57	-9	-27	0.10	0.06	22.6	22.6	0.41	0.41

61	Shengli Crude	48	18	12	1.75	1.4	19.3	19.8	1.71	1.69
62	Soroosh Crude	35	-21	-9	0.3	0.27	19.7	19.7	3.31	3.15
63	Souedieh Crude	39	-24	-30	0.10	0.2	24.4	24.3	3.98	4.00
64	South Arne Crude	14	6	9	0.05	0.11	37.2	37.1	0.21	0.20
65	St. Joseph Crude	31	30	9	0.2	0.17	29.3	30.4	0.11	0.09
66	Statjort Crude	6	3	-27	0.05	0.04	37.7	37.5	0.34	0.37
67	Syrian Light Crude	2	9	-9.02	0.10	0.03	36.0	37.7	0.74	0.75
68	Tanak Crude	8	12	-9.44	0.05	0.04	35.0	36.0	1.00	0.55
69	Tapis-Pulai Crude	50	12	12	0.15	0.14	45.6	45.8	0.03	0.03
70	TJP Crude	30	6	12	4.8	4.2	11.5	11.2	2.70	2.63
71	Troll Crude	67	-27	-48	1.10	1.06	27.7	27.5	0.28	0.26
72	Ukpokiti Light Crude	59	9	1.67	0.30	0.21	40.4	41.8	0.09	0.09
73	Ural Crude	17	-3	-3.89	0.10	0.17	31.6	31.4	1.35	1.34
74	Varg Crude	15	12	15	0.05	0.04	37.0	37.1	0.24	0.22
75	West-Nederland Crude	18	-27	-27	0.95	0.83	21.0	21.2	0.52	0.52
76	Zuata Crude	47	12	24	3.1	3.3	9.2	9.3	3.32	3.35

Table B.3: Cargo properties for the set of 76 crude oils. Two separate measurements exist; the first column was recorded by Shell researchers, the second is published on the Crude Services intranet (CSI). Large discrepancies exist between the two datasets.