ISBN 0-315-55471-1

THE UNIVERSITY OF ALBERTA

INDUCTIVELY COUPLED PLASMA ATOMIC EMISSION SPECTROMETRY USING AUTOMATED SOLUTION PREPARATION

by

MICHAEL STEWART ©

A Thesis

Submitted To The Faculty Of Graduate Studies And Research

In Partial Fulfilment Of The Requirements For The Degree

Of Doctor Of Philosophy

The Department of Chemistry

Edmonton, Alberta

Fall, 1989

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR:	Michael Stewart
TITLE OF THESIS:	INDUCTIVELY COUPLED PLASMA ATOMIC EMISSION SPECTROMETRY USING AUTOMATED SOLUTION PREPARATION.
DEGREE FOR WHICH THESIS WAS PRESENTED:	Ph.D.
YEAR THIS DEGREE GRANTED:	1989

M Stewart..................................
(signed)

Department of Chemistry,
Indiana University,
Bloomington, Indiana,
47405, U.S.A.

Date: 20th.....July.1989...

There is a theory which states that if ever anyone discovers exactly what the Universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable.

There is another theory which states that this has already happened.

Douglas Adams from "The Restaurant at the End of the Universe".

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled INDUCTIVELY COUPLED PLASMA ATOMIC EMISSION SPECTROMETRY USING AUTOMATED SOLUTION PREPARATION submitted by MICHAEL STEWART in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

[illegible]
Dr. G. Horlick (supervisor)

[illegible]
Dr. J. S. Martin

[illegible]
Dr. B. Kratochvil

[illegible]
Dr. J. A. Plambeck

[illegible]
Dr. R. W. Toogood

[illegible]
Dr. M. J. Stillman (external examiner)

Date: 20th July 1989

In memory of my father, Joseph R. Stewart and to
my mother Kathleen M. Stewart.

Abstract

The ideal instrument for chemical analysis should be able to accept any sample, provide any information desired and return the sample unharmed. The closest approach to this ideal, with present instrumentation, is to combine sample preparation and measurement in a single automated instrument. The capabilities of such an instrument extend beyond the automation of analysis. With suitable modification of the data analysis performed, such an instrument is able to automate some research tasks.

An instrument was developed which combined an automated solution preparation system with measurement by ICP-AES. The solution preparation system included a microprocessor controlled syringe pump to perform the dilutions required and a robotic arm to perform the necessary manipulation of objects within the system. All the electronic devices were connected to a central computer (IBM-AT) via an RS-232 multiplexer. A control language was developed, using FORTH, to allow programming of the system.

This instrument was used to develop a novel analysis procedure. This procedure was designed to provide an optimal set of calibration standards for the samples being analysed. Feedback was introduced by using previous measurement results to govern the compositions of the solutions prepared. The procedure was tested with multi-element solutions of known composition. Also well water samples, which had been analysed previously, were used for verification of the method.

The ability of the system to automated research related tasks is demonstrated. Two studies on matrix related interferences were performed

The instrument was used to prepare solutions with varying amounts of the matrix component (calcium or ethanol) and to collect the emission intensities. The effect of nebuliser gas flow rate was also studied. The interference due to calcium was found to be consistent with a reduction in the temperature of the plasma, whereas that due to ethanol was more complex.

Improvements to the design of the system and programming language can be made. The system can then be used to automate a wide range analytical procedures. The future developments of the system described all lead to the use of Artificial Intelligence. The combination of systems such as this one and AI looks very promising.

Acknowledgements

Firstly, I would like to thank my parents without whom I would not be possible.

The advice and direction provided by my supervisor, Dr G. Horlick has been greatly appreciated. Thanks must also go to my research group for many stimulating discussions, as well as to the honourary group member Charles Lucy who provided more than one good idea for this work.

This research would probably not have been possible without the skill of the staff of the Chemistry Department's Electronic, Machine and Glass shops. For equipment manufactured and repaired, I give them many thanks.

TABLE OF CONTENTS

CHAPTER PAGE

1. Automation in Analytical Chemistry 1
   1.1 Approaches to Automation 4
   1.2 Automation and Atomic Emission Spectroscopy 7
2. System Description 14
   2.1 Computing Hardware 18
   2.2 Communications System 26
   2.3 Robot 29
   2.4 Syringe Pump 62
   2.5 Peristaltic Pump 70
   2.6 ICP Spectrometer 74
   2.7 Balance 81
3. Solution Preparation 85
   3.1 Calibration of Pumps 88
   3.2 Factors Influencing Solution Preparation 95
   3.3 Solution Preparation 108
4. Automation of Analysis 118
   4.1 Development of Automated Analysis Procedure 126
   4.2 Analysis of Well Water Samples 142
5. Automation of Matrix Studies 145
   5.1 Matrix Study Procedures 152
   5.2 Matrix Effects Due to Ca 154

5.3 Matrix Effects Due to Ethanol 165

5.4 Matrix Effects and Their Possible Origin 174

6. Conclusions and Future Directions? 176

6.1 Hardware and Software Improvements 177

6.2 Further Developments in Automated Analysis 182

6.3 Points Arising from Automated Matrix Studies 186

6.4 Future Trends (Blue skies ahead) 188

Bibliography 198

APPENDIX PAGE

A. Summary of Commands in ACS 204

B. Listing of Solution Preparation and Measurement Routines 223

C. Listing of Automated Analysis Program 246

D. Listing of Matrix Study Program 272

LIST OF TABLES

TABLE PAGE

1. Definition of various solution types 12
2 Configuration of Serial Port on Multiplexing Unit 28
3 Summary of communication commands 29
4 Summary of Laboratory Robots available 35
5 Specifications of RM-501 Robot and Drive Unit 41
6 Summary of commands for RM-501 Robotic arm 42
7 Summary of commands for controlling RM-501 Robot 47
8 Table of Denavit-Hartenberg Link Parameters for a RM-501 Robot 56
9 Specifications of Syringe Pump 63
10 Summary of Syringe Pump commands 67
11 Summary of commands for control of syringe pump 69
12 Summary of commands for peristaltic pump and control of valves 71
13 Summary of commands for control of peristaltic pump 73
14 Typical plasma operating conditions 75
15 Wavelengths of spectral channels available on ARL 34000 76
16 Summary of commands used with the ARL 34000 spectrometer 79
17 Commands used by balance 82
18 Summary of commands for use with the balance 83
19 Regression coefficients for calibration of syringe pump in transfer and dispense mode 95

20 Concentration of elements in standard solutions and wash solution 102

21 Summary of F-ratios obtained from Analysis of Variance 107

22 Regression coefficients for logarithmic calibration plots of various elements 117

23 Summary of "analytical" results obtained from simulation of automated analysis procedure 126

24 Summary of calibration results for wide dynamic range calibration of spectrometer 130

25 Composition of test samples used to verify automated analysis procedure 134

26 First four standards for analysis of three samples 136

27 Summary of analysis results and estimated accuracy for test samples. 140

28 Results from previous well water samples analysis 143

29 Summary of analysis results for well water samples 144

30 Table of spectroscopic data for some channels of the ARL 34000 spectrometer 155

LIST OF FIGURES

FIGURE PAGE

1 Four steps of Chemical Analysis. 2

2 Structure of an inductively coupled plasma. 8

3 Block Diagram of automated analysis system 11

4 Schematic Diagram of Solution Preparation Station 15

5 Photograph of robotic system 16

6 Communication network used in Solution Preparation Station 17

7 Example of a word definition in the Forth programming language 21

8 Example of the definition of a defining word in the Forth programming language 22

9 Common geometries for laboratory robots 31

10 Steps required for motion control 37

11 Partial listing of a program for the RM-501 Robot 43

12 Timing sequence used by RM-501 Robot when transmitting a character on the RS-232 interface 45

13 Diagram of RS-232 cable used for interfacing to a RM-501 Robot 46

14 Example of low level robotic command 49

15 Composition of an homogeneous transform matrix 53

16 Definition of Denavit-Hartenberg link parameters 54

17 Definition of coordinate frames used with RM-501 Robot 55

18 Skeleton diagram of RM-501 Robot 60

19 Design of solution uptake probe for use with a robotic hand 64

20 Design of stand holder for solution uptake probes 65

21 Example of low level syringe pump command 68

22 Example of command which receives data from syringe pump 70

23 Example low level command for use with peristaltic pump 74

24 Example of command to transfer data between spectrometer and computer 78

25 Example of main routine of a Basic program run on spectrometer 80

26 Basic program used to execute robotic programs 81

27 Example of low level command used to control top loading balance 84

28 Illustration of solution preparation method 86

29 Calibration plot for peristaltic pump 88

30 Standard deviations of volume dispensed 89

31 Drift in weight of liquid transferred by peristaltic pump over the short term 90

32 Pseudo-code for main routine of syringe calibration procedure 92

33 Effect of size of air gap on volume of liquid transferred 93

34 Calibration plots for transfer and dispense mode 94

35 Uncertainty in volume of liquid transferred as function of volume 96

36 Structure of record for defining a solution 97

37 Photographs of potassium permanganate solution prior to homogenisation by shaking and following homogenisation by shaking 98

38 Photograph of potassium permanganate solution during homogenisation by shaking 99

39 Illustration of measurement procedure 101

40 Scatterchart for seven calcium solutions 104

41 Main routine of ANOVA experiment 106

42 Typical results obtained from an ANOVA experiment 108

43 Calibration plots constructed using solutions prepared manually and using robotic methods 115

44 Overlay of calibration plots constructed using solutions prepared manually and using robotic methods 116

45 Example of a typical linear calibration plot with 95% confidence bands 121

46 Plot of uncertainty in estimated sample concentration as function of relative concentration of next standard to be added to a calibration 122

47 Plots of estimated composition of test samples and standard and uncertainties in sample composition as function of number of points in calibration plots for simulation of automated analysis procedure 125

48 Typical quality control chart for semi-quantitative analysis of test samples 131

49 Partial listings of programs used for automated analysis procedure 133

50 Typical record for one calibration point of automated analysis procedure 135

51 Calibration plot for calcium obtained from automated analysis 138

52 Plots of estimated composition of samples and standard and uncertainties in sample composition as function of number of points in calibration plots for simulation of automated analysis procedure 139

53 Multi-element Saha-Boltzmann Plot 150

54 Variation of matrix effect with calcium concentration for various atom and ion emission lines 156

55 Variation in ionisation temperature as measured from ion/atom intensity ratio with calcium concentration 158

56 Relationship between relative sensitivity and energy of emitting state with respect to the ground state of the ion 159

57 Relationship between relative sensitivity and energy of emitting state with respect to the ground state of the ion with corrections made for partial ionisation for atom lines 160

58 Emission intensity as a function of nebuliser gas flow rate for various calcium concentrations 161

59 Ionisation temperature as function of nebuliser gas flow rate for various calcium concentrations 162

60 Relationship between relative sensitivity and energy of emitting state at different nebuliser gas flow rates 164

61 Variation of matrix effect with ethanol concentration for various atom and ion emission lines 166

62 Variation in ionisation temperature as measured from ion/atom intensity ratio with ethanol concentration 167

63 Relationship between relative sensitivity and energy of emitting state with respect to the ground state of the ion with corrections made for partial ionisation for atom lines 169

64 Emission intensity as a function of nebuliser gas flow rate for various ethanol concentrations 170

65 Ionisation temperature as function of nebuliser gas flow rate for various ethanol concentrations 171

66 Relationship between relative sensitivity and energy of emitting state at different nebuliser gas flow rates 173

67 Benchtop Metaphor for chemical workstation 189

68 The Recipe Book provides a method of storing procedures that can be used repeatedly 191

69 Information sources available to an Analytical Chemical Expert System 193

70 Division of Analytical Chemical Expert Sytem into two knowledge systems 195

71 Analytical Chemical Expert System modified for Chemical Research problems 196

Chapter 1

Automation in Analytical Chemistry.

A goal of analytical chemistry is the development of an instrument which would be able to analyse any sample, provide the user with the information desired plus any other interesting information acquired and return the sample unharmed [1]. As yet no such instrument has been developed. The closest that present instrumentation can approach this ideal is an automated system combining sample handling with several instrumental techniques and computer based data and information processing.

A dictionary definition of "automatic" is "working by itself without direct and continuing human operation" (Chambers 20th Century Dictionary). Within this definition is the concept of performing tasks previously done by humans using inanimate devices, either mechanical or electronic. Automation is the process of combining automatic devices to perform more complex tasks, again, previously done by humans.

Chemical analysis typically involves four steps; sample preparation, measurement, data analysis and data interpretation, as shown in Figure 1 [2]. When performed manually the procedure starts with the dissolution of the sample in a suitable solvent. A chemical reaction is performed and the amount of reagent or product is measured as typified by titration or gravimetric analysis. The results are recorded in a laboratory notebook and the results from standards are then used to calculate the composition of the samples.

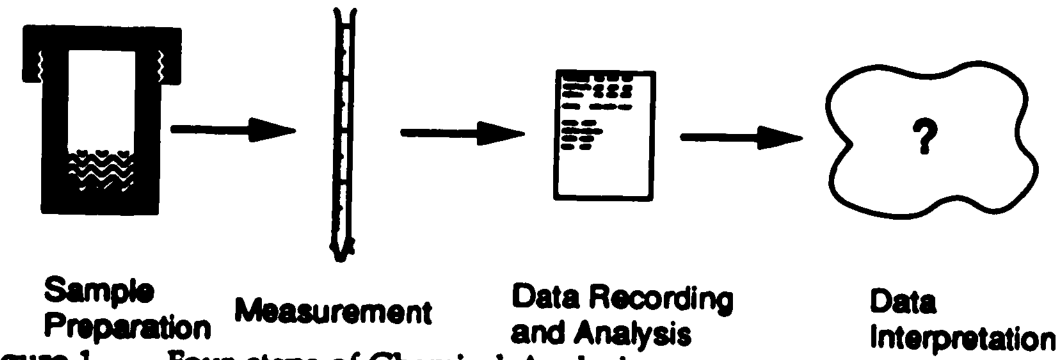


Figure 1 Four steps of Chemical Analysis.

Automatic analysis involves the automation of the measurement step and possibly some of the sample preparation as part of the sample introduction to the measurement device. The automation of the measurement step can be done by measuring a physical property which is related to the concentration of the chemical species present, for example, the amount of light absorbed at a particular wavelength or the potential developed at a surface. The measurement generates a voltage or a current which is read from a meter and then recorded. The calculations performed involve the calibration of the instrument and the determination of the sample composition.

The next level of automation is the automatic generation of results. This combines an automatic analyser with an automatic data recording device. This is achieved by digitising the signal and recording the result in a computer. The computer is then used to perform the calibration calculations and the determination of the sample composition. The choice of the calibration standards and the interpretation of the results are still tasks performed manually.

The total automation of analysis needs to be able to decide on the calibration standards to use and evaluate the results to test whether they meet the desired levels of precision and/or accuracy. It must then be able to modify the calibration to met the needs of the sample. This is done by choosing the calibration standards that are best suited to the sample. It will also require the ability to identify interferences from other components in the sample and to be able to correct for these.

The analysis procedure can be regarded as a special case of model testing. The calibration of the instrument assumes a particular model for the relationship between the signal and sample composition. The automatic analytical instrument is testing the results against this model and making a prediction based on the fit obtained. If interferences are found, the system has identified a deviation from the model and attempts to correct for this. The process of research is the generation and testing of models. The automation of research would involve the testing of models for relationships other than those between signal and concentration. For example, the definition of the relationship between the signal and the temperature of a source would be a typical research goal. The automatic analysis system could probably be modified to test models provided for it. Ultimately, a totally automated research system would be able to generate its own models prior to testing them.

1.1 Approaches to Automation.

There are two basic philosophies to automation. The first is to automate a manual method by mimicking each procedure. The alternative philosophy states that the mimicking of manual procedures will not necessarily produce an optimised system. Therefore the automated method approaches the analysis in a manner totally different to the manual method [1].

The solutions produced by these different approaches lead to very different types of systems. Those produced from the automation of a manual method can often retain the flexibility of the manual procedures. Such systems can have a capability that goes beyond that of the manual method automated. Whereas, those which optimise the procedure being automated will often have less flexibility. They have been designed for the immediate task and so are only capable of performing that task. If greater flexibility is required it has to be designed back into the system. These two approaches typify the compromise that must be made between flexibility and speed when automating a procedure.

The philosophy which mimics manual methods tends to lead to batch mode systems. The alternative philosophy is exemplified by continuous flow systems.

Continuous flow-analysers are dedicated systems capable of a very large number of analyses and so are standard pieces of equipment in most clinical laboratories where their large throughput is necessary [3]. These systems consist of a continuous flow of solvent into which samples are introduced along with air bubbles to maintain separation between the samples. As the samples flow through the system they can be mixed with reagents, or undergo various treatments such as dialysis, filtration, incubation etc. The product of the reactions is then measured by a suitable sensor, e.g. colorimeter or electrochemical detector [3]. The analysis is performed in a single line from the sample introduction to the detector. Once a line has be made for one analysis procedure it cannot be used for a different procedure. In order to perform multiple analysis procedures parallel systems are needed.

Alternatively, tne flexibility of continuous flow system can also be increased by removing the air bubbles and injecting the sample. This is done in flow injection analysis (FIA) [4]. The principle on which FIA depends for its versatility is the controlled dispersion of the samples in a flowing stream [4]. The use of double injection, where both the sample and reagent are injected and their zones allowed to merge downstream, permits many different reactions to be used [4]. The signal produced is always transient, the height of the signal being proportional to the sample concentration when a linear detector is used. The concentration profile can be used to provide useful information, specially when the whole profile is used rather than just the peak maximum or the integrated area.

The concentration profile generated by an FIA system has been used to automate an investigation of interferences and matrix effects in Inductively Coupled Plasma Atomic Emission Spectrometry (ICP-AES) [5]. As the solution that is being introduced into the ICP is inhomogeneous, this introduces some uncertainty about the concentrations of the species entering the ICP. This uncertainty is removed if homogeneous solutions are used instead.

For many analyses samples need to be homogeneous, and so must be prepared in bulk. The batch mode of automation is ideally suited for this form of sample preparation.

A typical example of batch mode analysis is titration. The first successful automatic titrator is that of Malmstadt and Fett [6]. The device simply automated the manual processes by using an electronically operated valve on the burette and included a sensor to provide feedback in order to detect the end-point of the reaction. This exactly mimicked the manual titration procedure. The development of autotitrators has seen the burette being replaced by a syringe which is controlled by a motor. The displacement of the syringe is then controlled by the autotitrator. From the displacement the volume of titrant added is therefore known. Further development of automatic titration will see more use of weight titrations [7,8]. These offer the possibility of greater precision, less reagent consumption and simplicity of design.

The possibility of completely mimicking manual procedures has been made possible by the introduction of laboratory robots. The use of robots in analytical chemistry was predicted in 1971 [9]. In this review, the robot was seen to be a possible solution to the problems caused by the increasing demands being made on clinical laboratories. It was not until 1982 [10,11] that the first analytical systems which included robots in them appeared. In one of the first applications the robot was used to measure the spatial distribution of physico-chemical properties of liquids [10] and in another application a sample preparation system was developed that was capable of preparing solutions and performing acid-base titrations [11].

Reviews have appeared in the chemical [12-18] and non-chemical [19] literature. These reviews have focussed on all aspects of robotics, from general background information [12,13,14,15], the differences between "hard" and "soft" automation [16] and the role of robots in different industries such as pharmaceutical [17] or clinical [18] to use in organic chemistry [19].

More recent applications of robotic systems have been their use for drug testing [20,21]. The automation of optimisation using a Simplex method has also been achieved [22,23]. The automation of a complexometric titration [24] was achieved by coupling an expert system with a robotic system. The application of a robotic system to a kinetic study [25] demonstrated the ability of such systems to automate research projects.

The systems mentioned above have all tended to use procedures which are very similar to the manual methods that they were replacing. They all retained a great deal of flexibility. For example the system used to perform the kinetic study [25] used many of the components used in the earlier optimisation system [22,23]. One of the main advantages of the batch mode systems over continuous flow is that they can be used to solve a more diverse set of problems.

1.2 Automation and Atomic Emission Spectroscopy.

The inductively coupled plasma (ICP) has long been used as a source for atomic emission spectrometry. The structure of an ICP source is shown in Figure 2. The plasma is formed by the outer argon gas in the region inside the coils by the interaction of the electrons in the plasma and the fluctuating

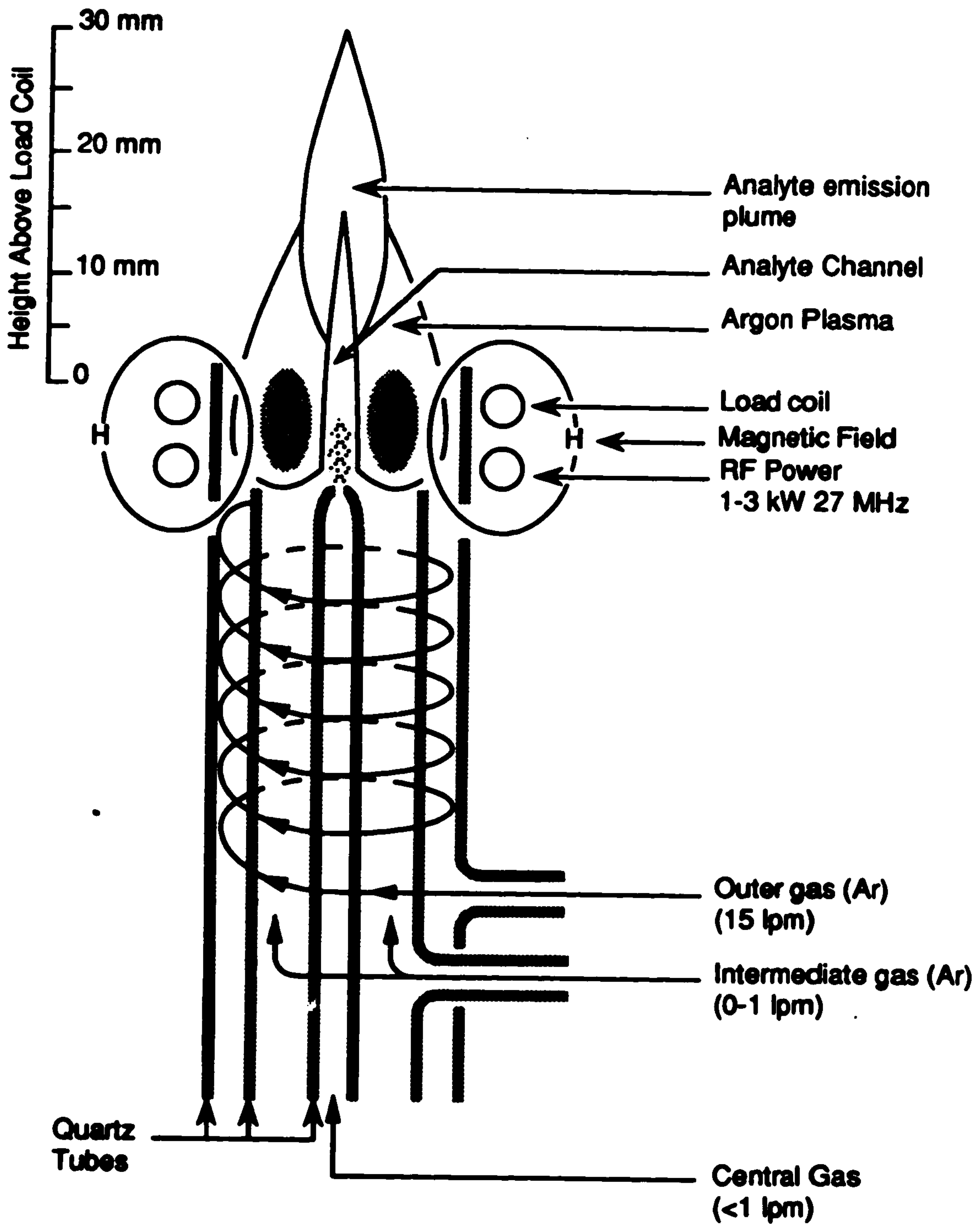


Figure 2 Structure of an inductively coupled plasma.

magnetic field from the rf current in the load coils. The intermediate gas allows small adjustments to be made in position by raising or lowering the plasma. Normally the sample is introduced as a solution via a nebuliser which forms an aerosol. The larger drops of the aerosol are removed during their passage through a spray chamber. The aerosol in the central gas penetrates the centre of the plasma discharge, here the sample is desolvated, vapourised and atomised. Emission from the analyte species is viewed at a distance of approximately 15mm above the load coils. One of the characteristics of the ICP as a source is that the intensity of the emission from it is linearly related to the concentration of the emitting species in the solution over a wide dynamic range [26].

Analysis by atomic emission spectrometry proceeds by initially dissolving the sample, typically by acid digestion. A series of calibration standards are then prepared by dilution of stock solutions. The measurement is then performed. Calibration of the instrument and the calculation of the sample composition are then done. It may be necessary to prepare more calibration standards if it is found that they did not bracket the samples. Once the analysis has been performed the results obtained are then documented. This analysis procedure can therefore be divided into three unit operations. These are dilution, measurement and control.

Two areas of active research in ICP-AES are improved understanding of the fundamental processes in the plasma and of the interferences caused by high concentrations of other species in the solution being measured. For these research applications the same dilution and measurement methods are used and only the control operation is different from the analysis application.

Therefore any system designed to automate analysis, with appropriate changes in the control mechanism, could perform some research tasks. The change in the control corresponds to changing the model being tested. For example, if a matrix effect is suspected to be due to a change in the plasma temperature then a model which describes th ange in intensity expected with a change in temperature could be used in the control system.

Automation of atomic emission spectroscopy therefore requires four sub-systems, a programmable dilution system, a solution delivery system, a measurement system and a control system. The relationship between these sub-systems is shown in Figure 3. The dilution system can be programmed to prepare solutions containing several components that can be varied independently. Conceptually, the components of a solution can be divided into two types; those which are included as standards and those included as other auxiliary components. The solution delivery system acts as an interface between the dilution system and the measurement system. The measurement system in this work is an ICP Atomic Emission Spectrometer which can be controlled by an external computer. It can be instructed to make a measurement and then send the results obtained back to the computer. The control system is a computer which coordinates the action of each sub-system, collects results and delivers analysis reports to the user.

This system is capable of performing the following analytical tasks;

<u>Semi-Quantitative Analysis</u> – The analysis of a sample and calculate the approximate composition of the sample using a stored calibration of the instrument.

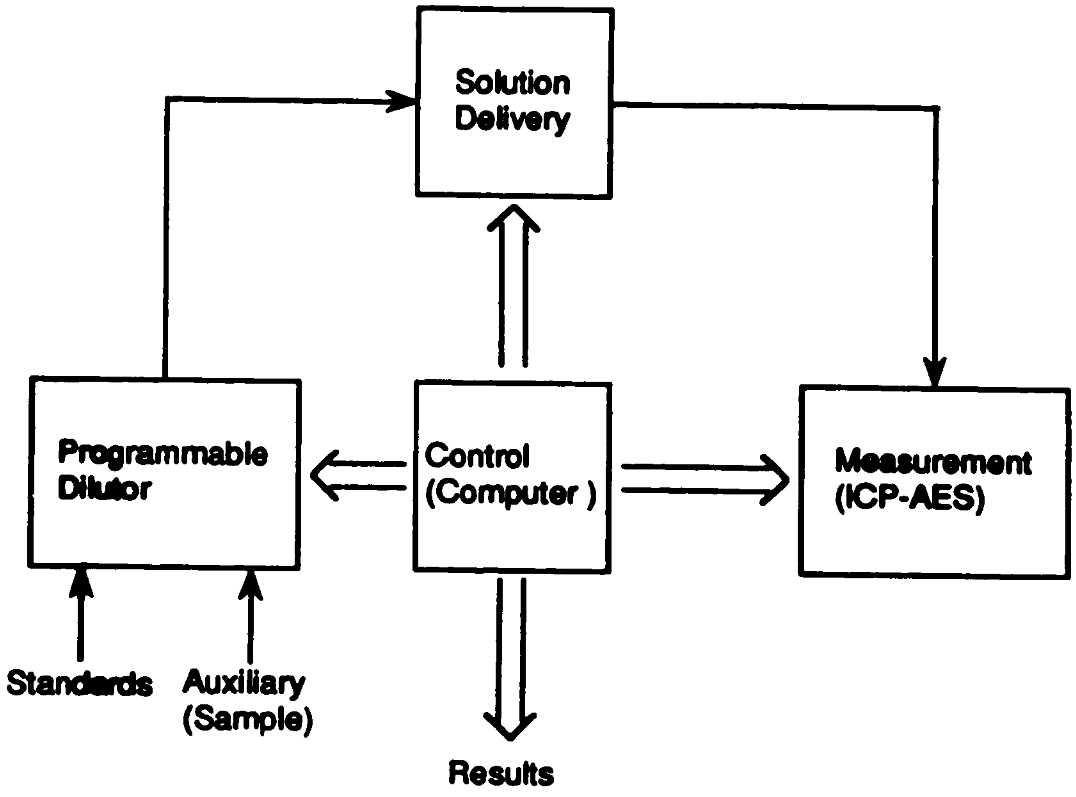


Figure 3 Block Diagram of automated analysis system.

Normal Calibration - The analysis of sample using a set of calibration standards which is predetermined.

Adaptive Calibration - The analysis of a sample using a set of calibration standards which is optimised for the sample.

Matrix Study - The measurement of the change in sensitivity due to the presence of a matrix component.

Standard Addition Calibration - The analysis of a sample using the standard addition method.

Solution Type	Standards	Auxiliary Components
Calibration Solution	Variable	None
Matrix Study	Fixed	Matrix - Variable
Standard Addition	Variable	Sample - Fixed
Internal Standard	Variable (None)	Internal Standard - Fixed (Sample - Fixed)
Matrix Matched Standard	Variable	Matrix - Fixed

Table 1 Definition of various solution types in terms of standard and auxiliary components.

<u>Calibration Using An Internal Standard</u> - The analysis of a sample with the inclusion of an internal standard.

<u>Adaptive Matrix Matching</u> - This involves the identification of any matrix interactions and analysis of the sample using matrix matched standards, the matrix matching being customised to the sample.

These tasks are made possible by the use of a programmable dilutor. The solutions used for each of the tasks each contain standards and auxiliary components. These can be either fixed or variable. The various combinations of fixed and variable standards and auxiliary components and how they relate to the tasks above is shown in Table 1.

In this work the system of the type proposed in Figure 3 will be developed. Also the viability of automating some of the analytical tasks mentioned will be demonstrated.

Chapter 2

System Description.

A schematic overview of the solution preparation system developed is shown in Figure 4. A photograph of it is shown in Figure 5. The overall system consists of a number of sub-systems as shown in Figure 6. These are a computer, a communications interface, a robot, a syringe pump, a peristaltic pump and a spectrometer. An electronic balance was included for the calibration and testing of the syringe pump. Each of these sub-systems will be described in more detail in this chapter.

All the sub-systems were controlled from an IBM computer. This was done by using RS-232 interfaces, since most manufacturers provide this either as standard or as an option for their equipment. The number of sub-systems precluded them all being connected directly to the computer, since the IBM-AT can only support directly 4 serial devices [27]. Communication boards are available which allow more than four serial ports. However since these boards will use non-standard system calls there may not be sufficient support for software development. A port expander which could switch communications from one host device to several peripheral devices was chosen, since the switching of communication between devices would require only an appropriate message to be sent. The communications network developed is shown in Figure 6.

In addition to these main sub-systems several ancillary components

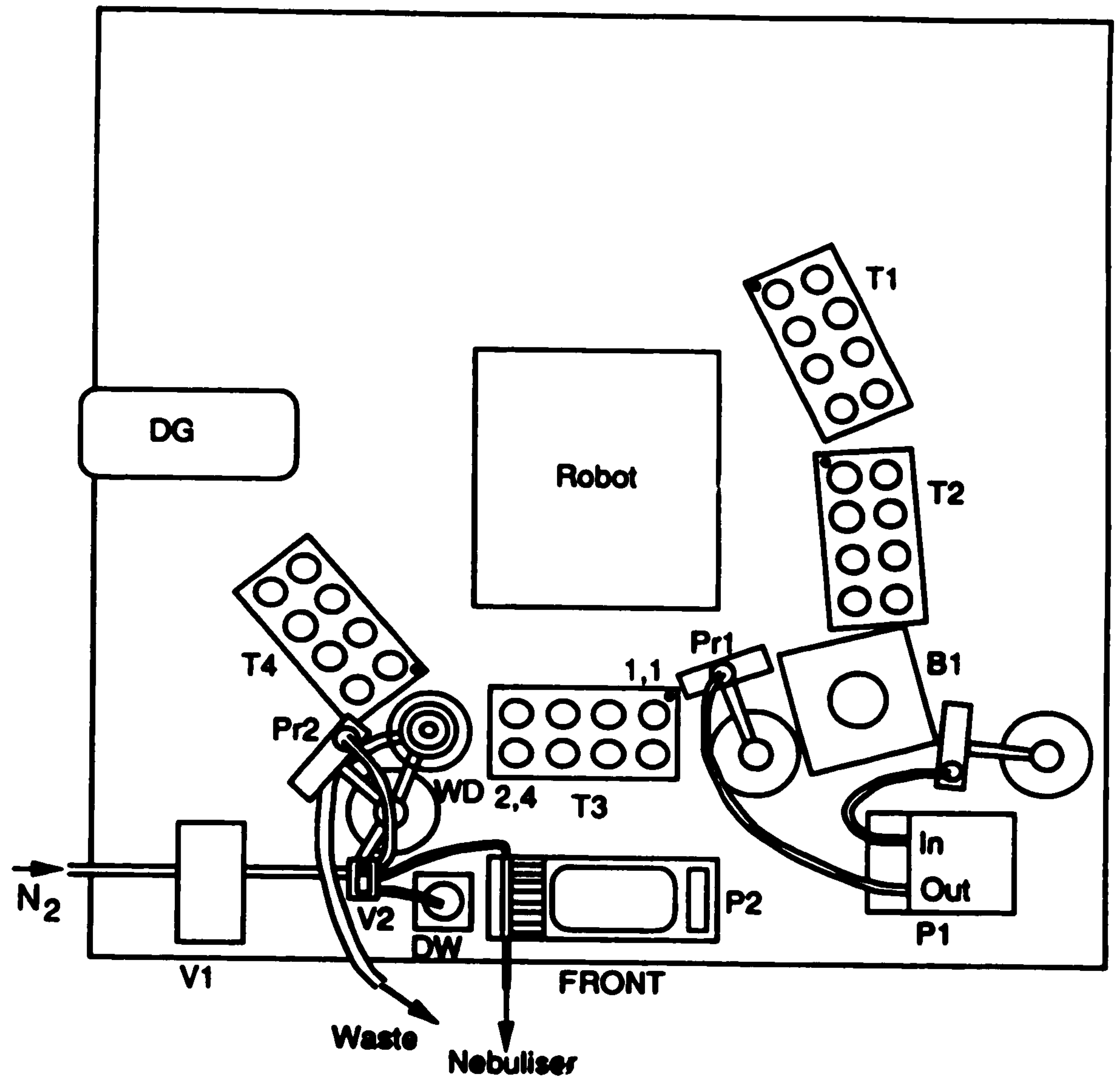


Figure 4 Schematic Diagram of Solution Preparation Station. P1 Syringe Pump, P2 Peristaltic Pump, B1 Top Loading Balance, V1 Solenoid Valve, V2 Pneumatically Operated 2-way Valve, Pr1 Syringe Pump Uptake Probe, Pr2 Nebuliser Uptake Probe, T1 Sample Tray, T2 Intermediate Solution Tray, T3 Standard Solution Tray, T4 Measurement Tray, DW Distilled Water Reservoir, WD Waste Disposal, DG Dirty Glassware Depository.

Figure 5 Photograph of robotic system.

were required. These included trays for storage of solutions, equipment for waste disposal, probes and storage racks for these probes. The last two devices are described in more detail with the syringe pump.

For storage of the stock solutions and for holding containers several trays were manufactured. These trays had eight positions arranged in two rows of four positions. The numbering of the position is shown in Figu.e 4 by T3. Tray T1 was used to hold empty containers in which the solutions will be made. Tray T2 was used to hold the containers in which the intermediate solutions of a serial dilution would be made. Stock solutions were stored in tray T3. The solutions to be measured by the ICP were held in tray T4.

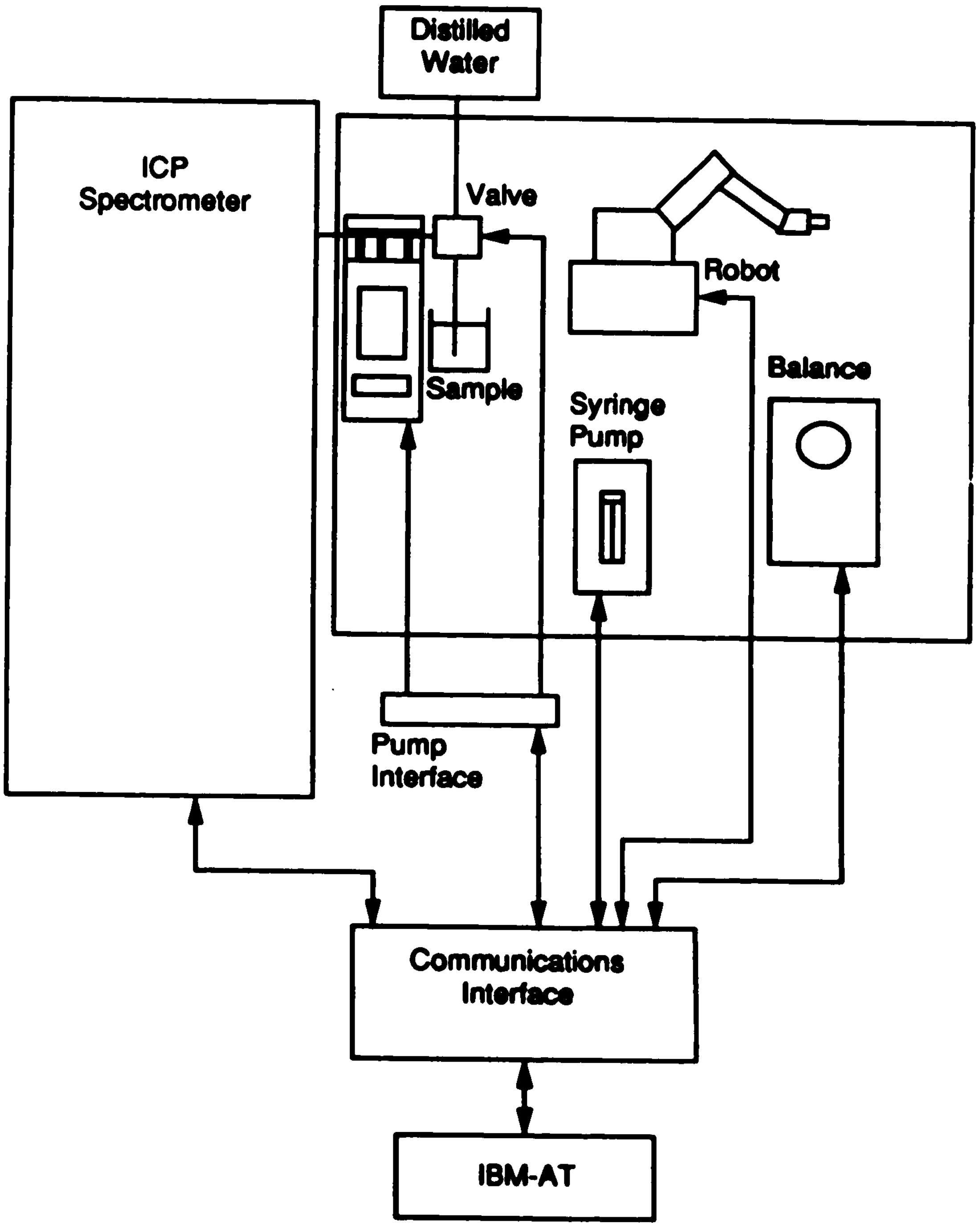


Figure 6 Communication network used in Solution Preparation Station

Once a solution had been measured the unused solution was emptied into a 100-ml glass funnel (WD in Figure 4) connected to the waste container to which the nebuliser drain was also connected. Once emptied, the bottle was deposited in a large, plastic container (DG). The dirty glassware was then washed by hand.

2.1 Computing Hardware.

The system used two computers, one for control of all the devices used in preparing solutions in addition to user interaction. The second was an integral part of the spectrometer and was used to collect spectral data as well as processing of that data.

The former was an IBM-AT with 640KBytes of RAM, 512KBytes of which are on the motherboard and the other 128KBytes on an AST Advantage board which can be further expanded to 3.5MBytes. The motherboard was also fitted with a 80287 numeric coprocessor.

The video display was with an IBM Enhanced Color Display connected to an Enhanced Graphics Adapter card on which was installed 64KBytes of video RAM which can be expanded to 256KBytes. The present configuration allows a maximum resolution of 640 x 200 pixels with 16 colours; with the expanded memory the resolution can be increased to 640 x 350 pixels.

Data storage was on an internal 20Mbyte hard disk and two internal 5 1/2" disk drives. Disk drive A was an high capacity 1.2MByte drive, drive B was a double-sided double-density 360KBytes drive which allowed transfer of data to other IBM computers in the laboratory and if necessary to the Macintosh computers via a serial link.

An IBM serial/parallel adapter was installed to provide for output to an IBM Graphic printer and an RS-232 serial port. An AST Advantage card provided a second serial port and an additional parallel port. A Microsoft Mouse Bus Version was also installed which could be used with software supporting graphical interfaces. The operating system used was IBM DOS Version 3.20.

The computer associated with the spectrometer was a Digital PDP11/03-L with 28K core memory. The CPU was based on the LSI-11 chip set. Data storage used a RX-02 dual 8" floppy disk drive unit. Normally the system would use a DECWriter teletype, however this terminal interface was modified to operate at 1200 baud instead of the normal 300 baud and was used to interface the spectrometer to the rest of the system. The operating system was provided by the spectrometer's manufacturer under the name ARLEB Version 1.2.

2.1.1 Control Software

The program "Automation Control System" (ACS) was designed as a programming language for the automation of experiments which could be carried out with the equipment. The program consists of low level device drivers which control individual pieces of equipment by sending the appropriate command messages and interpreting any messages that may be returned. Some driver routines maybe combined to provide functions not provided in the basic command set of the equipment. In addition to the driver routines, functions and procedures are provided to simplify program development.

The program was designed to provide an environment to support the interactive nature of developing robotic applications. This excluded the use of a compiled language in favour of a semi-compiled or interpreted language which could provide an interactive development environment and so remove the need to build the user interface. The programming language would then become an extension of the host language and become an integral part of it. The host language would then pr.)vide all the mechanisms for interpreting commands and for executing programs. The control structures of the host language would be an integral part of ACS as well.

New device drivers can be designed and tested easily in such an environment. They can be loaded in as source code and tested directly by passing the necessary parameters and obtaining immediate results, without the need to develop test programs which may introduce unwanted side effects which make interpretation of the results difficult.

The development system used was an implementation of Forth (HSFORTH, Harvard Softworks, Springboro, Ohio). Forth is a semi-compiled language and as such combines some of the speed advantages of a fully compiled language with the interactive nature of an interpreted language. Forth is a stack based language using reverse Polish notation. Commands in Forth normally expect parameters to be passed on the stack and leave the results on the stack [28]. Commands are stored as words in vocabularies. The vocabulary can be extended by the definition of new words, an example is given in Figure 7.

```
: GET.OTHER.NEXT ( 1 or 2 - 2 or 1 )
    DUP          ( Duplicate number on the top of the stack )
    2 MOD        ( Calculate remainder on division by two )
    2*           ( Multiply remainder by two )
    1-           ( Subtract one )
    +            ( Add to original number )
;
```

Figure 7 Example of a word definition in the Forth programming language.

The definition starts with a colon followed by the name of the word being defined. Comments are separated from the rest of the program by parentheses. It is normal practice to show the parameters expected on the stack and the results that will be left on the stack by using a comment statement immediately following the name of the word. In this example, if a 1 is on the stack on entering then a 2 will be left; similarly a 2 will be replaced by a 1. The definition is terminated by a semi-colon.

Constants are words which leave a specified value on the stack. Variables are words which leaves the address of a memory location on the stack; two operators then use the address to fetch a number from memory or to store a value on the stack in memory. Hence Forth is inherently a pointer based language.

In addition to defining new procedures or functions, Forth allows the definition of words which can be used to define words. This allows for the definition of more complex data structures than simple constants or variables. An example is shown in Figure 8. The definition of a definer consists of two parts. The first specifies the action to be taken when compiling

```
: MOVEMENT ( posN..pos1 N - )
( Store a sequence of positions to carry out an operation )
  CREATE                       ( New entry in dictionary )
    DUP ,                      ( Create copy on n and store )
    0
    DO                         ( Start loop )
      ,                        ( Store positions in definition )
    LOOP
  DOES> ( direction - )
( Carry out operation in direction specified )
    SWAP                       ( Address of movement on top of stack )
    BEGIN-CASE
      FRWRD  CASE-OF
             ( Execute moves in forward direction )
                DUP @
             ( Get number of positions in movement )
                0
                DO
                  2+ DUP @ MOVE.TO
                  ( Increment pointer, get posn. and move )
                LOOP
                DROP
              ELSE
      BCKWRD CASE-OF
             ( Execute moves in reverse direction )
                DUP @ SWAP 2+ OVER 2* + SWAP
             ( Put pointer to end )
                0
                DO
                  2- DUP @ MOVE.TO
                  ( Decrement pointer, get posn. and move )
                LOOP
                DROP
    END-CASE
;

ARL.PROBE.REMOVE
ARL.PROBE.LIFT
ARL.PROBE.CLOSED
ARL.PROBE.OPEN
PRE.ARL.PROBE
5 MOVEMENT ARL.PROBE
```

Figure 8 Example of the definition of a defining word in the Forth programming language.

a new word into the vocabulary, and is known as the compile-time action. This is the code between the name and DOES>. The second specifies what is to be done when a defined word is to be executed, the run-time action, this starts with DOES> and ends with a semi-colon. When a word defined in this way is executed the address of the base of the memory allocated to the word definition is left on the top of the stack and so the data stored in the definition can be accessed.

As an example of such a definer, the definition of a routine which stores a sequence of positions in memory and then executes that sequence in either the forward or reverse direction based on a single parameter is shown in Figure 8. Also shown is an example of a definition using this definer. The positions to be stored are placed on the stack in reverse order followed by the number of positions in the sequence. The definer stores the number of positions and the address of the positions in memory when ARL.PROBE is being defined. When the command FRWRD ARL.PROBE is executed, either after being typed from the keyboard or as part of the definition of a routine, the robot will pick up the probe from its stand by moving through the sequence positions from PRE.ARL.PROBE. Similarly BCKWRD ARL.PROBE will cause the robot to replace the probe in its stand by moving through the sequence in the reverse order.

A brief description of the commands used to control the various devices in the system will be given later in this chapter. Appendix A lists the main commands that would be used in program development. Complete documented listings of these routines are available in a supplementary volume. Once these low level commands were developed and the code was found to be stable, the software system was stored in a compiled form as ACS.EXE and formed the kernel for the development of the programs described in later chapters. The program can be executed from DOS by typing ACS. This compiled version of the software was designed so that it would first load a configuration file (ACS.SYS) in which the setting of system variables such as the port numbers of the devices, the loading of robotic positions to be used and the loading of programs was done. Any commands that would normally be typed at the beginning of a session can be included in the configuration file. In this way it is analogous to the AUTOEXEC.BAT file of PC-DOS.

The version of Forth used did not have a complete set of utility words suitable for the system that was being developed. In order to simplify development a number of utility words were defined. For the most part these were added to the existing vocabulary though there were some words redefined in order to make them more general.

The utility words fell into two main groups, those necessary to support serial communications and those for string handling. Some software tools were also defined which allowed for listing of words included in a vocabulary and for file manipulation.

The routines to support serial communications included the redefinition of the words which sent and received characters. As these only used the one serial port, they were modified so that both serial ports could be used. Routines were included to allow the serial ports to be initialised from within the program. These could also be used to change any of the communication parameters such as baud rate, word length, or parity.

Routines to send and receive complete lines of text were also defined. It was found that the routine to receive text could not execute fast enough to receive a line of text without losing some characters, so machine code routines were included to perform this task.

The string handling routines included words to find the position of particular characters or digits. These were then used in routines to strip all characters prior to the first occurrence of a particular character or up to the first occurrence of a digit. These were later used to interpret messages returned from various devices. A routine was included to convert a string to a number and leave the result on the stack.

Other utilities included routines to check the existence of a file on disk, the renaming of files and the making of back-up copies with a BAK extension. A routine to provide a timed delay was also written.

Extensive use of the 8087 co-processor was made. Forth does not normally support floating point calculations. A set of routines which used the co-processor were supplied with this implementation of Forth and were used. Some additional extensions were also necessary. These were mainly words which extended the manipulation of numbers on the floating point stack of the co-processor. The arc cos function was also written along with a routine to convert a string to a number on the floating point stack.

Floating point numbers use their own separate stack, which is located in the 8087. This stack is limited to a maximum depth of 8 numbers [27]. There was no stack overflow detection so great care was necessary in order to ensure that at no point could the floating point stack overflow. If this did happen then the data that was on the top of the stack would be corrupted and nonsense results were generally obtained. Stack overflow could be avoided by storing intermediate results in memory using the internal 80 bit format so that no loss in precision would result. However, this will result in a reduction in speed.

2.2 Communications System.

All communication between devices in the system was by means of RS-232 links. The IBM-AT BIOS is limited to supporting only 4 serial links. In order to avoid the non-standard patches to the BIOS necessary to support more RS-232 devices internally, an external multiplexing device was used. The device used was a Multiport Controller 528H supplied by Bay Technical Associates Inc. (Bay Saint Louis, Mississippi). The 528H multiport has 9 RS-232 ports, one for the host device and eight peripheral devices. In addition to providing the multiplexing, the unit also has a 256 byte buffer.

All ports are configured as DCE devices [29]. Each port can be individually configured for baud rate (110 - 9600 baud), word length (5 - 8 bits with 1, 1 1/2 or 2 stop bits), and parity (even, odd or none). The unit can also be programmed to recognise XON/XOFF characters for software handshaking, in addition to hardware handshaking through CTS and DTR signal lines.

The unit can be operated in one of six modes [29]:

1) Full duplex communication - allows the host full duplex communication with a selected port;

2) All messages from all ports - buffers messages from peripheral devices and transmits them to the host with an identification code when a terminating character has been received or when the buffer is full;

3) Single message from all ports - buffers messages as in 2) but only transmits them to the host when requested;
4) All messages from selected port - operates exactly as mode 2) but a single port is specified;
5) Single message from selected port - is a combination of modes 3) and 4) transmitting a buffered message from a selected port when requested;
6) Time Division Multiplexer - automatically multiplexes all peripherals by scanning all ports to check if any data is buffered, if so, it is transmitted to the host with an identification code until the buffer is empty or a user-specified block length is read.

The first mode was used exclusively, though the other modes could be used if extensive multitasking was necessary. The termination character for modes 2) - 4) is programmable though the default is CR.

The configuration of the ports is shown in Table 2. This can be altered using a terminal emulation program to issue the configuration command (Control-T C). The unit responds with a menu driven configuration routine. This routine allows changes to be made permanent, in which case they are stored in non-volatile memory. An alternative method of changing parameters which avoids going through the nested menus is also available. This non-verbose method allows only temporary changes to be made.

All commands to the multiplex unit are prefixed by Control-T. Selection of a port in mode 1) is by sending the message Control-T followed by the character 1 - 8. Once a port has been selected communication proceeds normally and the multiplex unit is transparent except as a buffer.

Port	Device	Baud Rate	Word Size	Stop Bit	Parity
1	Robot	4800	7	2	Even
2	Peristaltic Pump	1200	7	2	Even
3	ARL	1200	7	2	Even
4	Balance	1200	8	1	None
5	Syringe Pump	1200	8	1	None
6	Not Used	9600	8	1	None
7	Not Used	9600	8	1	None
8	Not Used	9600	8	1	None
9	IBM-AT	4800	7	2	None

Table 2 Configuration of Serial Port on Multiplexing Unit

In addition to the routines for sending and receiving information through the serial ports, routines for control of the routing of information by the multiplexing unit were written. The commands sent to the multiplex unit were comprised of a Control-T followed by the port number. A variable was defined for each device and was set to the port number of a device in the configuration file. The switching routine had the device variable passed to it as a parameter, the port number was read from the variable converted to a character then transmitted following a Control-T. The port number of the current device was stored in a variable. When switching devices, the previous port number was also stored in a separate variable. This allows the previous device to be reselected at the end of the transaction.

To configure the multiplexing unit a terminal emulation program is necessary. This is provided by running a terminal program as a child process. The command RUN.TERM will execute a program called "ZTerm.Exe". This allows direct communication with any device in the system and provides a method of fault diagnosis It can also be used for transferring files to and from the computer associated with the spectrometer.

Command	Stack activity	Description
INIT.PORT	b p s l prt -	Initialise port to given baud rate, parity, stop bits and word length.
RESTORE.PORT	prt -	Reinitialise serial port.
CKEY	- char	Get character from serial port.
CEMIT	char -	Send character to serial port.
CTYPE	addr$ n -	Send string to serial port.
CTEXT	delim -	Receive string using delim as terminator.
SELECT.DEVICE	device -	Switch serial output to device.
RESELECT.DEVICE	-	Select previous device.
GET.RETURN	addr$ com - addr$	Receive message from serial port and store in string.
WAIT.RETURN	addr$ com - addr$	Receive message without time out.
RUN.TERM	-	Run terminal emulation program and re-initialise ports after quitting terminal program.

Table 3 Summary of communication commands.

When transferring a program file to the spectrometer it was necessary to slow the transmission speed in order to allow time for the spectrometer to interpret the information. This was done by slowing the baud rate of the multiplexer/host connection to 300 baud. This was achieved by sending a short text file (Set300) to the multiplex unit containing the commands to change the baud rate using the non-verbose mode. Once the change in speed had been made the baud rate used by the terminal program must also be lowered. A similar text file (Set4800) contained the commands to reset the speed back to 4800 baud after the transfer was complete. A summary of the words associated with the control of communications is given in Table 3.

2.3 Robot.

The Robot Institute of America gave the following definition for a robot "A reprogrammable multi-functional manipulator designed to move material, parts, tools or specialized devices through variable programmed motions ..." [14]. The function of a robot is to move things. The design of the robot, however, imposes many constraints on the environment in which it works. The choice of the robot used will greatly influence the layout and even the choice of components used with it.

When choosing a robot there are several factors to be considered. These include;

Geometry - There are five general types of robots. They can be classified by the types of motion the joints can execute and the geometric arrangement of those joints. The two most common geometries used in laboratory robotics are revolute (e.g. Mitsubishi RM-501) and cylindrical (Zymate II). The revolute geometry consists of entirely rotational joints arranged in a body, shoulder, elbow and wrist configuration, (Figure 9a). The cylindrical geometry has a mixture of sliding and rotational joints, a central vertical slider which can rotate, and a hand attached to a horizontal sliding member, (Figure 9b). A third geometry which is becoming more available is the Selectively Compliant Articulated Robot Arm (SCARA, Perkin-Elmer AS-80 Robotic Autosampler). This can be thought of as a combination of the cylindrical and revolute geometries. It consists of a vertical slide as in the cylindrical robot but the horizontal ember consists of a two link arm similar to a horizontal revolute robot, (Figure 9c). The hand is often limited to vertical movements. This geometry of robot has less rigidity in the horizontal plane than the revolute geometry and so can adjust better to mis-alignments between components (compliance). Thus it would be well suited to function as an autosampler. Another geometry is the Cartesian which has sliding links

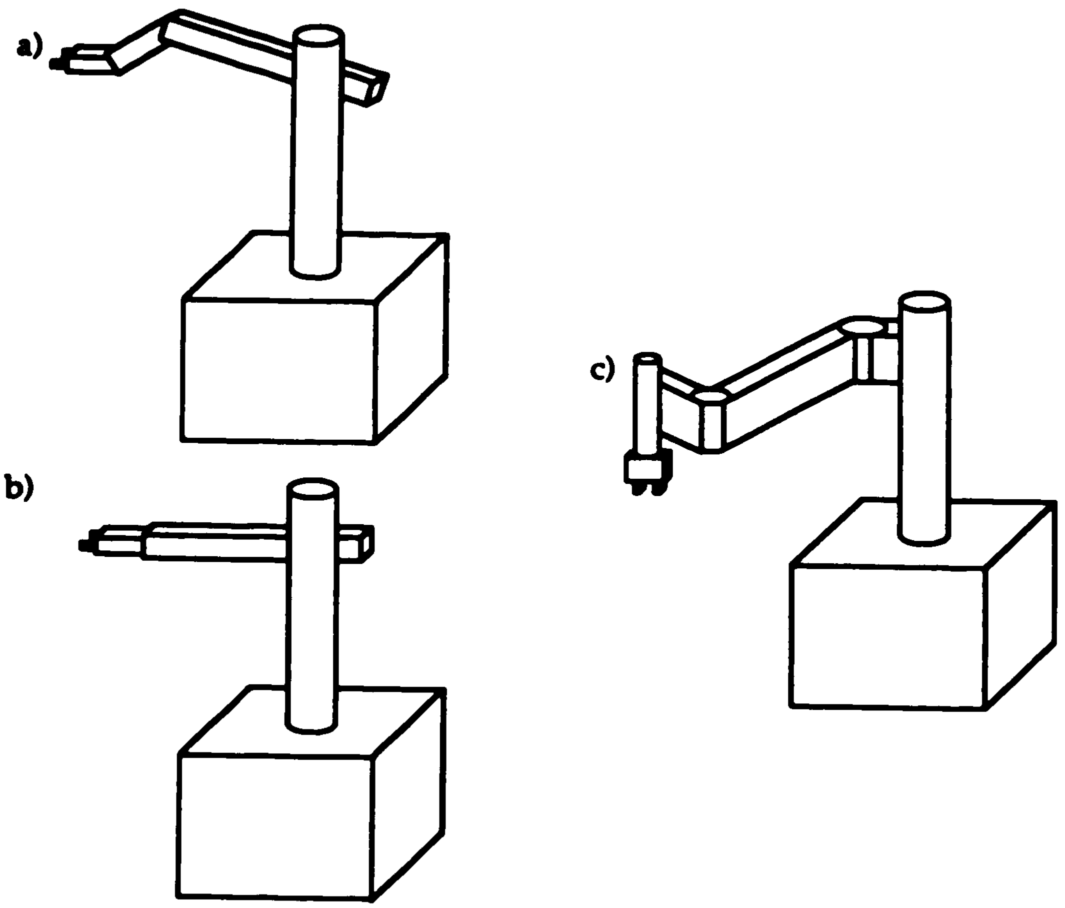


Figure 9	Common geometries for laboratory robots, a) Revolute, b) Cylindrical, c) SCARA

in the vertical and two horizontal directions, the traditional xyz-autosampler could be included in this class. Finally, the polar geometry has a sliding member on a base which can rotate horizontally and vertically.

Degrees of Freedom	-	The flexibility and manoeuvrability of the robot depends on the number of joints that can moved independently. For useful work the absolute minimum number of degrees of freedom would be 3. As the position and orientation an object can be described by the three

Cartesian coordinates for position and three angles for the orientation then the maximum number of degrees of freedom required are 6. Typically 4 (Zymate II) to 6 (Puma 260) degrees of freedom are available. The position of the gripper could, but is not normally, be regarded as an extra degree of freedom.

Reach - The reach of a robot is the distance between the centre of the base to end of the arm when the arm is fully extended. The reach combined with the geometry and number of degrees of freedom define the region of space, or the work envelope, that can be accessed by the robot. The shape and size of the work envelope will govern where objects can be placed with respect to the robot. The work envelope can be further extended by the mounting the robot on a track. For revolute robots mounted on the base the space immediately above the robot is difficult to use. This limitation can be overcome by mounting the robot in an inverted position and allowing the arm to work over the bench.

Repeatability - This is a measure of the ability of the robot to return to a position repeatedly. It will govern the precision required for positioning the objects with which the robot will work as well as the clearance required between objects in order to avoid collisions. The repeatability will be a function of the speed of motion, load and drive system used.

Speed - The speed of the robot will effect the throughput of the system. It is desirable to have the highest speed possible. However, the speed of the robot has to be a compromise between reducing the time required to perform a task and the ability to perform the task accurately. With increased speed the repeatability will suffer, also the forces exerted on the objects being handled could cause errors, such as spillage from open containers during acceleration or deceleration [18].

Drive system - This is the mechanical system used to move the joints of the robot. For laboratory robots electrical motors are the main source of motive force. For smaller robots stepping motors are sometimes used, while larger devices use DC servo motors. The feedback for these motors may be provided by either potentiometers (Zymate II) or by means of optical encoders (RM-501). The repeatability can be affected by the gearing system used to transmit the motive force from the motor to the joint. Older systems used conventional gear trains, but the repeatability of these systems was limited by the high degree of backlash in the gearing. Some newer laboratory robots (CRS Plus and Mitsubishi RVM1) uses harmonic drives which have less backlash than traditional gears [30].

Capacity - This is the weight that the robot is capable of carrying, including the weight of the hand. This has to be a compromise between being able to carry as large a load as would be desirable and the cost in terms of requiring a more powerful drive system, poorer repeatability and decreased speed.

Hands - These are devices which can be attached to the end of a robot arm and used to manipulate objects in the work envelope. The most common type of hand is some form of mechanical gripper. Typically this hand consists of two opposing fingers. Control of the hand can vary from a simple open/close to control of the distance between the fingers. It may be important to control the pressure that the fingers can exert on an object that is held in the fingers. The flexibility of the robot can be increased by providing some means of changing the type of hand that is being used, either by changing the complete hand unit (Zymate II) or the type of fingers (Perkin-Elmer Masterlab) attached.

Control unit — This is the device responsible for interpreting the commands issued by the user or host computer. The success of the robot will depend largely on the ease with which the robot can be integrated into the system being built. Important factors influencing the success of this integration are the means of communication between the robot and other components, the functions that are under the control of the control unit, those functions that have to be provided by the host computer and the ease of programming the control unit. For example, if a standard communication interface is available then the robot will be easily connected to the host computer. The ability to send commands using an XYZ coordinate system will greatly simplify the control software written for the host system. Similarly, features such as straight line interpolation and the definition of via points, that is points that the robot should pass through without stopping, make the control of the robot smoother (CRS Plus and Mitsubishi RVM1). At the other extreme it may be desirable for the host computer to able to control acceleration of the arm. This could allow overall higher speeds to be used by programming accelerations to avoid undue stress on any object being held. The method by which the robot is taught will be important. Teaching should not be by one method, such as by programming of positions from a terminal alone, rather there should also be a method of taking the arm to a desired position and programming that position into the control unit and/or host. The leading of the arm could be by means of a teaching pendant which allows

Supplier	Robot	Type	Reach/mm	Payload/kg	Degrees of Freedom
Zymark Ltd.	Zymate II	Cylindrical	660	1.4	4
Unimation Inc.	Puma 260 MK3	Revolute	406	0.9	6
Unimation Inc.	Puma 552 MK3	Revolute	1000	4.0	5
Unimation Inc.	Puma 562 MK3	Revolute	1000	4.0	6
Perkin-Elmer Ltd	RM-501	Revolute	660	1.2	5
Perkin-Elmer Ltd	AS-80 Autosampler	SCARA	130	–	3
Hudson Robotics Inc.	RVM-1	Revolute	483	1.2	5
Hudson Robotics Inc.	CRS Plus SRS-M1A	Revolute	559	2.0	5
Cybernetic Applications	Mentor	Revolute	420		
Cybernetic Applications	Naiad	Revolute	500	0.5	5
Cybernetic Applications	Serpent	SCARA	400	2.0	4
Cybernetic Applications	Neptune	Revolute	1120	2.5	5/6
Spectro Analytical Instruments	RM-501	Revolute	660	1.2	5
Peerless Systems Ltd.	Peerless	Cylindrical	500	2.0	6
Universal Machine Intelligence Ltd.	RTX	SCARA	685	4.0	6

Table 4 Summary of Laboratory Robots available.

manual control of each joint individually or by means of a simulator. A simulator is a small model of the robot with potentiometers at each joint which allow the simultaneous control of all joints (Cybernetic Applications). An alternative is to use the robotic arm directly by means of a lead-by-hand mode (CRS Plus).

There are several suppliers of laboratory robots. Reference 18 provides a table of most of these, however the list is not complete. A more complete list of suppliers, including those of Reference 18, is shown in Table 4.

A robot is the mechanical equivalent of the CPU in a computer system. The success of it depends not just on the performance abilities of the device but also on the ability to program the device efficiently. The design of the control language of the robot is probably as important as the mechanical design of the arm itself. The programming facilities of the control unit are the equivalent of the assembly language of a CPU. At present there is a trend towards providing a more complex command set which will allow the use of XYZ Cartesian coordinates, straight line interpolation and continuous path control. Whether this will induce a reaction towards a RISC-type (Reduced Instruction Set Computer) of philosophy remains to be seen. This would result in a smaller, more powerful command set for the robot, and possibly more control of the basic mechanical components.

The robot is a "manipulator designed to move material..." hence the main property to be controlled is the motion of the robot. The control of the motion of a robot can be viewed as an hierarchy as shown in Figure 10 [31].

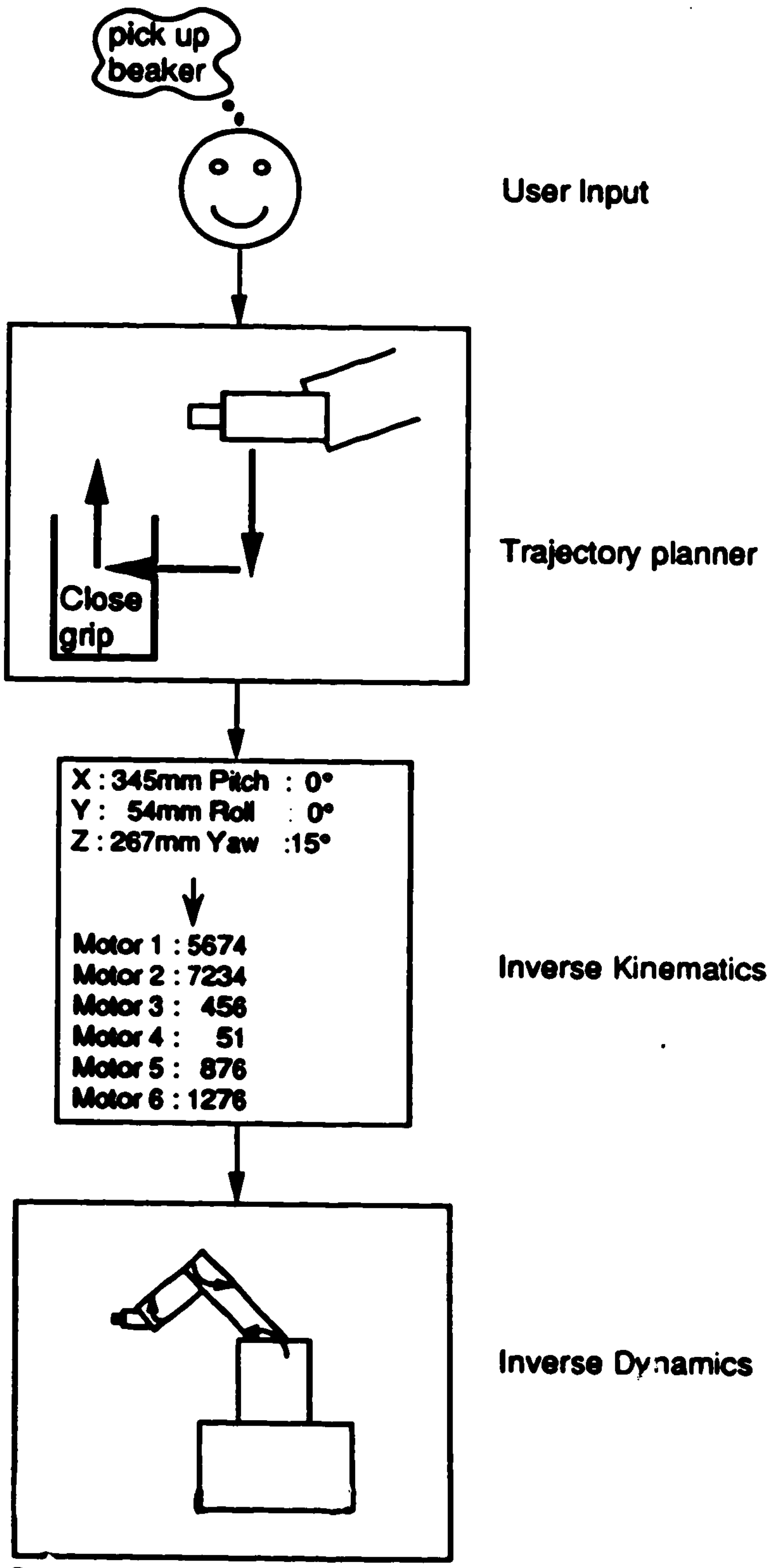


Figure 10 Steps required for motion control.

The dynamics of the robot, that is, the control of the torques on the joints and the accelerations of the links, are usually controlled by the control unit and are not programmable. Therefore, for most users of laboratory robots this is not a problem.

The kinematics of the robot, that is, conversion from the joint angles to positions and orientation in space and the reverse, are now being controlled by the control unit as with the CRS Plus and the Mitsubishi RVM1. However for older robots these conversions had to be made by the host computer. For the cylindrical geometry, the coordinate system of the robot is intuitive enough to the human operator that it can be used directly. However for the revolute and SCARA geometries the joint angles do not form an intuitive coordinate system for an human operator, so the conversion between XYZ coordinates and the robotic joint space is necessary. The equations for the conversions can be derived by a number of methods, the most general being the use of homogeneous transform matrices [30,32]. Alternatively for simple geometries the problem can often be reduced to a problem of plane geometry and solved by suitable trigonometric identities [33]. These conversions will be discussed in more detail with reference to the robot used later.

The ability to convert from joint space to Cartesian space and back also allows the possibility of running graphic simulations of the robot operation [34]. This will allow the robot to be programmed off-line, i.e. the program can be developed without the computer being connected to the robot and so allowing the robot to be used for other tasks [35]. The programs can be executed by a graphic simulation and the actions of the robot can be viewed before the program is executed on the real robot. The development of such simulation software for a small laboratory robot has been demonstrated by Tabani and Montaser [34].

Ideally the robot should be controlled by specifying the action to be taken, e.g. "pick up the beaker", rather than programming the motions to perform that action. The function of the trajectory planner is to interpret the command and plan the movements necessary to perform the task required. The plan would be a set of positions and orientations describing the path of the arm. This planning of the path is a non-trivial problem and attempts to solve it generally use artificial intelligence techniques [31]. A recent system [36] used several sub-systems, each capable of planning a different type of manoeuver, e.g. sliding, rotating, moving around. The overall problem being broken down into smaller sub-goals, each separate "expert" would then work on a sub-goal which was suited to it. For a fairly complete review of the role of artificial intelligence in robotics Reference 37 is recommended.

The high level programming of robots has been approached from two diametrically opposite directions [35]. One approach has been to solve the trajectory planning problem assuming that the implementation of the plan formed will be trivial, i.e. a top down approach. The other has started with designing the software to control the robot and building up from there. Eventually these two approaches should meet in the middle.

The language in most common use in chemical laboratories is Easylab®, supplied by Zymark for the Zymate robots. This programming language resembles more modern versions of Basic [2]. Perkin-Elmer supplies a language PERL (Perkin-Elmer Robotic Language) for use with their systems, it too is based on modern versions of Basic. CRS Plus supplies a language

RAPL for use with the SRS-M1A robot; further information is not available about this language.

There have only been two programming languages for laboratory robots published in the chemical literature [38,39]. The first was an attempt to bridge the gap between an AI system and a real robot [38]. This system was written in a dialect of LISP for an IBM PC-AT. The robot had a revolute geometry. Only the most basic commands of the robot's command set were implemented. This system represents only the beginning of a full programming environment for the development of robotic applications. The choice of LISP as the programming language should allow integration into AI systems but more higher level commands need to be defined.

A complete implementation of a robot control language is ARTS (Analytical Robot Telecommunications Software) [39]. This interpreter was written in C for IBM and IBM compatibles and ephasises the communication requirements of a robotics system. The system runs a Zymate I robot; the robot controller executes an Easylab program which is a simple command interpreter. ARTS was designed to execute either as a stand alone program or can be executed by other programs running under MS-DOS. This will allow it to be executed from within an expert system.

2.3.1 Robotic Hardware

Based on these considerations and the robot devices available, the robot chosen was a Mitsubishi Move Master II Model RM-501 (Mitsubishi Electric Corporation, Tokyo, Japan). It is a five axis articulated arm in a revolute geometry, the specifications of which are shown in Table 5a. The arm is

a)

Item	Specification
Waist Rotation Shoulder Rotation Elbow Rotation Wrist Pitch Wrist Roll	300° 130° 90° ±90° ±180°
Maximum Load	1.2 kg (including hand)
Maximum Speed	400 mm/sec
Position Reproducibility	±0.5 mm

b)

Item	Specification
Control System	Point to Point
Position detection	Optical encoder
Speed settings	0 - 9
No. of positions	maximum 629
No. of program steps	maximum 2048
Memory	ROM 32K RAM 32K
Interfaces	1 Parallel 1 Serial
External I/O lines External I/O handshaking	Input 8 bits Output 8 bits Input STB, ACK Output BSY, RDY

Table 5 Specifications of RM-501 a) Robot and b) Drive Unit

controlled by a separate drive unit, the specifications of which are shown in Table 5b [40].

The drive unit has 32KBytes of RAM in which can be stored positions of the arm and programs. The drive unit uses two letter mnemonic commands which can take a variable number of parameters. These commands are shown Table 6. Commands which can be used in programs are indicated in the final column.

Name	Command	Function	Program
Nest	NT	Reset robot	Yes
Home	HO	Set origin	Yes
Origin	OG	Move to origin	Yes
IMove	MI a1, a2, a3, a4, a5, a6	Immediate move	No
Move	MO a	Move to set position	Yes
Increment	IP	Move to next position	Yes
Decrement	DP	Move to prev. position	Yes
Set Position	PS a0, a1, a2, a3, a4, a5, a6	Store coordinates	Yes
Here	HE a	Store current coords.	Yes
Clear	PC a1, a2	Delete positions	No
Grip set	GP a1, a2, a3	Set closing parameters	Yes
Grip open	GO	Open the grip	Yes
Grip close	GC	Close the grip	Yes
Grip flag	GF a	Used in PS command	Yes
Speed	SP a	Speed arm speed	Yes
Time	TI a	Delay for a x 0.1 sec	Yes
Input	IN	Handshake read	Yes
Input data	ID	Read data	Yes
Output	OT a (&b)	Handshake output	Yes
Output data	OD a (&b)	Send data	Yes
Read position	PR a	Send coords to host	?
Read data	DR	Send input data to host	?
Test bit	TB a1, a2	If bit a1 set goto line a2	Yes
Error flag	EF a	Signal error on bit 7	Yes
If larger If equal If smaller If not equal	LG a1 (&b), a2 EQ a1 (&b), a2 SM a1 (&b), a2 NE a1 (&b), a2	Jump to line a2 if true	Yes
Gosub	GS a	Execute subroutine	Yes
Return	RT	Return subroutine	Yes
Repeat	RC a	Execute code a times	Yes
Next	NX	End of repeat loop	Yes
Goto	GT a	Unconditional jump	Yes
End	ED	Marks end of program	Yes
New	NW	Clear all data in RAM	No
Delete	DL a1, a2	Deletes program lines	No
Run	RN a	Begin execution from a	No
Write	WR a	Write RAM into ROM	No
Transfer	TR a	Read ROM into RAM	No
Reset	RS	Reset after error	No

Table 6 Summary of commands for RM-501 Robotic arm.

```
GP 7, 3, 5                    ; Set grip closing profile
GF 0                          ; Next positions have grip open
; Start defining some positions
PS 1, -4160, 1850, -3600, 750, 750, 0
PS 2, -4160, 950, -3600, 800, -800, 0
GF 1                          ; Next positions have grip closed
PS 3, -4160, 950, -3600, 800, -800, 0
PS 4, -4160, 1300, -3600, -1200, 800, 0
  .
; Start of program code
1 SP 9                        ; Set speed to maximum
5 GS 1100                     ; Call subroutine at line 1100
10 RC 4                       ; Repeat instructions four times
15 IP                         ; Move through next four positions
20 NX                         ; End of loop
25 GS 1200                    ; Call subroutine
30 GS 1300
35 SP 1                       ; Set speed to slowest value
40 MO 17                      ; Move to position 17
45 SP 7                       ; Set speed to intermediate value
50 RC 3
55 IP
60 NX
62 TI 15                      ; 1.5 second delay
  .
270 ED                        ; End of program

1100 MO 1                     ; Move to position 1
1110 RC 3
1120 IP
1130 NX
1140 RT                       ; Return from subroutine
  .
```

Figure 11 Partial listing of a program for the RM-501 Robot.

The drive unit can be programmed using the mnemonic commands indicated preceded by a line number. These programs are stored in the memory of the drive unit and can be executed by issuing a Run command followed by an optional starting line number as a parameter, otherwise the program would start execution from the line with the lowest line number.

An example of such a program is given in Figure 11. This is part of an early program that made tea. The comment statements would not be understood by the drive unit and if sent they would cause an error. They have been included for clarity.

For each joint, except for the wrist movements, the resolution was 0.025° per step. For the wrist joint the resolution was 0.075° per step [40]. The parameter ranges for the joints were therefore -12,000 - 0 for the waist, -5,200 - 0 for the shoulder, 0 - 3600 for the elbow. These values are for the origin being in the position of the robot after executing a NEST command. If the origin has been set to another position then the range of values these parameters may take will be altered accordingly.

Communication with the robot was by means of an RS-232 port which used a non-standard connector and an unusual handshaking protocol. The drive unit operated the handshaking lines on each character transfer and expected the host device to do likewise. The timing sequence expected is shown in Figure 12 [41]. The 528H multiport unit only operates the handshaking lines when the buffer is almost full hence it could not generate the signals required by the drive unit. This caused the system to hang when data was being read from the drive unit, as in a position read, though communication operated normally when commands were sent to the drive unit. In order to generate the signals expected a simple RC circuit was installed in the connector at the drive unit end, as shown in Figure 13.

The circuit is used to feed the RTS signal of the drive unit back to the DSR input. This allows the drive unit to send the next character. The time constant ($\tau = 0.01$ms) was selected so that the DSR input will be close to ground potential before the character transfer was complete even at 9600 baud rate. Once character transmission is complete, the drive unit senses that the

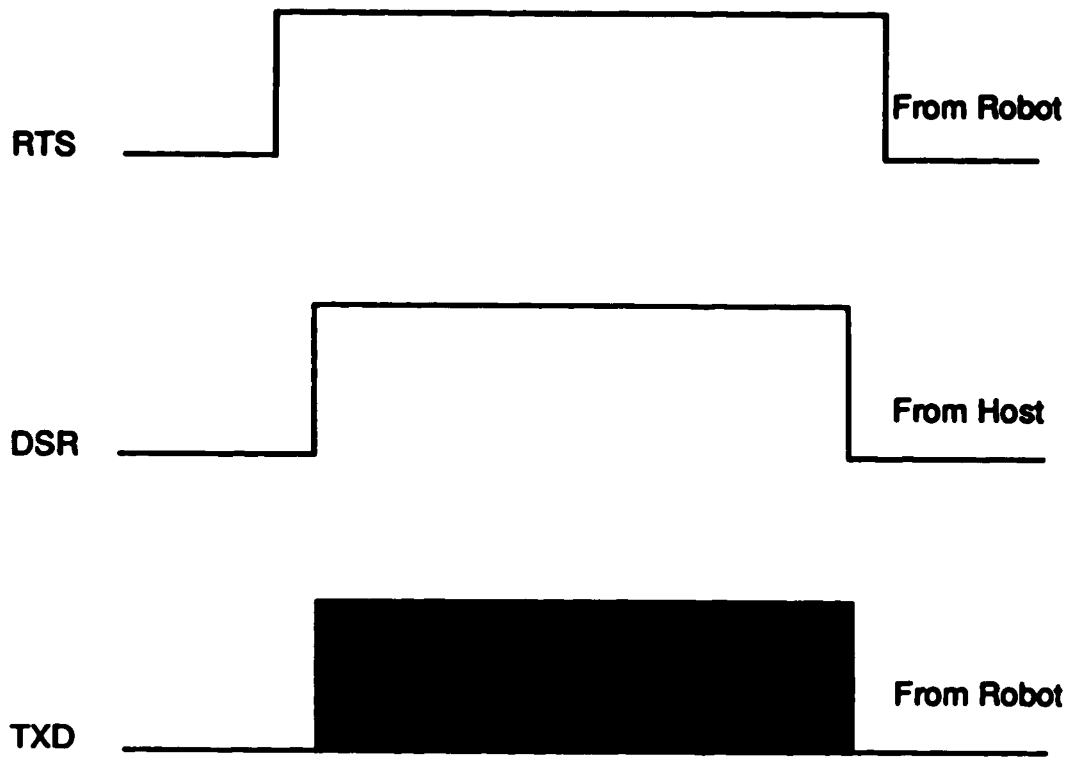


Figure 12 Timing sequence used by RM-501 Robot when transmitting a character on the RS-232 interface. Data is transmitted during the shaded portion of the TXD signal.

DSR input is low and then lowers the RTS output; the diode was included to protect the DSR input circuits when this happens by clamping the input to ground. The drive unit then raises the RTS output when the next character is ready for transmission. When this modified cable was used complete communication was possible using the multiplex unit in place. In earlier versions of the system the robot was controlled on a separate serial port and an assembly language routine was used in place of the BIOS serial driver (INT 14H) to provide the handshaking protocol expected. This was abandoned in

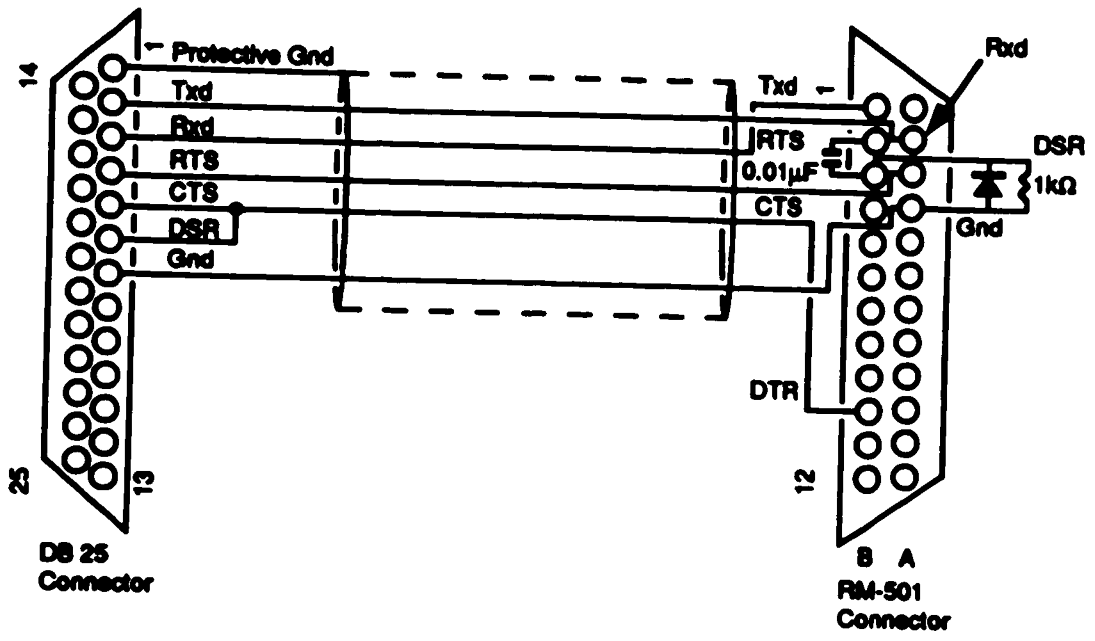


Figure 13 Diagram of RS-232 cable used for interfacing to a RM-501 Robot.

favour of a more standardised approach to the software provided by the hardware modification.

All the commands listed in Table 6 were implemented as low-level driver routines. A summary of the more important commands relating to the control and programing of the robot is given in Table 7. A complete list of robotic commands is given in Appendix A. In Figure 14 is shown some examples of the source code of low level drivers, one for a routine which sends parameters to the robot (PMOVE) and secondly how data from the robot is received (POS.READ).

Command	Stack activity	Description
END.PROGRAM	-	Stop programming of robot drive unit.
LABEL: name	-	Define label for current line number.
F.LABEL: name	n -	Define label n lines ahead of current line number.
NEST	-	Move to mechanical nest position.
GRIP.SET	c2 t1 c1 -	Set current curve for opening and closing hand.
GRIP.OPEN	-	Open hand.
GRIP.CLOSE	-	Close hand.
SPEED.SET	s -	Set speed of arm.
POS.READ	p -	Read coordinates of position p and store in buffer.
->START	- :: γ β z y x -	Transfer coordinates to variable START.
POSITION name	type mode - :: [γ β z y x] -	Define position using specified mode and of specified type, use coordinates if using MAKE mode, use current arm position if using IMMED mode.
TO.MOVE	a5 a4 a3 a2 a1 -	Move to position given by joint parameters supplied.
MOVE.TO	posn -	Move to defined position.
GO.THERE	- :: γ β z y x -	Move to coordinates given.
MOVEMENT name	posn ... pos1 n -	Define a sequence of positions.
RACK name	n - :: Δz Δy Δx -	Define a one dimensional array of positions starting a coordinates stored in START with offset between positions given on f.p.stack and with n positions.
TRAY name	m n - :: Δzm Δym Δxm Δzn Δyn Δxn	Define two dimensional array of positions mxn the 1,1 position being given in START with offsets between rows and columns given on f.p. stack.
UPDATE.CURRENT.POSITION	-	Update current position variable using data read from drive unit.
SHOW.CURRENT.POS	-	Display coordinates and joint parameters of current position.
SHOW.POSITION	posn -	Display coordinates and joint parameters of position.

Table 7a Summary of commands for controlling RM-501 Robot.

EDIT.POSITION	posn -	Modify joint parameters stored in position to be those of the current arm position.
GO.HOME	-	Move to home position.
MOVE.RELATIVE	- :: Δz Δy Δx -	Move relative to current position by amount given.
MOVE.UP	- :: Δ -	Move up by amount given.
MOVE.DOWN	- :: Δ -	Move down.
MOVE.FORWARD	- :: Δ -	Move forward.
MOVE.BACK	- :: Δ -	Move back.
MOVE.LEFT	- :: Δ -	Move left.
MOVE.RIGHT	- :: Δ -	Move right.
ROLL.BY	- :: Δγ -	Roll wrist through angle specified.
ROLL.TO	- :: γ -	Roll wrist to angle specified.
PITCH.BY	- :: Δβ -	Change pitch of hand by amount given.
PITCH.TO	- :: β -	Move hand to pitch angle given.
PFOPEN [filename]	-	Open file if no name specified then one will be asked for.
PLIST	-	List position defined in current vocabulary.
PFSAVE posnname	-	Save position in file previously opened.
PFSAVE.ALL [filename]	-	Save all positions in current vocabulary in file named.
PFPRINT [filename]	-	Print contents of a position file.
PFBROWSE [filename]	-	Display contents of position file on screen.
PFLOAD [filename]	-	Load contents of position file.
INIT.ROBOT	-	Initialise robot and move to home position.
GOODNIGHT	-	Put robot to bed and leave program.

Table 7b Summary of commands for controlling RM-501 Robot.

```
: PMOVE ( Pos# - )
( Move to position specified )
   CHECK.PROGRAM               ( Check Program Flag )
   $" MO " COUNT CTYPE         ( Send command )
   S->D <# #S #> CTYPE         ( Convert parameter to string )
   13 CEMIT                    ( Send CR )
   10 CEMIT                    ( Send LF )
;

: PREAD ( a - )
( Request position coordinates )
   $" PR " COUNT CTYPE
   S->D <# #S #> CTYPE
   13 CEMIT
   10 CEMIT
;

: POS.READ ( a - )
( Read position from D/U memory )
   PREAD                       ( Send command )
   LINE.BUFF COMPORT @ WAIT.RETURN
( Get position return and store in buffer )
   DROP
;
```

Figure 14 Example of low level robotic command.

2.3.2 Programming of the Drive Unit

Those commands that could be used in program mode first checked the value of a program flag to decide if the command was being sent as part of a program or for immediate execution. If the program flag was true then a variable containing the current line number was read, the line number was sent as the first part of the command line, and then the line number variable was incremented. The definer PROGRAM would store the current line number in a variable with the name specified as well as set the program flag to true. The word defined can then be used to execute a program stored in the drive unit. The command END.PROGRAM resets the program flag to false, hence any robot commands between PROGRAM and END.PROGRAM will be compiled into the drive unit memory and in this way a simple one-pass compiler has been constructed.

The word LABEL behaves similarly and allows the defining of labels in a program which are used with the various branching commands. Forward referencing of labels is not supported directly. The word FLABEL takes a single offset parameter which is added to the current line number in order to generate the line number of the destination. This can then be used in a forward branch. When this label is defined the number of programmable commands between the current line and the destination must be known. This is a major limitation to the utility of this simple compiler. Since all these programs can be defined as normal Forth words using the richer variety of control structures and being free of the memory size limitations of the drive unit there has been no need to use this compiler in this work.

2.3.2 Definition of Positions

The definition of a named position is also supported. A position may exist solely in the memory of the host computer and not in the drive unit. Such positions are regarded as virtual positions. Those which exist in the memories of both the drive unit and the host computer are regarded as real. The advantage of virtual positions is that the number of positions is limited only by the memory restrictions of the host computer. However, the disadvantage is that all the joint parameters have to be stored and the relative motion of each joint has to be sent as the parameters of an IMOVE command. In the case of a real position only the position number needs to be sent as the parameter of a MOVE command. Since both virtual and real positions are supported it is necessary to store the joint parameters of a real position as well so that a record of the current position of the robot can be maintained.

In addition to the two types of positions, positions can be defined in two ways. The current position of the robot can be defined as the position, or the parameters of the position can be defined and stored. All positions contain information on the type of position, the joint parameters in terms of the number of steps from the origin and whether the hand is open or closed.

Since the coordinates of a position specified in terms of robotic joint parameters are not easily comprehended by a human user, the definition of a position was specified in Cartesian coordinates with respect to the origin and the orientation of the hand was specified by two Euler angles corresponding to pitch and roll. Similarly, in the routine which displayed the coordinates of a position, both the joint parameters and the Cartesian coordinates with Euler angles were displayed. Hence if the current arm position is to be defined as a new position then the command IMMED VIRTUAL/REAL POSITION *name* will create a virtual or real position with the name "*name*" at the current arm position the joint parameters are obtained from the drive unit itself. If MAKE VIRTUAL/REAL POSITION *name* is used the Cartesian coordinates and Euler angles of the position must be present on the floating point stack. These are then converted to joint parameters and used to define a virtual or real position with the name "*name*".

Movement of the arm could be to any position defined, to any arbitrary position specified by either joint parameters or in Cartesian coordinates. Also movements relative to the current position could be made. These included movements along any of the three principle axes of the wrist, i.e up/down, forward/back or left/right.

2.3.4 Coordinate Transforms

In order to allow control of the robot using changes in the position specified in terms of Cartesian coordinates, or to report on a position as its Cartesian coordinates, it is necessary to be able to convert from the motor positions to this coordinate system and back. These conversions are achieved by a set of kinematic equations [30,32]. Full details of their derivation for any robot can be found in several robotic textbooks [30,32].

The first step in converting the motor positions to a Cartesian position and orientation is to calculate the joint angles. For the first three joints the angles are given by the motor position divided by the number of steps per degree. The motions of the wrist are achieved by two motors which control the pitch and roll in a concerted manner. The angles of pitch and roll are calculated from the difference and sum of the motor positions respectively.

The forward kinematic equations are used to calculate the position and orientation of the wrist from the joint angles. These can be derived by use of plane geometry. However, a more general derivation is possible by using homogeneous transform matrices [30,32,34,42]. The inverse kinematic equations will be derived using plane geometry.

Homogeneous transform matrices are 4x4 matrices which contain information to allow the transformation of positions in one coordinate frame to another from the rotations and translations of the frames [42]. The transform matrices can be subdivided into four sub-matrices, a 3x3 rotation

$$ {}^{i}_{j}T = \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ \hline p_1 & p_2 & p_3 & s \end{array}\right] $$

$$ p = \left[\begin{array}{c} p_1 \\ p_2 \\ p_3 \\ \hline s \end{array}\right] $$

Figure 15 Composition of an homogeneous transform matrix **T** and position vector **p**. The elements r_{ij} form the rotation sub-matrix, t_i are the translational vector, p_i are the perspective vector and s is the scale factor. The elements p_i of **p** are the cartesian coordinates of a position and s is the scale factor.

matrix, a 3x1 translation vector, 1x3 perspective vector, and a 1x1 scaling factor as shown in Figure 15. A position vector is a 4x1 vector and can also be subdivided into two sub-vectors, a 3x1 position and a scaling factor. In robotics the scaling factors are set to unity and the perspective vector is set to (0,0,0). To calculate the overall transform matrix for a series of rotation and translations the product of the transform matrices for each individual operation is taken [30,32].

To derive the kinematic equations for a robot a coordinate frame is defined for each joint [30,32]. Associated with each link between the joints of the robot are the four Denavit-Hartenberg (D-H) link parameters, α, a, d, and θ, as shown in Figure 16. These correspond to the twist angle (α) around the x-axis of the first frame between the frames on neighbouring joints, the length of the link along the x-axis (a), the displacement of two frames along

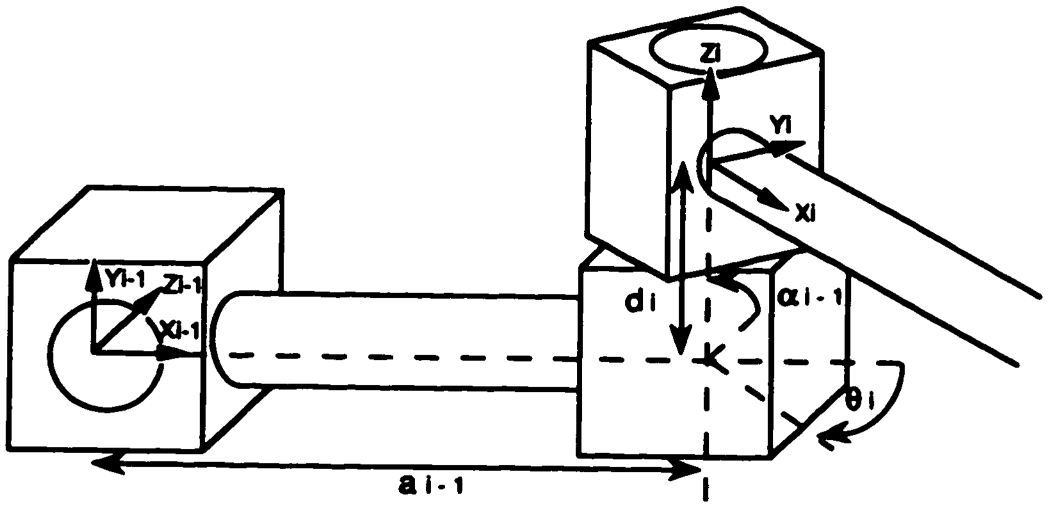


Figure 16 Definition of Denavit-Hartenberg link parameters.

the z-axis of the second frame (d) and the angle between two links around the z-axis (θ) [30,32]. The assignment of the coordinate frames for a RM-501 robot are shown in Figure 17 and the link parameters are shown in Table 8 and the lengths of the links of an RM-501 robot are; from base to shoulder (r_{BS}) 290mm, from shoulder to elbow (r_{SE}) 220mm, from elbow to wrist (r_{EW}) 160 and from wrist to tip of the fingers (r_{WT}) 214.6mm [40].

Equation 1 can be used to derive the transform matrix for a joint from a line in a D-H table [32]*.

$$ {}^{i-1}_{\ \ i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1) $$

*The following abbreviations for trigonometric functions will be used in order to simplify expressions:

$c_i = c\theta_i = \cos(\theta_i)$, $s_i = s\theta_i = \sin(\theta_i)$,

$c_{ij} = \cos(\theta_i + \theta_j)$, and $s_{ij} = \sin(\theta_i + \theta_j)$.

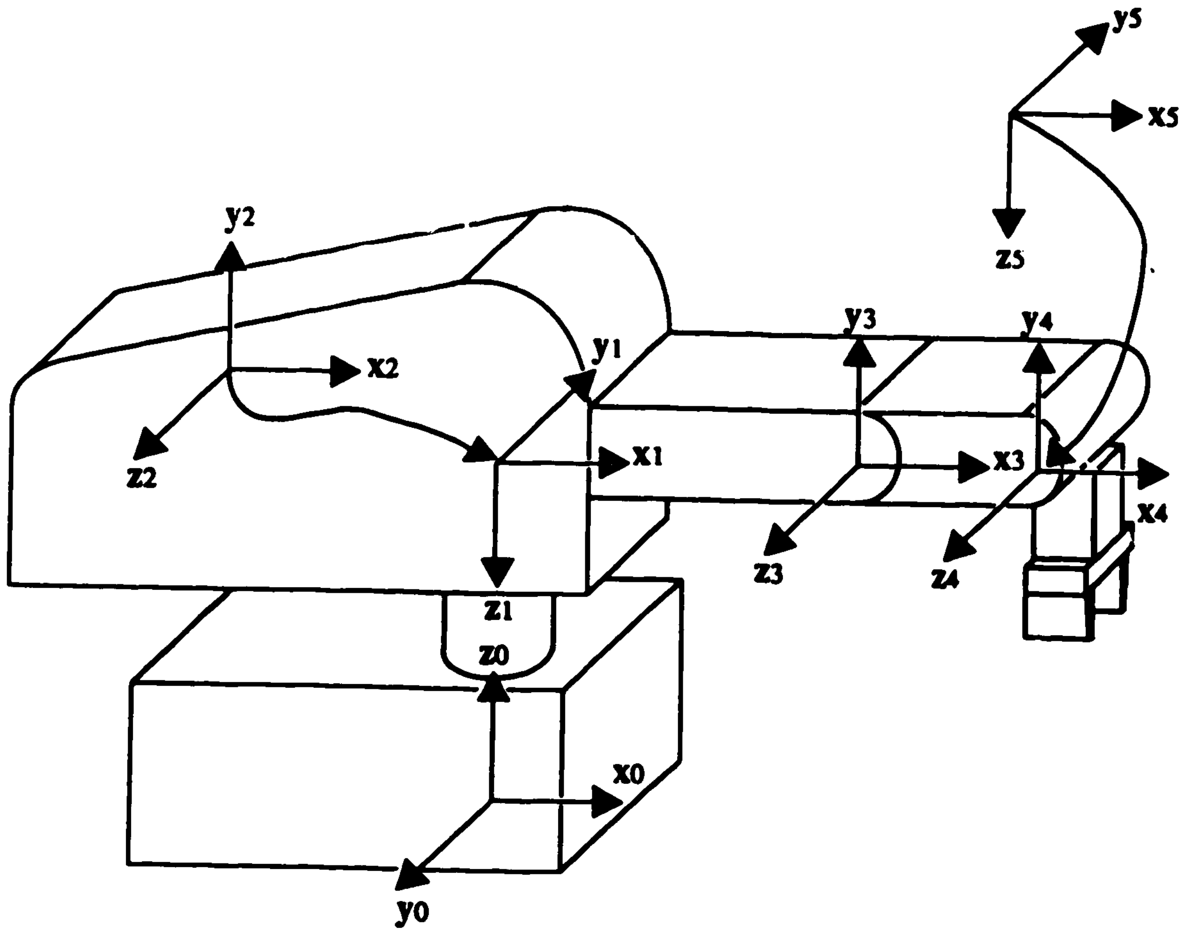


Figure 17 Definition of coordinate frames used with RM-501 Robot.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	180	0	$-r_{BS}$	θ_1
2	-90	0	0	θ_2
3	0	r_{SE}	0	θ_3
4	0	r_{EW}	0	θ_4
5	90	0	0	θ_5

Table 8 Table of Denavit-Hartenberg Link Parameters for a RM-501 Robot.

Applying the above to Table 8 the transform matrices in Equations 2 are obtained.

$$
{}^{0}_{1}T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ -s_1 & -c_1 & 0 & 0 \\ 0 & 0 & -1 & r_{BS} \\ 0 & 0 & 0 & 1 \end{bmatrix}
\quad
{}^{3}_{4}T = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ 0 & 0 & 1 & r_{EW} \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{1}_{2}T = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\quad
{}^{4}_{5}T = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\qquad (2)
$$

$$
{}^{2}_{3}T = \begin{bmatrix} c_3 & -s_3 & 0 & r_{SE} \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

The transform relating the base of the robot to the wrist is found by application of each transform in sequential order and results in Equation 3.

$$
{}^{0}_{5}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T \qquad (3)
$$

Multiplying in the order:

$$ {}^{1}_{4}T = {}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T $$
$$ {}^{0}_{5}T = {}^{0}_{1}T\,{}^{1}_{4}T\,{}^{4}_{5}T $$

results in Equations 4 and 5[*]:

$$ {}^{1}_{4}T = \begin{bmatrix} c_{234} & -s_{234} & 0 & r_{SE}c_2 + r_{EW}c_{23} \\ 0 & 0 & 1 & 0 \\ -s_{234} & -c_{234} & 0 & -(r_{SE}s_2 + r_{EW}s_{23}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4} $$

$$ {}^{0}_{5}T = \begin{bmatrix} c_1c_{234}c_5 - s_1s_5 & -(c_1c_{234}s_5 - s_1c_5) & c_1s_{234} & c_1(r_{SE}c_2 + r_{EW}c_{23}) \\ -(s_1c_{234}c_5 - c_1s_5) & s_1c_{234}s_5 - c_1c_5 & -s_1s_{234} & -s_1(r_{SE}c_2 + r_{EW}c_{23}) \\ s_{234}c_5 & -s_{234}s_5 & -c_{234} & r_{BS} + r_{SE}s_2 + r_{EW}s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5} $$

From translational vector part of ${}^{0}_{5}T$ (right hand column) above the coordinates of the wrist are given by Equations 6:

$$ \begin{aligned} x_w &= c_1(r_{SE}c_2 + r_{EW}c_{23}) \\ y_w &= -s_1(r_{SE}c_2 + r_{EW}c_{23}) \\ z_w &= r_{BS} + r_{SE}s_2 + r_{EW}s_{23} \end{aligned} \tag{6} $$

Z-Y-Z Euler angles (α,β,γ) describe the orientation of one frame with respect to another by describing three rotations which for the RM-501 robot correspond to the rotation about the waist (α), followed by a rotation of the wrist (β) either up or down and finally a rotation around the centre axis of the wrist (γ). The Euler angles [32,41,42] can be derived from a rotation matrix **R** by Equations 7[†]:

[*] The sum of angle formulae;

$$ \cos(\theta_i + \theta_j) = \cos\theta_i\cos\theta_j - \sin\theta_i\sin\theta_j $$
$$ \sin(\theta_i + \theta_j) = \cos\theta_i\sin\theta_j + \sin\theta_i\cos\theta_j $$

were used to simplify expressions involving rotations about parallel axes.

[†] Atan2(y,x) is the four quadrant arctan function and returns an angle in the same quadrant as the point (x,y)

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{23} & r_{33} \end{bmatrix}$$

then;

$$\begin{aligned} \alpha &= \operatorname{atan2}(r_{23}, r_{13}) \\ \beta &= \operatorname{atan2}(\sqrt{r_{23}^2 + r_{32}^2}, r_{33}) \\ \gamma &= \operatorname{atan2}(r_{32}, -r_{31}) \end{aligned} \qquad (7)$$

From the rotation sub-matrix of ${}^{0}_{5}T$ the Euler angles for the robot are given by Equations 8;

$$\begin{aligned} \alpha_w &= -\theta_1 \\ \beta_w &= -(\theta_2 + \theta_3 + \theta_4) \\ \gamma_w &= \theta_5 \end{aligned} \qquad (8)$$

The Cartesian coordinates for the tip of the standard hand are calculated using Equations 9;

$$\begin{aligned} x_T &= x_W + r_{WT} \cos\beta_W \cos\alpha_W \\ y_T &= y_W + r_{WT} \cos\beta_W \sin\alpha_W \\ z_T &= z_W + r_{WT} \sin\beta_W \end{aligned} \qquad (9)$$

where r_{WT} is the length of the standard hand.

Equations 8 and 9 are used for the display of the hand position and its orientation. They are also used in the first step of calculating how to make the relative movements described. The second step is to calculate the coordinates of the desired position and then back calculate the motor positions for that position.

This inverse calculation starts by calculating the Euler angle α using the relation;

$\alpha = \text{atan2}(y_T, x_T)$

The coordinates of the wrist are then calculated from the hand coordinates and orientation using Equations 10;

$$\begin{aligned} x_W &= x_T - r_{WT}\cos\beta\cos\alpha \\ y_W &= y_T - r_{WT}\cos\beta\sin\alpha \\ z_W &= z_T - r_{WT}\sin\beta \end{aligned} \qquad (10)$$

Since all the joints lie in a plane, the remaining angles can be solved for by using plane geometry [32,33]. The skeleton of the robot is shown in Figure 18. A vector of **h** exists between the wrist and the shoulder of the robot. This forms the third side of a triangle formed by the shoulder, elbow and wrist. The three angles can be solved for by successive applications of the cosine rule [33]. The resulting equations for the joint angles, in degrees, are given in Equations 11;

$$\begin{aligned} h^2 &= (z_W^2 - r_{BS})^2 + y_W^2 + x_W^2 \\ \theta_1 &= -\alpha \\ \theta_2 &= \text{atan}((z_W - r_{BS}) \,/\, \sqrt{(x_W^2 + y_W^2)}\,) + \text{acos}((r_{SE}^2 + h^2 - r_{EW}^2) / 2hr_{SE}) \\ \theta_3 &= \text{acos}((r_{SE}^2 + r_{EW}^2 - h^2) / 2r_{SE}r_{EW}) \\ \theta_4 &= \text{acos}((z_T - z_W) / r_{WT}) - (\theta_2 + \theta_3) - 90 \\ \theta_5 &= \gamma \end{aligned} \qquad (11)$$

This set of equations are then used to calculate the joint angles from a given orientation. The angles are then converted to joint parameters by dividing by the degrees/step. The wrist parameters are solved by the use of simultaneous equations.

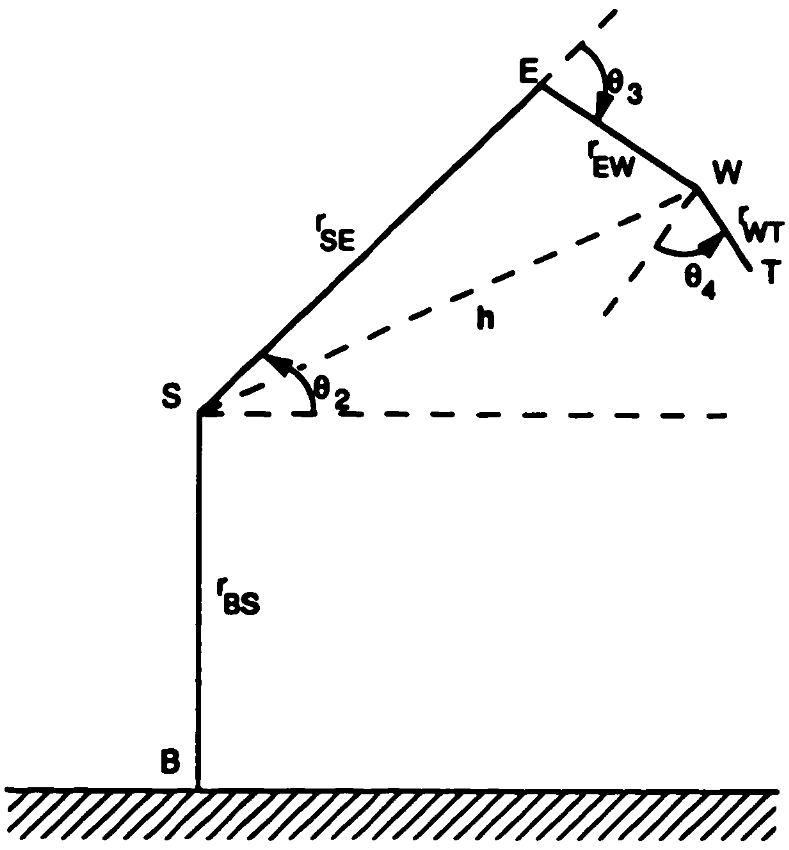


Figure 18 Skeleton diagram of RM-501 Robot used to derive inverse kinematic relations.

2.3.5 Definition of Racks, Trays and Movements

In addition to working with single positions the system has to be able to work with groups of related positions as would be found with a rack or a tray. Hence a one or two dimensional array of positions can be defined. This is done by specifying the Cartesian coordinates of one corner of the tray stored in the variable START and the displacement vectors between rows and columns along with the number of positions in each row and column. From this information the coordinates of the other positions are calculated, converted to joint parameters and stored. When a position is accessed the indices of the position along with the name of the tray are used. From the indices the corresponding position is calculated and the address of that position is left on the stack to be used like any other position. A rack is a one dimensional tray and only one displacement vector and the number of positions in the rack are needed in order to define the rack.

A sequence of positions can be stored, the sequence can be executed in either the forward or reverse direction, based on a single parameter passed at run time using the definer MOVEMENT. This is useful for performing related tasks where the only difference is the direction of the movements. For example, the removing and replacing of a probe in a stand. These movements are also used for storing the paths between objects in the system as the forward and reverse path are stored simultaneously.

2.3.6 Maintenance and Storage of Positions

There are a set of commands associated with the editing of positions, storing them on disk and the maintaining of the positions files. The editing of positions is accomplished by moving the arm to the new position and issuing the EDIT.POSITION command. The position to be edited is passed as a parameter on the stack. The current position of the robot is read from the drive unit, the coordinates are displayed and the user asked if these new coordinates are to be stored prior to the final change being made. This change must also be made to the position file. The position file is a TAB delimited text file and so can be easily modified with any text editor. Borland

International's Sidekick proved to be most useful for this as it was memory resident and the coordinates displayed could be easily typed into the file.

Position files stored on disk could be printed or viewed on screen. The information was displayed in a standard format of showing the robotic joint parameters as well as the Cartesian coordinates. The browser allows the contents of a file to be checked prior to loading. A routine similar to the vocabulary listing routine was defined, which allows listing of the positions already defined, their type, i.e. virtual or real, whether they are defined as a rack or tray and, if so, the index ranges. When positions are saved on disk, in addition to the information stored in the memory, i.e. position name, type, joint parameters and hand open or closed, a code indicating how the position was defined is included. This code is necessary for the loading of the positions back into memory, as the routine which loads a position has to recompile the information and during the recompilation it needs to relink the position definition to the run-time action of the original definer used so that the position will behave exactly as before.

2.4 Syringe Pump

The syringe pump used was supplied by Scientific Manufacturing Industries Inc. (Emeryville California). The model used was a Unipump 300 fitted with a 1-ml syringe (a 5-ml syringe was also available). The pump was supplied with a hand held probe which was suitable for manual use. A summary of specifications is given in Table 9. [43]

Item	Specification
Liquid Transfer Functions	Dilute Dispense Transfer/wash Titrate Serial Dispense
Auxilliary Functions	Prime/purge Save reagent Progr 'm Air gap Volume Program speed Change syringe Diagnostics
Display	16 digit fluorescent in 4 fields
Volume range	1µl to 10ml entered in µl
Syringe sizes	100, 250 and 500 µl 1, 2.5 5 and 10 ml
Tubing sizes	22, 18, 13 and 12 gauge
Probe Handle	Aspirate/Dispense program Allows single-hand operation

Table 9 Specifications of Syringe Pump

The hand held probe supplied was unsuitable for use with the robot. A probe of simpler design better suited to use with the robot hand was made. The design is shown in Figure 19. The body was constructed from polypropylene, the teflon dispensing tube being held in place at the top and bottom by epoxy resin. The bottom part of the probe was coated with teflon to minimise the amount of liquid adhering to the external surfaces. A collar manufactured from aluminium at the top of the probe gave additional stability to the probe when it was stored in a stand as well as preventing the polypropylene body from being damaged by the pressure exerted by the robot hand.

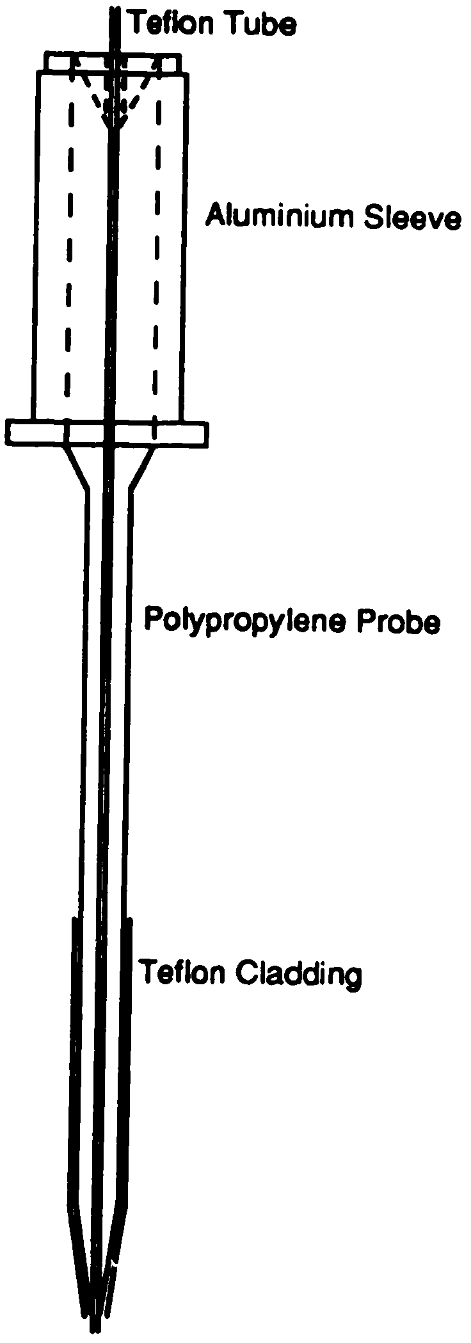


Figure 19 Design of solution uptake probe for use with a robotic hand.

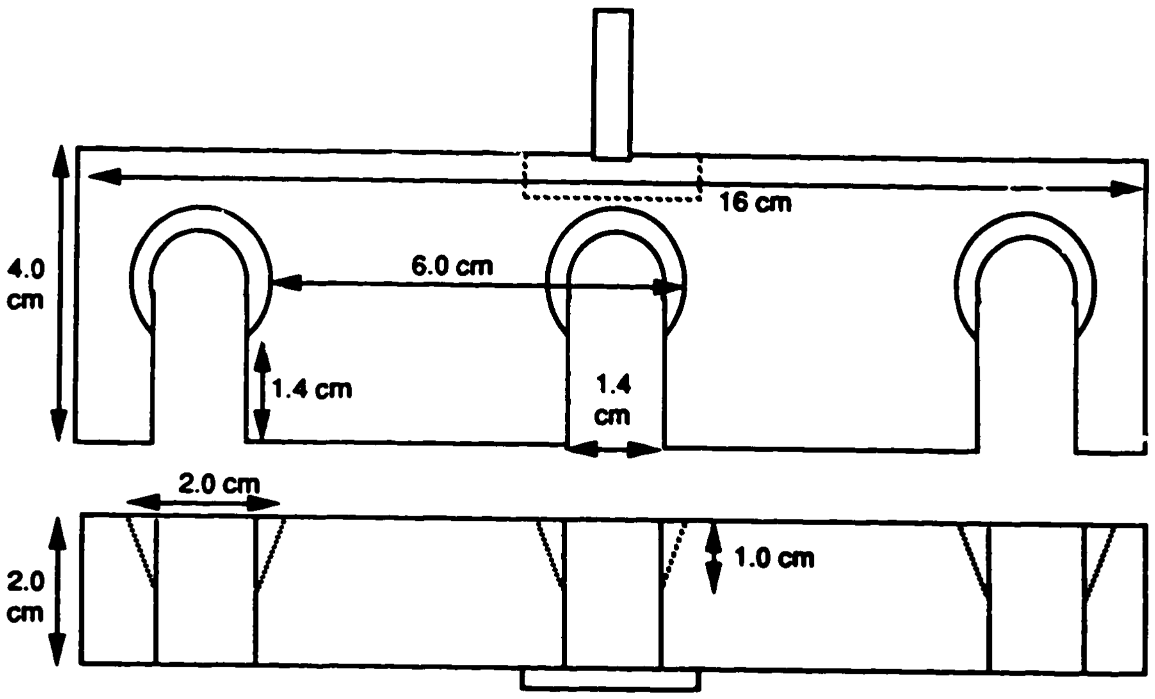


Figure 20 Design of stand holder for solution uptake probes.

Simple stands were designed for storing robot probes. The design is shown in Figure 20. Each stand can accommodate up to three probes. Probes can be easily removed from the stand by the robot.

The syringe pump unit included an RS-232 serial interface as a standard feature. This interface supported a variety of baud rates (150 - 9600) which were switch selectable. A baud rate of 1200 was chosen as a compromise between speed and the possibility of losing characters.

Commands were sent to the syringe pump in a formatted message. The message format was:

[ttffccdata] xx CR LF

where:

[	– start of message
tt	– two character code for pump, normally "PD"
ff	– two character code to identify sender
cc	– two command code
data	– variable length data string
]	- end of message
xx	– hexadecimal checksum, if no checksum used send "**"

The two character command codes are listed in Table 10. The parameter ranges for selecting a unit; 1 to 9, for setting the speed; 1 to 31, for loading the registers; i = 1 to 61, for indexed syringe movements; 1 to 11. The contents of the registers are; 1 contains the syringe size, 2 contains the reagent volume, 3 to 10 contain sample volumes, 11 contains the air gap size, 13 contains the speed, 14 contains an acceleration factor and 30 to 61 contain a deceleration table.

When the pump receives a message it immediately sends a response message which has the same format as the command message except the contents of the tt and ff fields are reversed and the data field contains a 0 character. When the command has been executed or has been attempted the pump sends a completion message. This has the same format as the response message except that the data field contains an error code plus any additional data relevant to the error state. An error code of 1 corresponds to no error, 2 signifies an invalid command, 3 indicates that the data was out of range and 5 tells of an hadrware malfunction.

All the commands in Table 10 were implemented. This was done by first sending the invariant, initial part of the message followed by the two letter command and any parameters; there are passed to the routine on the stack. Finally, a routine sends the end of message symbol and two asterisks (to

Name	Command	Function
Online	PO	Bring pump online
Select Unit	PNi	For future use by manufacturer
Display message	PXaaaaaa	Clear display and show message string
Set syringe size	PBnnnnn	Set syringe size to nnnnnµl
Set speed	PSnn	Set syringe speed to between 01 - 31
Load parameter	PLiinnnnnn	Load register ii with data nnnnn
Read parameter	PIii	Read value stored in register ii
Wait for event	PWi	Wait for event specified by i
Offline	PF	Take pump offline
Operate valve	PVi	Set valve to reagent if odd else sample
Change syringe	PC	Goto change syringe position
Home	PH	Move syringe to home position
Syringe Up	PUnnnnn	Move syringe up nnnnnµl
Syringe Down	PDnnnnn	Move syringe down nnnnnµl
Syringe Up Index	UPi	Move syringe up by amount stored in i
Syringe Down Ind.	DNi	Move syringe down by amount in i
Prime/purge	PPnnnnn	Prime/purge pump nnnnn cycles

Table 10 Summary of Syringe Pump commands

indicate no checksum being calculated), then receives the acknowledgment message. Since the acknowledgement message contains no useful information it is normally ignored. If the message needs to be inspected this can be done since it is still stored in the variable PAD. An example of such a command is shown in Figure 21. A second routine which waits for the completion message is then called; once the completion message has been received it is interpreted by an error checking routine. If an error has been detected an error message is printed on the screen and control is returned to the user.

The error checking routine works by finding the first occurrence of a numeric character in the message; the ASCII code of that character is converted to a number; if it is greater than one then an error has occurred and

```
: SYRINGE.PUMP.PREFIX ( - )
   ( String prefixed to all messages sent to pump )
   $" [PDAT" COUNT CTYPE
;

: SYRINGE.PUMP.RESPONSE ( - )
   ( String that terminates all messages with no checksum )
   $" ]***" COUNT CTYPE
   PAD COMPORT @ GET.RETURN ( Get acknowledgement string )
   DROP          ( Acknowledgement string not interpreted )
;

: SYRINGE.UP ( n - )
   ( Move syringe plunger up n microlitres )
   SYRINGE.PUMP.PREFIX
   $" PU" COUNT CTYPE     ( Send command )
   S->D <# #S #> CTYPE    ( Send parameter )
   SYRINGE.PUMP.RESPONSE
;
```

Figure 21 Example of low level syringe pump command.

an error message is printed based on the value of the error code. This completion message may also contain information, such as the values stored in a register. Routines which return data from the pump on the stack will interpret the part of the message with the necessary data. An example is shown in Figure 22.

In addition to the low level routines, commands were defined which combined the low level functions in order to provide operations not directly supported. These included routines to dispense solvent where the volume is greater than the syringe capacity, by repeatedly filling the syringe and dispensing. A routine to empty the syringe was also defined; this performed the reverse of the priming function. A summary of commands used to control the syringe pump is given in Table 11.

Command	Stack activity	Description
SYRINGE.COMPLETION.MSG	- pad	Wait for pump complete operation and read message returned.
SYRINGE.ERROR.CHECK	pad - pad	Check message for error, display error message and abort if necessary.
SYRINGE.PUMP.ONLINE	-	Start communication with pump.
SYRINGE.PUMP.SELECT	n -	Select pump for next commands.
SYRINGE.PUMP.DISPLAY.MSG	addr$ -	Display message on front panel of pump.
SET.SYRINGE.SIZE	n -	Inform pump of size of syringe installed.
SET.SYRINGE.SPEED	n -	Set speed of stroke.
SET.SYRINGE.PUMP.PARAMETER	n i -	Set parameter i to n.
GET.SYRINGE.PUMP.PARAMETER	i - n	Get value of parameter i.
WAIT.FOR.SYRINGE.EVENT	n -	Wait for event specified to occur.
SYRINGE.PUMP.OFFLINE	-	Stop communications with pump.
SYRINGE.VALVE.OPERATE	r	Operate valve to position specified.
CHANGE.SYRINGE	-	Move plunger to position in preparation for changing the syringe.
SYRINGE.HOME	-	Move plunger to home position.
SYRINGE.UP	n -	Move syringe up n µl.
SYRINGE.DOWN	n -	Move syringe down n µl.
INDEXED.SYRINGE.UP	i -	Move syringe up by amount specified in register i.
INDEXED.SYRINGE.DOWN	i -	Move syringe down by amount specified in register i.
SYRINGE.PRIME.PURGE	n -	Prime pump with n strokes.
INIT.SYRINGE.PUMP	-	Execute initialisation procedure for syringe pump.
SYRINGE.DISPENSE	n -	Dispense volume of n µl, where n is less than syringe volume.
SYRINGE.DEPRIME	n -	Empty syringe with n strokes.
SOLVENT.DISPENSE	n -	Dispense n µl of diluent, n may exceed syringe capacity.

Table 11 Summary of commands for control of syringe pump.

```
: GET.SYRINGE.PUMP.PARAMETER ( Parameter_Index - n )
   ( Get value of parameter )
   SYRINGE.PUMP.PREFIX
   $" PI" COUNT CTYPE      ( Send Command )
   DUP
   10 <
   IF
    ASCII 0 CEMIT          ( Send leading 0 )
   THEN
   S->D <# #S #> CTYPE
   SYRINGE.PUMP RESPONSE
   SYRINGE.COMPLETION.MSG   ( Get message with data )
   GET.SYRINGE.DATA       ( Interpret message for the data )
;
```

Figure 22 Example of command which receives data from syringe pump.

2.5 Peristaltic Pump.

A peristaltic pump (Wiz Pump, ISCO Inc. Lincoln Nebraska) was used to regulate the flow of solutions into the nebuliser of the ICP. The speed of the pump could be controlled from the computer via an interface unit (Cheminterface, ISCO Inc.) with an RS-232 interface. The pump could operate in three modes; pump, dispense or dilute. In pump mode the flow could be varied from 1.4 to 500 ml/hr/channel; in dispense mode it operated at 1500 ml/hr/channel. Up to four channels could be used. The accuracy was quoted as ±1% for pump mode and ±25 µl in dispense mode. A special serial input connected the pump to the interface unit [44]. The inlet tubing of the pump was fitted with a probe similar to that used with the syringe pump, the only difference being that the tip was not tapered.

The interface unit allows computer control of the peristaltic pump as well as other devices that the manufacturers supply. The commands can often be full English commands; however, only the first two or three characters are necessary in order to specify the command, the extra characters

Name	Parameter Range	Function
BB		Use big bubble, large air gap
BUBBLE		Draw in air gap
CCW		Set counterclockwise direction
COUNT n	1 - 65535	Count n volume pulses, or end pulses
CW		Set clockwise direction
DECR		Decrement flow rate, volume, or dilution factor
DELIVER		Start delivery in dilute mode
DILUTE		Select dilute mode
DISPENSE		Select dispense mode
INCR		Increment flow rate, volume or dilution factor
LB		Use little bubble
PICKUP		Start pick up portion of dilution
PUMP		Select pump mode
RECAL		Reset pump calibration to default value
RUN		Start pump or dispense
SELECT n	1 or 2	Select on of two pumps
SPEED n	0 - 8191	Set speed and start pump
STOP		Stop pump
TONE		Enable or disable end signal of dilute or dispense
*TEN		Increment range factor
/TEN		Decrement range factor
OFF		Turn relay off
ON		Turn relay on
RELAY n	1 or 2	Select relay for next ON/OFF commands
VALVE		Select relay 1 for next ON/OFF commands
ERROR n	1 - 99	Send error message with error code n
PAUSE n	1 - 65535	Pause for n tenths of a second
TRAP n	1 - 99	Execute following instructions on error n
(		Start of a repeat group
)REPEAT n	1 - 255	Repeat instructions enclosed between () n times

Table 12 Summary of commands for peristaltic pump and control of valves.

being ignored. A summary of the commands relevant to the control of the peristaltic pump are given in Table 12 [44,45], with the characters necessary to specify the command underlined.

A series of commands can be sent to the interface unit on a single line of less than 80 characters terminated by a CR. Once the commands have been executed the interface unit will send back a message depending on the results of execution. If no errors occurred then the message "OK" is returned otherwise an error message is returned (with an instruction number if necessary). An error code 1 was returned if the line was too long, error code 10 was returned if an unrecognised instruction was received, error code 11 was returned if a parameter was out of range, error code 12 was returned when an unexpected parameter was received, error code 20 was returned if the pump was not accepting data and error code 40 was returned if the execution of a command was halted [44].

In addition to providing control of the peristaltic pump, the interface unit includes two relay switches which can be used to operate valves or other electrical devices. The relays were suitable for controlling devices operated using 117 Vac. Both relays circuits included 3A fuses.

In order to switch the solution entering the nebuliser between a reservoir of distilled water and the sample to be measured, a pneumatically actuated two way valve (Rheodyne Model 5302, Cotati, California) was included. The pneumatic actuator (Rheodyne Model 5300) was operated by nitrogen at a pressure of 90psig controlled by an electrically operated solenoid valve (Skinner C3, Scarborough, Ontario). The solenoid valve was operated by Relay 1 of the interface unit. When the solenoid is not active the nebuliser is switched to the distilled water reservoir; when energised the contents of the container being sampled by the probe will be delivered to the nebuliser.

The commands listed in Table 12 were all implemented plus some not listed which handled digital I/O through the Cheminterface unit. In all these routines the minimum number of characters were used. These routines send the command followed by any necessary parameters. A series of commands are terminated using the PUMP.RESPONSE routine, which sends a Carriage Return and waits for the response message. The response message can then be

Command	Stack activity	Description
PUMP.RESPONSE	- pad	Send CR and get message returned store in PAD.
RELAY.ON	-	Turn on relay.
RELAY.OFF	-	Turn off relay.
VALVE.SELECT	-	Select Relay1 for next relay commands.
RELAY.SELECT	r -	Select Relay1 or 2 for next relay commands.
SET.CCW.DIRECTION	-	Set counterclockwise rotation.
SET.CW.DIRECTION	-	Set clockwise rotation.
RECAL	-	Set pump to default calibration.
DILUTE	-	Set to dilute mode and RECAL.
DISPENSE	-	Set to dispense mode and RECAL.
PUMP	-	Set to pump mode and RECAL.
PUMP.RUN	-	Start pump running.
PUMP.SPEED	n -	Set pump speed and start running.
STOP.PUMP	-	Stop pump running.
SET.VIRTUAL.MULTIPLIER	- :: value -	Set pump to range and multiplier corresponding to value, (default = 10.0).
SET.PUMP.RANGE.MULT	range mult -	Set range and multiplier to values given.
CHECK.RETURN.MESSAGE	addr$ - flag	Check if error has occured, true if no error.

Table 13 Summary of commands for control of peristaltic pump.

interpreted by an error checking routine. In Table 13 a summary of the routines for use with the peristaltic pump is given, and an example of the source code of a driver routine is shown in Figure 23.

The basic routines were then combined to provide functions not directly supported. In particular, commands to set the flow rate or dispense volumes were defined using repeated applications of the increment or decrement functions. A variety of methods for performing this task were defined, since the settings are a combination of a multiplier and a range parameter. These were combined as a virtual multiplier which was the product of the multiplier and the appropriate power of 10. Commands were

```
: PUMP.RESPONSE ( - PAD )
( Terminate Command line to pump and wait for a return message)
  PAD COMPORT @ 13 CEMIT WAIT.RETURN
;

: PUMP.SPEED ( n - )
( External speed set and start, n=0-8191 )
   $" SP " COUNT CTYPE     ( Send command )
   S->D <# #S #> CTYPE     ( Send parameter )
;
```

Figure 23 Example of low level command for use with peristaltic pump.

available which would set the pump by specifying the value of either the virtual multiplier or the values of the multiplier and the range.

A set of commands were defined to switch the nebuliser from aspirating the blank solution (typically distilled water) to the sample solution, along with a routine to run the pump at a higher speed during the first 2/3rds of the pre-flush time. This ensured that the solution entering the nebuliser when the integration started was truly representative of the sample being measured. The remaining time of the pre-flush allowed time for the system to equilibrate following the high speed pumping.

2.6 ICP Spectrometer.

The measurement system used was an Inductively Coupled Plasma Atomic Emission Spectrometer (ARL34000 Applied Research Limited, Sunland California). Most of the functions of the spectrometer were under the control of a PDP11/03-L computer.

The plasma was an argon plasma, Typical operating conditions are shown in Table 14. The values for the plasma and auxiliary gas flow were not available because no flow meter was fitted.

Operating Conditions	Value
Forward Power	1.2kW
Reflected Power	10W
Torch	Fassel
Nebuliser	Meinhard TR-30-A2
Spray chamber	Conical single pass
Gas Flows/pressure	
Plasma	60psi
Auxiliary	20psi
Nebuliser gas	1.0 l min^{-1}
Nebuliser Uptake	1 ml min^{-1} pumped
Preflush time	90 secs 60 secs high flow rate 30 secs normal flow
Integration times	30 secs 5 replicates

Table 14 Typical plasma operating conditions.

The spectrometer was a 1-m direct reading spectrometer based on the Paschen-Runge geometry. The dispersive element was a 1080 lines/mm holographic concave grating. The spectrometer has 34 exit slits in the focal plane positioned to measure the spectral lines indicated in Table 15 [46].

The signal from the photomultiplier tubes was integrated using a capacitor based integrating circuit. The integrated voltage was read by the PDP11/03-L using an 5 digit charge transfer analogue to digital converter. The output of the A/D converter was stored in an array in a Basic program. Basic also provided control of some of the indicators on the diagnostic panel on the instrument; these included the pre-flush indicator and a two digit LED display.

The spectrometer was originally designed to be operated with a DECwriter teletype at 300 baud through a 20mA current loop. This serial

Channel No.	Element	Wavelength/nm	Order	Atom/Ion
1	Zr	343.82	1	II
2	Sr	407.78	1	II
3	Ba	455.4	1	II
4	Ni	231.6	2	II
5	Al	237.34	2	I
6	B	249.68	2	I
7	Mn	257.61	2	II
8	Fe	259.94	2	II
9	N	174.27	3	I
10	P	178.29	3	I
11	S	180.73	3	I
12	Hg	184.95	3	I
13	Mg	279.08	2	II
14	As	189.04	3	I
15	Sn	189.99	3	II
16	Si	288.16	2	I
17	C	193.09	3	I
18	V	292.4	3	II
19	Na	589.59	1	I
20	Mo	202.03	3	II
21	Cr	205.55	3	II
22	Sb	206.83	3	I
23	Ge	209.53	3	
24	Ca	317.93	2	II
25	Zn	213.86	3	I
26	Cu	324.75	2	I
27	Ag	328.07	2	I
28	Pb	220.35	2	II
29	Li	670.78	1	I
30	Ti	337.28	3	II
31	Cd	226.5	3	II
32	Cd	228.8	3	I
33	In	230.61	3	II
34	K	766.49	1	I

Table 15 Wavelengths of spectral channels available on ARL 34000.

interface was modified by including a 1.2kΩ resistor in the cable to operate with RS-232 interfaces. Also the baud rate was increased to 1200 baud. Thus data exchange could be accomplished using the PRINT and INPUT statements of Basic.

A series of routines was written to facilitate communication with the computer associated with the spectrometer. These used a handshaking protocol which involved the exchange of synchronisation characters prior to the data exchange. Communication with the spectrometer is initiated by the host computer by sending a CR character. The spectrometer responds with a single character which varies depending on the transaction expected to be carried out. When the host computer verifies that the character sent is the one expected it sends an acknowledgement character. Once this synchronisation procedure is complete the data is transferred.

This procedure was adopted because the two computers independently run separate programs and it is necessary that prior to data transfer the computers be checked to determine whether they are at equivalent points in their programs. The synchronisation routines are able to retransmit characters if a time out occurs. This may arise when one computer is delayed by a long process, such as when the spectrometer is performing an integration. Routines were defined for the transfer of string, integer and floating point data. Floating point data originates or is left on the floating point stack of the 8087 co-processor. An example of a data transfer command is shown in Figure 24, and a summary of commands used with the spectrometer is given in Table 16.

```
: SYNC.ARL ( Send_sync_char Recv_sync_char - )
    ( Get char from ARL )
    CKEY
    BEGIN
      DUP 0=
      (If character is NUL then Time Out has occurred )
    WHILE
      DROP
      13 CEMIT CKEY
      ( Send CR to prompt for Sync char to be resent )
    REPEAT          ( Repeat till a non-NUL char. is received )
    = NOT
    ( Check if correct )
    IF
      ( If not then abort with error message )
      CRT
      ABORT" Lost Sync with ARL"
    THEN
    500 WAIT        ( 500mS Delay )
    CEMIT         ( Send acknowledgement character )
;

: N.SEND.ARL ( n Recv_char Send_char - )
   ( Send Integer data )
   SYNC.ARL
   ASCII ? WAIT.ARL        ( Wait for Basic Input Prompt )
   S->D <# #S #>           ( Convert data to string )
   CTYPE
   13 CEMIT
   PAD COMPORT @ GET.RETURN
   DROP
   CKEY DROP
;
```

Figure 24 Example of command to transfer data between spectrometer and computer.

In addition to the definition of routines for data transfer, variables for the storage of pre-flush and integration times were defined, as well as routine for transmitting these times to the spectrometer. A series of constants were defined to allow the spectrometer channels to be referred to by element symbols. An array to store the elements being analysed was defined as well as a routine to transfer this information to the spectrometer.

Command	Stack activity	Description
RESTART.ARL	-	Restart program running on ARL.
BREAK.ARL	-	Halt program currently running on ARL.
WAIT.ARL	char -	Wait till ARL sends char.
SYNC.ARL	char2 char1 -	Perform synchronisation check with ARL.
RECEIVE.ARL	char2 char2 -pad	Receive message from ARL.
SEND.ARL	addr$ char2 char1 -	Send message to ARL.
N.SEND.ARL	n char2 char1 -	Send integer.
N.RECEIVE.ARL	char2 char1 - n	Receive integer.
F.SEND.ARL	#p char2 char1 - :: f -	Send real number with precision of p digits.
F.RECEIVE.ARL	char2 char1 - :: - f	Receive real number.
STORE.ELEMENTS	0 en...e1 -	Store elements to be measured in ELEMENT.ARRAY.
LIST.ELEMENTS	-	Display symbols of channels to be measured.
SEND.PREFLUSH	-	Send preflush time to ARL.
SEND.INTEGRATION	-	Send integration time.
SEND.REPLICATES	n -	Send number of replicate measurements required.
SEND.ELEMENTS	-	Send channel numbers to be measured.
ASPIRATE.SAMPLE	-	Switch valve to aspirate sample.
ASPIRATE.WATER	-	Switch valve to aspirate distilled water.
?ARL	-	Print name of program running on ARL.
ARL.PROG.RUN	n -	Run program on ARL.
ARL.QUIT	-	Stop running program and return to command processor.

Table 16 Summary of commands used with the ARL 34000 spectrometer.

The programs for the spectrometer were written in Basic. They tended to follow the standard structure of an initialisation step during which instrument specific data was loaded from a file, and variables were declared and initialised. The version of Basic used allowed a statement to be included which would cause the program to branch to a particular line of the program whenever a Control-X character was received. This was used to restart the program from a known point. After restarting, the program would receive the elements to be studied. This routine included an escape option to allow

```
10 Rem Readout Program
11 Rem Load instrument specific information from SYMB file
12 Open #SYMB, 0, X5: Rem open file on drive zero, X5 has error code
13 DIM T(21): Rem dimension array to store instrument information
14 Fetch(1,0) (21)T(0): Rem Get information from record 1
15 Let T9=T(0): Let L9=T(1): Rem T9=No. of Channels, L9=Line freq.
19 Abort 30: Rem Restart program from line 30 when receive Crtl-X
20 Dim M(T9), X(T9): Rem Dimension arrays for intensity data and
channels
30 Print "Hello": Rem Print program title
40 Gosub 4500: Rem Get elements
50 Gosub 1000: Rem Get preflush
60 Gosub 1050: Rem Get integration time
70 Gosub 1100: Rem Get number of replicates
80 Wait: Rem Wait for IBM to signal start of measurement
90 Print "I": Rem Send I to signal ready to start
100 Key @A,0,10: Rem Get character from IBM
110 If @A <> "S" Then 100: Rem Loop till get character
120 Gosub 2000: Rem Do preflush
130 For I = 1 To N: Rem Do N replicates
140   Gosub 2100: Rem Do integration
150   Print "I": Rem Send I to tell IBM integration complete
160   Gosub 2300: Rem Send error code
170   Gosub 2200: Rem Send data
180 Next I
190 Goto 30: Rem Goto start of program

1000 Rem Get Preflush time
1010   Wait: Print "P";: Rem Wait for IBM then send P
1020   Key @A,0,10: Rem After short delay read char sent from IBM
1030   If @A <> "F" Then 1020: Rem If not F then loop till get F
1040   Input T1: Rem Get preflush time
1042   Call(9,T1): Rem Display preflush time on two digit panel
1045 Return
```

Figure 25 Example of main routine of a Basic program run on spectrometer.

the program to be terminated and the return to an executive program from hich any other program designed to be used in this system could be run.

Other experimental data were then be transferred, starting with data common to all experiments such as pre-flush and integration times and number of integration replicates. This was then followed by information specific to a particular type of experiment. The main part of the experiment

```
10 Entry Robot Command Processor
20 Rem This program inputs a menu choice from the IBM
30 Rem and then RUNS the program corresponding to that choice
40 Abort 50: Etrap X,Y,50: Rem Always come back to line 50 even after
an error
50 Print "Command": Rem Program title
60 Wait: Rem Wait for activity from the IBM
70 Print "P";: Rem Send synchronisation character
80 Key @A,0,10: Rem Get synchronisation character
90 If @A <> "R" Then 80
100 Input N: Rem Get program number
110 Goto N: Rem Goto code that RUNS the program

200 Entry Run Readout Program
210 @P =AURDV
220 Rem Assign program filename to variable NOTE no space between =
and name
230 Load @P: Rem Load and run program
```

Figure 26 Basic program used to execute robotic programs.

would then begin. An example of the main routine of a typical program is shown in Figure 25.

The executive program differed from this format. It was simply designed to accept a program code from the host computer, that caused the program to branch to that part of the program which ran the desired program. The main routine and an example of how a program file is run are shown in Figure 26.

2.7 Balance.

In order to check the calibration of the syringe pump a top-loading balance was included (Model L 420 P+, Sartorius GmbH, Goettingen, West Germany) in the system. The maximum capacity of the balance was 424g, with the readibility varying between 0.001 and 0.005g depending on the weight range. The precision was quoted as being ≤±0.001 to 0.0025g, again depending on the range [48]. The stabilisation time was 2s [48]. The balance included an

Name	Command	Function
Change Units		Change weight unit
grams	ESC A	
kilograms	ESC B	
carats	ESC C	
pounds	ESC D	
ounces	ESC E	
troy ounces	ESC F	
taels Hong Kong	ESC G	
taels Singapore	ESC H	
taels Taiwan	ESC I	
grains	ESC J	
pennyweights	ESC U	
mommes	ESC V	
milligrams	ESC W	
karats	ESC X	
/lb	ESC Y	
Stability		Set settling time
Environment type		
Very stable	ESC K	
Stable	ESC L	
Unstable	ESC M	
Very Unstable	ESC N	
Lock keyboard	ESC O	Stop keyboard input
Unlock keyboard	ESC R	Release keyboard
Print	ESC P	Send data to host
Tare	ESC T	Tare
Self check	ESC S	Check circuits
Internal calibration	ESC Z	

Table 17 Commands used by balance.

RS-232 option which allowed the setting of various parameters by the computer as well as the transfer of weight data from the balance to the computer.

The commands that the balance recognised were all preceded by an ESC character followed by a letter. The commands are summarised in Table 17. The weight data returned was always in the format;

Command	Stack Activity	Description
SELECT.WEIGHT.UNIT	unit -	Select unit of weight to be used.
SET.STABILITY	code -	Set stability of reading required.
LOCK.BALANCE.KEYBOARD	-	Prevent keyboard operation, not functional.
RELEASE.BALANCE.KEYBOARD	-	Allow keyboard operation.
BALANCE.CHECK	-	Perform internal check of electronics.
TARE.BALANCE	-	Set tare weight.
INTERNAL.CALIBRATE	-	Calibrate balance.
READ.BALANCE	- :: - weight	Get weight from balance.
GET.WEIGHT	- :: - weight	Get weight after a 5 second delay and when two weights which agree have been obtained.

Table 18 Summary of commands for use with the balance.

```
"      ± 9999.999 KKK"
```

where the message had 6 leading spaces and where the ± could also be replaced by a space. The numerical information was printed in a floating point format and the message was terminated with the weight unit being used.

All the commands listed in Table 17 were implemented even though not all were effective because the presence of a programmed keyboard locks out some of the commands [49]. Table 18 summarises the commands that can be used with the balance. In Figure 27 an example of one such command is shown. The keyboard allows an extra set of commands to be sent which would emulate the keypresses of the keyboard, but these were not implemented.

The function to print a weight received the weight data and interpreted it, leaving the weight on the floating point stack. It was found that when

```
ASCII A CONSTANT g

: SELECT.WEIGHT.UNIT ( unit - )
    27 CEMIT ( Send ESC character )
    CEMIT    ( Send Unit Code character)
;
```

Figure 27 Example of low level command used to control top loading balance.

small weight changes are made that data will be sent even when the readings are not completely stable, i.e. when a slow drift is still present. This could result in weights being recorded which were systematically low. A command was defined which would only start taking data after a 5 second delay, so that even a slow drift will have settled, and would repeatedly read the weight until two weights which agree to within 1 mg have been obtained.

Chapter 3

Solution Preparation.

The solutions to be prepared will be made from single element stock solutions of typically 1000 µg ml^{-1} concentration. The typical volume of solution to be prepared will be 10 ml. The concentration range of the final [illegible] will, ideally, range from about 10 ng ml^{-1} to close to 1000 µg ml^{-1}. Theretore the lowest concentration will require a volume of 0.1 µl of stock solution for a single step dilution. This volume is too small to be delivered by the pump available, therefore a serial dilution method will have to be used.

The liquid handling will be performed by computer controlled pumps. Two types were available, a peristaltic and a syringe pump. The method by which a multi-element dilution can be achieved is demonstrated using the syringe pump in Figure 28. Initially the pump will be filled with the diluent (Figure 28a). Before a component is drawn into the pump, it is separated from the diluent by an air gap (Figure 28b). The probe is then lowered into the stock solution (Figure 28c) and the required volume drawn up. Another air gap is then drawn (Figure 28d). Steps c and d are repeated for each component. Once the all the components are in the tube, or the capacity of the pump is filled (Figure 28e), they are emptied into the receiving container (Figure 28f). Steps b to f are repeated as often as is necessary to transfer all the components. Once all the components have been transferred the diluent is added (Figure 28g). This usually requires several strokes of the syringe. The solution is then homogenised (Figure 28h).

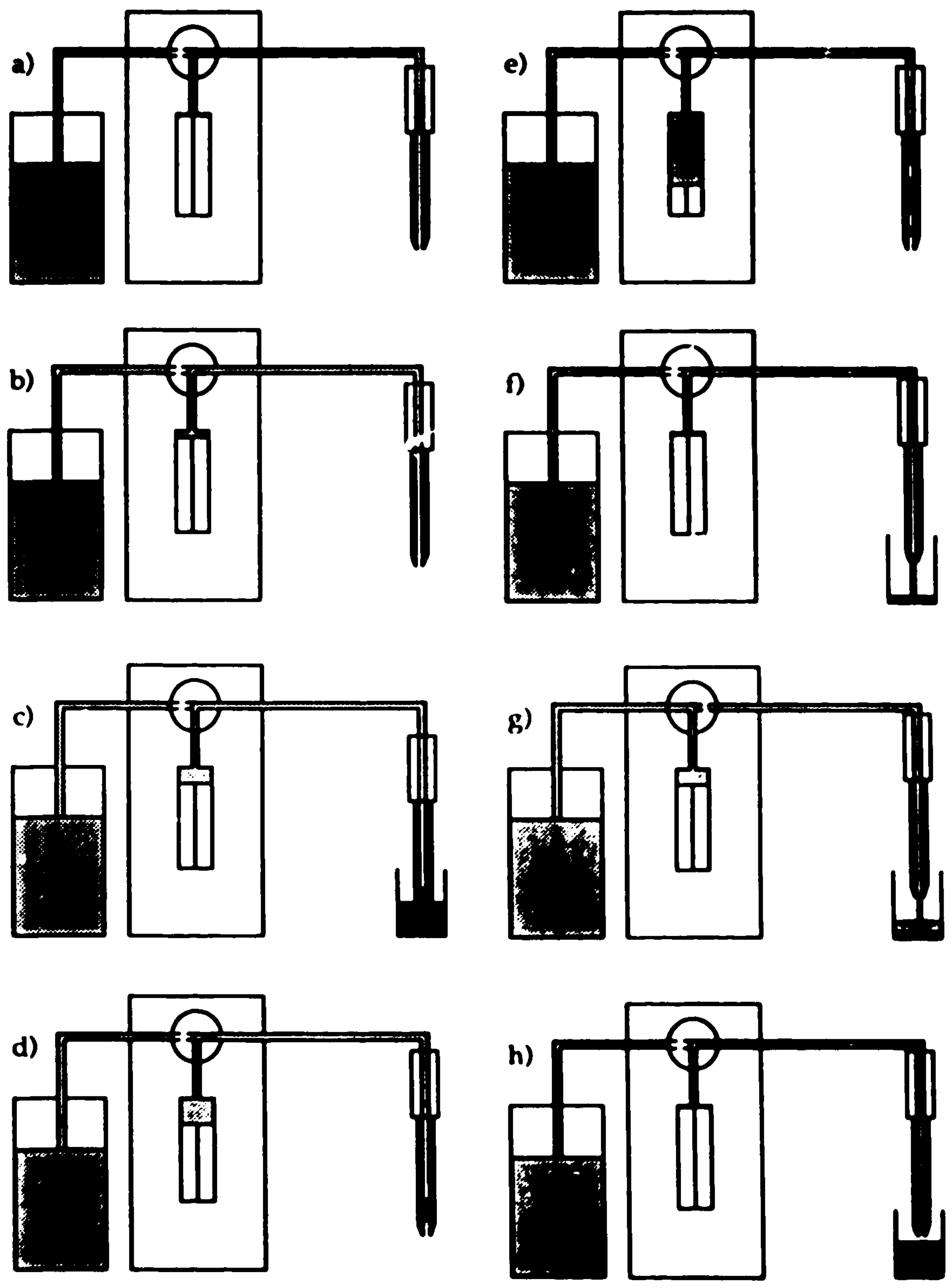


Figure 28 Illustration of solution preparation method. See text for explanation of sequence.

The procedure for a peristaltic pump is similar. The direction of the rollers can be reversed to draw up solution. The capacity of the pump would be limited by the length of tubing used since it will be undesirable to have the stock solution enter the peristaltic tube.

The accuracy, precision and reproducibility of the solutions prepared will depend on the performance of the pump used with respect to these parameters. The accuracy of the solution preparation is also dependent on the completeness of the transfer of the standard solution to the receiving vessel. Not only is it important that all the solution measured be delivered to the receiving vessel but also that no unmeasured volume of liquid be transferred in addition. If precision and accuracy are to be maintained over a long period of use then the integrity of the standard solutions must also be maintained. Contamination of the standard solutions must be minimised. The main source of contamination is from the carry over of one standard to another on the outside of the probes. The probes were coated with teflon to minimise the adherence of solution but drops of liquid still remained.

The precision required for the solution preparation will depend on the precision of the measurement made on the solutions. The more precise the measurement system then the more care will be necessary to ensure that the solution is prepared correctly. The ultimate test of the accuracy of the solution preparation is a comparison with solutions prepared by methods known to yield correct results.

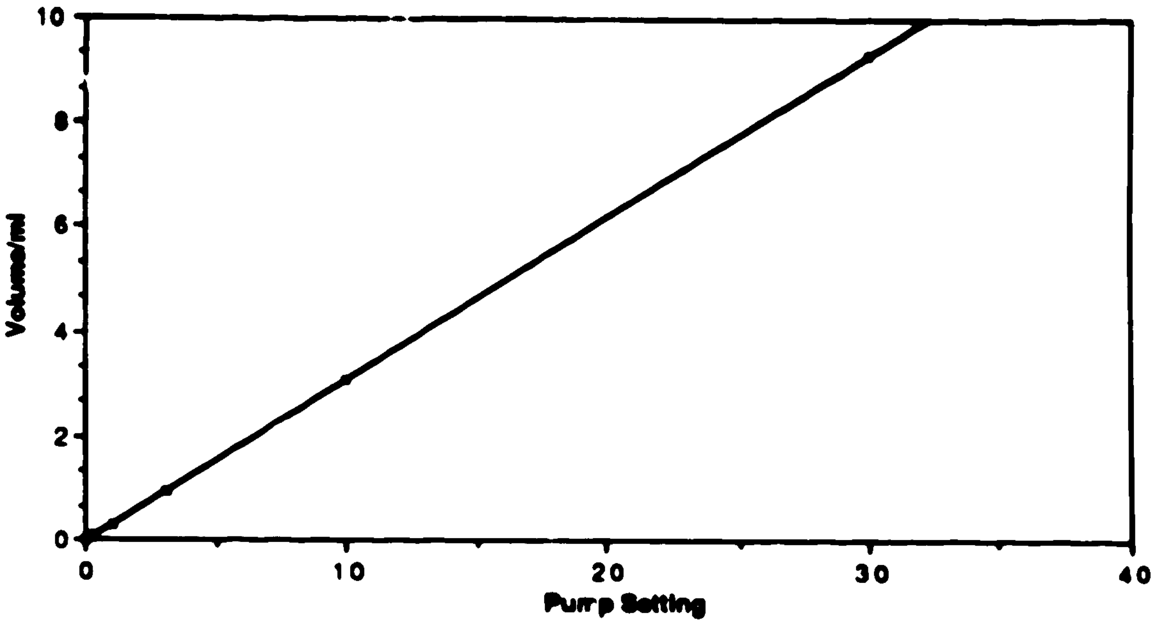


Figure 29 Calibration plot for peristaltic pump.

3.1 Calibration of Pumps.

3.1.1 Peristaltic Pump

Initially a peristaltic pump was preferred over a syringe pump. The advantages expected were a wide dynamic range of volumes that could be transferred in a single operation. This would lead to simpler operation. Also multiple channels would be available which could allow a high flow rate channel to be used for transfer of diluent and a lower flow rate channel for the standards. Potential problems would be chemical compatibility with the solvents and the day-to-day reproducibility in positioning the pump tubing. The syringe pump can transfer a lower range of volumes in a single operation but should have better reproducibility.

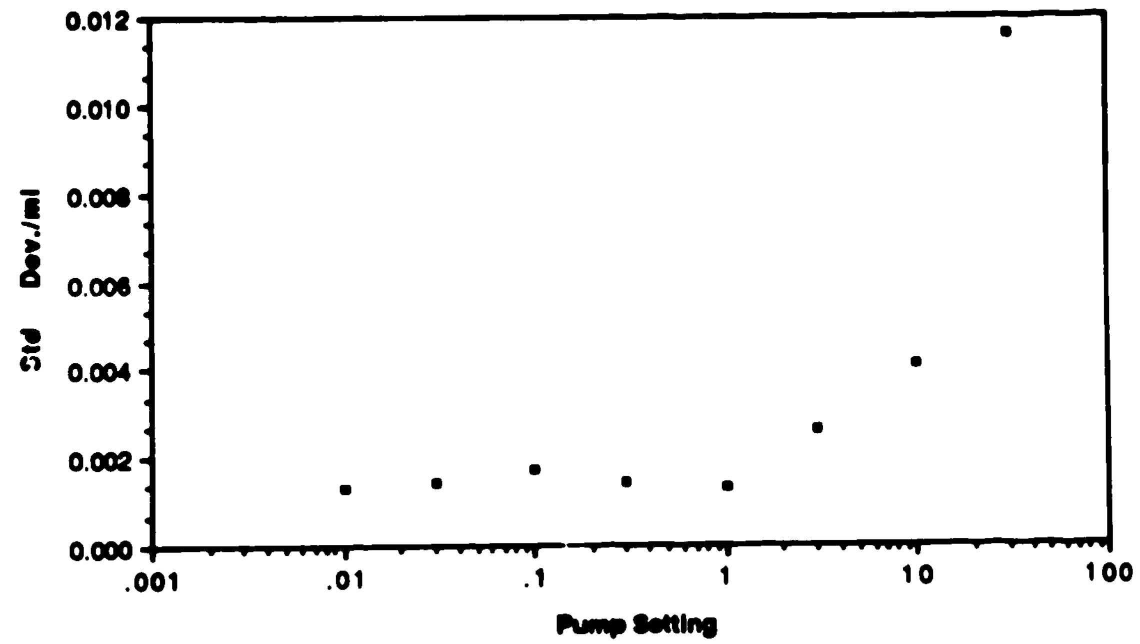


Figure 30 Standard deviations of volume dispensed as function of volume.

Calibration of the peristaltic pump was performed by transferring a variable amount of distilled de-ionised water to a vial on an analytical balance. The volume transferred was calculated from the weight and the temperature of the water. This was repeated five times and the mean and standard deviation calculated. A calibration of volume dispensed against the pump setting is shown in Figure 29. The variation in standard deviation as a function of pump setting is shown in Figure 30. The calibration was found to be linear over three orders of magnitude; a logarithmic plot had a slope of 1.001. It was also found that at larger volumes, above 0.3 ml, the standard deviation increases approximately linearly with volume. At 10 ml the relative standard deviation was about 0.1%.

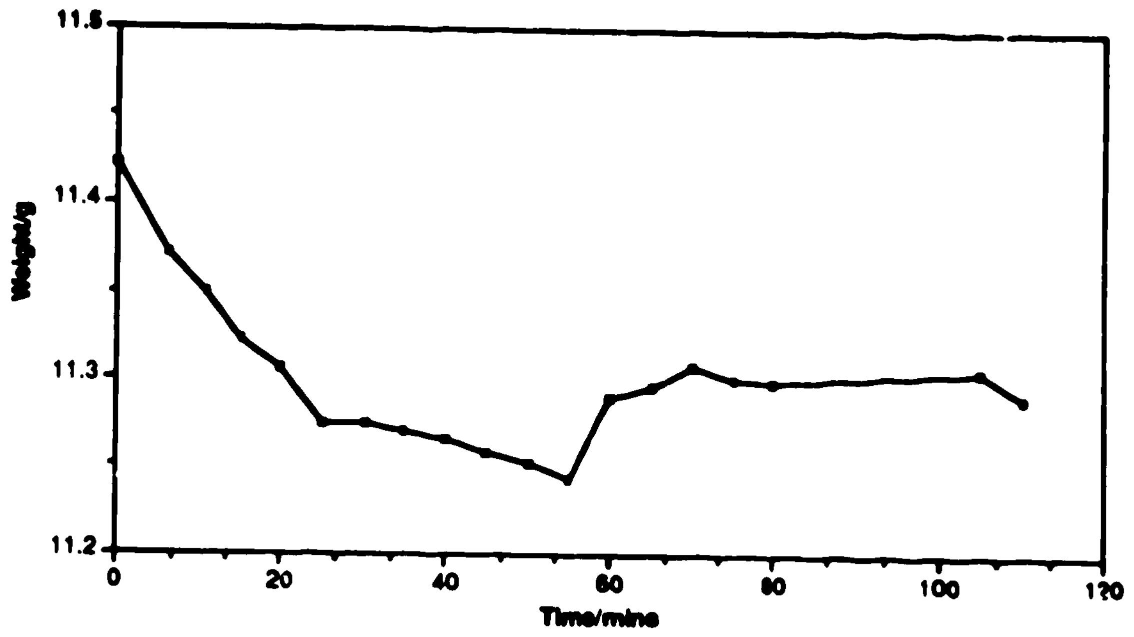


Figure 31 Drift in weight of liquid transferred by peristaltic pump over the short term.

The reproducibility of the dispensing of a volume was tested by pumping continuously for a period of approximately two hours, the effluent from the pump being returned to the reservoir. Periodically, the pump was stopped and used to transfer a fixed volume of water to a vial on an analytical balance. A plot of the weight of water transferred over the two hour period is shown in Figure 31. It was found that initially there was a drop in the volume of ·'ater transferred, followed by a more constant region. However, the size of the drift (~2%) is such that frequent recalibration of the pump would be necessary. The source of the drift could be due to either creep in the position of the tubing or deformation of the tubing between the rollers of the pump. It was decided that the peristaltic pump would be unsuitable for the sample preparation though it would still be useful for pumping solutions into the nebuliser.

3.1.2 Syringe Pump

Movement of the syringe of the syringe pump was controlled using microlitre units. In order to verify the accuracy of this the syringe pump was calibrated. If necessary, the calibration could have been used to correct systematic error that may result from any discrepancy between the volume used in the command and the volume dispensed. This calibration would also provide a measure of the precision of the pump and how that precision varied with the volume being dispensed. This precision will then be used in the design of the method for solution preparation. When transferring the standard solution an air gap was used to separate the components of the solution. The effect of the size of the air gap on the accuracy of the transfer was also investigated.

The effect of the size of the air gap was measured by varying the air gap size and transferring a 200 µl volume of water to a vial on the balance. The weight of water was measured, each weight being the average of ten readings, and the transfer was repeated sixteen times from which the average weight and standard deviation was calculated. The temperature of the water was measured and the density of water at this temperature was used to convert the mean value of the weight to the volume of water delivered. The mean value was plotted against the size of the air gap.

The calibration of the syringe pump was performed using the same procedure except that prior to drawing the air gap the position of the syringe was randomised. This was done in order to more closely model the typical

```
: TRANSFER.TEST ( volume no_of_replicates : )
  Get_and_open_file
  Print_Header
  Get_Averaged_Weight  ( Average of ten readings )
  Clear_Summation_Variables
  0
  DO
    Goto_Position_of_Water
    Generate_Random_Number
  ( Random number is between 0 and 900-Volume )
  ( This ensures that there is room for the )
  ( water to be drawn up )
    Syringe_Down ( Move to random position )
    Draw_Air_Gap
    Go_Down_Into_Water
    Draw_Volume_of_Water
    Goto_Balance
    Get_Averaged_Weight
    Deliver_Water
    Reset_Plunger_Position
    Get_Averaged_Weight
    Update_Summation_Variables
    Come_Out_of_Bottle
  LOOP
  Calculate_and_Print_Statistics
  Close_File ;
```

Figure 32 Pseudo-code for main routine of syringe calibration procedure.

use of the pump where the volumes to be transferred will be in an essentially random order. Also it will remove any bias that may result from sampling only the top part of the syringe barrel. The pump was calibrated by transferring water from the standards tray and also by dispensing diluent. In Figure 32 is shown a pseudo-code listing of the main routine of the program used to calibrate the syringe pump by transfer of water.

The effect of the size of the air gap is shown in Figure 33. It can be seen that the volume of liquid transferred decreases with an increase in the size of the air gap. This is probably due to the expansion of the air gap as the water vapour content of the air increases during the drawing up of the liquid. For

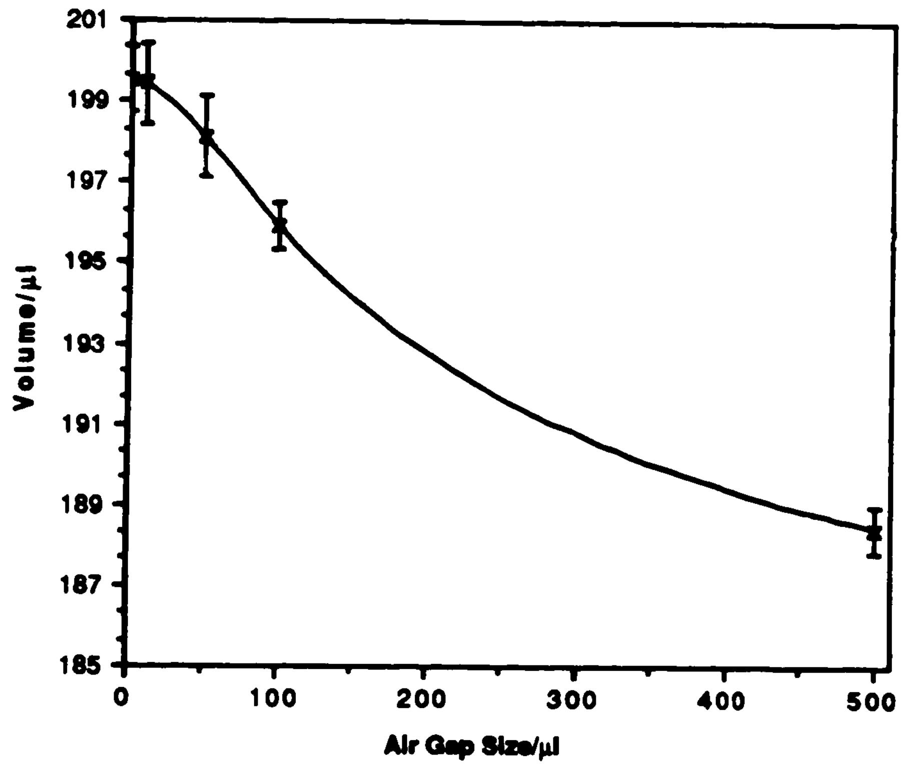
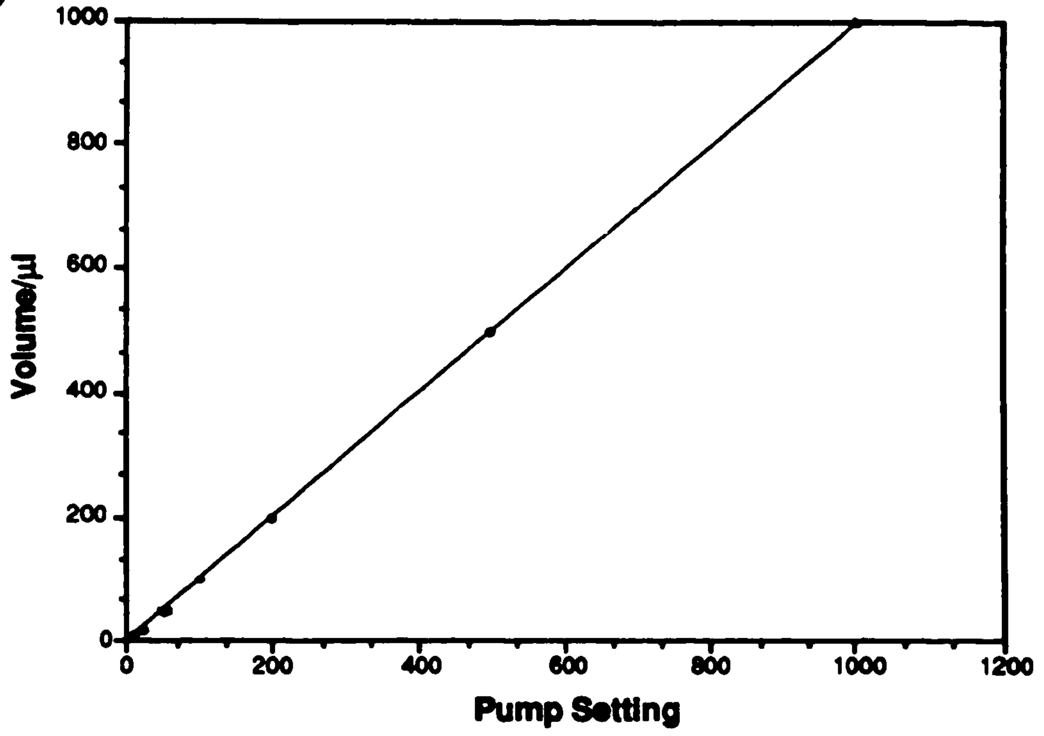


Figure 33 Effect of size of air gap on volume of liquid transferred.

water at a temperature of 25°C the vapour pressure is 23.8 Torr. If there was no water content in the air gap then when fully saturated the volume of the air gap would have increased by 3%. The expansion would then displace some of the liquid. It was found that the volume increase was not always as large as might be expected. For example, at 500 µl the error represents an increase in the air gap volume of only 2%. This will be due to the fact that the air in the air gap will not be completely dry and that the air gap will not get completely saturated with water vapour during the time of the transfer. Since

a)



b)

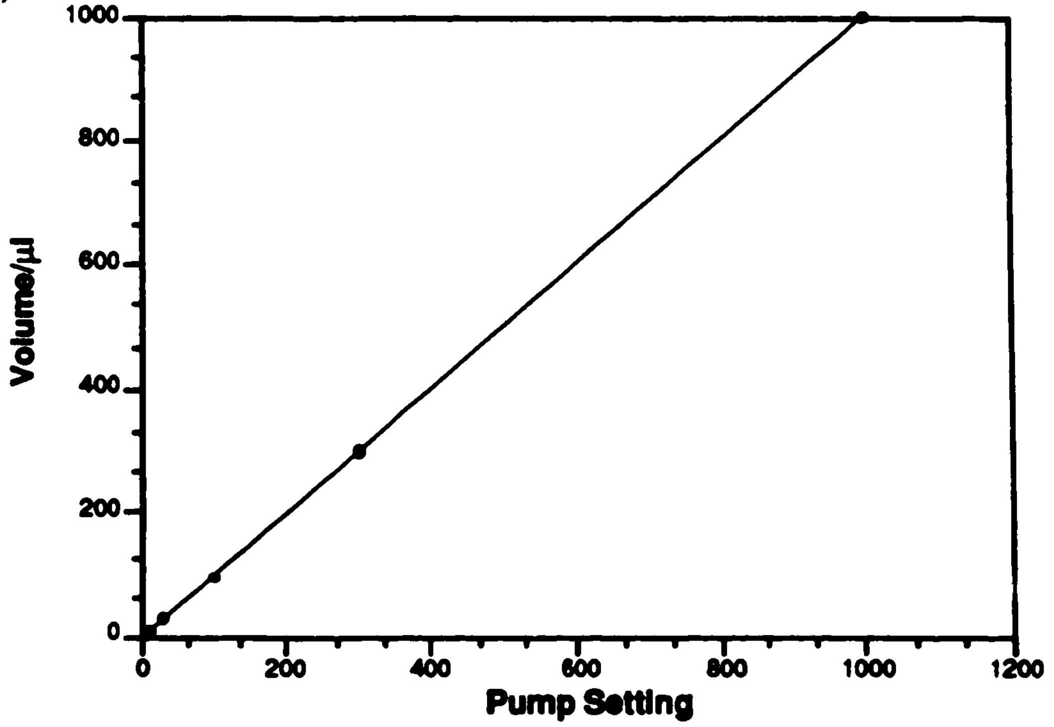


Figure 34 Calibration plots for a) transfer and b) dispense mode.

Coefficient	Transfer Mode	Dispense Mode
Slope	0.998±0.0007	0.9995±7x10^{-5}
Intercept	-0.1±0.3	-0.12±0.03

Table 19 Regression coefficients for calibration of syringe pump in transfer and dispense mode.

the size of the expected error is directly proportional to the size of the air gap the error can be minimised by reducing the size of the air gap. The air gap will help to reduce contamination of the standard solutions by preventing diffusion of the last standard from the end of the tube. As a compromise between reducing contamination and introducing an error due the effect observed above an air gap size of 10 µl was used for all experiments.

The calibration plots for the dispensing of diluent and for transferring of solutions are shown in Figure 34. These are both linear with unit slope and zero intercept, as shown by the regression coefficients in Table 19. This shows that no systematic error will arise from using microlitre volume units in the syringe pump commands. The corresponding uncertainties in the calibration points are shown in Figure 35. These are essentially at a constant value of about 0.8 µl.

3.2 Factors Influencing Solution Preparation.

In order to prepare a solution the program must be informed of the number of components in the solution, the volume and position of the standards, the volume of diluent to be added and the position of the bottle in which the solution is to be made. This information is stored in a record structure, the base address of which is passed to the solution-making routine on the stack. The structure of the record is shown in Figure 36. A solution can be created using the definer SOLUTION. Solutions defined in this way can be

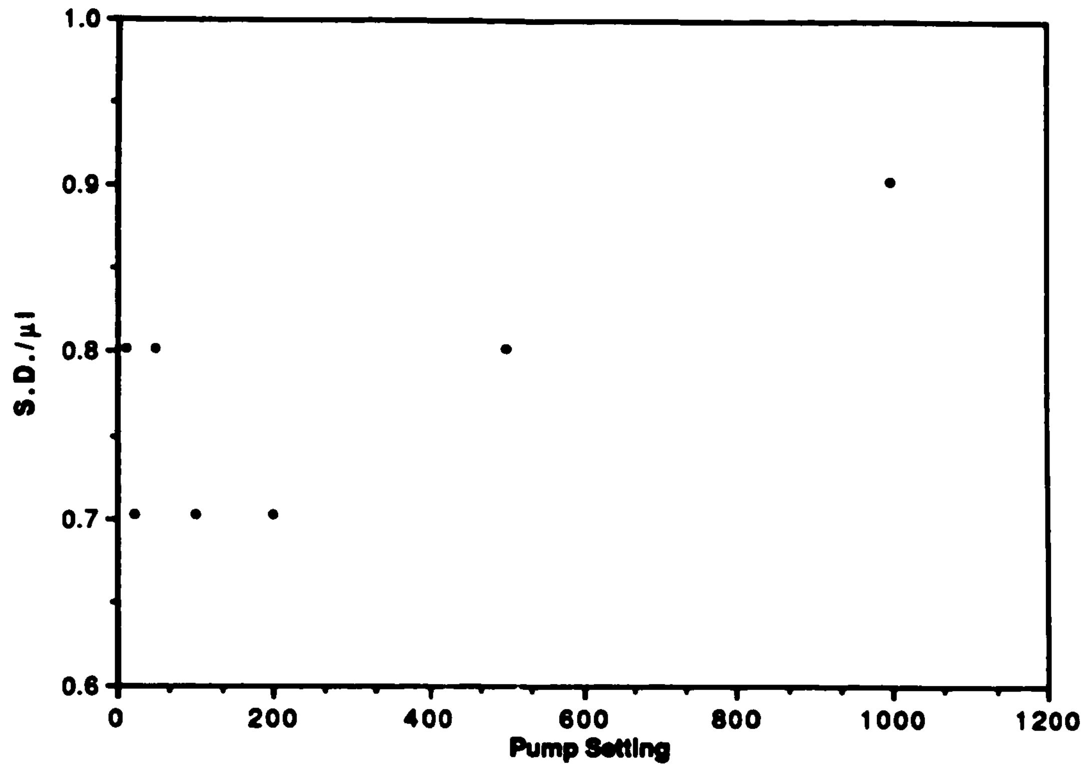


Figure 35 Uncertainty in volume of liquid transferred as function of volume.

made repeatedly, however the position of the receiving vessel must always have an empty bottle present. Also, the dilution ratios and the volume of diluent must be calculated manually. A procedure which automates the definition and the preparation of a solution will be described later.

The method for preparing a solution was that outlined in Figure 28. In order to control the amount of contamination the outside of the probe tip was washed in a container of distilled water at the 2,4 position of the standards tray. The homogenisation of the solution was performed by the robot using the procedure described below.

```
Solution
   No of Components        : integer
   Position of solution    : position
   Volume of diluent       : integer
   Position of standard 1  : position
   Volume of standard 1    : integer
          .
          .
          .
   Position of standard n  : position
   Volume of standard n    : integer
```

Figure 36   Structure of record for defining a solution.

Once the diluent has been added the probe is returned to the stand. The robot then picks up the bottle with the newly made solution and mixes the contents with a slow wrist shake. The homogenisation of the solution occurs by starting with the vial in a vertical position, moving to an angle 60° from the vertical, swinging through 120° and back ten times then returning to the vertical position. The average speed of rotation was 70°/s, or approximately 20rpm, the whole mixing procedure typically took 36 seconds. The completeness of homogenisation was verified by simulating a worst case scenario by adding a concentrated potassium permanganate solution in the bottom of a vial with 10 ml of distilled water. As can be seen from Figure 37a the permanganate solution forms a layer on the bottom. After the mixing procedure the colour of the solution was found to be uniform throughout as shown in Figure 37b. Normally, the solution is partially homogenised by the adding of the diluent, hence this procedure will be more than sufficient for a typical solution. It might have been expected that more vigourous mixing would be required such as can be provided by a vortex mixer or more rapid shaking. However, from Figure 38 it can be seen that during the main part of

a)

b)

Figure 37 Photograph of a) potassium permanganate solution prior to homogenisation by shaking. b) potassium permanganate solution following homogenisation by shaking.

Figure 38 **Photograph of potassium permanganate solution during homogenisation by shaking, showing disruption of flow pattern during turnaround.**

the swing a regular flow pattern is established, but at the top of the swing when the wrist slows down for turnaround this flow pattern is disrupted and mixing can occur quite effectively.

Once a solution was prepared, if it was to be measured it would be transported to the measurement tray in front of the spectrometer. When a solution was ready to be measured the procedure illustrated in Figure 39 was executed. Normally distilled water, or solution blank, was being aspirated into the plasma (Figure 39a). First the valve was switched to the sampling position (Figure 39b), this allows an air gap to be drawn into the tube. The sampling tube was then lowered into the solution to be measured (Figure 39c) and the speed of the pump was increased to reduce the time required for pre-flushing. Some time (typically 30 seconds) after the pump had returned to the normal speed the first integration was started. Normally, 5 replicate integrations of 30 seconds were measured (Figure 39d). After the last replicate the sampling probe was removed from the solution (Figure 39e) and the outside was washed in the same manner as for the solution preparation probe. The wash solution was then drawn into the sampling tube (Figure 39f) to prevent the solution just measured from being carried over to the next measurement. The sampling probe was returned to its stand and, normally, the used solution container would be emptied.

The washing of the probe in distilled water reduced direct contamination of one standard by another but there is still a chance of contamination by carry over in the wash solution which adheres to the side of the probe. That is, the standard washed off the probe is diluted by the distilled water, but now the wash solution is contaminated and can carry this contamination to another standard. With prolonged use this contamination will increase and hence so will the contamination of the standard solutions.

This was tested by preparing solutions for a typical working day (about 30 to 40 multi-element solutions). Then a single element solution for each element was prepared, each being a ten fold dilution. The elements used were calcium (1000 $\mu g\ ml^{-1}$ in HCl from Baker Analysed Reagent, Calcium Carbonate powder), iron (1000 $\mu g\ ml^{-1}$ in aqueous solution from SRM 2124-3 NBS), barium (1000 $\mu g\ ml^{-1}$ in 2% HNO_3 Custom Standard, Inorganic Ventures Inc, Lot DO5WG), boron (1000 $\mu g\ ml^{-1}$ in 2% HNO_3 Custom Standard, Lot DO5WG), zinc (1000 $\mu g\ ml^{-1}$ in 2% HNO_3 PLZN-2, Spex Industries Inc, Lot 06B6MP) and sodium (1000 $\mu g\ ml^{-1}$ in 2% HNO_3 Custom Standard, Lot DO5WG). The concentrations of the contaminating elements in

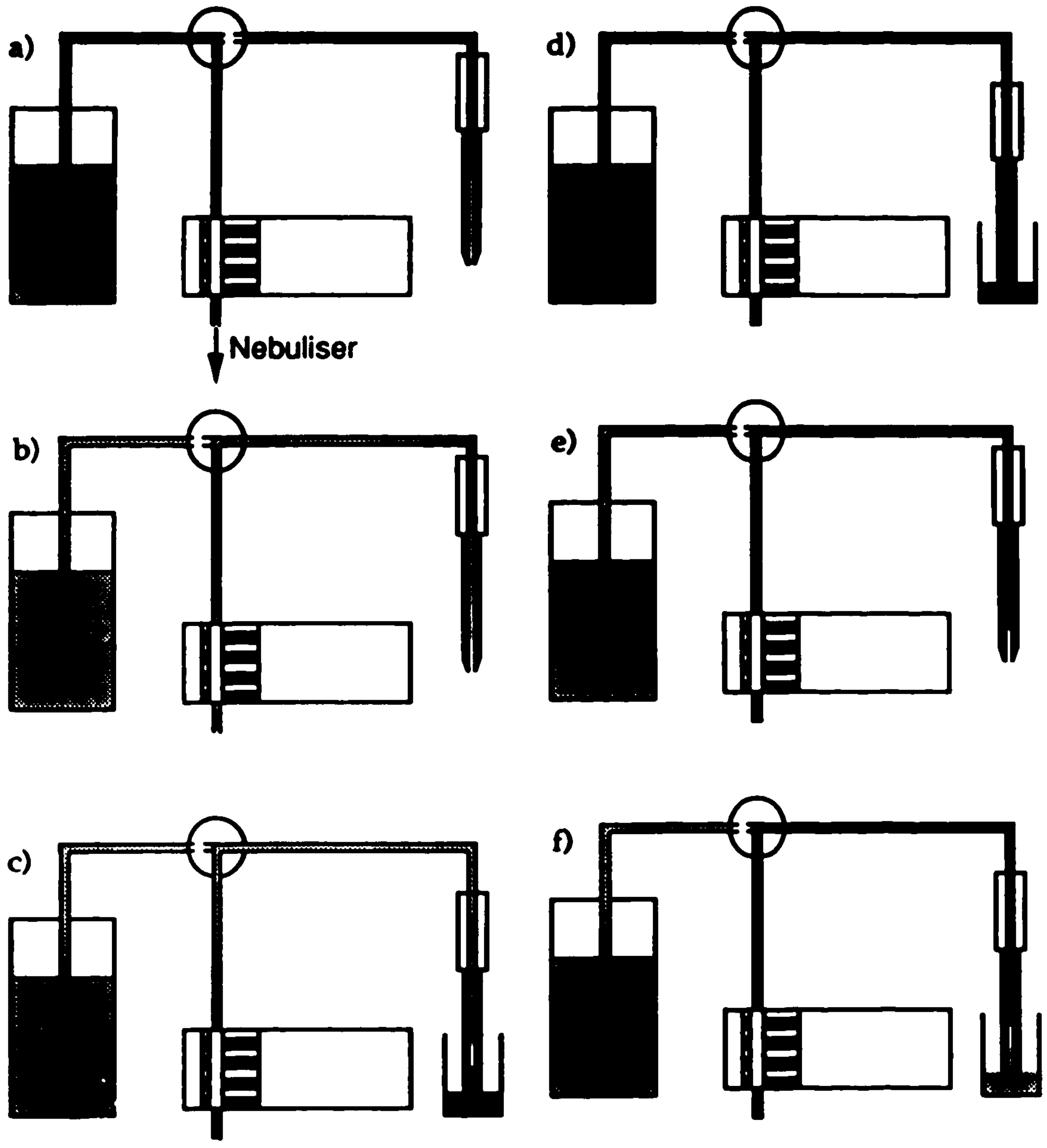


Figure 39 Illustration of measurement procedure. See text for explanation of sequence.

	Concentration/$\mu g\ ml^{-1}$					
Solution	Ca	Fe	Ba	B	Zn	Na
Ca	1000	0.11	0.05	0.14	0.00	0.57
Fe	0.82	1000	0.025	0.27	0.08	0.58
Ba	0.10	1.22	1000	0.17	0.09	0.62
B	0.15	0.16	1.11	1000	0.20	1.14
Zn	0.17	0.11	0.13	49.3	1000	1.34
Na	0.23	0.87	0.16	15.74	1.67	1000
Wash Solution	0.49	0.32	0.24	0.20	0.57	0.74

Table 20 Concentration of elements in standard solutions and wash solution following typical use.

the original standard solutions were calculated by an one point calibration. The solution prepared from the stock solution for each element was used for the standardisation. It was assumed that the standard had undergone negligible dilution during previous use. The concentration of the contaminants are shown in Table 20, from which it can be seen that there is still significant contamination of the standards over one day's use. Also the contamination is distibuted over every solution, whereas if the contamination was from one standard to the next, then when the order of the addition of the elements is fixed, the contamination would be expected to be localised among a few solutions. Therefore, this indicates that contamination is probably through the wash solution. The wash solution was also analysed (calibration was by a one point calibration using the concentration of the elements in the prepared solutions). From Table 20 it can be seen that the concentration of the elements analysed in the wash solution is on the same order of magnitude as in the standards.

In order to minimise the contamination of solutions, fresh standard solutions were used each day and only a minimal amount of the solution was used so as to minimise contact of the solution with probe. Similary, a large amount of fresh distilled water was used each day so as to increase the dilution of the solutions washed from the probe. Also the probe was drawn through the wash solution ᵇy modulating the pitch angle of the wrist several (5) times. This helped to dislodge drops of solution adhering to the sides of the probe. A similar procedure was employed for rinsing the probe used in the measuring of the solutions in order to minimise carry over of solution from one sample to another.

In order for reliable results to be obtained from the measurements made on the solutions the uncertainty in the concentration of the solution should be no larger than the uncertainty in the measurement. From the calibration of the syringe pump it was found that the uncertainty in the volume of liquid dispensed was a constant value of about 0.8 µl for all volumes dispensed. Therefore the relative u::certainty of the concentration will increase with decreasing size of the volume of standard used. The noise characteristics of the ICP are known to be proportional to signal at a level of about 1%. Therefore the uncertainty of the concentration should be no larger than 1%. For a 0.8 µl uncertainty in the volume this suggests that the minimum volume dispensed should be 80 µl. A minimum volume of 100 µl was used to provide for some margin of error.

This procedure was tested by an Analysis of Variance (ANOVA) experiment. Several identical solutions were prepared and replicate measurements were made on each solution. The ANOVA analysis compares the amount of variation between the solutions to the variation within the set of replicate measurements of each solution. This is illustrated in Figure 40 which shows the results of the measurement of seven Ca solutions, all of the same concentration. Each point is the mean of six replicate integrations.

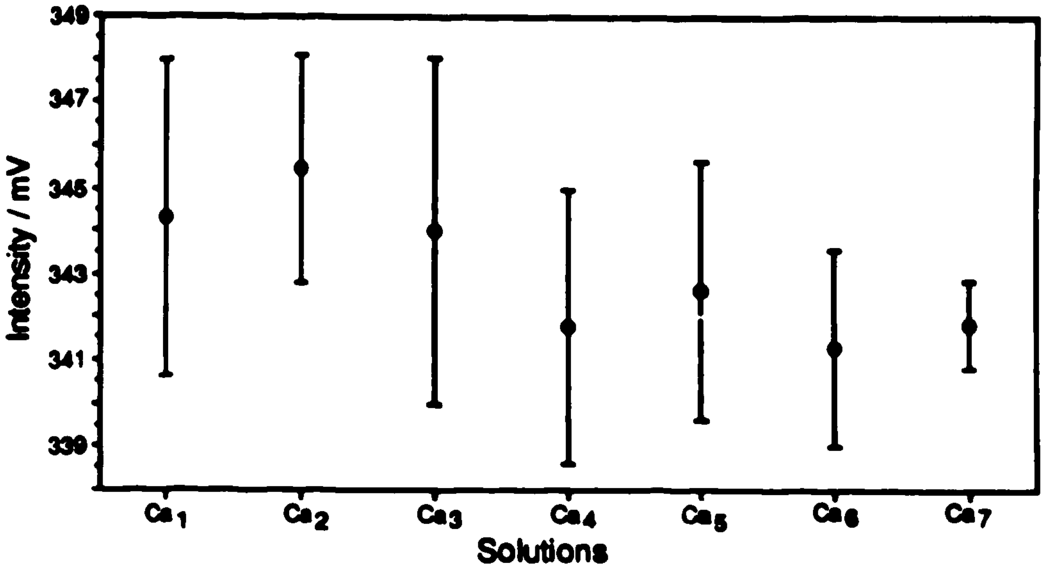


Figure 40 Scatterchart for seven calcium solutions. Error bars are for one standard deviation.

Analysis of Variance is used to compare the amount of variation between the means to the deviations. This is done by calculating the ratio of the variance between the solutions to the variance within the replicate measurements. This ratio is denoted F^*. The F^* ratio will increase as the solutions become more dissimilar; the critical value of F^* is given by F ratio tables for the required confidence level and degrees of freedom. If F^* is less than the critical value then there is no discernible difference between the solutions, whereas, if it is greater than the critical value then at least one solution is detectably different from the rest.

The experiment involved making several series of identical multi-element solutions, each series differing in the amount of standard used. The seven solutions each contained the elements; aluminium (1000µg ml^{-1} in 2% HNO_3 PLAL-2, Spex Industries Inc), calcium (1000µg ml^{-1} in HCl), iron

(1000μg ml^{-1} in aqueous solution), manganese (1000μg ml^{-1} in 2% HNO_3 PLMN-2), magnesium (1000μg ml^{-1} in 2% HNO_3 PLMG-2) and potassium (1000μg ml^{-1} in aqueous solution AQK2-500, Spex Industries). The volume of standards used were 10, 30 and 100μl. Once a series of solutions were made they were analysed by the ICP with a 90 second preflush and two sets of triplicate, 30 second integrations. The order in which the solutions were measured was randomised for each set of replicate integrations. The integrated readings were stored in a file and transferred to a Macintosh computer where the ANOVA analysis was performed. The main routine for the program used to perform the experiment is shown in Figure 41. A typical ANOVA analysis is shown in Figure 42 for one element (Ca). The number of degrees of freedom for this experiment were 6 and 35, however F tables only list values for 6 and 30 or 40 at the 95% confidence level Since the listed value of F(0.95, 6, 30) was less than that for F(0.95, 6, 40) the value of F(0.95, 6, 40) was used to determine if the solutions were identical to within the experimental error.

It would be expected that when small volumes of standard were used to prepare a solution there would be a greater likelihood that at least one solution would be prepared which is discernibly different from the rest, thereby causing a large value of F^*. However, when a large volume of standard is used then the likelihood of making all the solutions appear the same will be increased and so there will be less likelihood of a large F^*. Table 21 shows the values of F^* for each element in each solution; the value of F(0.95, 6, 40) is also shown in Figure 42. The values marked are above the critical value and so represent elements where at least one solution is

```
: ANOVA ( n - )
  2/
  ( Replicate measurements will be done in two
  groups )
  NUMBER.OF.MEAS !
  [ ARL ]
  CR ." Please enter filename " IN$
  MAKE-OUTPUT
  >FILE
  ." Anova Analysis" CR
  DATE. 9 EMIT TIME. CR        ( Date TAB time )
  9 EMIT LIST.ELEMENTS CR      ( Element symbols )
  NUMBER.OF.MEAS @
  INIT.MEASUREMENT
  ( Send measurement data to ARL )
  INIT.SAMPLE.SEQUENCE
  ( Store sample positions in order )
  RANDOMISE.SEQUENCE
  [ ROBOT ]
  FRWRD ARL.PROBE    ( ?ick up measuring probe )
  2 0
  DO
    SAMPLE.SEQUENCE
    7 0
    DO
      DUP
      I 2* I@
      ( Get position of next sample to measure )
      DUP
      1 1 ARL.SAMPLES 1 15 / 1+
                     ( Calculate sample number )
      .              ( Print sample number )
      START.MEASURE.SAMPLE
      INTEGRATE      ( Start measurement )
      FORCE.FLUSH
    ( Run pump at higher speed during pre-flush )
      NUMBER.OF.MEAS @ 0
      DO
        GET.DATA     ( Get intensity data )
      LOOP
    LOOP
    DROP             ( Address of sample sequence )
    RANDOMISE.SEQUENCE
    ( Measure samples in a different order )
  LOOP
  CLOSE-OUTPUT       ( Close file )
  [ ROBOT ]
  BCKWRD ARL.PROBE   ( Replace measuring probe )
  [ FORTH ] ;
```

Figure 41 Main routine of ANOVA experiment.

Volume/μl	Al	Ca	Fe	Mn	Mg	K
100	0.59	1.64	2.1	1.02	1.01	0.68
30	1.04	3.8*	1.26	0.80	0.60	12.9*
10	0.96	4.5*	10.4*	4.8*	1.18	5.5*

Table 21 Summary of F-ratios obtained fron ılysis of Variance.

measurably different from the rest. It can be seen that when 10 μl of standard was used, four of the six element concentrations were not correct. When 30 μl was used this was reduced to only two in six elements, showing that the probability of producing a correct solution has increased. Only when 100 μl was used was the error rate sufficiently low that all elements were prepared with the proper concentrations. From this result and the standard deviation data obtained from the calibration of the syringe pump the minimum volume of standard used in the preparation of solutions was set at 100 μl. This will ensure that the solutions will be correct to within the measurement error present in the ICP. This will also limit the dilution range that can be covered by a single step for 10 ml of solution to 1/100.

Sample No.	1	2	3	4	5	6	7
100μl	338.2	343.7	337.9	338.1	338.6	337.3	339.8
	346.9	347.3	341.8	340	339.5	343.4	342
	344.3	345.8	344.9	343.7	344.4	341.9	342.2
	341.9	341.3	342.8	339.3	342.6	340	342.2
	347.3	348.7	348.3	346.5	345.5	342.2	342.2
	347.1	346	348.4	343	345.3	343	342.8
Total	2065.7	2072.8	2064.1	2050.6	2055.9	2047.8	2051.2
Mean	344.3	345.5	344.0	341.8	342.6	341.3	341.9
Count	6	6	6	6	6	6	6
%R.S.D.	1.1	0.8	1.2	0.9	0.9	0.7	0.3
Sum Squares	711186	716083	710085	700827	704454	698914.	701237

ANOVA Table

	SS	df	MS
Between Samples	87.4	6	14.6
Within Samples	309.9	35	8.9
Total	397.3	41	
F*	1.64		
F(0.95,6,40)	2.34		

Figure 42 Typical results obtained from an ANOVA experiment.

3.3 Solution Preparation.

A basic method for making solutions has already been discussed. However there were some limitations to this method in that the minimum volume is not used, and that the volume of standards and diluent have to be calculated manually. This becomes more of a limitation when very low concentration solutions have to be made and the intermediate solutions of a serial dilution have to be calculated and defined as well.

To eliminate these limitations a procedure was defined which would allow the user to specify the elements to be included in the solution and the concentration of each element. The volume of the standards required and the volume of diluent would be calculated using a single step serial dilution if necessary and all the information required to prepare the solution would be recorded in a record in the memory space which had been reserved. If a component of the solution had to be diluted more than a hundred fold then an intermediate solution would be required. A definition of that solution would be prepared as well. There was an option that if one solution had already been prepared, a second solution with the same composition could be prepared. In this case the position of the receiving vessel would be changed so that the next empty vial in the sample tray (and also in the intermediate tray, if the intermediate solution was also to be remade) would be used. Once it has been established that a new solution is to be defined, the number of elements to be included in the solution is asked for. Any number less than eight is accepted (the system only has space for eight standard solutions). Once a valid value has been entered the total volume of solution is entered; the last volume used or 10 ml is offered as a default value. A maximum volume of 18 ml has been set so as to avoid overflowing the vials. The first element is then entered by typing the element symbol. These are limited to the elements that can be analysed by the spectrometer. A check is made to see whether a standard solution for that element is in the system, if not a message is displayed informing the user that this element cannot be used and another element should be entered. If an element that is in the system has been entered the concentration will be asked for. The numerical value would be entered followed by the units of the concentration. The concentration can be entered in w/v (%, ppm, ppb) or in molarity (M, mM or μM) units. A check is made to see whether the standard solution can be diluted to the concentration entered using a single intermediate solution; if not, a message is displayed and another concentration can be entered.

The first step in entering an element into the definition of a solution is to calculate the overall dilution ratio;

$$R = \frac{C_{Required}}{C_{Standard}} \qquad (1)$$

The volume of standard required is then given by

$$v = RV_T \qquad (2)$$

where V_T is the total volume of solution required, typically 10 ml. If v is greater than 100 µl then the dilution can be done in a single step. In this case v and the position of the standard solution are entered into the record for the definition of the final solution. Otherwise, it is necessary to go through a single step serial dilution.

The overall dilution after a single step serial dilution would be given by;

$$R = \frac{v_1}{V_T}\frac{v_2}{V_T} \qquad (3)$$

where v_1 is the volume of standard and v_2 is the volume of intermediate solution used in the second dilution. The variance in R is given by the propagation of errors as;

$$\left(\frac{\sigma_R}{R}\right)^2 = \left(\frac{\sigma_v}{v_1}\right)^2 + \left(\frac{\sigma_v}{v_2}\right)^2 \qquad (4)$$

It is assumed that V_T is large compared to v_1 and v_2 so that the relative uncertainty is negligible and it has been shown that σ_v is constant. v_2 is related to v_1 through R and so Equation 4 becomes;

$$\sigma_R^2 = R^2\left(\left(\frac{\sigma_v}{v}\right)^2 + \left(\frac{v\sigma_v}{RV_T^2}\right)^2\right) \tag{5}$$

Differentiating with respect to v;

$$\frac{d\sigma_R^2}{dv} = 2\left(\left(\frac{v}{R^2V_T^4}\right) - \left(\frac{1}{v}\right)^3\right) \tag{6}$$

Setting the derivative to zero and solving for v gives;

$$v = \sqrt{R}\,V_T \tag{7}$$

where v is the volume of standard and of the intermediate solution. For example, if R was 0.001 then a simple first approximation would be to dilute 1 ml of the stock solution into 10 ml of intermediate followed by 100 µl of intermediate solution into 10 ml of the final solution. The dilution could equally well be performed in the reverse order, i.e. 100 µl of stock solution and 1 ml of intermediate. From Equation 7 it would be better to use about 316 µl of stock solution and intermediate solution. This result for a two step dilution could be generalised for an n step dilution by using the nth root of the overall dilution ratio at each step of the dilution.

When more than one component is included in the intermediate solution; the volume of the intermediate solution (v_2) is determined by the component at the lowest concentration. Using the above example, if component 1 requires a dilution of 10^{-3} the volume of the intermediate solution used is 316 µl and the volume of stock solution is also 316 µl. If a second component is entered which requires a dilution of 10^{-4} then the volume of stock solution for component 2 would be 32 µl, (if the volume of intermediate solution remained unchanged). This is less than the minimum volume allowed, therefore the volume of intermediate solution must be reduced to 100 µl and the volume of the stock solution for component 2 required is also 100 µl. The reduction of the volume of the intermediate solution requires a proportionate increase in the volume of the stock solution for component 1 to 1 ml. So when a component with a lower concentration than those already defined is entered, the volumes of the other components have to be updated for the new dilution ratio. Since the volumes for standards are stored as integer values, round-off errors will result in a loss in accuracy if the updating procedure is repeated often. Under these conditions the concentrations in the final solution may not be as close to the desired concentration as would be possible if no updating had been required. In the above example the volume of component 1 would be 999 µl instead of 1 ml due to round-off error.

In order to avoid this readjustment the components present at lower concentrations should be entered first. Also, only 100, 200, 500, 1000, 2000 and 5000 µl volumes are allowed for the intermediate solution in the final dilution. When the ideal volume has been calculated, the allowed volume [illegible]est to this is found. Hence in our first example, instead of using 316 µl, 200 µ' of intermediate solution will be used. The volume of standard required [illegible]ain the desired overall dilution is calculated from Equation 3 and used [illegible]e definition of the intermediate solution. Now if a lower concentration component is entered there is less chance that a different volume of the intermediate solution will be used and so fewer updates of volumes will be necessary.

Once the volume of a component has been calculated and entered on the definition of the solution in which the standard is to be included (either the final or intermediate solution), then the concentration of the next component is entered. After all the concentrations of the various elements have been entered the definition of the intermediate solution, if it is to be used, is completed by calculating the volume of diluent required from the difference between the total volume of the standards used and the total volume of solution, as given by

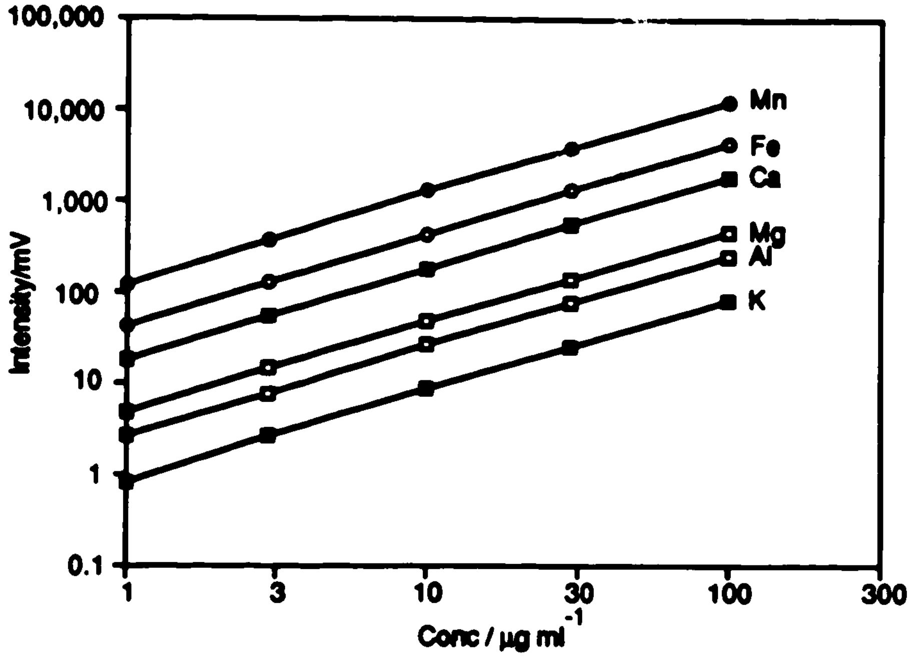
$$v_{diluent} = V_T - \sum v_{standard} \qquad (8)$$

Then the intermediate solution is entered on the definition of the final solution as another component by using the dilution ratio for the intermediate solution to calculate the volume of the intermediate solution to be used from Equation 2. This volume is entered into the definition along with the position where the intermediate solution is to be made. The definition of the final solution is completed by calculating the volume of diluent required using Equation 8. Once the solutions have been defined the intermediate solution, if it is being used, is prepared followed by the final solution.

Once a solution has been defined it can be remade by the same procedure. A check is first made to see whether a solution has already been defined. If one has been then the user is asked whether the solution is to be remade. If an intermediate solution was used then the user is asked whether it is also to be remade. When a solution is to be remade the position of the vial(s) used is the only part of the definition that is changed. The position of the next empty vial is obtained (if all the positions in the tray have been used a message is displayed asking the user to refill the tray) and entered into the definition. The solutions for the ANOVA study were made by this method, the minimum volume allowed was temporarily set to 10 µl. Once the definition of one solution had been entered, the other six solutions were just the same solution remade.

The routine described above was tested by preparing a multi-element calibration plot which covered about 2 orders of magnitude, with points at decade and half decade intervals. The lowest concentration used was close to the detection limit of the least sensitive element. The solutions prepared by the robot were compared to those prepared manually using the same standard solutions and diluted using traditional serial dilution methods where each step of the dilution reduced the concentration by one decade. Each solution contained the same elements as were used in the ANOVA study earlier. These elements were chosen because the sensitivity covers two orders of magnitude. Each element in a calibration solution was present at the same concentration. The solutions were measured by alternating between a manual solution and a robotic solution. A measurement of the blank was made between each pair of calibration solutions. The blank-subtracted measurements were then plotted on a log/log scale. The calibration plots, shown in Figure 43, are overlaid in Figure 44, from which it can be seen that the results from both sets of solutions are identical. There are only very slight mismatches in the plot of iron and possibly in the 3 $\mu g\,ml^{-1}$ aluminium

a)



b)

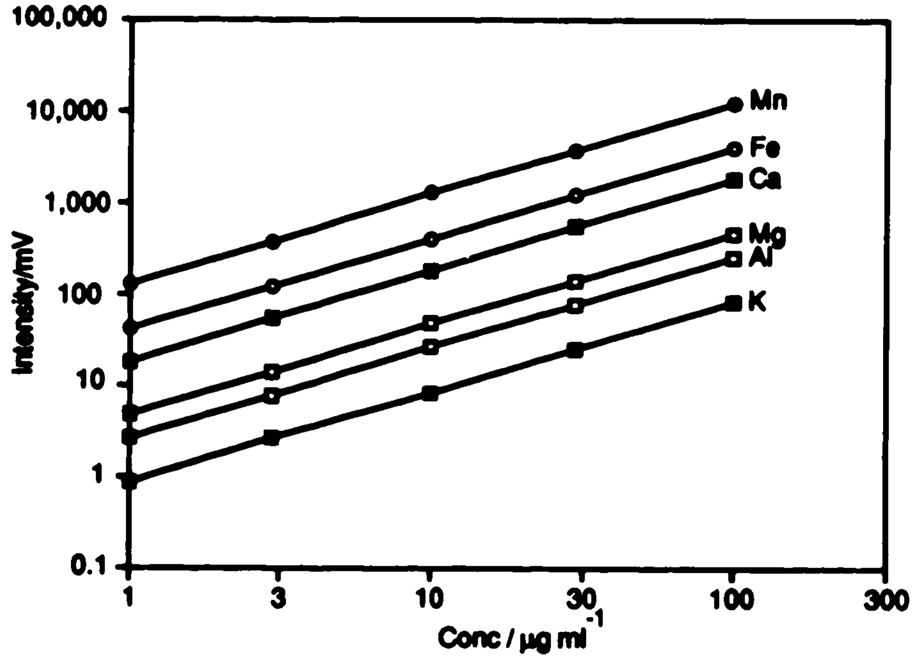


Figure 43 Calibration plots constructed using solutions prepared manually and using robotic methods.

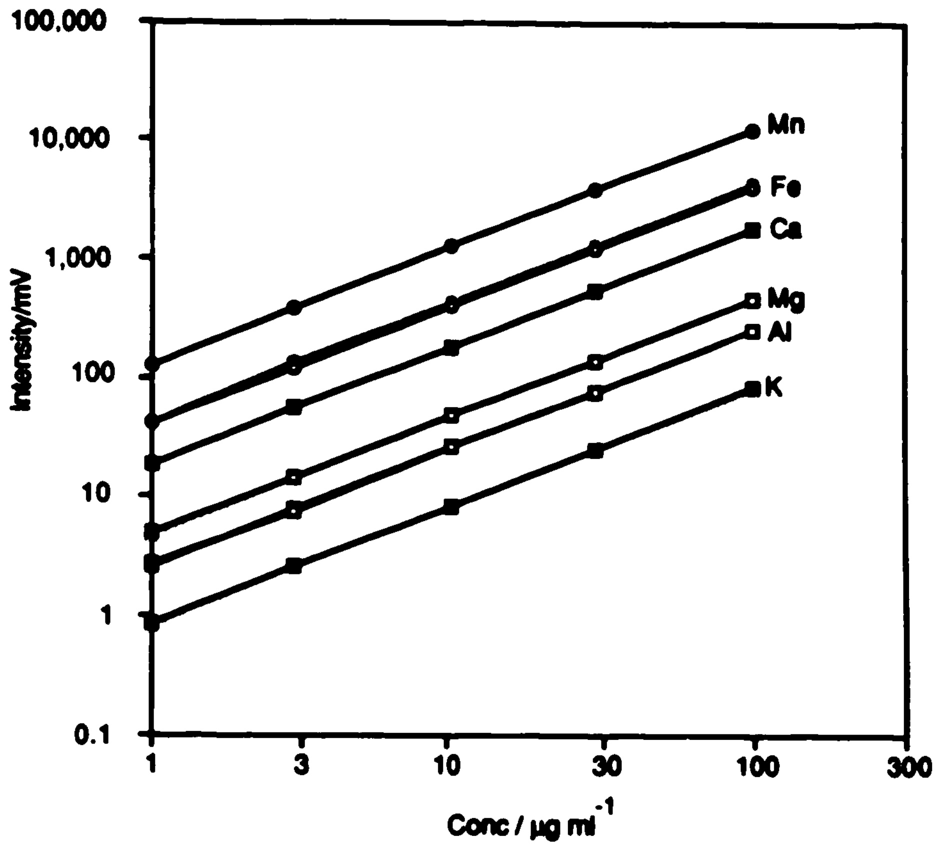


Figure 44 Overlay of calibration plots constructed using solutions prepared manually and using robotic methods.

solutions; otherwise the data points are exactly superimposed. The regression coefficients for the logarithmic plots are given in Table 22 from which it can be seen that the calibration curves obtained are indistinguishable and that the calibration curves are linear over the range covered.

From this it can be concluded that this solution preparation method was reliable. The only element that seemed to behave anomalously was iron, for which the robotic solution gave results that were systematically lower than those obtained from solutions prepared manually. This was reflected in the intercept of the logarithmic plot, where the two values do not agree to

	Al	Ca	Fe	Mn	Mg	K
Intercept						
Robot	0.42±0.009	1.26±0.002	1.62±0.005	2.11±0.004	0.68±0.005	-0.067±0.004
Manual	0.42±0.002	1.27±0.006	1.63±0.004	2.10±0.006	0.68±0.004	-0.077±0.011
Slope						
Robot	0.99±0.019	1.00±0.004	0.99±0.010	1.00±0.010	0.99±0.010	0.99±0.008
Manual	0.99±0.004	1.00±0.012	1.00±0.009	1.00±0.013	0.99±0.009	1.00±0.022

Table 22 Regression coefficients for logarithmic calibration plots of various elements using calibration solution prepared by manual and robotic methods.

within one standard deviation. However, the difference is not significant at the 95% confidence level. The discrepancy could be due to a small error in the manual solution preparation, since if a small additional amount of iron standard were included in the initial calibration standard, then this would result in all the other standards being proportionately high.

Chapter 4

Automation of Analysis.

It has been shown that the system can reliably produce solutions over a wide range of concentrations using serial dilution methods. So far the solutions produced have been defined by the user. The next step is to tie the definition of the solution to the analysis procedure and have the composition of the solution decided by the analysis program.

The analysis procedure should be designed to produce the best possible precision and accuracy in the minimum time and effort. One possible method for performing the analysis is to do an one point calibration where a single standard of known concentration is measured and the concentration of the sample is calculated by multiplying the concentration of the standard by the ratio of the net emission intensities of sample and standard. This method is quick and can be used for a large number of samples. However, it requires that all the samples be close in composition along with some knowledge of the likely concentration of the samples so that the standard can be closely matched to the samples.

If more precision is required, then more calibration standards could be run and a calibration plot constructed. If the approximate composition of the samples is known, then the standards can be chosen so that the calibration plot brackets the samples. However, if the composition of the samples is completely unknown, then the calibration plot needs to cover a wider range of concentrations to ensure that the samples are bracketed.

In the case of the ICP this may require the concentration to cover several orders of magnitude. The wide dynamic range of the signal and noise characteristics in the ICP means that linear least squares methods cannot be used to estimate the sample composition. Linear least squares methods assume that the uncertainty is independent of the signal, whereas in the ICP the uncertainty in the signal increases linearly with the signal [50]. The use of weighted least squares could be used to correct for the non-uniformity of the uncertainties, though another problem would remain due to the wide dynamic range of the ICP. The wide range of concentrations used in the calibration mean that the results of linear regression tend to be determined by the standards with the higher concentrations. Using a log/log plot gives more equal significance to each point in the calibration plot, also the uncertainties of each data point become equal so that least squares methods can be used to perform the regression.

An alternative approach to the above would be to obtain an initial estimate of the sample composition and then to limit the concentration range of the calibration standards. In this way the uncertainty in each data point will be more closely equal and the weighting of each point in a least squares regression will be equal. Also, it can be applied to measurement systems which exhibit significant curvature if the range of the calibration plot extends too far. The initial estimate of sample concentration can be obtained from a wide dynamic range calibration plot, the regression parameters of which have been stored in a file and used as a semi-quantitative calibration of the instrument. As long as the results obtained using this data are within an order of magnitude of the actual concentration then the calibration should be able give reliable results. This holds specially if the plot is constructed in an iterative manner and the calibration plot is used to estimate the sample composition after each calibration point is added. The current estimate can then be used to calculate the concentration of the standard which leads to the greatest improvement in the calibration plot in terms of reducing the uncertainty in the sample composition.

The intensity of emission from an ICP is a linear function of concentration. The calibration is performed by measuring the emitted intensity for a series of standards of known concentration. Least squares regression is then used to estimate the equation of the line. This equation is then solved for concentration at the intensity of the samples. The uncertainty in the concentration is then given by [51];

$$s_{Sam} = \frac{s_{y/x}}{b}\left[(1/m + 1/n) + \frac{(\bar{I}_{Sam} - \bar{I}_{Std})^2}{b^2 (\sum c_{std}^2 - (\sum c_{std})^2/n))} \right]^{1/2} \qquad (1)$$

where n,m are the number of standards in the calibration plot and the number of replicate measurements made on the sample respectively, $\bar{I}_{Std}$ and $\bar{I}_{Sam}$ are the average intensities of all the standards and the sample, $s_{y/x}$ is the standard deviation of the calibration points about the line. The factors which influence the precision of the analysis are therefore the number of replicate measurements (m) and the number of points used to construct the plot (n), the closeness of the sample to the centre of the calibration plot (the numerator in the last term), the concentration range of the standards (the denominator of the last term) and the amount of scatter present in the data points ($s_{y/x}$).

A typical calibration plot with 5% noise in the data points is shown in Figure 45, along with 95% confidence bands. These bands can be used to

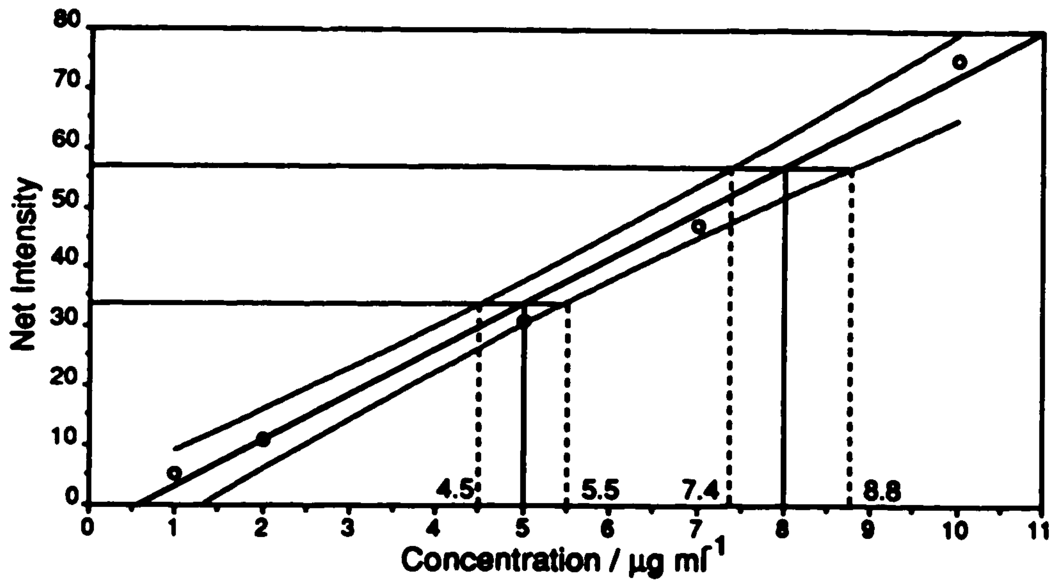


Figure 45 Example of a typical linear calibration plot with 95% confidence bands.

estimate the uncertainty in the concentration as shown [52]. The further the sample is from the centre of the plot the larger will be the uncertainty, hence in order to reduce uncertainty the calibration plot should ideally be constructed so that the sample will lie close to the centre.

To design a procedure which will produce such an optimised calibration plot, it is necessary to have an estimate of the concentration of the sample. This can be done by iterating the construction of the plot, using the existing results to calculate the optimal concentration of the next standard. The goal is to reduce the uncertainty in the estimated concentration of the sample. Therefore the concentration of each succeeding standard should be at the minimum of a plot of the uncertainty in the concentration of the sample after the next standard has been added to the calibration plot against the

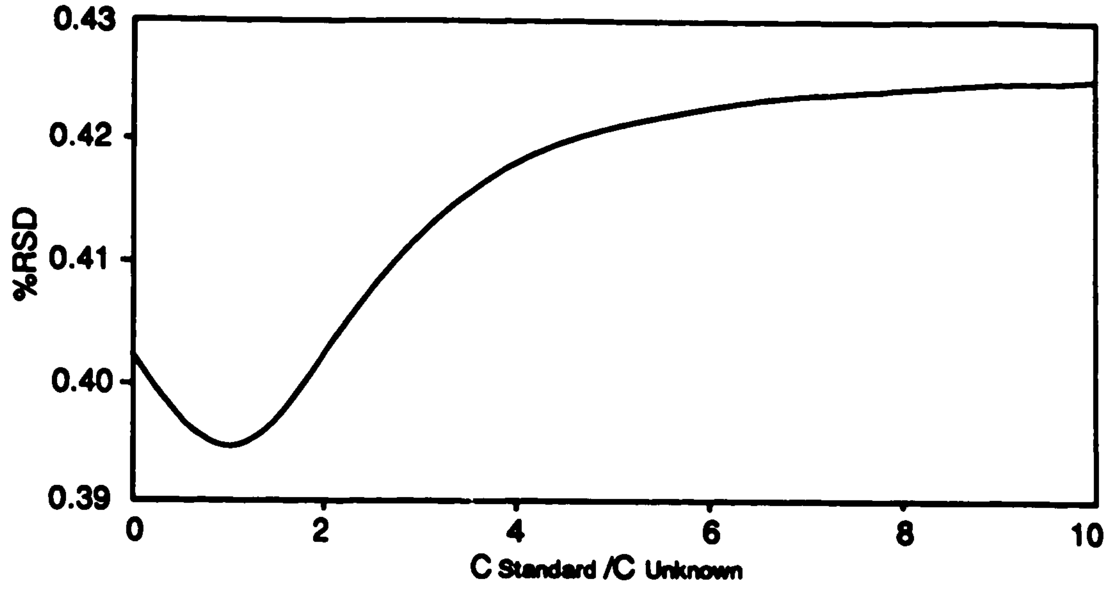


Figure 46 Plot of uncertainty in estimated sample concentration as function of relative concentration of next standard to be added to a calibration.

concentration of the next standard. From Equation 1 s_{sam} can be calculated for the next standard by updating the summations with the concentration of the next standard and the intensity of emission expected on the basis the estimated values of slope and intercept. A plot of the predicted uncertainty against the concentration of the next standard is shown in Figure 46. The minimum for adding the fifth data point in a calibration plot is found to be close to unity. This indicates that once the calibration plot is started the ideal standard to use is one which is equal in concentration to that of the sample.

This can be explained by considering the term $(\bar{I}_{Std} - \bar{I}_{Sam})^2$ in Equation 1. This term can be replaced by $b^2(\bar{c}_{Std} - \hat{c}_{Sam})^2$, where $\bar{c}_{Std}$ is the average concentration of all the standards used in the construction of the calibration plot, and $\hat{c}_{Sam}$ is the current estimate of the concentration of the sample. The ideal concentration c of the next standard would reduce this term to zero and so would satisfy the relation

$$\hat{c}_{Sam} = \frac{n\bar{c}_{Std} + c}{n + 1} \qquad (2)$$

If this term is already close to zero, then

$$n\hat{c}_{Sam} \approx n\bar{c}_{Std}$$

This implies that $c \approx \hat{c}_{Sam}$.

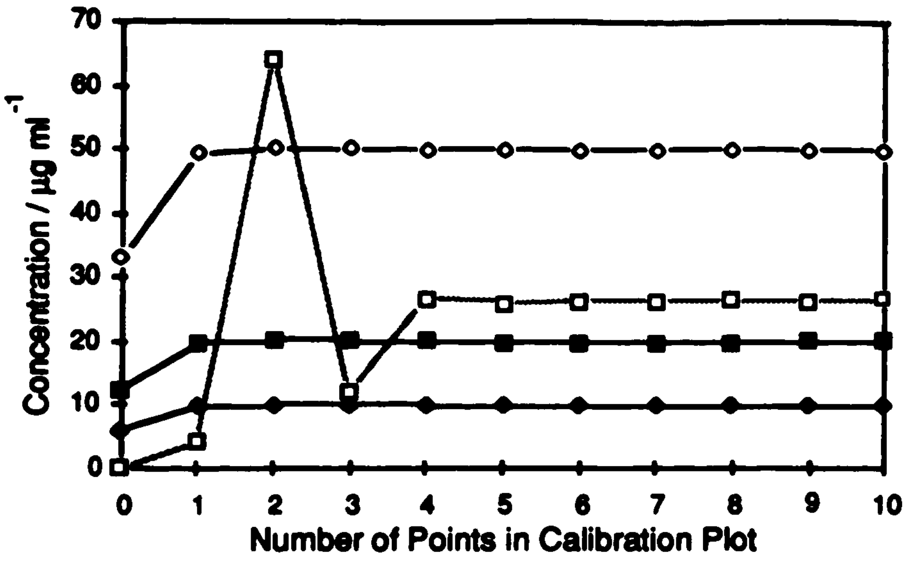
Hence, if Equation 2 were used to calculate the concentration of the standards in an iterative analysis procedure, then, when analysing a single sample, the concentration of the standards used at the end of the analysis would be equal to the concentration of the sample. However, it is usually necessary to analyse several samples simultaneously, and so it is not desirable to have to construct a calibration plot for each sample, especially if the composition of the samples are very similar. Ideally a single calibration plot should be designed for the analysis of a set of samples. Equation 2 will find the ideal set of standards for a single sample but in order to analyse several samples a compromise set of standards needs to be found.

Equation 2 was designed to minimise the $(\bar{I}_{Std} - \bar{I}_{Sam})^2$ term in Equation 1. The compromise set of standards would minimise the sum of these terms for all the samples because only this term varies for each sample. This condition is met when $\bar{I}_{Std} = \bar{\bar{I}}_{Sam}$, where $\bar{\bar{I}}_{Sam}$ is the average of the average intensities of all the samples. The optimal set of standards will therefore be found when $\bar{\hat{c}}_{Sam}$ is used in Equation 2, where $\bar{\hat{c}}_{Sam}$ is $\hat{c}_{Sam}$ averaged over the samples.

To check whether this procedure can be used to successfully perform an analysis, a computer simulation of the experimental procedure was performed. The simulation used regression parameters obtained from a calibration for the elements which are used in later experiments. The concentrations of "samples" were the same as those of the solutions used to test the actual analysis routine. Also, the initial estimates for the simulation used the same semi-quantitative calibration as the actual experiment. Normally distributed noise, at a level of 1%, was included in the simulated "signals".

The simulation generated "signals" for the "samples". A semi-quantitative calibration was used to obtain the initial estimate and the first standard was at a concentration 30% below the lowest sample concentration. This standard was used to perform a one point calibration, from which new estimates of the sample composition were obtained. The second standard, at a concentration 30% above the highest sample concentration, was used in a two point calibration and new estimates of sample composition were calculated. The iteration starts by calculating the concentration of the next standard using the modified version of Equation 2, as described earlier. A least squares procedure is used to estimate the sample composition and the uncertainties in the composition. This iteration was repeated ten times, and the resulting sample composition and uncertainties were plotted against the number of points in the calibration plot as shown in Figure 47.

a)



b)

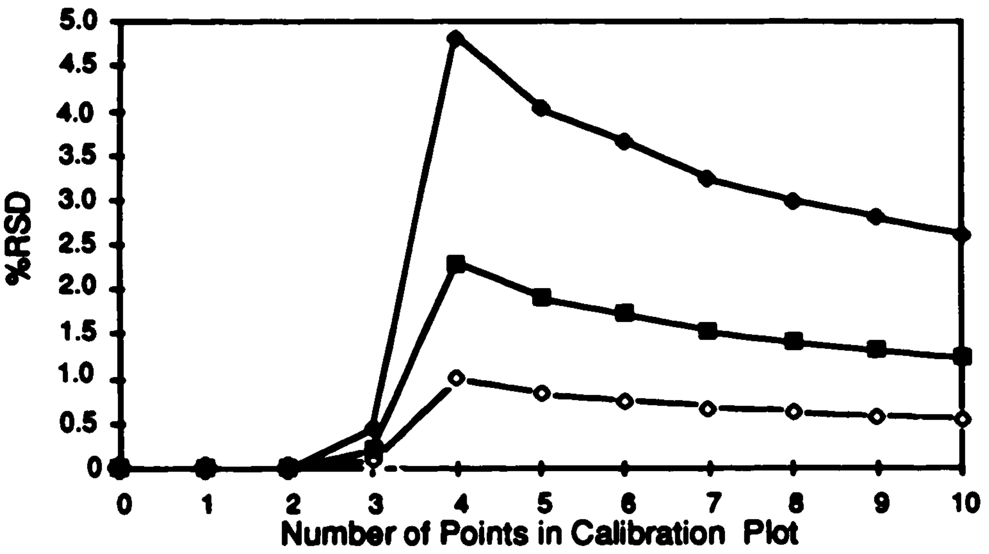


Figure 47 Plots of a) estimated composition of test 1 (◆), test 2 (◇), test 3 (■) and standard (□). b) Uncertainty in sample composition as function of number of points in calibration plots for simulation of automated analysis procedure.

	Concentration/μg ml^{-1}					
	Ba	Fe	Zn	B	Ca	Na
Sample 1	0.050±9%	1.00±0.8%	0.094±8%	0.18±7%	9.8±3%	51±3%
Sample 2	0.02±22%	0.491±1%	0.210±3%	0.50±3%	50.0±0.5%	100±1%
Sample 3	0.029±15%	0.10±8%	0.50±2%	0.31±4%	19.9±1%	197±0.7%

Table 23 Summary of "analytical" results obtained from simulation of automated analysis procedure.

The estimated concentration of the samples very quickly comes to an essentially constant value as shown in Figure 47a, whereas the uncertainties decrease with the number of points in the calibration curve (though the rate of decrease is less when more points are in the plot as can be seen from Figure 47b). From these simulations it was found that acceptable results could be obtained from this procedure. The precision that could be expected was also as predicted, the relative standard deviation being found to be less for the highest concentration sample, as might be expected if the noise in the signal is approximately constant for all samples (an assumption implicit in least squares methods). A summary of the results that could be expected is given in Table 23

4.1 Development of Automated Analysis Procedure.

The solution preparation program defined previously, in Chapter 3, was found not to be general enough. Two problems with it were that the concentration range was insufficient and that the use of a single intermediate solution led to problems when two components were at opposite ends of the concentration range for the intermediate solution.

ICP-AES has a dynamic range of up to five orders of magnitude, with detection limits going down to the level of ng ml^{-1}. Therefore it may be necessary to prepare solutions at the level of tens of ng ml^{-1}. The present solution preparation procedures with only one intermediate solution could only prepare solutions to 100 ng ml^{-1} from a 1000 µg ml^{-1} stock solution. Therefore an extra intermediate step needs to be included in the solution preparation routines. Another problem arises from the fact that only one intermediate solution is used for components whose concentrations may differ by up to two orders of magnitude. This gives rise to a problem when the concentrations of two components lie at opposite ends of this range. For example, if a solution required component 1 to have a concentration of 100 ng ml^{-1}, component 2 to have a concentration of 9 µg ml^{-1} and a total volume of 10 ml from stock solutions of 1000 µg ml^{-1}, then the intermediate solution would include 100 µl of component 1 and 9 ml of component 2, leaving only 900 µl for a third component if needed. This severely limits the generality of the routine.

To extend the dynamic range of the routine and to improve the generality, four intermediate solutions were allowed; the concentration range of the components of each solution being limited to only one order of magnitude. For example, if the solution to be prepared had four components which required dilution factors of 10^{-6} for component 1, 10^{-5} for component 2, 10^{-4} for component 3 and 10^{-3} for component 4 then the preparation of this solution would proceed as follows (all solutions have a total volume of 10 ml). Firstly, 100 µl of component 1 would be included in intermediate solution 4 and 250 µl of component 2 would be included in intermediate solution 3. Secondly, 100 µl of intermediate solution 4 and 100 µl of component 3 are included in intermediate solution 2. Similarly, 200 µl of intermediate solution 3 and 500 µl of component 4 are included in intermediate solution 1. Finally, 200 μl of intermediate solution 2 and 100 μl of intermediate solution 1 are included in the final solution. The calculation of the volumes of standards and intermediate solutions used at each stage of the dilution is an extension of that used previously. The ideal volume is calculated so that the dilution factor of each stage is equal. For a two step dilution, such as for components 1 and 2 in the example above, the dilution ratio at each stage is the cube root of the overall dilution ratio (for component 1 this is 10^{-2} and for component 2 this is 2.15×10^{-2}). Once the ideal has been calculated, the closest allowed volume of intermediate is found (100 μl of intermediate 2 and 200 μl of intermediate 1), and from the dilution factor for this volume the dilution ratio remaining is calculated (10^{-4} for component 1 and 5×10^{-4} for component 2). From this ratio the volumes for the first step of the dilution are calculated exactly as before. The overall dilution ratio is used to calculate which of the four intermediate solutions will be used to make the initial dilution of the standard. With this modified solution preparation procedure the system will be able to prepare solution of arbitrary concentration, as would be required when the composition of the standards is being calculated during the analysis procedure.

To start the analysis it was necessary to obtain an initial estimate of the composition of the samples. This was done by recording a wide dynamic range calibration of most of the spectral channels of the spectrometer in a disk file. The data in this file was read at the start of the analysis procedure and once the samples had been measured, it was used to get the initial estimate required.

The calibration routine used to prepare the semi-quantitative calibration data prepared a fixed set of standards for the elements being calibrated. Blank subtracted measurements were made and the regression coefficients of a log/log plot, using natural logarithms, were stored. The user could specify the elements to be calibrated, the initial concentration of the calibration, the concentration range to be covered (specified by the number of decades to be calibrated) and the number of data points to be included in each decade; the sequence of data points could be chosen to be one of the following sets (1,10), (1,3,10) or (1,2,5,10). This allows the calibration curve to be prepared from close to the detection limit of the element to the highest concentration that could be prepared.

The elements shown in Table 24 were calibrated. These could be divided into two groups according to the detection limit. Those of high sensitivity and low detection limit were calibrated over the concentration range 0.1 to 100 $\mu g\ ml^{-1}$ using the (1,3,10) sequence of concentrations. Those of lower sensitivity were calibrated over the range 1 to 100$\mu g\ ml^{-1}$ using the (1,2,5,10) sequence of concentrations. The regression coefficients for the log/log plots are also shown in Table 24. The slope for most elements was found to be close to unity, indicating that the intensity is a linear function of concentration over the range of the calibration plot prepared (as expected for an ICP). Elements for which the slope differed significantly from unity (>5%) were recalibrated. The results of the second calibration were within 1% of the first calibration. Therefore, the calibration results were retained.

To test that the semi-quantitative calibration using this data would yield a reasonable estimate of concentration over the long term, test samples from the analysis routine developed later were used. The samples were measured over a period of approximately two months and the estimated concentrations recorded. The mean and standard deviation of these

Element	Intercept	Slope
Zr	-1.09	1.12
Ba	1.34	1.01
Ni	-0.75	0.99
Al	-2.6	0.96
B	-0.21	1.01
Mn	1.26	0.99
Fe	0.176	0.98
Hg	-0.34	0.80
Mg	-2.0	0.99
As	-0.98	0.98
Sn	-1.27	1.07
V	-0.40	1.13
Na	-2.9	0.99
Cr	0.26	1.01
Sb	-1.74	1.02
Ge	-2.1	0.97
Ca	-0.65	0.94
Zn	1.09	0.99
Cu	0.092	0.99
Pb	-2.0	0.99
Li	-0.23	1.00
Ti	-0.17	1.03
Cd1	0.66	1.00
In	-1.30	1.00

Table 24 Summary of calibration results for wide dynamic range calibration of spectrometer.

concentrations give an idea of the accuracy and precision that could be expected.

The analysis procedure was to measure the blank followed by the samples. The logarithm of the blank-subtracted intensity was then used to calculate the concentration of each sample from the data stored in the calibration file. The concentrations obtained were then entered in a table of results with the date of analysis. This table was used to construct a set of quality control charts. The chart for Ca is shown in Figure 48, the mean and

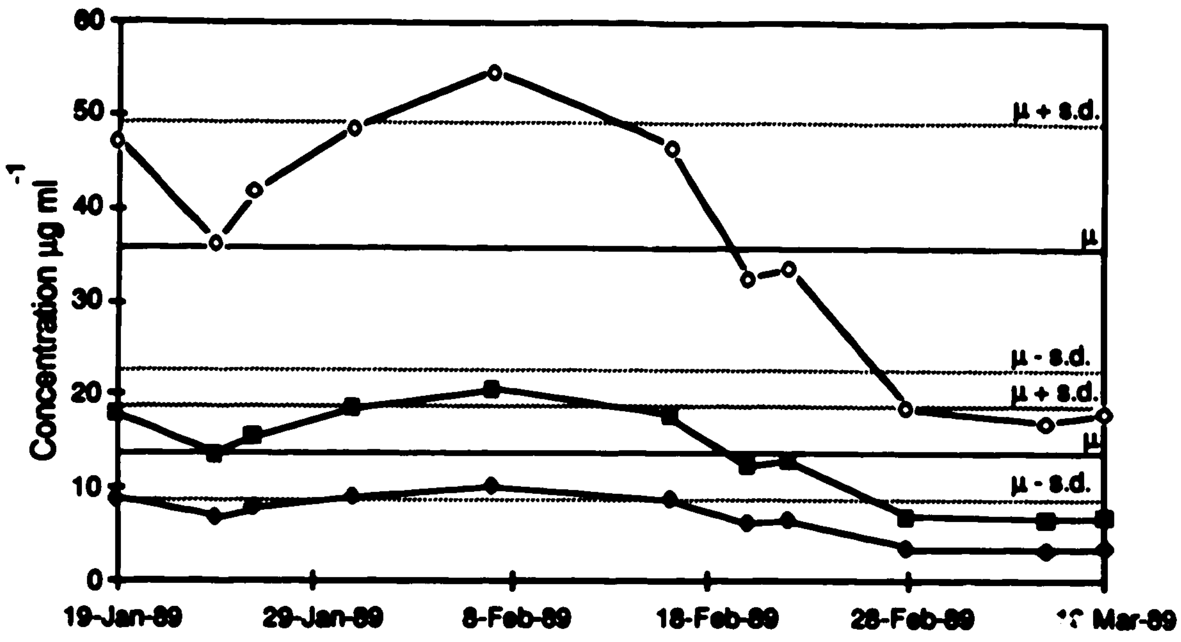


Figure 48 Typical quality control chart for semi-quantitative analysis [illegible] st samples Sample 1 (◆), Sample 2 (○) and Sample 3 (■).

one standard deviation limits for the lowest sample are not shown for clarity. These charts show the mean value of the measured concentrations, from which systematic error can be determined. From the way the pattern of the variation in the measured concentrations relate to each other, it can be determined if the measured concentrations are in the correct proportions to each other.

The actual concentrations of Ca in these solutions were 10, 20 and 50 µg ml^{-1}. The mean values obtained over the period studied were 7, 14 and 36 µg ml^{-1}. The relative standard deviation obtained was 37%. Therefore this semi-quantitative calibration can be used to obtain an estimate of the composition of the samples which will remain within an order of magnitude of the true value, as is required for the automated analysis procedure, for an extended period. The current calibration tends to give results which underestimate the true composition; this was found to be true for all the elements in these samples.

The underestimation of the semi-quantitative calibration is probably due to a loss in sensitivity between the time of the calibration and the start of the analyses. It was also found that the pattern for each of the samples showed the same trends, suggesting that these were a result of variations in the sensitivity of the measurement.

The analysis procedure was the same as that used in the simulations done earlier. The sample solutions were measured first, after which a semi-quantitative estimate of the sample composition was made from the blank-subtracted intensities. Since the semi-quantitative estimate was always lower than the true composition, the first standard was prepared at a concentration 30% below that of the lowest sample and used in a one point calibration. Using the improved estimate, which will not be as biased as the initial estimate, a second standard was prepared at a concentration 30% above the concentration of the highest sample. These two standards should now bracket all the samples. The factor of 30% used will ensure that the highest and lowest samples are well removed from the ends of the calibration plot, where the uncertainty in the measured concentrations is increasing. If the semi-quantitative analysis had shown a bias in the opposite direction the standards would have been prepared in the opposite order, i.e. the first standard would be prepared at a concentration 30% above the highest sample and the second at a concentration 30% below the lowest.

a)

```
50 Print "Auto Calibration Analysis":  Rem Print program
                                title
60 Gosub 4500:                  Rem Get elements
70 Gosub 1000:                  Rem Get Preflush
80 Gosub 1050:                  Rem Get Integration time
90 Gosub 1100:                  Rem Get Replicates
   .
   .
130 Gosub 1500:                 Rem Measure samples
140 Let N3 = 0:                 Rem Zero standards counter
150 Gosub 3200:                 Rem Calculate means
155 Gosub 3000:                 Rem Calculate sample
                                concentrations
160 Gosub 4300:                 Rem Send samples composition
165 Entry Repeat:               Rem Entry point to analysis
                                loop
170   N3 = N3 + 1:              Rem Increment counter
180   Gosub 3300:               Rem Calculate std.
                                composition
190   If N3 = 1 Then Gosub 3100:        Rem Bubble sort
200   Gosub 4000:               Rem Send std composition
210   Gosub 4200:               Rem Get std composition
220   Gosub 2000:               Rem Blank measure
230   Gosub 2500:               Rem Measure solution
240   Gosub 5000:               Rem Update summation
                                variables
250   Gosub 5500:               Rem Calculate sample
                                composition
255   Gosub 6500:               Rem Send regression data
260   Gosub 4300:               Rem Send sample composition
265   Gosub 6000:               Rem Check done
270 If R1 <> 1 Then 170:        Rem Repeat until done
280 Goto 30:                    Rem return to start of
                                program
```

b)

```
: AUTO.ANALYSIS
   GET.SAMPLES.INFO
   GET.FILE
   INIT.ARL
   MEASURE.SAMPLES
   BEGIN
     MAKE.STANDARD
     MEASURE.STANDARD
     GET.REGRESSION.DATA
     GET.SAMPLES.RESULTS
     ?DONE
   UNTIL ;
```

Figure 49 Partial listings of programs used for automated analysis procedure. For a) ARL spectrometer and b) IBM-AT

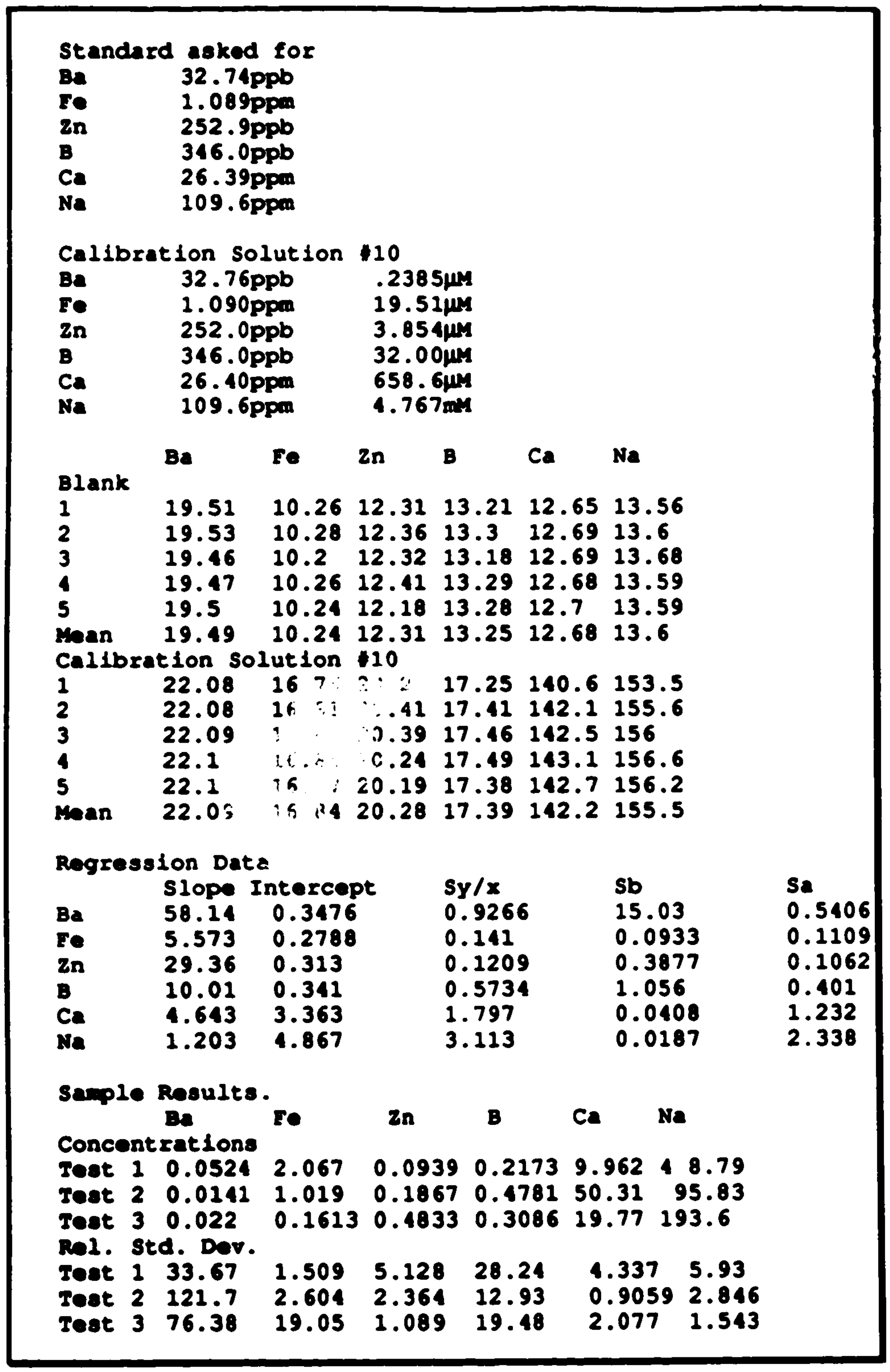
Concentration/$\mu g\ ml^{-1}$						
	Ba	Fe	Zn	B	Ca	Na
Sample 1	0.05	1	0.1	0.2	10	51.9
Sample 2	0.02	0.5	0.2	0.5	50	101.5
Sample 3	0.03	0.1	0.5	0.3	20	198.7

Table 25 Composition of test samples used to verify automated analysis procedure.

Subsequent standards were calculated using the modified version of Equation 2. Estimation of the sample composition was by linear least squares. The chemical analysis would terminate either when the relative standard deviation of all components for all samples was below some user specified value or when a certain number of standards had been prepared. A partial listing of the programs used are shown in Figure 49. The program in Figure 49a was run on the ARL spectrometer whilst that in Figure 49b was run on the IBM-AT.

The composition of the test samples is given in Table 25. These were prepared in 1 l batches from 1000 $\mu g\ ml^{-1}$ standards by a single step dilution, except for Na which was added as solid NaCl (Analytical reagent grade, BDH Chemicals, Toronto). The standard solutions used for calibration were all 1000 $\mu g\ ml^{-1}$ except for Fe, which was 500 $\mu g\ ml^{-1}$. (However, the value of 1000 $\mu g\ ml^{-1}$ was erroneously entered in the configuration file and used by the analysis routine in the calculations.)

An example of a typical record for one calibration point is shown in Figure 50. The first entry is the ideal composition of standard required. The standard was prepared and the actual composition recorded. Actual



```
Standard asked for
Ba        32.74ppb
Fe        1.089ppm
Zn        252.9ppb
B         346.0ppb
Ca        26.39ppm
Na        109.6ppm

Calibration Solution #10
Ba        32.76ppb      .2385µM
Fe        1.090ppm      19.51µM
Zn        252.0ppb      3.854µM
B         346.0ppb      32.00µM
Ca        26.40ppm      658.6µM
Na        109.6ppm      4.767mM

          Ba      Fe     Zn     B      Ca     Na
Blank
1         19.51   10.26 12.31 13.21 12.65 13.56
2         19.53   10.28 12.36 13.3  12.69 13.6
3         19.46   10.2  12.32 13.18 12.69 13.68
4         19.47   10.26 12.41 13.29 12.68 13.59
5         19.5    10.24 12.18 13.28 12.7  13.59
Mean      19.49   10.24 12.31 13.25 12.68 13.6
Calibration Solution #10
1         22.08   16 [illegible] [illegible] 17.25 140.6 153.5
2         22.08   16[illegible] [illegible].41 17.41 142.1 155.6
3         22.09   [illegible] [illegible]0.39 17.46 142.5 156
4         22.1    [illegible] [illegible]0.24 17.49 143.1 156.6
5         22.1    16[illegible] 20.19 17.38 142.7 156.2
Mean      22.0[illegible]   16.[illegible]4 20.28 17.39 142.2 155.5

Regression Data
          Slope Intercept       Sy/x          Sb            Sa
Ba        58.14   0.3476        0.9266        15.03         0.5406
Fe        5.573   0.2788        0.141         0.0933        0.1109
Zn        29.36   0.313         0.1209        0.3877        0.1062
B         10.01   0.341         0.5734        1.056         0.401
Ca        4.643   3.363         1.797         0.0408        1.232
Na        1.203   4.867         3.113         0.0187        2.338

Sample Results.
          Ba      Fe       Zn      B       Ca     Na
Concentrations
Test 1  0.0524  2.067   0.0939 0.2173 9.962 4 8.79
Test 2  0.0141  1.019   0.1867 0.4781 50.31  95.83
Test 3  0.022   0.1613 0.4833 0.3086 19.77 193.6
Rel. Std. Dev.
Test 1  33.67   1.509   5.128   28.24   4.337  5.93
Test 2  121.7   2.604   2.364   12.93   0.9059 2.846
Test 3  76.38   19.05   1.089   19.48   2.077  1.543
```

Figure 50 Typical record for one calibration point of automated analysis procedure.

	Concentration/μg ml^{-1}					
	Ba	Fe	Zn	B	Ca	Na
Standard 1	0.0076	0.021	0.023	0.074	2.4	27
Standard 2	0.070	1.15	0.46	0.75	62	250
Standard 3	0.041	2.2	0.28	0.28	17	71
Standard 4	0.00	1.09	0.26	0.121	26	114

Table 26 First four standards for analysis of three samples.

compositions were typically within 0.5% of those originally chosen. Before the first standard was prepared the order of the elements in the definition of the standard were sorted from low concentration to high, this minimised the record updating that could occur in the definition of the intermediate solutions if the order was left random. This sorting also ensured that actual composition was close to that required by reducing the round off error that could occur when the intermediate solution records were updated. A distilled water blank was measured (5x30 second integrations were recorded and the average was calculated). The standard was measured and the value of the blank subtracted. The results from a linear regression, for this and all previous standards, were recorded, values calculated for the slope and intercept, along with the standard deviations of each (sb and sa respectively). The standard deviation of the data about the line $\sigma_{y/x}$ (syx) was also calculated. Finally, the current estimate of the sample composition with the relative standard deviation was recorded. Each record was time stamped so that the time required to perform the analysis could be estimated.

From these records a complete picture of how the analysis develops can be made. The data can also be used for diagnostic purposes if any problems should arise. The composition of the first four standards is shown in Table 26.

From the table it can be seen how the calibration plot for each element was customised to the amount present in the samples. If the concentration required was less than 1 ng ml^{-1}, then that element was omitted from the solution and a concentration of zero was used in the least squares calculations.

A calibration plot for Ca constructed from this data is shown in Figure 51. The first data point is at the lowest concentration on the plot and can be seen to be well below the lowest sample. Similarly the highest point on the plot is well above the highest sample so the calibration plot does properly bracket all the samples as intended. The third calibration point lies between the lowest standard and the swarm of data points in the centre of the plot. This point will always tend to be at the low end of the plot since the semi-quantitative calibration underestimates the concentration of the samples so the first calibration point tends to be lower than necessary. The average of the first three data points should equal the average of the samples, so the third point will be below this in order to compensate for the underestimate in the first standard. Once these standards have been prepared, the rest of the standards tend to be close to the average concentration of the samples; this can be seen from the swarm of points around 27 µg ml^{-1} shown in the scale expanded plot.

The estimate of the concentration of Ca as a function of the number of points in the calibration plot and the uncertainty in that estimate are shown in Figure 52. The same general trends in these plots were found in the simulations except that the uncertainty shows an increase after 7 points in the calibration plot. This increase is probably due to instrumental drift. A

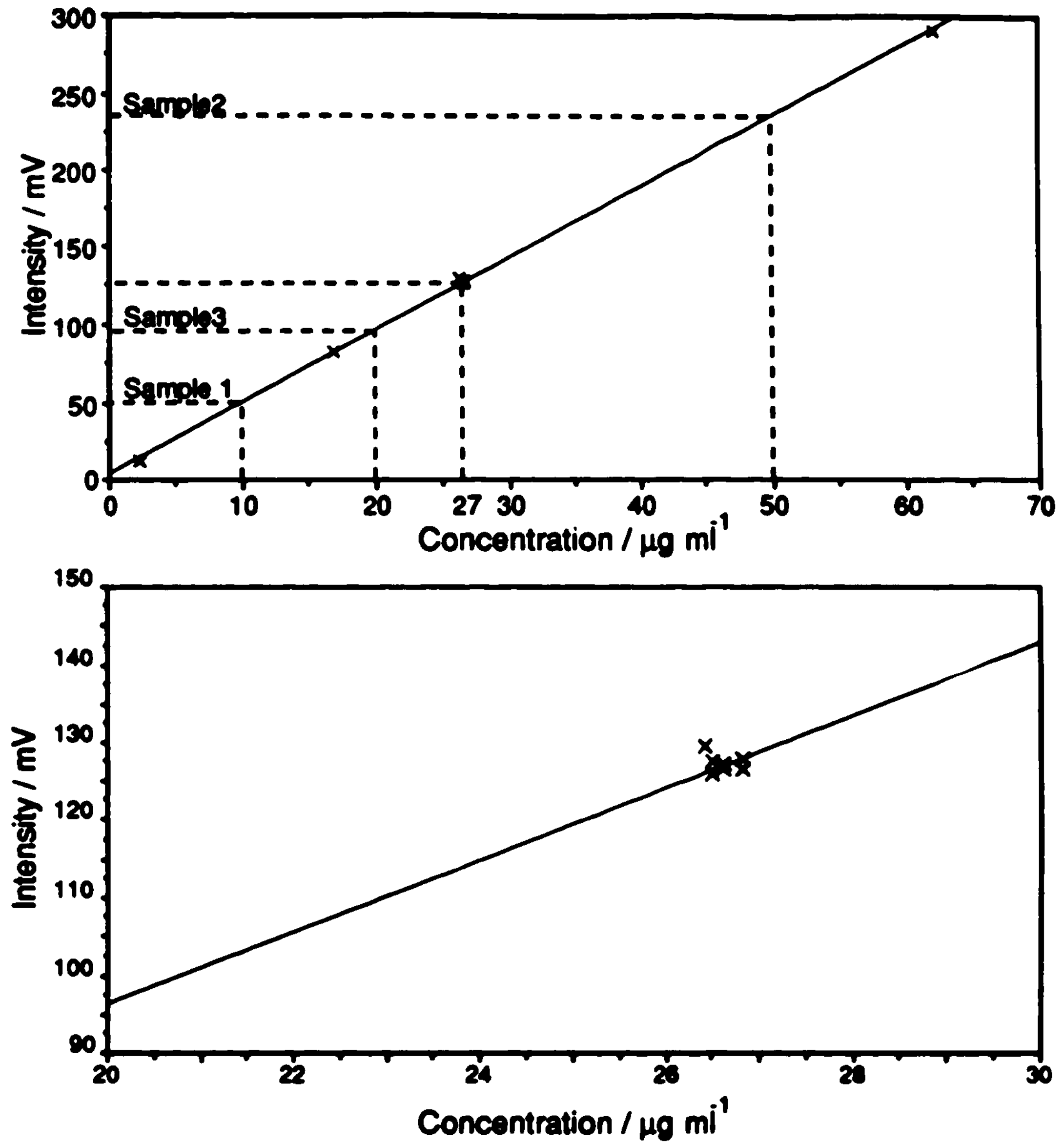
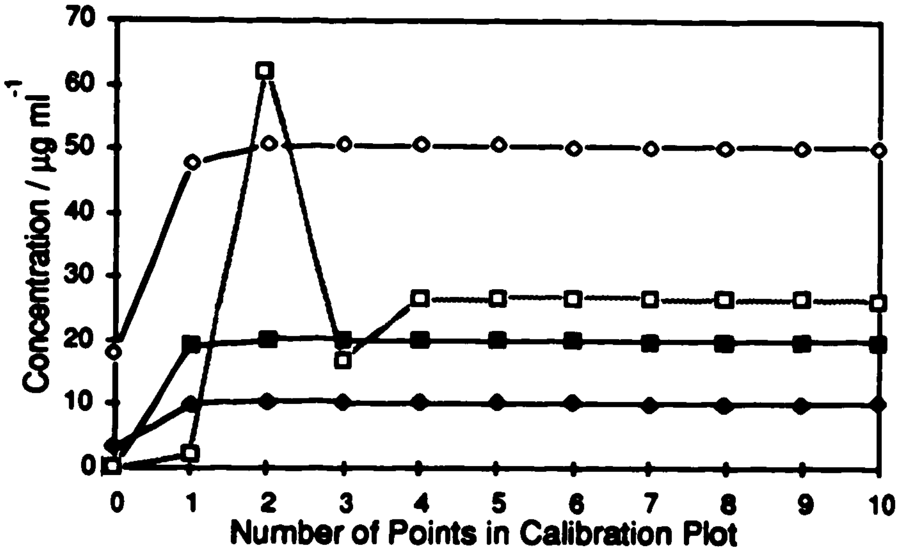


Figure 51 Calibration plot for calcium obtained from automated analysis. Detail of the range 20 to 30 µg ml^{-1} is shown below.

a)



b)

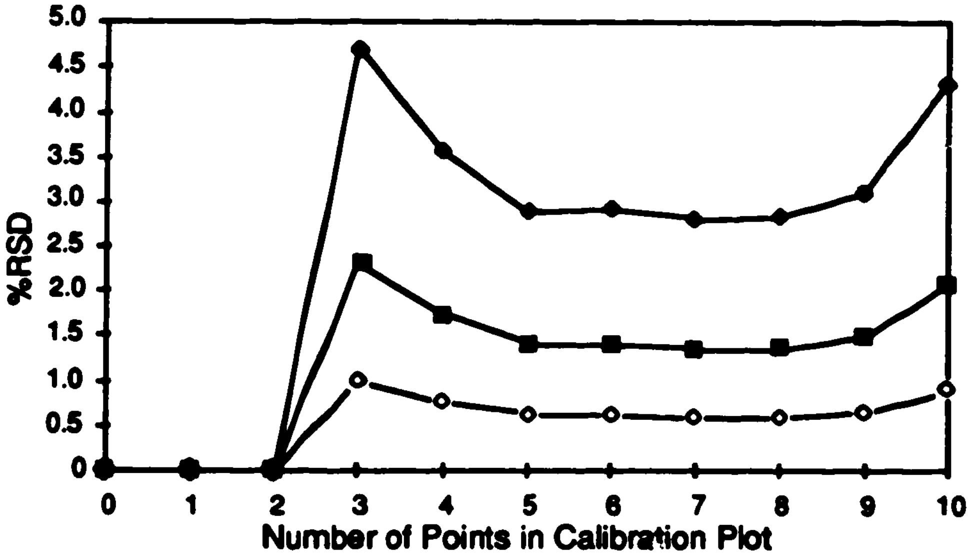


Figure 52 Plots of a) estimated composition of Sample 1 (◆), Sample 2 (◇), Sample 3 (■) and standard (□). b) Uncertainty in sample composition as function of number of points in calibration plots for simulation of automated analysis procedure.

a)

	Concentration/μg ml^{-1}					
	Ba	Fe	Zn	B	Ca	Na
Sample 1	0.05±32%	1.03±2%	0.094±5%	0.22±28%	10.0±4%	49±5%
Sample 2	0.01±122%	0.51±3%	0.187±2%	0.48±13%	50.3±1%	96±3%
Sample 3	0.02±76%	0.08±19%	0.483±1%	0.31±19%	19.8±2%	194±2%

Total Analysis Time : 4:30:36

b)

	Ba	Fe	Zn	B	Ca	Na
Sample 1	4.80%	3.30%	-6.10%	8.65%	-0.38%	-5.99%
Sample 2	-29.50%	1.90%	-6.65%	-4.38%	0.62%	-5.59%
Sample 3	-26.67%	-19.35%	-3.34%	2.87%	-1.15%	-2.57%

Table 27 Summary of analysis a) results and b) estimated accuracy for test samples.

summary of the results obtained for these test samples is shown in Table 27a. The values for Fe have been corrected for a standard solution concentration of 500 μg ml^{-1}. The accuracy of the results, estimated by expressing the difference between these results with the actual sample composition as a percentage of the actual concentration, is given in Table 27b.

The results for Ba showed a large amount of scatter. This was due to the unknown sample concentrations being close to the 3σ detection limit of 28 ng ml^{-1} as calculated from $3s_a/b$. Therefore, in Sample 2 and Sample 3 Ba is present at concentrations below the detection limit, and if the limit of quantitation is defined as being approximately 3 times the detection limit [53] then all three samples are below it.

The large uncertainties in the results for B are probably due to instrumental factors. They may arise from excessive carryover of the standards in the nebuliser/spray chamber, which is manufactured from borosilicate glass. It is interesting to note that though the precision is not very good, the results still show reasonable accuracy.

The Fe concentrations covered approximately an order of magnitude. This resulted in good precision being obtained for the samples with higher concentrations of Fe; those with low Fe, however have large uncertainties associated with them. It was found that the relative standard deviation was roughly inversely proportional to the concentration in all samples, which is consistent with the noise in the data being approximately constant. However, this is probably an artefact arising from the use of a least squares method to estimate the uncertainties.

In general, when concentrations are well above the limit of detection (at least 10x) and when the sample composition does not vary over more than an order of magnitude, then reasonable results (about 5% R.S.D.) can be obtained. A large number of calibration points is not necessary and, in view of the length of time required (4 hrs 30 mins for 10 calibration points), may be undesirable because drift in the instrument can reduce the precision. A compromise in the present system would be to limit the number of standards to 5.

The main factors that influence the analysis time are the number of samples, their complexity and the speed of the robot and pump. For each additional sample the analysis time would increase by about five minutes, corresponding to 1.5 minutes preflush, 2.5 minutes for measurement and about 1 minute for rinsing and the transfer of data to the IBM. As the complexity and range of concentrations of elements present in the samples increases, the need to use more intermediate solutions will increase. Each intermediate solution requires up to 5 minutes to prepare. Since for analyses performed on test samples there are typically three intermediate solutions required, the total preparation time is about 20 minutes for three intermediate solutions and the final solution. With an additional 5 minutes per standard for the measurement of the standard, a total of 25 minutes per calibration point is needed. The total analysis time for the test samples can thus be broken down to about 20 minutes for the measurement of the samples and 4 hours for the preparation and measurement of the ten standards. This can be reduced by using samples with components which are all at higher concentrations and so eliminate the need to prepare intermediate solutions.

4.2 Analysis of Well Water Samples.

A set of ground water samples had been analysed in the laboratory previously. These samples showed a wide range of compositions, as shown in Table 28 [54], and included some with high sodium content which may pose difficulty in normal analysis. From these samples three (marked with * in Table 28) were chosen as having a composition suitable for analysis by this procedure. The elements selected were those which showed small variation among the samples and covered a wide range of concentration between the elements. All three had a moderately high sodium content.

Rather than performing a single analysis with ten calibration points as had been done for the earlier test samples, two analyses with five calibration points were done. A summary of the results obtained is given in Table 29 with the results from the previous analysis. The results from each analysis is given and the mean values are in parentheses.

	Concentration/μg ml^{-1}								
Sample	B	Ba*	Ca*	Fe*	Li	Mg*	Na*	S	Zn*
ww1	0.57	0.04	24	2.6	0.047	12.4	260	43	<0.02
ww2*	0.55	0.042	33	0.34	0.069	16.3	300	70	0.23
ww3*	0.57	0.043	17.7	0.22	0.05	9.0	290	42	<0.02
ww4	0.19	0.14	77	0.059	0.091	25	98	17.1	2.3
ww5	0.16	0.19	83	1.3	0.095	27	86	5.3	0.022
ww6	1.5	0.14	5.4	0.23	0.057	1.29	610	<0.13	0.2
ww7	0.65	0.21	15	<0.01	0.047	5.2	500	0.42	<0.02
ww8	0.82	0.44	11.2	0.36	0.017	4.8	200	0.39	0.027
ww9	1.6	0.2	9.6	0.13	0.065	2.3	550	<0.13	0.053
ww10	0.22	0.051	300	<0.01	0.35	115	290	430	4.4
ww11	1.5	0.48	18.7	0.58	0.12	5.0	1170	0.59	0.058
ww12*	0.46	0.073	41	0.63	0.1	15.6	189	32	<0.02
ww13	1.1	0.12	4.5	0.37	0.057	1.17	470	0.26	0.044
ww14	1.2	0.17	6.1	0.064	0.068	1.57	540	<0.13	0.098
ww15	0.62	0.19	7.8	0.33	0.084	2.3	620	<0.13	0.031
ww16	0.31	0.2	86	7.3	0.084	34	220	3.4	0.16
ww17	0.3	0.19	85	4.6	0.085	35	220	4.2	0.23
ww18	0.57	0.062	25	2.4	0.07	11.4	230	36	0.099
ww19	0.31	1.1	115	5.2	0.089	46	250	10.2	1.9
ww20	1.1	0.14	8.4	0.30	0.04	3.4	260	0.41	<0.02
ww21	1	0.45	40	1.75	0.04	18.2	290	0.90	0.023
ww22	0.76	0.048	3.6	<0.01	0.065	1.32	350	45	0.032
ww23	1.5	0.15	7.6	0.23	0.068	1.97	480	0.36	<0.02
ww24	0.12	0.18	101	<0.01	0.041	30	35	26	0.12
ww25	0.22	0.1	60	2.1	0.11	18.9	106	9.9	0.1
ww26	0.42	0.055	92	2.7	0.095	37	180	99	<0.02

Table 28 Results from previous well water samples analysis.

For most of the results agreement is good. In the absence of known uncertainties for previous results, they are assumed to be in the range 1-5%. There are occasions where the difference between the replicate runs exceeded the estimated uncertainty (for example Ba in WW12), though the average of the two values is fortuitously close to the previous result. This suggests that the discrepancy in the results is due to random fluctuations in the initial

	Concentration/$\mu g\ ml^{-1}$					
	Zn	Ba	Fe	Mg	Ca	Na
Current Results						
WW 2	0.14±25%	0.042±7%	0.37±4%	15.6±0.6%	33.0±0.8%	284±0.5%
	0.16±22%	0.040±5%	0.36±2%	15.6±0.8%	33.1±1%	290±0.4%
	(0.15±24%)	(0.041±6%)	(0.37±3%)	(15.6±0.7%)	(33.0±1%)	(287±0.4%)
WW 3	–	0.042±7%	0.25±6%	8.8±1%	18.7±1%	283±0.5%
	–	0.046±4%	0.24±4%	8.7±2%	18.1±2%	280±0.4%
	–	(0.044±6%)	(0.24±5%)	(8.8±1%)	(18.4±2%)	(282±0.4%)
WW 12	–	0.088±4%	0.64±2%	15.1±0.6%	42.2±0.6%	183±0.8%
	–	0.068±3%	0.63±1%	14.6±0.9%	40.8±0.9%	181±0.6%
	–	(0.078±3%)	(0.64±2%)	(14.8±0.8%)	(41.5±0.8%)	(182±0.7%)
Previous Results						
WW 2	0.23	0.042	0.34	16.3	33	300
ICP-MS	0.21	0.043				
WW 3	<0.02	0.043	0.22	9.0	17.7	290
ICP-MS	0.02	0.045				
WW 12	<0.02	0.073	0.63	15.6	41	189
ICP-MS	0.011	0.070				

Table 29 Summary of analysis results for well water samples.

measurement step and that the uncertainty has been underestimated by the least squares method. The values obtained for Na are consistently low, suggesting there may be some interference. A few of the other results show large positive and negative deviations (e.g. Ca in WW3, Mg in WW2 and Fe in WW3). These again suggest that the uncertainty is being underestimated.

Chapter 5

Automation of Matrix Studies.

The accuracy of the results of an analysis is dependent on the ability of the measurement system to produce the same signal for the standards and the sample when both are at the same concentration. There may be two reasons why this is not so. First, another component in the sample may produce an additional signal which is indistinguishable from that of the sought-for component. For example, this can be due to spectral overlap. Second, the other component may somehow change the sensitivity of the measurement system to the sought-for component. This would result in a change in the slope of the calibration plot by causing the sought-for component to produce either proportionately more or less emission.

The first source of error can be reduced by various background correction methods that attempt to estimate the extra signal and subtract it from the measured signal. This may be done by measuring the emission away from the spectral line and estimating the emission under the line by interpolation.

The second source of error is more difficult to correct. One of the reasons for the widespread use of the ICP has been its relative freedom from matrix effects. Recently, however, there have been reports of significant matrix effects being observed [55 – 59] especially in the analysis of geological [55 – 57] and biological samples [58,59]. Initial matrix effect studies focused on interferences caused by the alkali metals [60, 61]. The effects of these elements were found to be minimal under normal operating conditions. More recently, the effect of some alkaline earths (especially Ca) and other elements has been found to more severe [55,58]. In the case of biological samples some severe matrix effects have been reported for organic components of the matrix [58].

The analysis of trace metals in organic matrices is of importance in the food and beverage industries [62 – 66]. Several analyses of inorganics in such samples have been reported [62 – 66]. These include alcoholic beverages, where the presence of ethanol can cause an interference [65,66]. The matrix effect of ethanol in the ICP over a wide range of ethanol concentrations has been reported [65].

All the matrix effects considered affected the sensitivity of the measurement. The intensity of the emission signal in the ICP is given by [67];

$$I = f \frac{hc\,gA}{4\pi \lambda Q} n \exp\left(-\frac{E}{kT}\right) \qquad (1)$$

where I is the measured intensity; f is an instrumental parameter which combines the collection efficiency of the optics, the throughput of the spectrometer, sensitivity of the detector and the gain of the electronics; h is the Planck constant; c is the speed of light; g is the degeneracy of the emitting state; A is the transition probability of the observed emission; λ is the wavelength of the emission; Q is the partition function for the emitting species (atom or ion); n is the number density of the species in the plasma; E is the energy of the emitting state; k is the Boltzmann constant and T is the excitation temperature. Matrix components can influence only two of the parameters in Equation 1, n and T.

Factors that influence n are the concentration of the species in solution, the nebulisation efficiency (which governs what fraction of the solution will be transferred into the plasma), the vapourisation efficiency (which governs the fraction of the material which appears in the gaseous form), atomisation efficiency and the ionisation temperature (which governs the amount of the atomic gas that is in the form responsible for the emission) and the spatial distribution of the species (which governs the amount of material in the observation zone). All of these could potentially be affected by the sample matrix. The excitation temperature could be affected by anything which can add or remove energy from the plasma, which is responsible for the excitation of the analyte. Hence matrix effects can arise from two causes, those which affect the transport of material into the plasma and those which affect excitation processes. Matrices which alter the bulk properties, such as the surface tension or the viscosity of the solution, will cause matrix effects due to changes in the transport of the solution by modifying the nebulisation efficiency, or the liquid flow rate in unpumped systems. Matrices which do not effect the solution properties may cause matrix effects through changes in the excitation environment.

The matrix effects induced by metals are due to changes in the excitation processes because the presence of a metal ion in solution has little effect on solution properties, especially at the typical concentrations studied (<1% w/v). On the other hand, the presence of organic solvents (such as ethanol or acetone) in aqueous solutions can cause significant changes in solution properties. In this study the effect of Ca is investigated as an example of a matrix which acts through changes in the excitation process. Also, ethanol was studied as an example of a matrix which could cause changes in both the solution transport and the excitation processes.

Equation 1 can be used to predict how the emission intensity measured will vary with temperature. Both the exponential term and the number density will change with temperature. The number density will be proportional to the molar concentration of the analyte and will also depend on the degree of ionisation. The emission intensity will be given by;

$$I_i = f' \frac{gA}{\lambda Q} \alpha \, c_m \exp\left(-\frac{E}{kT}\right) \qquad (2)$$

$$I_a = f' \frac{gA}{\lambda Q} (1 - \alpha) \, c_m \exp\left(-\frac{E}{kT}\right)$$

where f' is a constant combining the instrumental parameters of Equation 1, along with the fundamental constants h, c and π and the transport efficiency of the nebuliser and spray chamber (the latter governs how much of the analyte species in the solution will be transferred to the plasma), α is the degree of ionisation of the analyte and c_m is the molar concentration. I_i is the emission intensity for ions whilst I_a is the emission intensity for atoms.

The degree of ionisation (α) can be determined from the Saha Equation [68];

$$K = \frac{\alpha}{1 - \alpha} = \frac{4.83 \times 10^{15}}{n_e} \frac{Q_i}{Q_a} T^{3/2} \exp\left(\frac{E_i}{kT}\right) \qquad (3)$$

where K is the Saha equilibrium constant, n_e is the electron density in the plasma, Q_i and Q_a are the partition functions for the ion and atom respectively, T is the ionisation temperature and E_i is the ionisation potential of the analyte.

If local thermal equilibrium (LTE) is assumed to exist, then a single temperature can be used in both Equations 3 and 2. If it is further assumed that the elements are close to being completely ionised, then $\alpha \approx 1$ and $(1 - \alpha) = 1/(1+K) \approx 1/K$. With these assumptions α and $(1-\alpha)$ from 3 can be substituted into 2 to give the following;

$$I_i = f' \frac{gA}{\lambda Q_i} c_m \exp\left(-\frac{E}{kT}\right)$$
$$I_a = f' \frac{gAn_e}{\lambda Q_i T^{3/2}\, 4.83\times10^{15}} c_m \exp\left(-\frac{E - E_i}{kT}\right) \qquad (4)$$

The assumption that $K \gg 1$ is equivalent to the analyte being essentially completely ionised (hence $\alpha \approx 1$). It also results in the ground state of the ion being the natural reference state, so that the effects of temperature will depend on the distance a state is from the ground state of the ion. For ion emission this corresponds to the excitation energy of the state whilst for atom emission it is the difference between the excitation energy and the ionisation potential.

Rearranging 4 to bring all element-dependent terms to the left hand side gives;

$$\frac{I_i \lambda Q_i}{gAc_m} = f' \exp\left(-\frac{\Delta E}{kT}\right)$$
$$\frac{I_a \lambda Q_i}{gAc_m} = f' \frac{n_e}{4.83\times10^{15}\, T^{3/2}} \exp\left(-\frac{\Delta E}{kT}\right) \qquad (5)$$

where ΔE is the distance the emitting state is in energy from the ground state of the ion, as defined by Equation 4. If the detector sensitivity and the

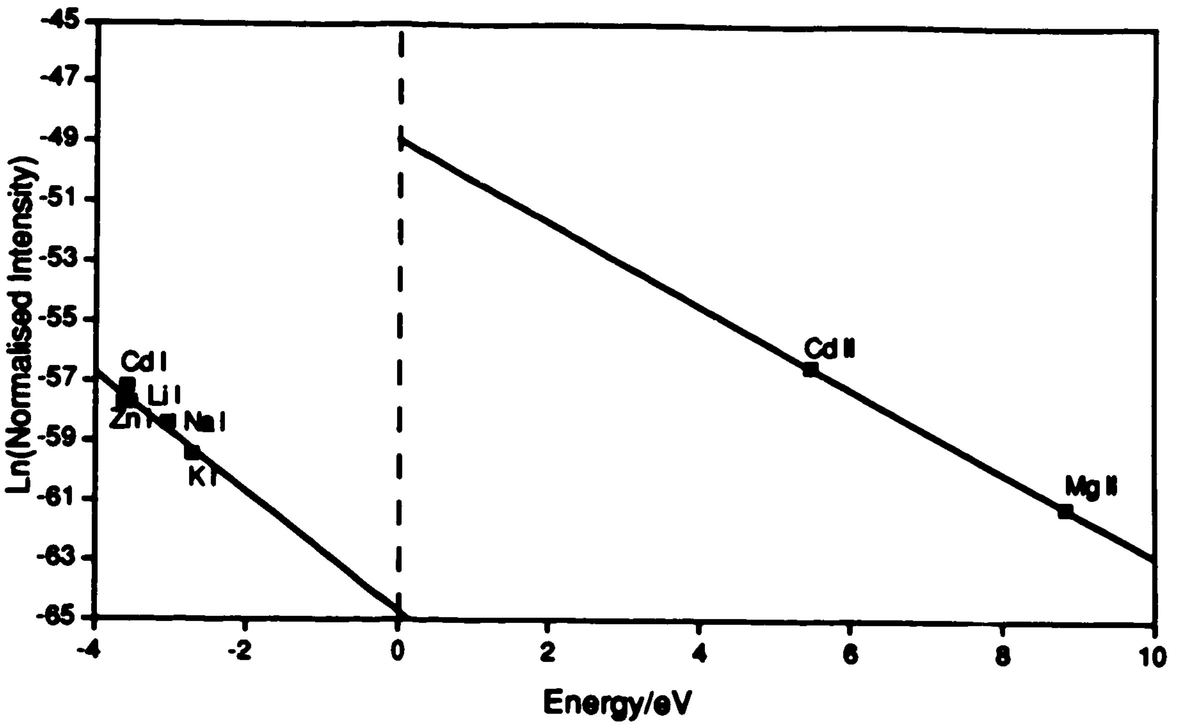


Figure 53 Multi-element Saha-Boltzmann Plot.

amplifier gains are equal for each channel used, and if local thermal equilibrium exists in the plasma, then a plot of the natural logarithm of the left hand side of Equation 5 against ΔE should yield two parallel lines with slope $-1/kT$. The difference in the intercepts is related to the electron density. An example of such a plot for those elements for which gA values are available [68 – 70] is shown in Figure 53. The plot shows the trends that could be expected. The atom lines, below zero on the energy axis, show a decrease in intensity due to the decrease in the population of the excited states following a Boltzmann distribution. An increase in intensity is observed at the ionisation potential followed by a decrease in intensity due the Boltzmann distribution. However, the absence of the gA values for most of the ion lines measured doesn't allow a true picture to be seen of the behaviour of the ion lines. Similar Saha-Boltzmann plots for iron lines have been constructed by Blades et al. [68] that show the same type of trends observed here. The difference between the plots of Blades et al. and those presented here is that since Blades et al. used a single element the energy was defined with respect to the ground state of the atom. Hence the energy of the ion lines were defined as the sum of the excitation energy and the ionisation potential. This results in a shift in the origin of the energy axis.

An alternative to this plot for calculating the temperature is the two line method using the two Cd channels. The ion/atom intensity ratio is given by [68];

$$\frac{I_i}{I_a} = \frac{4.83 \times 10^{15}}{n_e}\left(\frac{gA}{\lambda}\right)_i\left(\frac{\lambda}{gA}\right)_a T^{3/2}\exp\left(\frac{-E_i-E_{ei}+E_{ea}}{kT}\right) \tag{6}$$

where E_i is the ionisation potential and E_{ei} and E_{ea} are the energies of the emitting states of the ion and atom respectively. This equation also assumes the existence of local thermal equilibrium. Equation 6 was solved numerically using the Newton-Raphson iteration method and was used to calculate all ionisation temperatures.

The relative change in the emission intensity due to a change in the temperature can be obtained from Equation 4;

$$\frac{I_i(T_2)}{I_i(T_1)} = \exp\left(\frac{E\Delta T}{kT_2T_1}\right) \tag{7}$$

$$\frac{I_a(T_2)}{I_a(T_1)} = \exp\left(\frac{(E - E_i)\Delta T}{kT_2T_1}\right)$$

If the temperature change is small then these can be simplified to;

$$\frac{I(T_2)}{I(T_1)} = \exp\left(\frac{\Delta E \Delta T}{kT^2}\right) \tag{8}$$

where ΔE is as defined for Equation 5. A plot of the natural logarithm of the relative sensitivity against ΔE will yield a straight line with slope $\Delta T/kT^2$, T being estimated from the ionisation temperature calculated from the Cd ion/atom line ratios.

As with Equation 4, there is the assumption that $K \gg 1$; for $\alpha > 0.95$ this assumption leads to no more than a 5% error. However, for four of the elements chosen (As, Cd, Zn, Cu), $\alpha < 0.95$. For these elements the equation;

$$\frac{I_a(T_2)}{I_a(T_1)} = \exp\left(\frac{E\Delta T}{kT^2}\right)\frac{1 + K(T_1)}{1 + K(T_2)} \tag{9}$$

has to be used. Here K(T) is the Saha equilibrium constant at temperature T and E is the excitation energy. The second term is due to the change in the degree of ionisation. The above four elements were used to verify the theory. The atom lines for the essentially completely ionised elements, K, Na, Li, and all the ion lines, were used to estimate ΔT from Equation 8. From this estimated value of ΔT, T_2 was calculated and the relative change in the degree of ionisation was calculated. Equation 9 was then used to correct the intensity ratio and the resulting values were plotted. These corrected values should lie on the same line as the rest of the points.

5.1 Matrix Study Procedures.

The experimental procedures used for the two matrix types were identical. The solution preparation method used for the analysis routine was modified slightly to allow the use of a matrix component which is not necessarily a channel on the spectrometer. This was done by associating with this extra component the dummy channel number 35, so the solution preparation record would refer to this component by this dummy channel number. A special record for recording the position and concentration of the matrix component was defined as well as a string variable for the matrix label and a variable for the molecular mass of matrix component so that the composition of the final solution could be printed in the same format as previously.

The experimental procedure for the matrix study involved the preparation and measurement of a series of solutions with varying concentrations of the matrix component whilst keeping that of the analyte components constant. The matrix component was prepared from a concentrated stock solution (typically 2% w/v for Ca and 2% v/v for ethanol) and the analytes were prepared from 1000 μg ml^{-1} solutions. The procedure was to first prepare a solution containing only the analyte components and then to measure a distilled water blank and the prepared solution. The blank-subtracted signal from this solution was used as the reference signal for all other measurements. Then two solutions were prepared for each concentration of matrix component required, the first containing only the matrix component, to serve as the matrix blank, and the second containing both the matrix component and the analyte components. These two solutions were measured, the signals obtained subtracted from each other, and the differences ratioed to the reference signal obtained earlier. This ratio was stored for later data processing. The complete record stored in the data file contained the composition of all the solutions, integrated intensities of each solution measured and the relative sensitivities for each concentration of the matrix component.

Twelve elements (Zn, As, In, Mn, Cr, Cd, K, Na, Li, Cu, Ba, Mg) were used in the matrix studies. Since only six elements could be studied at a time the matrix study program had to be run twice for each study. The results from each part were combined during the data processing stage.

In order to study the effect that the nebuliser gas flow rate has on the matrix effects a mass flow controller (Model 8240, Matheson, New Jersey) was used to control the nebuliser gas flow rate. This allows for better control of the gas flow than the needle valve on the spectrometer. The mass flow controller allowed the flow rate to be set; a digital readout of the flow rate was also provided.

5.2 Matrix Effects Due to Ca.

In a previous study [57] the behaviour of the matrix effect was consistent with a decrease in the excitation temperature of the plasma. It was found that the magnitude of the temperature decrease was related to the energy required to decompose the solid form of the matrix to the equilibrium mixture of atoms and ions.

For this study analyte lines were chosen to cover a wide range of excitation energies and ionisation potentials, and to provide a mixture of atom and ion emission lines. A summary of the spectroscopic data for the channels available is shown in Table 30. The channels chosen are indicated.The two Cd channels were especially chosen since one was an atom line and the other was an ion line; this made it was possible to use data from

Element	Wavelength	Atom/Ion	I.P./eV	Exc. Energy/eV a
* K	766.49	I	4.33	1.62
* Na	589.59	I	5.12	2.1
* Ba	455.4	II	5.2	2.83
* Li	670.78	I	5.38	1.85
Sr	407.78	II	5.68	3.05
* In	230.61	II	5.76	5.37
Al	237.34	I	5.98	5.24
Ca	317.93	II	6.11	7.05
V	292.4	II	6.72	4.63
* Cr	205.55	II	6.77	6.03
Ti	337.28	II	6.85	3.68
Zr	343.82	II	6.93	3.69
Mo	202.03	II	7.19	6.13
Sn	189.99	II	7.33	7.05
Pb	220.35	II	7.41	7.37
* Mn	257.61	II	7.42	4.81
Ag	328.07	I	7.59	3.78
Ni	231.6	II	7.63	6.39
* Mg	279.08	II	7.63	8.86
* Cu	324.75	I	7.72	3.82
Fe	259.94	II	7.9	4.77
Si	288.16	I	8.15	5.08
B	249.68	I	8.28	4.96
Sb	206.83	I	8.63	5.98
* Cd2	228.8	I	8.98	5.41
* Cd1	226.5	II	8.98	5.48
* Zn	213.86	I	9.41	5.8
* As	189.04	I	10	6.56
Hg	184.95	I	10.5	6.67
C	193.09	I	11.3	7.69

Table 30 Table of spectroscopic data for some channels of the ARL 34000 spectrometer. a) From Reference 71.

these channels to calculate the ionisation temperature by the two line method [68].

The results of a matrix effect study for which the concentration of Ca was varied over the range 100–10,000 μg ml^{-1} are shown in Figure 54. The

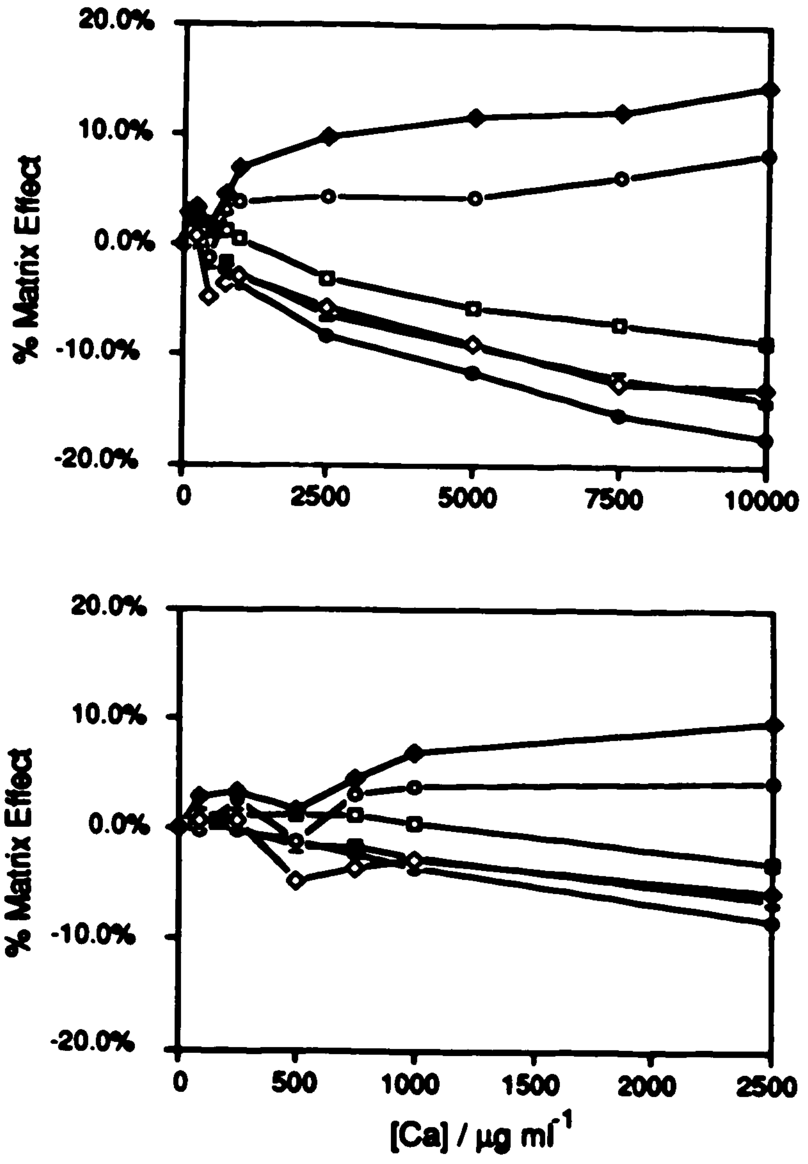


Figure 54 Variation of matrix effect with calcium concentration for emission lines for (◆) Li atom, (○) K atom, (□) As atom, (◇) Mg ion, (■) Zn atom, (●) Cd ion.

general trends observed are in agreement with those obtained by Thompson and Ramsey [55]. The alkali metals all show an enhancement with increasing Ca concentration whilst most other elements showed a suppression which increased with Ca concentration. This suppression was most severe for Cd and can lead to an error of approximately 20% for a Ca concentration of 1% w/v.

These observations are consistent with a decrease in plasma temperature. The alkali metals are analysed by relatively weak atom emission even though they are all at least 99% ionised [72]. These atom lines are therefore very sensitive to changes in the degree of ionisation, increasing in intensity as the degree of ionisation decreases with decreasing temperature. For the other atom lines the increase in the total atom population is offset by the decrease in the excited state population and so leads to an overall decrease in intensity. The ion lines are affected by a slight decrease in the ion population and in the population of the excited states.

From the Cd ion/atom ratio the ionisation temperature at each concentration of Ca was calculated; this data is shown in Figure 55. It was found that the temperature does indeed decrease with increasing concentration of Ca. A plot of the natural logarithm of the relative sensitivity against the energy difference from the ground state of the ion, as suggested by Equation 8, was constructed and is shown in Figure 56 for a Ca concentration of 10,000 $\mu g\ ml^{-1}$. The slope of this plot is $\Delta T/kT^2$. If T is assumed to be equal to the the ionisation temperature measured from the Cd ion/atom ratio with no Ca present, then the change in temperature is approximately equal to 150K. The points plotted with open squares were not included in the

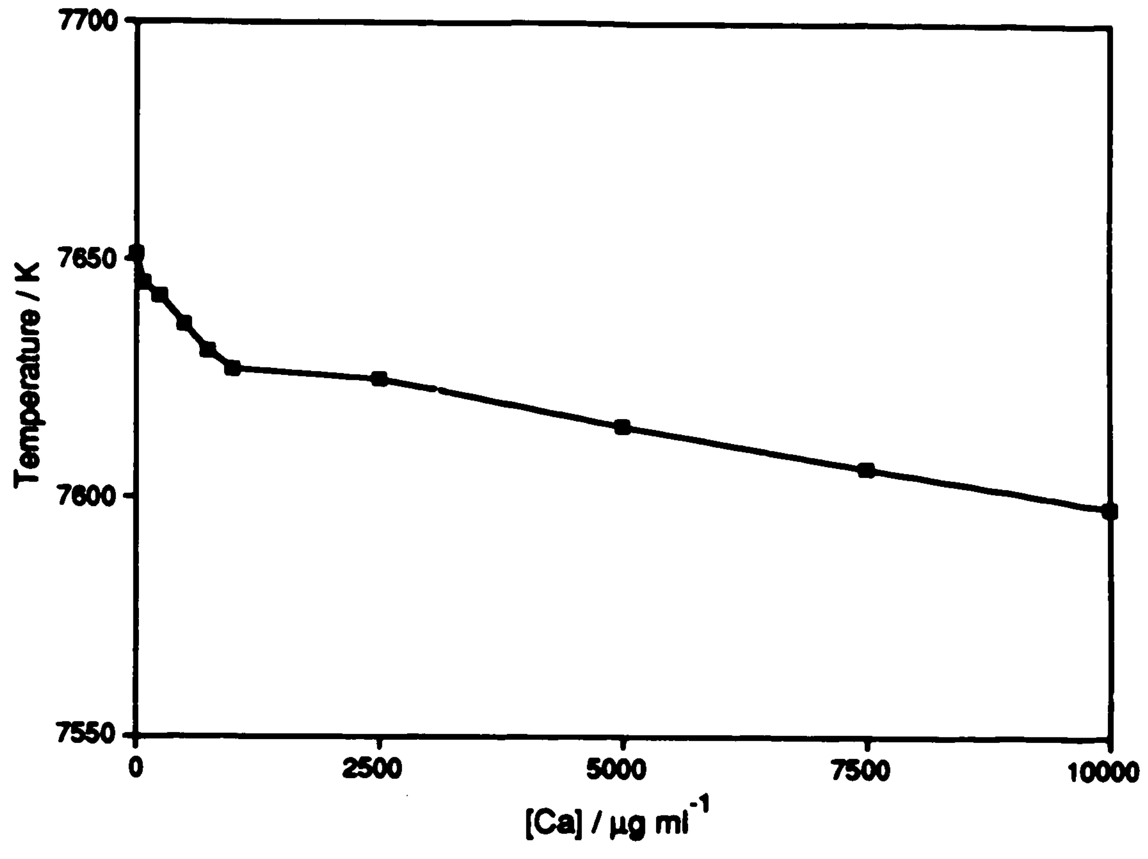


Figure 55 Variation in ionisation temperature as measured from Cd ion/atom intensity ratio with calcium concentration.

regression as these are due to the four elements which are less than 95% ionised [72]. After correction using Equation 9, these points are shifted as shown in Figure 57. In Figure 56 the point for Mg would appear to be an outlier. When it is excluded from the regression the correlation coefficient improves from 0.94 to 0.99.

The high correlation in this plot indicates that the matrix effect is probably due to a decrease in the ionisation/excitation temperature. The change in temperature is relatively small (~2%) but can lead to relatively large

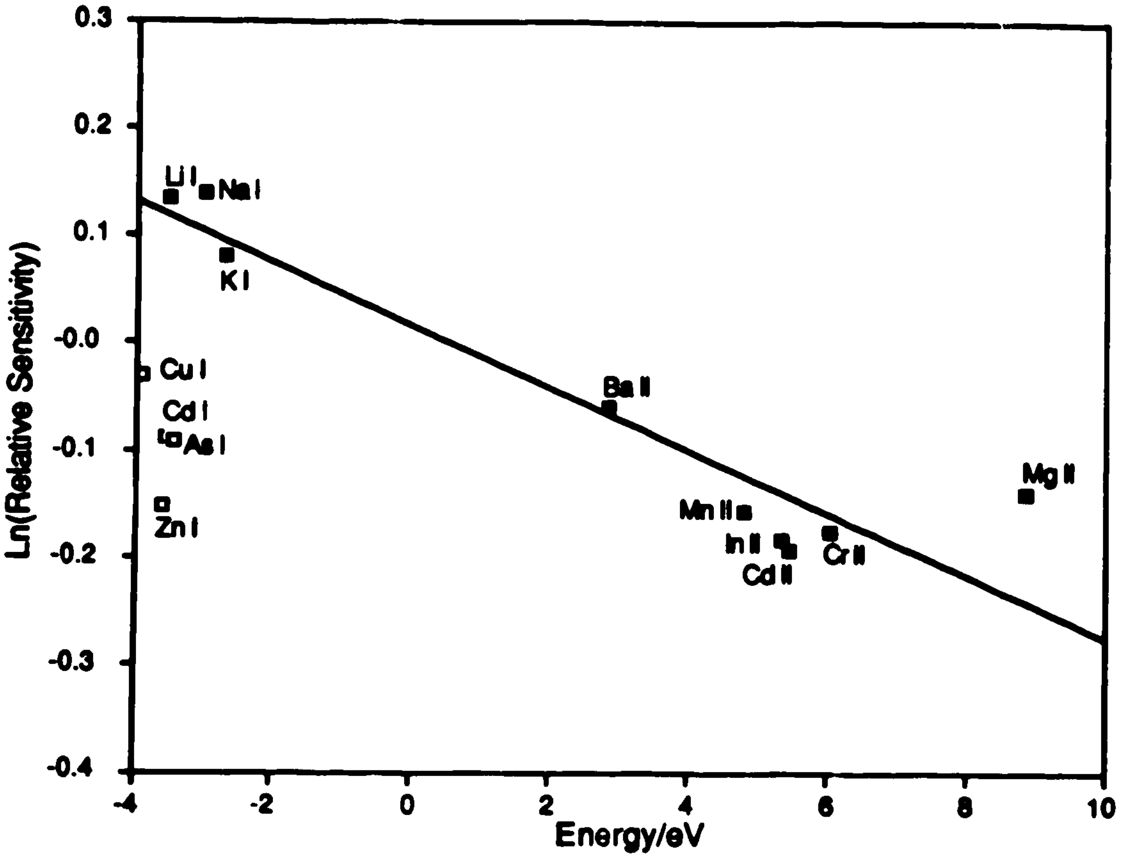


Figure 56 Relationship between relative sensitivity and energy of emitting state with respect to the ground state of the ion for elements that are almost completely ionised (■) and only partially ionised (□).

errors (~20%). When the matrix effects due to the alkali metals were originally being studied it was found that under certain operating conditions large effects were observed whilst under others virtually no effect was observed [60].

The effect of the nebuliser gas flow rate on the Ca matrix effect was studied for two Ca concentrations (1000 and 10,000 μg ml^{-1}). The nebuliser gas flow rate was varied from 0.8 to 1.5 l min^{-1} in 0.1 l min^{-1} increments. The

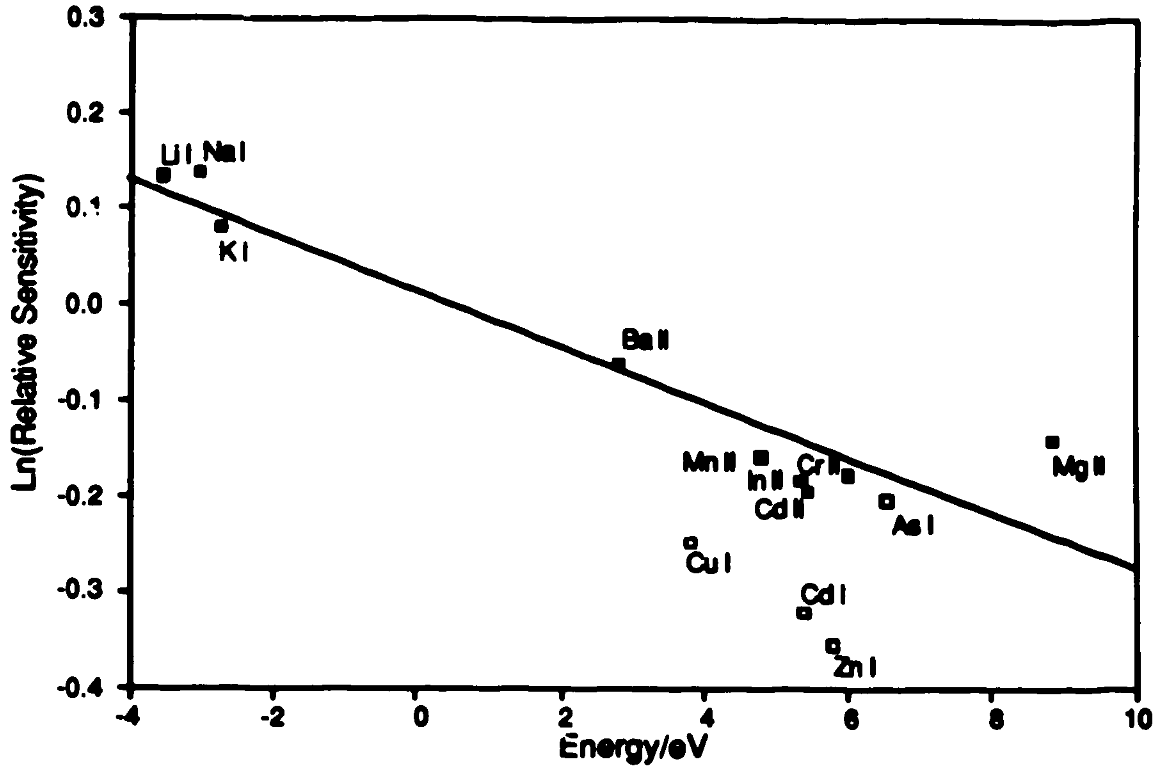


Figure 57 Relationship between relative sensitivity and energy of emitting state with respect to the ground state of the ion after correction for partial ionisation for atom lines. Energy measured with respect to ground state ion for highly ionised elements (■) and for partially ionised elements (□) energy is measured with respect to the ground state of the atom. See Equation 7 and Equation 9.

emission intensity as a function of flow rate is shown in Figure 58. The ionisation temperature as function of flow rate is also shown in Figure 59.

From the emission profiles it can be seen that for most elements the matrix effect was minimal at a flow rate of approximately 1.3 l min^{-1}, with the exception of the alkali metals. For the atom lines with higher ionisation potentials and for the ion lines the profiles shown for Zn, As and Cd ions are

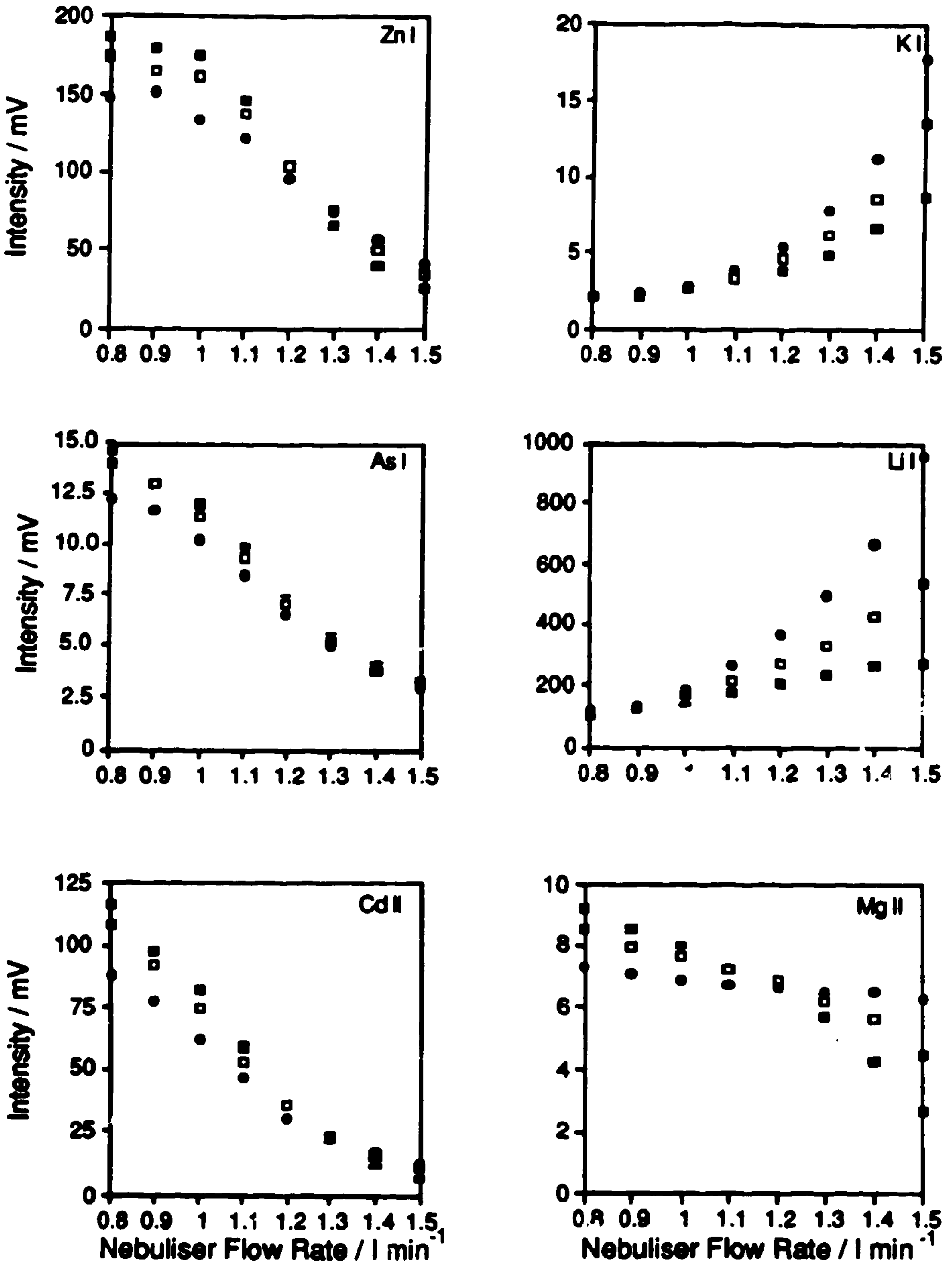


Figure 58 Emission intensity as a function of nebuliser gas flow rate for calcium concentrations of (■) 0 µg ml^{-1}, (□) 1000 µg ml^{-1} and (●) 10,000 µg ml^{-1}.

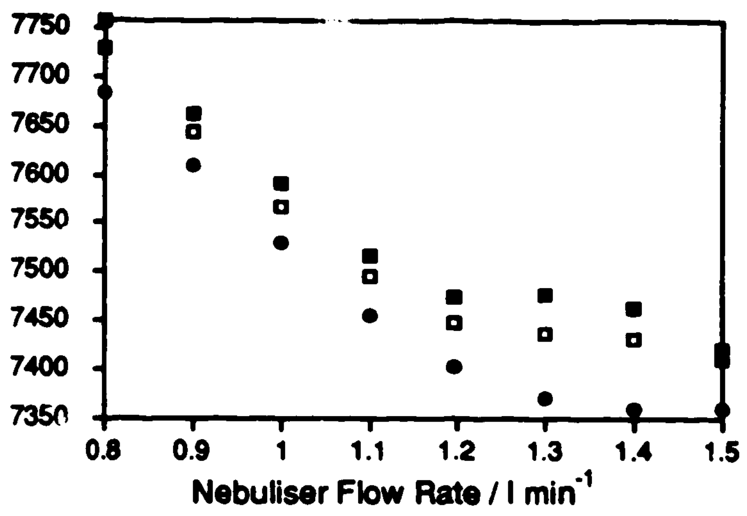


Figure 59 Ionisation temperature as function of nebuliser gas flow rate for calcium concentrations of (■) 0 μg ml^{-1}, (□) 1000 μg ml^{-1} and (●) 10,000 μg ml^{-1}.

typical. At low flow rates a high emission intensity which changes little with flow rate was found. This signal was suppressed with the addition of Ca. At higher flow rates a lower signal was observed which was enhanced with the addition of Ca. For the alkali metals there is a very small signal at low flow rates which increases with flow rate. At all flow rates an enhancement is observed. Magnesium ion shows similar trends to the other ions except that the profiles are much flatter, so that at the highest Ca concentration used the signal did not change appreciably with changes in flow rate. The temperature profile did not show this trend. It was found that the temperature of the plasma in the presence of 1% Ca was consistently about 50K less than the temperature of the plasma without Ca present.

The relationship between the relative sensitivity and the energy difference between the emitting state and the ground state of the ion as a function of nebuliser gas flow rate was studied using a series of plots similar to that of Figure 57 at each flow rate. These are shown in Figure 60. It was found that the four atom lines of the partially ionised elements in each case clustered with the ion lines. The change in the LTE temperature measured from these plots ranged from about 110K to about 270K, the largest change in temperature being found at a flow rate of 1.20 l min^{-1}.

In addition to the change of temperature observed from the changing slope of these plots, there was also a change in the intercept of the plots. Although Equation 8 predicts an intercept of zero, a non-zero intercept could result from a change in the number density of the analyte species on adding the matrix component. Hence, on adding Ca to the solutions there would appear to be two effects. The first is a decrease in the excitation/ionisation temperature and the second a change in the number density of the analyte species in the center channel of the plasma. The extent of these effects varies with the nebuliser gas flow rate. At low flow rates there is a small decrease in temperature (~100K) and net decrease in the number density. The magnitude of the decrease in temperature increases with increasing flow rate to a maximum value of ~270K at a flow rate of 1.20 l min^{-1}. The change in number density decreases with increasing flow rate till about 1.10 L. min^{-1}, where the intercept of the plot is close to zero. With increasing flow rate the number density shows a net increase when Ca is added. At a flow rate of about 1.30 l min^{-1} the increase in number density almost completely offsets the decrease in temperature for most elements and so the matrix effect is minimised. At higher flow rates an enhancement is observed due to the increased number density of the analyte.

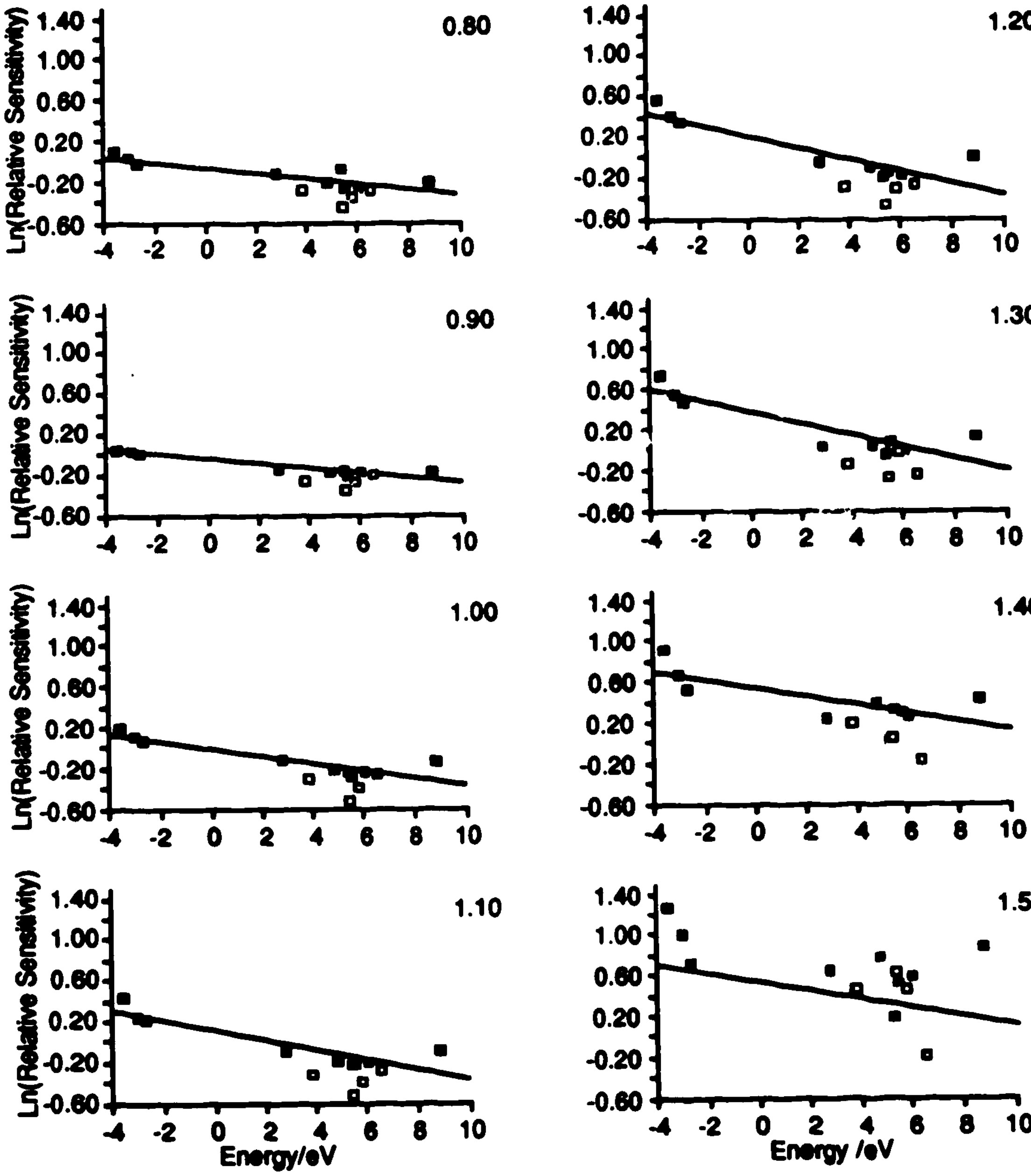


Figure 60 Relationship between relative sensitivity and energy of emitting state at different nebuliser gas flow rates. Energy is measured with respect to ground state ion for highly ionised elements (■) and for partially ionised elements (□) energy is measured with respect to the ground state of the atom.

The plots used all suggest a decrease in temperature as accounting for the magnitude of the matrix effect. However, it has been shown that the ICP cannot be characterised by a single temperature [68] and that the electronic excitation temperature is significantly lower than the ionisation temperature [68]. In the case of the Ca matrix effect it is likely that these temperatures are changing in a concerted manner and that this simultaneous change in temperature accounts for the correlation observed in the plots used.

5.3 Matrix Effects Due to Ethanol

There has been one previous study of the matrix effects of [illegible] n the ICP reported in the literature [65]. In this report the concentration of ethanol was varied from a completely aqueous solution to completely ethanolic. The instrument, which used a free running rf generator, was able to operate in a completely organic matrix without the addition of oxygen, as is common with a fixed frequency rf generator [73]. The ICP used in the present study uses a fixed frequency rf generator and is not able to operate with large amounts of organic solvent present. Accordingly, the concentration range of ethanol was kept to a maximum of 1% v/v. For this reason, and because many operating parameters may have varied in the free running generator which are fixed in the system used here, results of this work are not readily comparable with the Xiaoru study.

The same analyte lines were chosen for this matrix study as for the Ca study. These are shown in Table 30. The ethanol stock solution was prepared to a nominal concentration of 2% v/v from 98% ethanol by dilution with distilled de-ionised water.

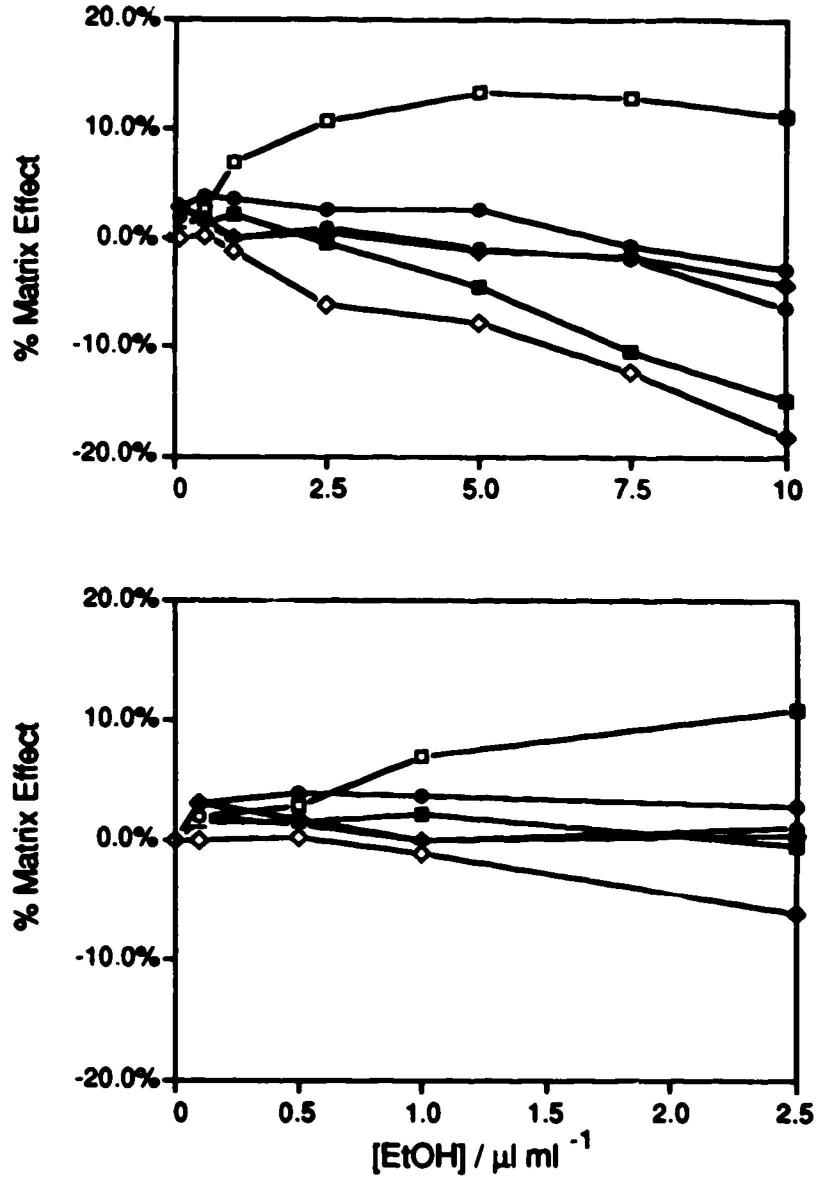


Figure 61 Variation of matrix effect with ethanol concentration for emission lines for (□) As atom, (●) Cd ion, (◆) Li atom, (○) K atom, (■) Zn atom, (◇) Mg ion.

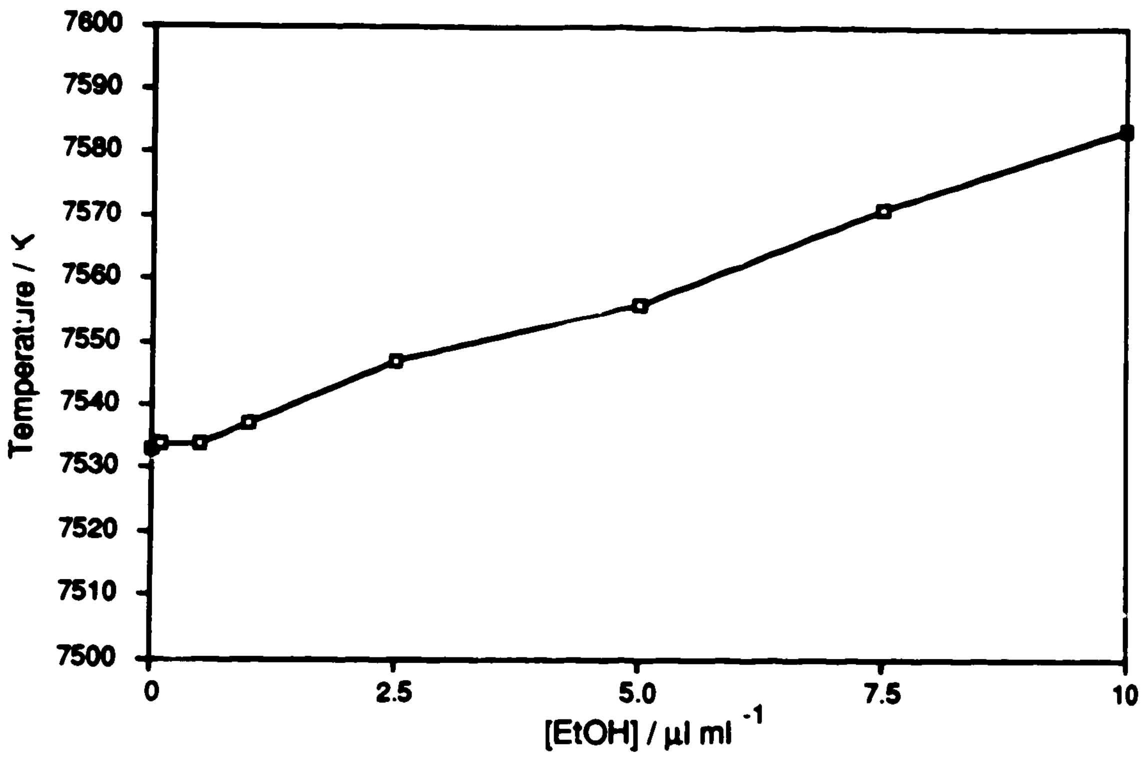


Figure 62 Variation in ionisation temperature as measured from ion/atom intensity ratio with ethanol concentration.

The initial study of the matrix effect was done by varying the ethanol concentration from 0.1 µl ml^{-1} to 10 µl ml^{-1} while maintaining a constant concentration of the analyte species. A typical set of results are shown in Figure 61.

The ionisation temperature was determined using the two line method from the Cd lines. The change in temperature with changing ethanol concentration is shown in Figure 62. It was found that the ionisation temperature increased with concentration of ethanol. This is the opposite from the trend observed for Ca.

The trends shown are markedly different for those found for Ca. There are few elements which show enhancements; some show severe suppressions. The behaviour cannot be explained by a simple change in temperature as for Ca. The greatest enhancement is found for As, the element with the highest ionisation potential, whereas the alkali metals and most other ions show almost no matrix effect. The increase in ionisation temperature can explain the lack of enhancement of the alkali metals. The trends found for the ion lines, however, would be consistent with a decrease in the excitation temperature.

This separation of the ionisation temperature and the excitation temperature is more apparent in the energy plot shown in Figure 63. There is a very low correlation ($r=-0.005$) between the natural logarithm of the relative sensitivity and energy in the presence of ethanol indicating that some of the assumptions made in deriving Equation 8 are not valid here. In particular, the assumption that the system can be described by a single change in temperature does not apply. All points that have positive energy in this plot are plotted against the excitation energy of the emitting state. Changes in the intensity of these states will therefore depend solely on changes in the excitation energy. The negative energy points include both the excitation energy and the ionisation potential and so will be influenced by changes in both the ionisation and excitation temperatures. The trend shown by the positive energy lines (ions and corrected atom lines) is consistent with a decrease in temperature, whereas the trend in crossing the zero energy axis suggests an increase in temperature. The negative trend amongst the ion lines is due to a decrease in the excitation temperature, while the rising trend in going from the atom lines to the ion lines is due an increase in the

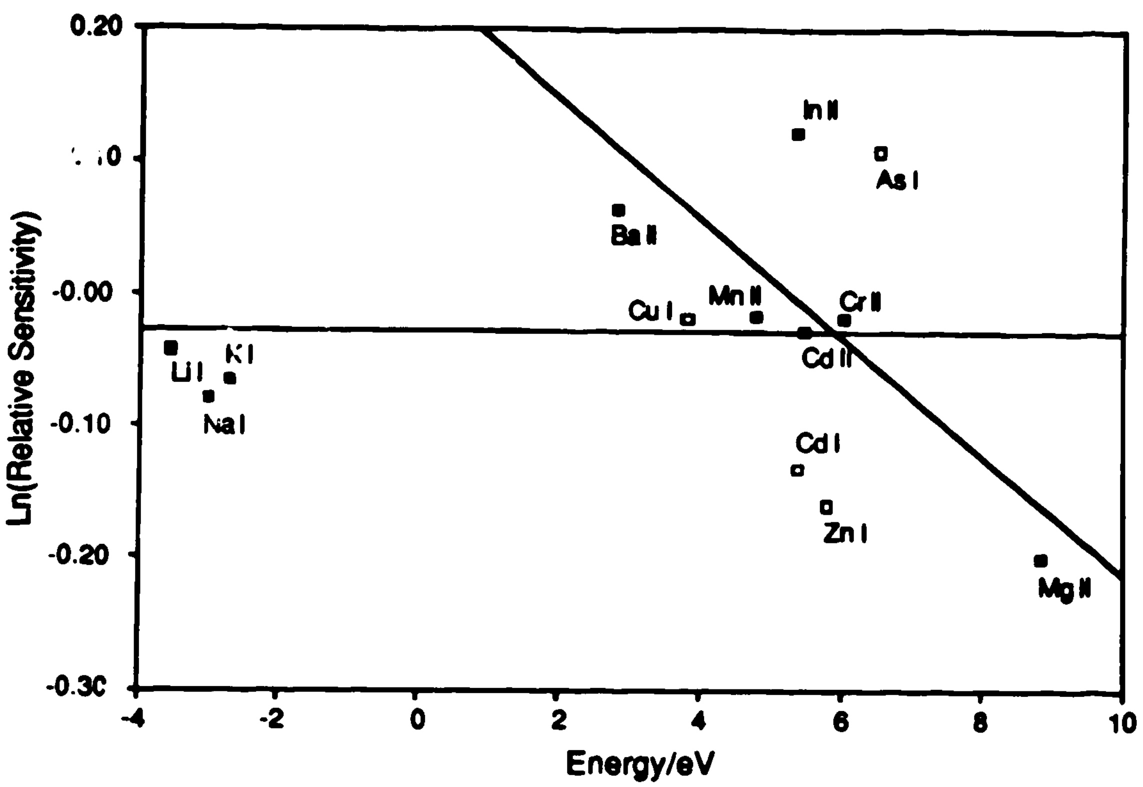


Figure 63 Relationship between relative sensitivity and energy of emitting state with respect to the ground state of the ion.Corrections have been made for partial ionisation for atom lines. Energy is measured with respect to ground state ion for highly ionised elements (■) and for partially ionised elements (□) energy is measured with respect to the ground state of the atom.

ionisation temperature. The corrections made to the partially ionised atom lines may not be complete but they do bring the atom lines into the same cluster as the ion lines.

The nebuliser gas flow rate was varied in the same way as for Ca. The flow profiles obtained are shown in Figure 64. The temperature profile is

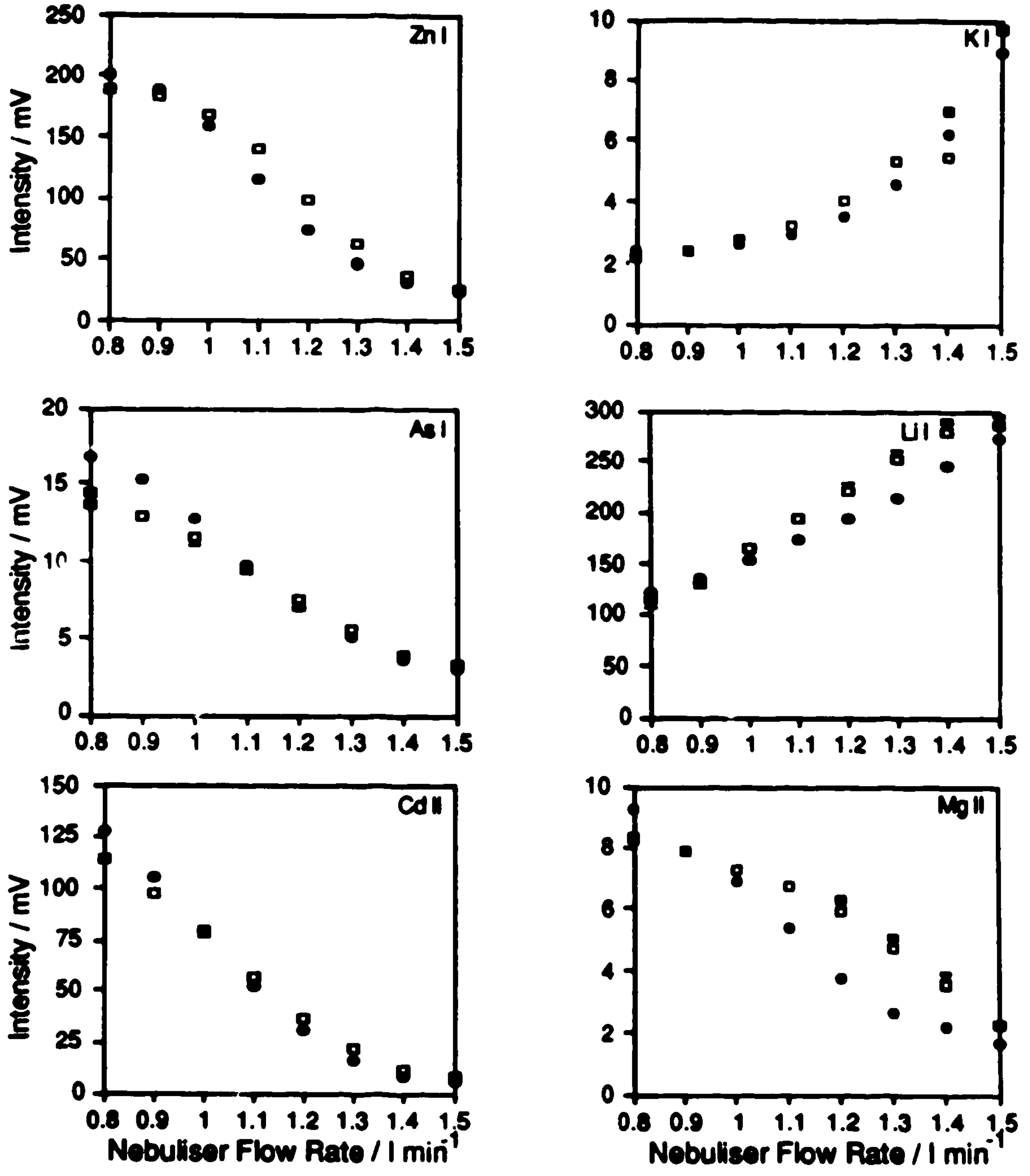


Figure 64 Emission intensity as a function of nebuliser gas flow rate for ethanol concentrations of (■) 0 µl ml^{-1}, (□) 1 µl ml^{-1} and (●) 10 µl ml^{-1}.

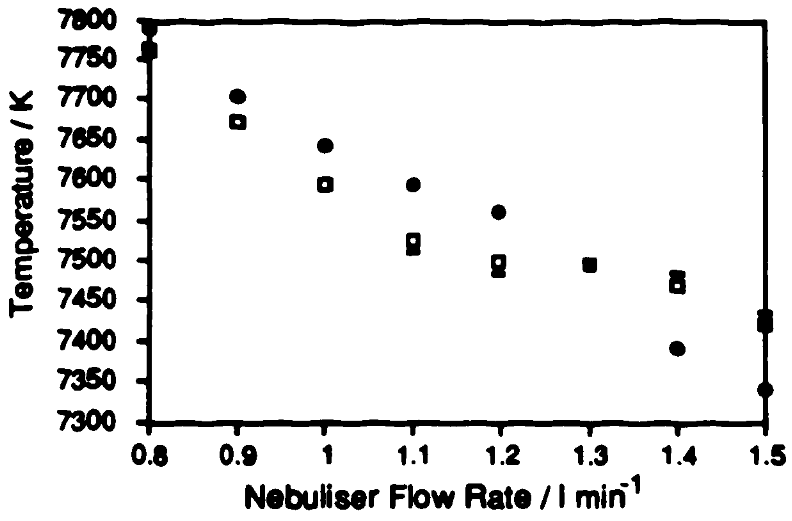


Figure 65 Ionisation temperature as function of nebuliser gas flow rate for ethanol concentrations of (■) 0 µl ml^{-1}, (□) 1 µl ml^{-1} and (●) 10 µl ml^{-1}.

shown in Figure 65. It is interesting to note that a plateau in the temperature at 1.30 l min^{-1} when no matrix component was present was obtained in both experiments.

It was found that at low flow rates that there was generally a signal enhancement with the addition of ethanol. This enhancement is reduced with increasing flow rate until at the higher flow rates it is replaced by a suppression. This behaviour is again the opposite from that of Ca, which showed suppressions at low flow rates and enhancements at higher flow rates.

The temperature profile also contrasts with that due to calcium in that at low flow rates there is an increase in temperature with the addition of ethanol. The change in temperature increases, then drops rapidly to zero at

1.30 l min^{-1} and thereafter is negative. The effect of calcium was always to decrease the temperature, the decrease was essentially constant except at the highest flow rates (>1.30 l min^{-1}).

The effect of flow rate on the energy plots is shown in Figure 66. These show that the ionisation temperature and the excitation temperature are decoupled at most flow rates. At the lower flow rates the magnitude of the temperature changes are small, then increase with flow rate till about 1.00 l min^{-1}, after which the change in ionisation temperature decreases until at higher flow rates the change in ionisation temperature is negative. As was found from the temperature profile calculated from the Cd ion/atom ratio, at the highest flow rates the changes in excitation and ionisation temperature are about the same.

The presence of two trends in these plots makes the identification of the intercept difficult. From the shifting of the plot it can be deduced that the intercepts of the plot are changing with flow rate. However, the changes are opposite to those observed for Ca. The intercept starts at a positive value and decreases with increasing flow rate. This can be interpreted, as was done before, as being due to a change in the number density of the analyte species in the observation zone.

Though the plots indicate that the assumptions on which they are based are not valid, they do, nevertheless, provide useful indications about the changes that occur in the plasma when ethanol is added to the analyte solutions. There would appear to be up to three effects. The first two are

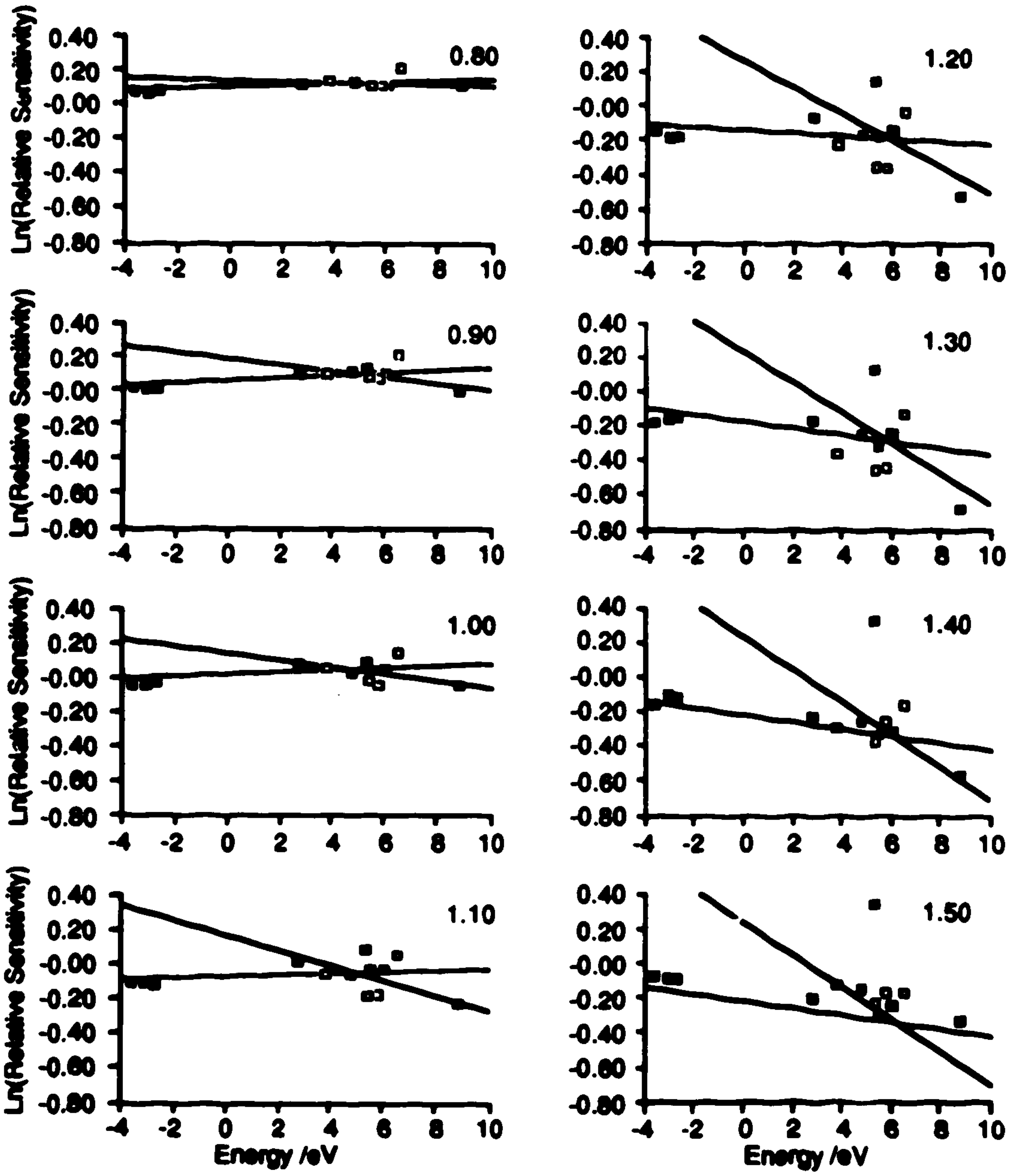


Figure 66 Relationship between relative sensitivity and energy of emitting state at different nebuliser gas flow rates. Energy is measured with respect to ground state ot the ion for highly ionised elements (■) and for partially ionised elements (□) energy is measured with respect to the ground state of the atom.

changes in the ionisation and excitation temperatures; at low nebuliser gas flow rates the former is increasing while the latter is decreasing. The third is a change in the number density of the analyte species in the observation zone.

5.4 Matrix Effects and Their Possible Origin.

The matrix effect observed for Ca is consistent with a concerted decrease in the excitation and ionisation temperatures. In other words, there is a suppression of both excitation and ionisation. From the work of Thompson and Ramsey [57] the degree of suppression is related to the energy required to convert the matrix from the solid state to that found in the plasma. They define the "total energy demand" as the sum of the crystal dissociation energy to the atomic state of the metal plus the first two ionisation potentials, each weighted by the degree of ionisation. A high correlation was found between the degree of suppression and this total energy demand. This would suggest that the suppressions observed are due to the energy required to dissociate the matrix not being available for excitation and ionisation of the analyte.

The results obtained here appear to confirm this observation. It could be further verified by the direct measurement of excitation temperatures using iron spectra in the presence and absence of a Ca matrix and also by extending the range of matrices studied. The effect of refractory oxide forming metals, e.g. La, would give information about the rôle of the precursors (crystal or molecular) to the atom in the matrix effect. The solutions used in this study had chloride present as the counter-ion. This leads to doubts as to whether it is chloride or oxide which is causing the suppression. By changing the counter-ion to other halides, nitrate, sulphate or phosphate, the importance of the oxide can be determined. If the observed suppression is found to be independent of the counter-ion, then the oxide is the main cause of the suppression, since the oxide would be formed by interaction with oxygen arising from the dissociation of water. If the dissociation of the crystal is the main factor, then calcium phosphate would be expected to give the largest suppression since it has the highest crystal lattice energy.

On the other hand the matrix effect observed for ethanol is more complex. The apparent behaviour is that there is a decrease in the amount of excitation and an enhancement of the degree of ionisation. The suppression of excitation could be due to the energy demand for dissociation of the ethanol molecule in a manner analogous to that of Ca. The increase in ionisation may be due to a charge transfer process involving either C^+ (1st Ionisation Potential 11.3 eV) or possibly a molecular carbon species, e.g. C_2^+. These highly energetic species may be formed in the hotter boundary regions of the plasma and diffuse into the central channel.

To verify this hypothesis it would be necessary to identify the carbon species present in the plasma when ethanol, or other organic solvents, are present. Also, a knowledge of their spatial distribution will be important. The effects of isomeric compounds would give information on the importance of the energy demand, since these will have different heats of formation and hence different energy demand, but the amount material introduced into the plasma would be the same and so the effect due to carbon on the degree of ionisation would be constant.

Chapter 6

Conclusions and Future Directions?

The automated ICP-AES solution sample analysis system is operational and meets many, though not all, the original design goals. It is still very much a prototype system. Though operation is reliable it still requires supervision. In view of the fact that use of an ICP requires the presence of an operator, this is more an inconvenience than a limitation. The reliability of the robotic operations could be improved with a redesign of some of the components. The software developed is a programming language which is reasonable for the ill defined problems for which the system may be used. However it is not necessarily user friendly. The use of an alternative programming language could lead to a more user friendly system, and by suitable choice of language also one with more flexibility. With suitable modifications the current system could be called by an external program which could be a possible method of integrating the system into an knowledge based program, e.g. expert system. However if the software were to be redesigned using an artificial intelligence language (e.g. Smalltalk), then integration of the robotic system into an expert system could be seamless.

The system was used to automate the analysis of aqueous samples. Rather than automate existing procedures the flexibility of the system was used to develop an alternative analysis strategy. Existing procedures use a batch mode method of analysis where the solution preparation and measurement operations are separated in time. The automated system could alternate between these two operations and so allow interaction between them. This was done by introducing feedback into the method of analysis, using the results of the measurements to guide the preparation of the solutions. The implementation of the feedback may not be optimal for all analysis methods and alternatives should be considered. Other analysis methods could be similarly automated. Standard addition methods are particularly tedious to perform manually and so would benefit from automation.

The automation of a matrix study shows that even some research procedures can be performed with this system. The method used was of an open loop type. The use of feedback in research tasks may require some artificial intelligence techniques to guide the decision making processes.

6.1 Hardware and Software Improvements.

In general operation of the system was reliable. However, there were occasional problems with alignment of the probe with the vials used to contain the solutions. The problem was a result of the width of the probe barrel and of difficulties in programming the eight positions of a tray to perform all the functions required; that of positioning the probe to dispense liquids into the vials and as a starting point for the picking up of the vials. A redesign of the probe could help to alleviate some of these difficulties. A design with more rigidity, with a narrower barrel and of shorter length, would have greater precision when entering the vials because the effective "target" area would be larger.

Also, the choice of a suitable coating material may help reduce cross contamination problems. These contamination problems could also be eliminated by a device which could wipe the outside surfaces of the probe, thus mimicking the procedures used in manual solution preparation when using a pipette. The use of a second robotic arm for this purpose would lead to problems coordinating the movement of both arms. A dedicated device would be simpler. Such a device might consist a loop of absorbent material into the centre of which the probe would be lowered. Once the probe had been lowered to the desired depth, the loop could be closed around it. Upon withdrawal the loop would wipe off any solution adhering to the outside. Once the probe had been removed the loop would open, and the absorbent material advanced so that fresh material would be used on the next operation of the device.

One of the operator's main tasks in the present system is the restocking of the trays with vials. A vial dispensing system would eliminate this task. Since the robot would remove only the vials it requires for the preparation of the solution, fewer tray positions would be necessary. This would simplify programming of the system. The other regular task of the operator is emptying of the dirty glassware. The ideal solution to this problem would be to have the glassware automatically washed, dried and returned to the vial dispenser, thus eliminating restocking of the dispenser.

The number of elements that can be analysed simultaneously is limited by the amount of space in the stock solution tray. The wash solution reduces the number from eight to seven. The inclusion of each auxiliary component (e.g. matrix component, sample in standard addition analysis or internal standards) further reduces the number of standards. The most obvious solution to this problem is to either increase the number of trays in the system or to increase the capacity of the trays. There is space to include more trays in the present system, but some reorganisation of the components in the system would be needed to maximise efficiency. For example, if two more trays for standards were made available up to 22 elements could be included. If all 22 elements were included in the solution, the robot would have to travel to the receiving container at least three times (the syringe capacity will allow a maximum of 9 components at 100μl with air gap). Alternatively, the capacity of the trays can be increased. This would require fewer trays in the system and so they could be placed closer together. However, this would require smaller capacity containers and thus more frequent refilling.

An alternative to using static trays would be to use trays mounted on indexing tables. This allows larger containers to be used and by keeping most of them outside of the work envelope a larger number can be stored. For example, when using a carousel type autosampling tray, a solution that is needed can be brought to the sampling position by rotating the tray. Static trays would then be used for solutions which need to be accessed frequently such as the wash solution.

The improvements proposed so far still have the problem of cross contamination of the stock solutions. One possible method of solving this would be to use two stage dispensing. The stock solutions could be stored in large reservoirs outside the work envelope of the robot. Dispensing from these reservoirs would be controlled by an electronically operated valve. The outlets from the valves could be shared by more than one stock solution if the common lines were properly flushed prior to dispensing. In this scenario the robot would remove a vial from the dispenser and take it to the appropriate outlet where a volume of stock solution, larger than required, would be dispensed. The vial would then be placed in the stock solution tray. Several stock solutions could be placed in the tray and the volumes required could then be transferred to the solution being prepared. If more components are to be added, the vials in the tray could be emptied and replaced by different stock solutions.

In the present system the unit operations of dilution and measurement are performed by the same robot. This leads to a bottleneck in processing in that preparation of the next solution, even when the composition is known, cannot proceed until measurement of the previous solution is complete. The inclusion of another robotic device, for example an intelligent autosampler, to control the sampling probe of the ICP would free the main robot for the task of solution preparation and transport to the measurement position. The software at present is designed for the single tasking nature of the hardware, with a small element of multitasking present in that commands are often sent to a different device before the action of the first has been completed. The serial multiplexer can be used to buffer messages and some of the other operational modes could also be used. The memory size of the unit may need to be increased to accommodate an increased volume of data. Also, the software would need to be modified so as to optimise the scheduling of different tasks and to identify the tasks that can be paralleled.

The programming language used in developing the system was not originally designed for multitasking systems and so does not include routines to simplify the scheduling of tasks. More modern languages may include these facilities. The integration of the robotic system into an artificial intelligence program would be smoother if the robotic system were written in one of the A.I. languages. These languages were also designed for rapid prototyping and flexibility, both attributes having been shown to be important requirements for robotic program development.

The natural view of the system developed is one of a collection of objects which achieve their tasks by sending messages to each other. This is the same paradigm used by the Object Oriented Programming System's (OOPS). The most complete implementation of an OOPS is Smalltalk. This program development system makes extensive use of windows and a graphical interface hence applications developed with it will inherit the same graphical interface, thereby making them more user friendly.

Object oriented programming systems do not rely on the traditional concept of a program operating on data but rather use the idea of objects to which messages are sent. The object is defined using an hierarchical structure of classes and sub-classes, in which a sub-class can inherit all the properties of its parent class. The class defines the properties of the object and the methods which define the action to take in response to a message. The objects in the robotic system used here include hardware such as the robot itself, the pumps, the spectrometer and the balance. Other objects that would be defined are solutions. For example, sending the message "make" to a solution would cause it to be made. This would be done by sending the appropriate messages to the robot and syringe pump objects. The messages sent to the logical object for the robot and the other hardware devices would cause appropriate command messages to be sent to the physical object through the serial ports.

6.2 Further Developments in Automated Analysis.

The analyses performed provided results which showed the feasibility of this feedback method. There were problems when the samples covered a wide dynamic range. Also, the length of time required for the analysis could lead to problems with instrument drift.

The method used to calculate the concentrations of the standards may not be optimal because it tends to produce essentially only four points on the calibration plot, one central cluster and three outlying points. If a very large numb[illegible]ration solutions were prepared following this algorithm then the analysis would tend to approximate that of an one point calibration.

The uncertainties calculated were always lower for the samples with higher concentrations and varied in an approximately linear fashion with sample composition. Hence if samples differ in composition by an order of magnitude the uncertainties similarly differ by an order of magnitude. Thus results for a more concentrated sample may be acceptable while those for a less concentrated one may not. One possible method of reducing problems due to this would be to use a weighted average of the sample concentrations instead of the simple average. The weights would be derived from the relative standard deviations of the sample compositions;

$$\bar{\hat{c}}_{sam} = \sum w_i \hat{c}_i \qquad (1)$$

where

$$w_i = \frac{s_i / \hat{c}_i}{\sum s_i / \hat{c}_i} \qquad (2)$$

where s_i is the standard deviation in the estimated concentration $\hat{c}_i$.

This would move the composition of the standards towards the sample with the largest relative uncertainty. This method of calculating the standards may possibly produce a calibration plot which has a more even distribution of points since as the composition of the standards moves towards one end of the range of samples the uncertainty in the composition at the other end will increase and so the composition of the solutions will start to move back towards the other end. Even if this method does not in itself cause a more even spread in the concentrations of the standards, a combination of this and the current method would produce two points on the curve, one close to the centre and one close to the sample with the largest uncertainty. An alternative would be to cycle through three standards, one prepared below the concentration of the lowest sample, one above the highest sample and one in the centre. A fourth could be added using the weighted average if desired.

These alternative strategies are only necessary if a large number of calibration points is envisaged, since they are used to counteract the excessive weighting of the central point of the calibration plot caused when the simple average value of the samples is used as the target composition of the standard. By limiting the number of standards to be prepared, a better balanced calibration plot can be obtained. The limiting of the number of standards is also desirable since it reduces the risk of errors due to instrumental drift.

Drift causes many problems in designing an analysis method. In the method used the samples were only measured once at the beginning of the analysis. This was necessary since the initial measurement is needed in order to perform the semi-quantitative analysis. No further measurement was made since if the samples are being remeasured due to drift, then the validity of the results obtained from previous standards must also be questionable and hence also the validity of the whole calibration plot. It is therefore more desirable to reduce the magnitude of the drift in the signals or to limit the analysis time by limiting the number of standards used.

The effect of drift could be reduced by using an internal standard. In this mode the analysis would probably start with a semi-quantitative analysis to identify the elements present. On the basis of this data, suitable internal standard(s) would be chosen. A solution containing the sample witl. added internal standard would be prepared and an iterative calibration would follow, possibly using calibration methods similar to those described above. With the present system this would reduce the number of elements that could be analysed to five, because space in the stock solution tray would be occupied by the internal standard solution and the sample. This again emphasises the need for an alternative method of storing the stock solutions.

When errors in the analysis are not due to drift but arise from a change in sensitivity induced by the sample matrix, then the method of standard additions is often used. A method to automate the standard addition method could be developed using the solution preparation methods used for the matrix studies, but with the matrix solution replaced by the sample. The method would use dilution to a constant volume rather than spiking of the sample by the standard. The procedure would be similar to that of the analysis method in that the size of the addition would be determined from the estimated concentrations of the elements present. An initial semi-quantitative analysis could be followed by an addition which is equal to the sample composition. Further additions would then be designed to prepare a standard addition calibration plot. Each point of the plot would be some multiple of the estimated concentration at the time that the point was added to the plot. The optimal set of multiples would have to be determined; an example of such a set might be 1x, 2x, 1/2x, 1/4x, 3/4x ...

The disadvantage of the standard addition method is that a calibration plot is prepared for each sample. It would be more efficient if a single calibration plot could be prepared for a batch of samples. However the results obtained may be inaccurate if there are matrix effects which invalidate the calibration plots obtained using aqueous standards. In this case matrix matched standards would have to be used. If the system was extended this could be done. There would be an initial semi-quantitative analysis of the sample, possibly including analytical methods other than the ICP to test for components for which there are no spectral channels, e.g. to test for the presence of organic solvents and possibly to identify the anions present.

From this semi-quantitative analysis possible matrix interferents could be checked for and a feedback controlled matrix effect study could be used to decide which components cause the most serious interference. The study would focus on the matrix concentrations actually present in the samples, and similarly would test the effect of the matrix on the components identified at the approximate levels present in the samples. The components and levels required for matrix matching the samples could then be determined. It may be expedient to compare the calibration obtained using matrix matched standards with that of a standard addition calibration for one of the samples. It may be necessary to run only a few points of the standard addition calibration to validate the matrix matching. This method would require access to a large number of solutions and a variety of instruments. Also, the decision making requirements may necessitate the use of an expert system. The expert system would decide which components are likely to cause interferences and, on evaluation of the matrix studies, which components are needed for matrix matching. It would also select the levels required to best correct for all the samples if the matrix composition differs among them. The system would only be required to correct for matrix effects and not to investigate the mechanisms responsible for causing them.

6.3 Points Arising from Automated Matrix Studies.

The matrix study showed the utility of the system in performing this type of research. As yet a full explanation of the effects observed is not possible. Further experiments are necessary. These include the measurement of the excitation temperatures by a standard method such as from the spectrum of iron. The excitation temperature changes may be too small to detect directly by this method, though it may be possible to detect them by the ratio of intensities. The effect that the matrices have on the spatial distribution of analytes in the plasma should also be studied. If carbon from the ethanol is causing increased ionisation by a charge transfer mechanism, then carbon ion emission may be visible at the boundary between the plasma and the analyte channel. The emission from other ions would also be increased in those regions.

With the current spectrometer it is not possible to perform these experiments. It would be necessary to integrate other instruments into the system. Ideally these added instruments should be controlled by their own computers and should be able to communicate with the rest of the system via a serial interface. With the serial interface and the control of the instrument distributed between the two computers, the definition of the control routines will be a relatively simple task. If the instrument requires direct interfacing to the host computer the definition of the control routines will be somewhat more complex and will be a move away from the design philosophy of basing the control of all the hardware or serial communications.

The results obtained also showed that it is possible to use this type of instrument for plasma diagnostics. It is normal to perform such diagnostics with single element solutions, e.g. iron, using instruments which provide a larger wavelength coverage for that element, such as a sequential scanning spectrometer, a photodiode array spectrometer or an interferometer. However, with appropriate manipulation of the data, to account for the different number densities and degrees of ionisation for different elements, it is possible to use multielement solutions to perform plasma diagnostics.

Multielement plasma diagnostics overcome some of the limitations that are present when using conventional single element methods. With single element methods the excitation energies that can be studied are limited to those present in the atom. Similarly, emissions from higher excited states are considerably weaker than those from lower energy states due the exponential nature of the population distribution This difference in emission intensity means that the measurement system requires a wide dynamic range. It also means that multiplexed measurement systems are of limited utility due to the multiplex disadvantage. However, by selecting appropriate elements so that the lower energy excitation energies are represented by one element and the higher energies by another, then by increasing the concentration of the weaker emitter a measurement system with a lower dynamic range, or even a multiplexed system, could be used.

The matrix study methods so far discussed have been of an open loop type. The introduction of feedback into this type of study is more problematic than is its use for analytical problems. In analysis a single figure of merit can be defined, e.g. precision, and optimisation of this can be used as a goal to direct how the feedback is used. The goal of a matrix study is either to minimise the interference or to identify the mechanism by which the interference occurs. In the former case an optimisation method could possibly be used. For the latter the goal is less well defined. An heuristic approach would probably be required. This could possibly proceed as follows. An initial experiment would be performed from which some trends would be sought. From these trends an initial model would be constructed and tested against the existing data. If the model is partially successful, further experiments could be designed and the model further refined. If it is unsuccessful an alternative model would be constructed from the trends found or other trends in the original data would be looked for. If still no successful model is found then an alternative set of initial experiments would be performed. This heuristic approach is essentially performing research using the scientific method.

6.4 Future Trends (Blue skies ahead).

John Donne said "No man is an Island, entire of itself". This is also true of automated systems. Even a simple automated system will be more useful when it is in communication with other systems in a network. A simple solution preparation system becomes a useful laboratory tool if it is

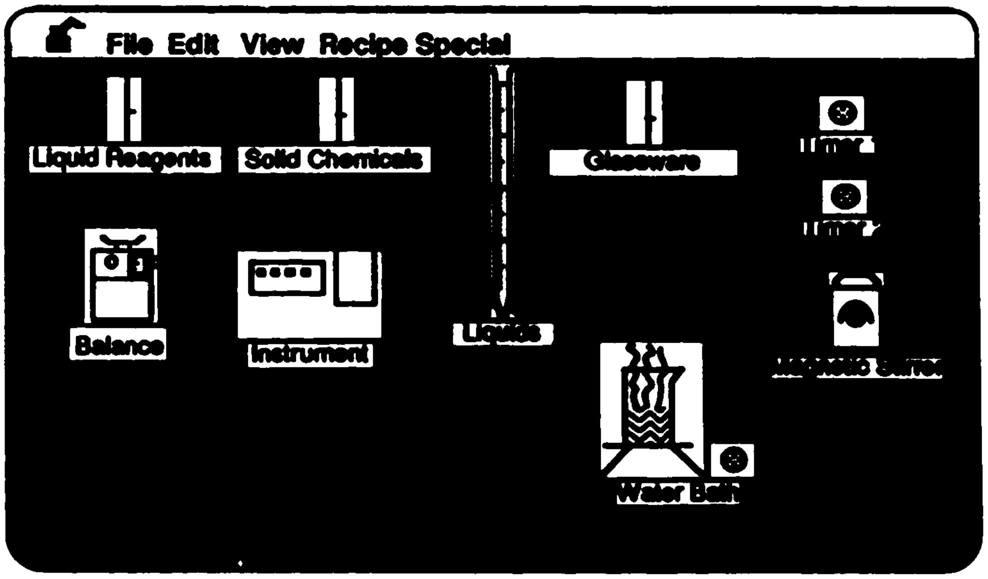
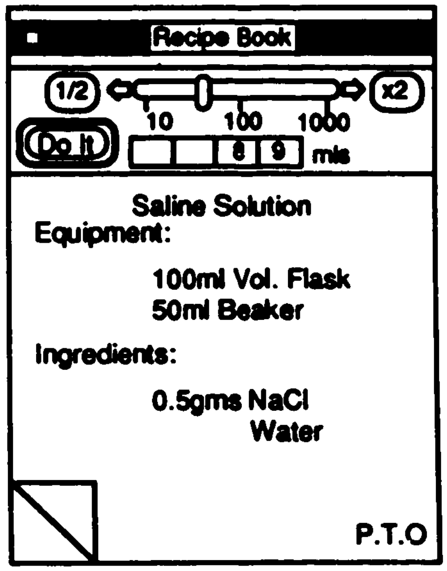


Figure 67 Benchtop Metaphor for chemical workstation.

connected into a computer network (e.g. AppleTalk) and requests for solutions could be sent to it from other computers in the network. The automated system becomes just like any other peripheral in the system, in that jobs could be sent to it, much like the print jobs sent to a Laserwriter.

The terminal in a network can be considered as a chemical workstation. Interaction with the chemical workstation will likely be by means of a graphical user interface similar to those on Macintosh computers. The desktop metaphor used on the Macintosh would be replaced by a benchtop metaphor such as the one shown in Figure 67. For example, the weighing of a chemical could be done by opening the chemical cupboard (solid chemicals) with a double click of the mouse on the icon. This would open a window showing the contents of the cupboard. The user would then proceed by selecting the chemical required, dragging it to the balance and typing in the weight required. The system would then transfer a weighed amount of the chemical into a beaker which had previously been transferred to the balance. The robot need not be shown on the benchtop because movement of the robot is implicit in the movement of objects on the benchtop. These operations are metaphors for the robot picking up the object selected and moving it to where the icon was dragged. The layout of the computer screen need bear no physica. elationship to the organisation of the automated system being controlled. This interaction with the system would allow the chemist to do many of the operations that he now does manually. Greater flexibility can be achieved by allowing these types of operations to be written down, stored and executed by the computer. To this end a recipe book could be used. An example of one such recipe is shown in Figure 68. Clicking on the "Do It" button would cause the recipe to be made. The amount prepared can be varied using the slider control.

Such a system could prove useful to the organic chemist, who could use the recipe book to store standard preparations. Manual interaction could be used for later stages of a synthesis. Methods for taking aliquots of the reaction mixture and transferring them to a spectrometer would allow the reaction to be monitored. The spectra returned would then be pasted directly into an electronic laboratory notebook. The electronic laboratory notebook would be a special type of wordprocessor where data entered could not be removed once it had been saved, and would provide options to date and time stamp each entry. This system could be used to automate some of the simpler analysis procedures. It would certainly be adequate to collect the initial data, which would then be transferred to the data analysis programs. The operator



Saline Solution
Equipment:
100ml Vol. Flask
50ml Beaker
Ingredients:
0.5gms NaCl
Water

Figure 68 The Recipe Book provides a method of storing procedures that can be used repeatedly. The volume required can be varied by changing the position of the slider or by clicking on buttons.

of this system would still be required to perform much of the interpretation of the data obtained.

The main limitation of the system described above is that the preparation and measurement steps have been separated. However, it has been shown that automation allows these to be combined in an analysis method by introducing a feedback loop. This allows information obtained from the measurement to be used to guide the preparation of the next sample. It has been shown that all the analysis methods considered are capable of being improved by the use of feedback. The inclusion of this feedback allows the system an element of adaptability in response to the sample which it is required to analyse. Thus the feedback introduces an element of intelligence to the automation of the analysis. A common element to the future development of this system is the inclusion of artificial intelligence techniques to guide the decision making element of the tasks, whether they be scheduling of the work load, deciding the composition of the matrix matched standards, or unravelling of the mechanism of matrix effects.

Robotic based automation and artificial intelligence are necessary companions in analytical chemistry. The experimental nature of the science mean that any expert system will require experimental data and so will greatly benefit from the ability to gather that data itself. Robotic based automation will require artificial intelligence for several reasons. As the systems become more complex the task of planning rapidly becomes non-trivial and algorithmic approaches become incapable of coping with the problem.

An expert system capable of performing some of the tasks of an expert in analytical chemistry will have four sources of information as shown in Figure 69. Primarily there will be information available from the user making the enquiry. The user can often furnish background information which can provide partial solutions to the problem; these can then be used to search for the final solution. The second source is an internal database which can contain fundamental chemical information and heuristic rules to direct a search for the solution by identifying the path which is most likely to end in a solution. The samples submitted for analysis are a vital source of information; the automated laboratory provides the means of accessing this information. Finally, in order to maintain the usefulness of the system access to the chemical literature is necessary. This is so that the system does not become obsolete when new analysis methods become available or when the

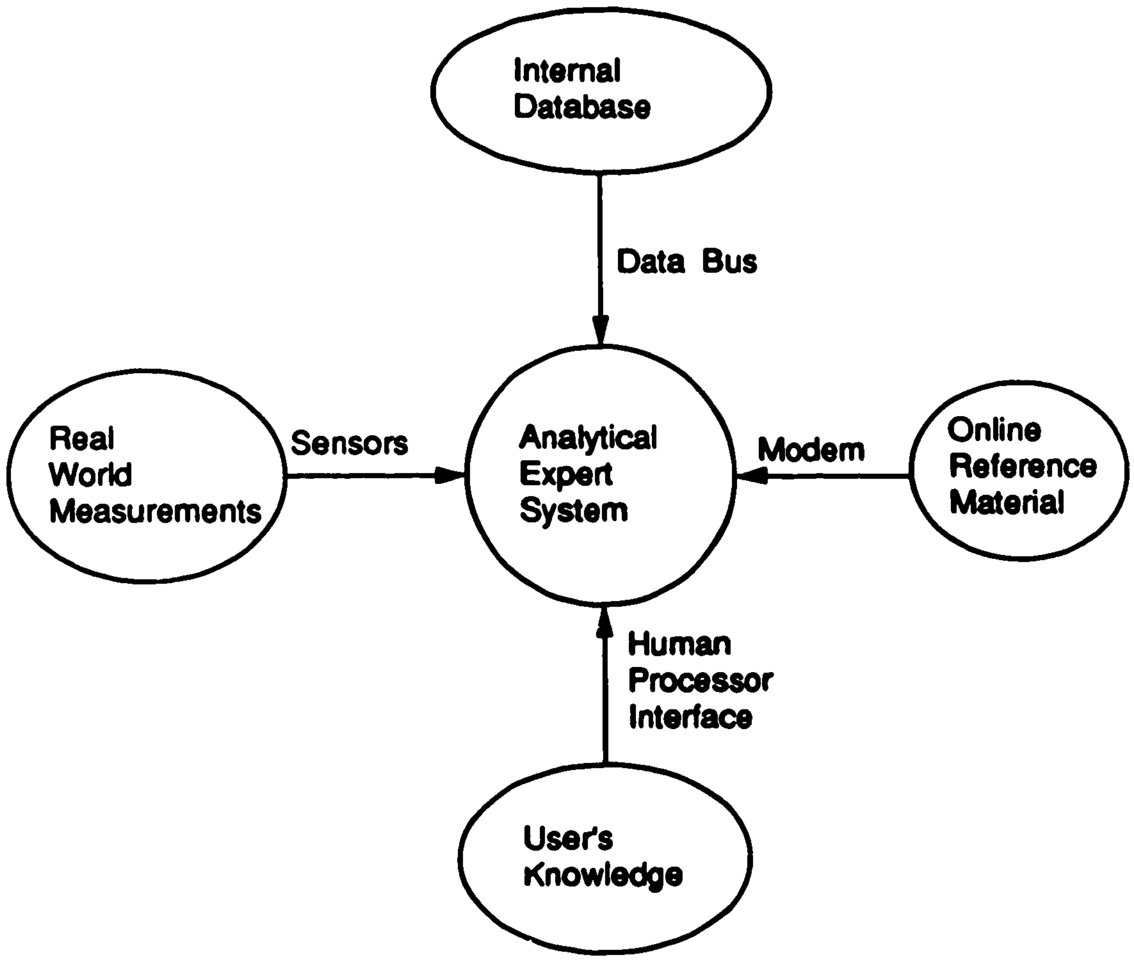


Figure 69 Information sources available to an Analytical Chemical Expert System.

fashions of the market change and the analysis of a new class of compounds is demanded.

The organisation and management of information from the first two sources is in the realm of artificial intelligence research. The knowledge necessary to produce an operational system can be supplied by chemists but the mechanisms used to manipulate this knowledge should be built by knowledge engineers. The design of the other components would require more chemical expertise. For the automated laboratory the design specifications would have to conform to the chemical needs of the system. The choice of instrumentation and the methods that can be employed would have to be able to meet the chemical goals of the system. In order to provide the fourth source of information the organisation of the chemical literature must be totally overhauled. Not only would Chemical Abstracts have to be organised so that it can be accessed by the automatic systems being proposed, but the information contained in the chemical journals will also have to be available in machine readable form.

In the analytical laboratory there is a division of labour between the laboratory supervisor and the technicians who perform the actual experimental work. Similarly, any complete knowledge-based system should consist of several smaller knowledge systems, each of which encapsulate the expertise necessary to perform the task required of it. Hence the analytical expert system of Figure 69 would consist of at least two separate sub-systems as shown in Figure 70. The supervisory part of the system would be responsible for deciding which method of analysis, e.g. a chromatographic or a spectroscopic method, would be most likely to produce the desired information. It would then analyse the results to determine if the solution has been found and also perform the necessary quality assurance analysis. This system would then start a decision making process to determine whether further analyses need to be performed to obtain further information and whether diagnostics are required to test the reliability of the instrumentation. If instrumentation is found to be non-functional, or performing below specification, then corrective action needs to be taken. This

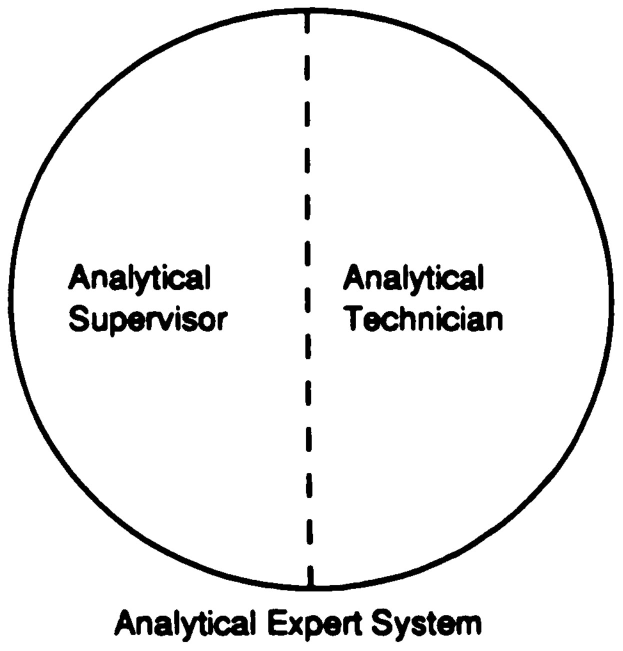


Figure 70 Division of Analytical Chemical Expert Sytem into two knowledge systems.

may mean modification of analysis methods or operating conditions. If modifications are ineffective then the instrument needs repair. A decision would have to be made between using an automatic repair system, if available, or calling a service engineer.

The technician part of the system will contain the expertise necessary to perform the experimental work and also to repair the instruments. It will be responsible for preparation of the sample and standards. It will have to decide on the most appropriate preparation methods. It would then perform the measurement and calibration of the instrument, selecting from a variety of calibration methods and instrumental techniques appropriate for the

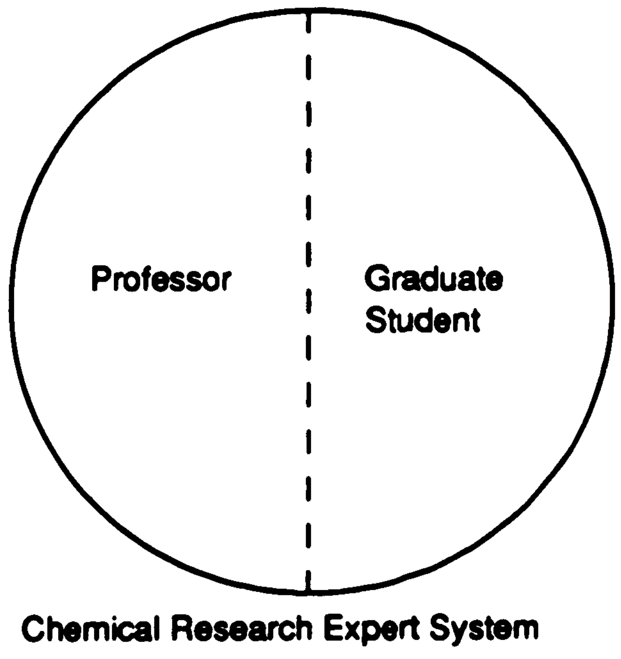


Figure 71 Analytical Chemical Expert System modified for Chemical Research problems.

problem. Initial data analysis would be performed and the results returned to the supervisor part of the system.

With some modifications of the analytical expert system it is conceivable that the system could be adapted to performing chemical research. The supervisory part would require major modifications in order to generate more speculative types of experimental requests, and the analysis of the results will be less routine. The modifications of the technician part will be less in that the main functions required will still be sample preparation and performing instrumental analysis. However, a larger variety of experiments be will required and it will be necessary to be able to customise the experiments. The modifications necessary to make the system more openended and capable of speculative action. The reasoning required will be less rigourous than that used in traditional expert systems. A model for such a system is shown in Figure 71.

The possibilities available with the combination of artificial intelligence and robotics are limited only by the imagination. For the systems envisaged here the A.I. techniques are not yet available to implement them. But when they are, it will be the equivalent to discovering "what the Universe is for and why we are here" since when this happens according to Douglas Adams "(the Universe) will instantly disappear and be replaced by something even more bizarre and inexplicable".

Bibliography

M. Bonner Denton; Analyst, 112, 347-353 (1987).

2) W.J. Hurst and J.W. Mortimer; "Laboratory Robotics" VCH Publishers Inc. New York. (1987).

3) W.E.Harris and B. Kratochvil; "An Introduction to Chemical Analysis" Saunders College Publishing, Philadelphia, (1981).

4) J. Ruzicka; "Flow Injection Analysis" 2nd Ed., J. Wiley and Sons, New York (1988).

5) J. A. Horner, A. P. Wade and M. W. Blades; J. Anal. Atom. Spectrsc., 3, 809-814 (1988).

6) H.V. Malmstadt and E.R. Fett; Anal. Chem., 26, 1348-1351 (1954).

7) B. Kratochvil and J. Nolan, Anal. Chem., 56, 586-589 (1984).

8) J.R. Chipperfield and D.E. Webster Anal. Chim. Acta, 197, 373-375 (1987).

9) S.R. Gambino; Anal. Chem., 43, 20A-27A (1971).

10) G. Beni; J. Electroanal. Chem., 140, 137-140 (1982).

11) G.D. Owens and R.J. Eckstein; Anal. Chem., 54, 2347-2351 (1982).

12) R. Dessy; Anal. Chem., 55, 1100A-1113A (1983).

13) R. Dessy; Anal. Chem., 55, 1232A-1242A (1983).

14) K.R. Lung and C.H. Lochmüller; J. Liq. Chrom., 9, 2995-3031 (1986).

15) B.J. McGrattan and D.J. Macero; Am. Lab., Sept. 1984, 16-24.

16) C.H. Lochmüller, K.R. Lung and M.R. Cushman; J. Chromatgr. Sci., 23, 429-436, (1985).

17) P.E. Last; Chem. in Brit., 1987, 1073-1075.

18) R.A. Bunce, P.M.G. Broughton, D.M. Browning, J.E.C. Gibbons and L.J. Kricka; J. of Auto. Chem., 11, 64-69 (1989).

19) G.W. Kramer and P.L. Fuchs; BYTE, 11(1), 263-284, (1986).

20) A.N. Papas, M.Y. Alpert, S.M. Marchese and J.W. Fitzgerald; Anal. Chem., 57, 1408-1411 (1985).

21) R.J. Eckstein, G.D.Owens, M.A. Baim and D.A. Hudson; Anal. Chem., 58, 2316-2320 (1986).

22) C.H. Lochmüller, K.R. Lung and K.R. Cousins; Anal. Lett., 18, 439-448 (1985).

23) C.H. Lochmüller and K.R. Lung; Anal. Chim. Acta, 183, 257-262 (1986).

24) W.A. Schlieper, T.L. Isenhour, and J.C. Marshall; Anal. Chem., 60, 1142-1145 (1988).

25) C.H. Lochmüller, T.L. Lloyd and K.R. Lung; Anal. Lett. 20, 1237-1246 (1987).

26) A. Montaser and D.W. Golightly (Editors); "Inductively Coupled Plasmas in Analytical Atomic Spectrometry", VCH Publishers, New York (1987).

27) Technical Reference Personal Computer AT 1502494, IBM Florida, USA (1984).

28) Leo Brodie; "Starting Forth" Prentice-Hall Inc., Englewood Cliffs, New Jersey (1981).

29) Multiport Controller H Series Owner's Manual #U140A019, Bay Technical Associates Inc., Mississippi, USA (1986).

30) A.J. Critchlow; "Introduction to Robotics" Macmillan Publishing Co., New York, (1985).

31) Y. Wang and S. Butner; AI Expert, 2(2), 26-32 (1987).

32) J.J. Craig; "Introduction to Robotics, Mechanics and Control" Addison-Wesley Publishing Co., Massachusetts, (1986).

33) J.R. Mühlbacher and F.X. Steinparz; J. of Microcomp. Appl., 7, 1-17 (1984).

34) I. Tabani and A. Montaser; Anal. Instr., 16, 385-398 (1987).

35) M. Gini; Robotica, 5, 235-246 (1987).

36) B.R. Donald; Artificial Intelligence, 31, 295-353 (1987).

37) M. Brady; Artificial Intelligence, 26, 79-121 (1985).

38) I. Tabani and A. Montaser; Anal. Instr., 16, 375-383 (1987).

39) W.A. Schlieper, T.L. Isenhour and J.C. Marshall; J. Chem. Inf. Comput. Sci., 27, 137-143 (1987).

40) Move master II Instruction Manual, Mitsubishi Electric Corporation, Tokyo, Japan (1983).

41) RM-501 RS-232 Supplement (80-01-0309), E&L Instruments, Connecticut, USA (1984).

42) Brian J M^{c}Grattan and Daniel J Macero; Am. Lab. 1986 16-24.

43) Unipump 300 Operator's Manual, SMI California USA.

44) WIZ Peristaltic Pump Instruction Manual 60-1613-124, Isco Inc., Nebraska USA. (1986).

45) Cheminterface Instruction Manual 60-1613-227, Isco Inc., Nebraska USA. (1984).

46) ARL Instrument Documentation, ARL, California USA.

47) L 420 P Installation and Operating Instructions, Sartorius GmbH, West Germany.

48) MP 8-4 Presentation Disk, Sartorius GmbH, West Germany.

50) E.D. Salin and G. Horlick; Anal. Chem., 52, 1578-1582 (1980).

51) J. Neter and W. Wasserman; "Applied Linear Statistical Models", Richard D. Irwin Inc., Homewood, Illinois. (1974).

52) M.A. Sharaf, D.L. Illman and B.R. Kowalski; "Chemometrics", John Wiley & Sons, New York. (1986).

53) L.H. Keith, R.A. Libby, W. Crummett, J.K. Taylor, J. Deegan and G. Wentler; Anal. Chem., **55**, 2210-2218 (1983).

52) Y. Shao; Private Communication.

55) M. Thompson and M.H. Ramsey; Analyst, **110**, 1413-1422 (1985).

56) M.Thompson, M.H. Ramsey, B.J. Coles and C.M. Du; J. Anal. Atom. Spectrosc., **2**, 185-188 (1987).

57) M. Thompson and M.H. Ramsey; J. Anal. Atom. Spectrosc., **1**, 185-193 (1986).

58) J. Lee, J.R. Sedcole and M.W. Pritchard; Spectrochim. Acta, **41B**, 217-225 (1986).

59) J. Lee, J.R. Sedcole and M.W. Pritchard; Spectrochim. Acta, **41B**, 227-235 (1986).

60) M.W. Blades and G. Horlick; Spectrochim. Acta, **36B**, 881-900 (1981).

61) G. Horlick and M.W. Blades; Appl. Spectrosc., **34**, 229-233 (1980).

62) M. Satake, G. Kano, B.K. Puri and S. Usami; Anal. Chim. Acta, **199**, 209-214 (1987).

63) M. Simonoff, C. Hammon, P. Moretto, Y. Llabador and G. Simonoff; Nucl. Inst. and Meth. in Phys. Res., **B31**, 442-448 (1988).

64) J.C. Sherlock, C.J. Pickford and G.F. White; Food Add. and Cont., 3, 347-354 (1986).

65) W. Xiaoru, H. Benli, S. Yaru and L. Chunlan; Fenxi Hanxue, 11, 1-7 (1983).

66) S. Yaru, L. Chunlan, W. Xiaoru, and H. Benli; Fenxi Hanxue, 11, 140-143 (1983).

67) P.W.J.M. Boumans; "Theory of Spectrochemical Excitation" Plenum Press, New York (1966).

68) M.W. Blades, B.L. Caughlin, Z.H. Walker and L.L. Burton; Prog. Anal. Spectrosc., 10, 57-109 (1987).

69) W.L. Wiese, M.W. Smith and B.M. Glennon; "Atomic Transition Probabilities: Hydrogen through Neon" Volume I NSRDS-MBS 4, Washington (1966).

70) W.L. Wiese, M.W. Smith and B.M. Miles; "Atomic Transition Probabilities: Sodium through Calcium" Volume II NSRDS-NBS 22, Washington (1969).

71) A.N. Zaidel, V.K. Prokof'ev, S.M. Raiskii, V.A. Slavnyi and E. Ya. Shreider; "Tables of Spectral Lines" IFI/Plenum Press, New York (1970).

72) J. Lam; Ph.D. Thesis, Department of Chemistry, University of Alberta, Edmonton, Alberta, (1988).

73) M.P. Lau; M.Sc. Thesis, Department of Chemistry, University of Alberta, Edmonton, Alberta, (1981).

Appendix A

Summary of Commands in ACS

This appendix contains a list of the comma. available in the ACS kernel. They are separated according to the devices which they control. The lists are in three columns, the first has the command as it would normally be used. If there are any parameters which appear after the command these have been italicised. Optional parameters appear in brackets. The second column shows how the word acts on the stacks, the parameters expected are shown first in the order in which they would be typed in, i.e. the top of the stack (TOS) is on the right, followed by an hyphen and then the results as they would appear on the stack, again TOS is on the right. When the floating point stack is used as well, it is separated from the integer stack by a double colon. The third column gives a brief description of what the word does.

General

INIT.PORT	b p s l prt -	Initialise port to given baud rate, parity, stop bits and word length.
RESTORE.PORT	prt -	Reinitialise serial port.
CKEY	- char	Get character from serial port.
CEMIT	char -	Send character to serial port.
$-LB	addr$ - addr$	Remove leading blanks from string.
CTYPE	addr$ n -	Send string to serial port.
CTEXT	delim -	Receive string using delim as terminator.
$-LX	addr$ char -	Remove all characters from string up to first occurance of char.
DIGIT?	char - flag	Test if char is a digit.
$NFIND	addr$ - addr$ n	Find find occurance of a digit in string.
$-LD	addr$ - addr$	Remove all characters up to first occurance of a digit.
VAL	addr$ - n	Convert string to number.
DATE.	-	Print date.

TIME.	-	Print time.
SELECT.DEVICE	device -	Switch serial output to device.
RESELECT.DEVICE	-	Select previous device.
GET.RETURN	addr$ com - addr$	Receive message from serial port and store in string.
WAIT.RETURN	addr$ com - addr$	Receive message without time out.
?Y/N	- flag	Get y/n reply from keyboard. Return true if y key pressed, false if n key pressed.
EXIST?	addr$ - flag	Test if file with name stored in string exists on disk.
MAKE.BAK	addr$ -	Make backup copy of file named in string if it already exists.
VLIST	-	List words defined in current vocabulary.
MU*	u2 u1 - u1*u2	Unsigned multiply.
WAIT	t -	Delay time in milliseconds.

Floating Point

F3PICK	f3 f2 f1 - f3 f2 f1 f3	Copy third number to TOS.

F4PICK	f4 f3 f2 f1 - f4 f3 f2 f1 f4	Copy fourth number to TOS.
F4ROLL	f4 f3 f2 f1 - f3 f2 f1 f4	Bring fourth number to TOS.
FARCCOS	Cos(θ) - θ	Calculate inverse cosine.
FVAL	addr$ - :: - f	Convert string to floating point number.
FP.STACK.CLEAR	fn ... f1 -	Clear contents of floating point stack.

Robot

END.PROGRAM	-	Stop programming of robot drive unit.
LABEL: *name*	-	Define label for current line number.
FLABEL: *name*	n -	Define label n lines ahead of current line number.
NEST	-	Move to mechanical nest position.
HOME	-	Set origin to current arm position.
ORIGIN	-	Move to origin.
IMOVE	a6 a5 a4 a3 a2 a1 -	Move joints by amounts given.
PMOVE	p -	Move to position number p stored in drive unit.
P1+	-	Move to next position stored in drive unit.

P1-	-	Move to previous position.
SET.POS	a6 a5 a4 a3 a2 a1 p -	Set position number p to coordinates specified.
PHERE	p -	Set position p to current arm position.
PCLEAR	p -	Clear position p from drive unit memory.
GRIP.SET	c2 t1 c1 -	Set current curve for opening and closing hand.
GRIP.OPEN	-	Open hand.
GRIP.CLOSE	-	Close hand.
GFLAG.SET	f -	Set open/close flag.
SPEED.SET	s -	Set speed of arm.
PAUSE	t -	Pause robot for t tenths of a second.
CLOCKED.IN	-	Read digital I/O with handshaking.
DATA.IN	-	Read I/O lines without handshaking.
CLOCKED.OUT	d -	Output data on I/O lines with handshaking.
DATA.OUT	d -	Output data without handshaking.

BIT.CASE	l ±t -	Branch to line l depending on state of bit t in input data.
FLAG.OUT	-	Toggle echoing of error signal to bit 7 of output.
GT	l d -	Branch to line l if input data greater than d.
EQ	l d -	Branch to l if data equal to d.
LE	l d -	Branch to l if data less than d.
NEQ	l d -	Branch to l if data not equal to d.
GOSUB	l -	Call subroutine starting at line l.
SUB.RETURN	-	Return from subroutine.
FOR.LOOP	n -	Start loop of n repeats.
FOR.NEXT	-	End of loop.
GOTO	l -	Unconditional branch to line l.
PROG.END	-	End of program.
PROG.NEW	-	Clear programs from memory.
RESET	-	Clear error state.
[illegible].LINE	l -	Remove line l from program.

DELETE.LINES	12 11 -	Remove lines from 11 to 12 from program.
PROG.RUN	-	Run program from start.
PROG.RUN.FROM	1 -	Start program from line 1.
PROG.WRITE	s -	Write contents of section s to ROM.
PROG.TRANSFER	s -	Read contents of ROM to section s.
PREAD	p -	Have coordinates of position p sent.
DREAD	-	Have input data sent.
POS.READ	p -	Read coordinates of postion p and store in LINE.BUFF.
DATA.READ	-	Get input data and store in LINE.BUFF.
WAIT.ROBOT	-	Wait until robot completes movement.
->START	:: $\gamma\beta$ z y x -	Transfer coordinates to variable START.
CART->ROBOT	- a6 a5 a4 a3 a2 a1 ::$\gamma\beta$ z y x -	Convert Cartesian coordinates to robotic joint parameters.
START->ROBOT	- a6 a5 a4 a3 a2 a1 :: -	Convert coordinates stored in START

		to joint parameters.
ROBOT->CART	a6 a5 a4 a3 a2 a1 - TEMPS	Convert joint parameters to Cartesian coordinates stored in variable TEMPS.
PROGRAM *name*	-	Create word for running program stored in drive unit, start compiling program.
CURRENT.POSITION		Variable for storing joint parameters of current arm position.
POSITION *name*	type mode - :: [$\gamma\beta$ z y x] -	Define position using specified mode and of specified type, use coordinates if using MAKE mode, use current arm position if using IMMED mode.
TO.MOVE	a5 a4 a3 a2 a1 -	Move to position given by joint parameters supplied.
MOVE.TO	posn -	Move to defined position.
GO.THERE	- :: $\gamma\beta$ z y x -	Move to coordinates given.
MOVEMENT *name*	posn ... pos1 n -	Define a sequence of positions.

RACK *name*	n - :: Δz Δy Δx -	Define a one dimensional array of positions starting a coordinates stored in START with offset between positions given on f.p.stack and with n positions.
TRAY *name* m n -	:: Δzm Δym Δxm Δzn Δyn Δxn	Define two dimensional array of positions mxn the 1,1 position being given in START with offsets between rows and columns given on f.p. stack.
UPDATE.CURRENT.POSITION -		Update current position variable using data read from drive unit.
SHOW.CURRENT.POS	-	Display coordinates and joint parameters of current position.
SHOW.POSITION	posn -	Display coordinates and joint parameters of position.
EDIT.POSITION	posn -	Modify joint parameters stored in position to be those of the

		current arm position.
GO.HOME	-	Move to home position.
MOVE.RELATIVE	- :: Δz Δy Δx -	Move relative to current position by amount given.
MOVE.UP	- ·: Δ -	Move up by amount given.
MOVE.DOWN	- :: Δ -	Move down.
MOVE.FORWARD	- :: Δ -	Move forward.
MOVE.BACK	- :: Δ -	Move back.
MOVE.LEFT	- :: Δ -	Move left.
MOVE.RIGHT	- :: Δ -	Move right.
ROLL.BY	- :: $\Delta\gamma$ -	Roll wrist through angle specified.
ROLL.TO	- :: γ -	Roll wrist to angle specified.
PITCH.BY	- :: $\Delta\beta$ -	Change pitch of hand by amount given.
PITCH.TO	- :: β -	Move hand to pitch angle given.
PFOPEN [*filename*]	-	Open file if no name specified then one will be asked for.
PLIST	-	List positions defined in current vocabulary.

PFSAVE *posnname*	-	Save position in file previously opened.
PFSAVE.ALL [*filename*]	-	Save all positions in current vocabulary in file named.
PFPRINT [*filename*]	-	Print contents of a position file.
PFBROWSE [*filename*]	-	Display contents of position file on screen.
PFLOAD [*filename*]	-	Load contents of position file.
INIT.ROBOT	-	Initialise robot and move to home position.
GOODNIGHT	-	Put robot to bed and leave program.

Balance

SELECT.WEIGHT.UNIT	unit -	Select unit of weight to be used.
SET.STABILITY	code -	Set stability of reading required.
LOCK.BALANCE.KEYBOARD	-	Prevent keyboard operation, not functional.
RELEASE.BALANCE.KEYBOARD	-	Allow keyboard operation.

BALANCE.CHECK	-	Perform internal check of electronics.
TARE.BALANCE	-	Set tare weight.
INTERNAL.CALIBRATE	-	Calibrate balance.
READ.BALANCE	- :: - weight	Get weight from balance.
GET.WEIGHT	- :: - weight	Get weight after a 5 second delay and when two weights which agree have been obtained.

Peristaltic Pump

PUMP.RESPONSE	- pad	Send CR and get message returned store in PAD.
INIT.PUMP	-	Initialise communications with pump.
RELAY.ON	-	Turn on relay.
RELAY.OFF	-	Turn off relay.
VALVE.SELECT	-	Select Relay1 for next relay commands.
RELAY.SELECT	r -	Select Relay1 or 2 for next relay commands.
WAIT.FOR.HIGH	-	Wait for selected input to go high before continuing.
WAIT.FOR.LOW	-	Wait for input to go low.

SELECT.INPUT	i -	Select input for next wait commands.
SELECT.OUTPUT	o -	Select output for next out commands.
BIG.BUBBLE	-	Use large air gap.
SMALL.BUBBLE	-	Use small air gap.
BUBBLE	-	Draw up an air gap.
PICKUP	-	Draw up solution for dispensing or dilution.
DELIVER	-	Deliver solvent when in dilute mode.
SET.CCW.DIRECTION	-	Set anticlockwise rotation.
SET.CW.DIRECTION	-	Set clockwise rotation.
PUMP.COUNT	n -	Wait for n pulses from pump.
PUMP1-	-	Decrement multiplier.
PUMP1+	-	Increment multiplier.
RECAL	-	Set pump to default calibration.
DILUTE	-	Set to dilute mode and RECAL.
DISPENSE	-	Set to dispense mode and RECAL.

PUMP	-	Set to pump mode and RECAL.
PUMP.RUN	-	Start pump running.
SELECT.PUMP	n -	Select pump 1 or 2 for next commands.
PUMP.SPEED	n -	Set pump speed and start running.
STOP.PUMP	-	Stop pump running.
BEEP.PUMP	-	Toggle end of operation beeps.
PUMP10*	-	Increment range.
PUMP10/	-	Decrement range.
PUMP.ERROR	n -	Force error condition.
PUMP.WAIT	n -	Pause for n tenths of a second.
PUMP.TRAP	n -	Execute following instructions when error n occurs.
PUMP.BEGIN	-	Start of a loop.
PUMP.REPEAT	n -	Repeat instructions in loop n times.
SELECT.INTERFACE	n -	Select interface unit for next commands.

REPEAT.RESPONSE	-	Send last message again.
HALT.ALL	-	Stop execution of all interface units.
HALT.PUMP	-	Stop execution of current interface unit.
CANCEL.LINE	-	Cancel current line.
SET.VIRTUAL.MULTIPLIER	- :: value -	Set pump to range and multiplier corresponding to value, (default=10.0).
SET.PUMP.RANGE.MULT	range mult -	Set range and multiplier to values given.
CHECK.RETURN.MESSAGE	addr$ - flag	Check if error has occured, true if no error.

Syringe Pump

SYRINGE.COMPLETION.MSG	- pad	Wait for pump to complete operation and read message returned.
SYRINGE.ERROR.CHECK	pad - pad	Check message for error, display error message and abort if necessary.
SYRINGE.PUMP.ONLINE	-	Start communication with pump.
SYRINGE.PUMP.SELECT	n -	Select pump for next commands.

SYRINGE.PUMP.DISPLAY.MSG	addr$ -	Display message on front panel of pump.
SET.SYRINGE.SIZE	n -	Inform pump of size of syringe installed.
SET.SYRINGE.SPEED	n -	Set speed of stroke.
SET.SYRINGE.PUMP.PARAMETER	n i -	Set parameter i to n.
GET.SYRINGE.PUMP.PARAMETER	i - n	Get value of parameter i.
WAIT.FOR.SYRINGE.EVENT	n -	Wait of event specified to occur.
SYRINGE.PUMP.OFFLINE	-	Stop communications with pump.
SYRINGE.VALVE.OPERATE	n -	Operate valve to position specified.
SYRINGE.DELIVER SYRINGE.UPTAKE		Constants to be used for operating the valve
CHANGE.SYRINGE	-	Move plunger to position in preparation for changing the syringe.
SYRINGE.HOME	-	Move plunger to home position.
SYRINGE.UP	n -	Move syringe up n µl.
SYRINGE.DOWN	n -	Move syringe down n µl.
INDEXED.SYRINGE.UP	i -	Move syringe up by amount specified in register i.

INDEXED.SYRINGE.DOWN	i -	Move syringe down by amount specified in register i.
SYRINGE.PRIME.PURGE	n -	Prime pump with n strokes.
INIT.SYRINGE.PUMP	-	Execute initialisation procedure for syringe pump.
SYRINGE.DISPENSE	n -	Dispense volume of n µl, where n is less than syringe volume.
SYRINGE.DEPRIME	n -	Empty syringe with n strokes.
SOLVENT.DISPENSE	n -	Dispense n µl of diluent, n may exceed syringe capacity.

ARL

RESTART.ARL	-	Restart program running on ARL.
BREAK.ARL	-	Halt program currently running on ARL.
WAIT.ARL	char -	Wait till ARL sends char.
SYNC.ARL	char2 char1 -	Perform synchronisation check with ARL.
RECEIVE.ARL	char2 char2 -pad	Receive message from ARL.
SEND.ARL	addr$ char2 char1 -	Send message to ARL.
N.SEND.ARL	n char2 char1 -	Send integer.
N.RECEIVE.ARL	char2 char1 - n	Receive integer.

F.SEND.ARL	#p char2 char1 - :: f -	Send real number with precision of p digits.
F.RECEIVE.ARL	char2 char1 - :: - f	Receive real number.
STORE.ELEMENTS	0 en...e1 -	Store elements to be measured in ELEMENT.ARRAY.
LIST.ELEMENTS	-	Display symbols of channels to be measured.
SEND.PREFLUSH	-	Send preflush time to ARL.
SEND.INTEGRATION	-	Send integration time.
SEND.REPLICATES	n -	Send number of replicate measurements required.
SEND.ELEMENTS	-	Send channel numbers to be measured.
ASPIRATE.SAMPLE	-	Switch valve to aspirate sample.
ASPIRATE.WATER	-	Switch valve to aspirate distilled water.
?ARL	-	Print name of program running on ARL.
ARL.PROG.RUN	n -	Run program on ARL.
ARL.QUIT	-	Stop runnning program and return to

command processor.

General Communications

RUN.TERM	-	Run terminal emulation program and re-initialise ports after quitting terminal program.

Appendix B

Listing of Solution Preparation and Measurement Routines

This appendix contains a program listing for the basic solution preparation and sample measurement routines.

The listing starts with the definition of the movements used, these pick up and replace the probes as well as move between fixed points. A simple path finding routine is then defined. Routines for system start up and shut down and routines to make solutions follow.

Finally there are routines to move a solution over to the measurement tray, routines to start the measurement of a sample in the tray, stop measurement and to empty the used solution. The final routine allows a forced flush of the solution uptake line to be performed during the pre-flush period.

```
CR
." Syringe Pump Solution Making Commands"
."  M. Stewart"
."  V1.11"
."  23rd August 1988" CR
DECIMAL
ROBOT

TO.PROBE.PATH.2
TO.PROBE.PATH.1
2
MOVEMENT TO.PROBE.PATH       ( Define movement from start position to probe stand )

HAND.PROBE.REMOVE
HAND.PROBE.LIFT
HAND.PROBE.CLOSED
HAND.PROBE.OPEN
PRE.HAND.PROBE.OPEN
5 MOVEMENT HAND.PROBE        ( Define MOVEMENT to remove hand probe from stand )

DUMP.WASTE.2
DUMP.WASTE.1
WASTE
PRE.WASTE
4 MOVEMENT EMPTY.CONTAINER   ( Define movement to empty container into waste )

PROBE.TO.WASTE.PATH.2
PROBE.TO.WASTE.PATH.1
2 MOVEMENT PROBE.TO.WASTE.PATH      ( Movement from stand to waste )

PROBE.IN.WASTE
PROBE.OVER.WASTE
PROBE.BEFORE.WASTE
3 MOVEMENT PROBE.WASTE              ( Movement to put probe into waste )

SAMPLE.TO.ARL.PATH.2
SAMPLE.TO.ARL.PATH.1
2 MOVEMENT SAMPLE.TO.ARL.PATH       ( Movement from sample tray to ARL tray )

SAMPLE.TO.ARL.PATH.2
TO.PROBE.PATH.1
2 MOVEMENT TO.ARL.PROBE.PATH        ( Movement from start to ARL probe stand )

ARL.PROBE.REMOVE
ARL.PROBE.LIFT
ARL.PROBE.CLOSED
ARL.PROBE.OPEN
PRE.ARL.PROBE
5 MOVEMENT ARL.PROBE                ( Movement to pick up ARL probe )

INNER.CIRCLE
DIRTY.DROP.PATH
DIRTY.DROP.OPEN
DIRTY.DROP.CLOSED
DIRTY.PATH.2
DIRTY.PATH.1
6 MOVEMENT DIRTY.GLASSWARE   ( Movement from waste to dirty glassware tray )

FORTH
CREATE PATH.POSITIONS        ( Waist positions to find best tie point )
1 1 STOCK.SOLUTIONS 3 + @ 1- ,
PROBE.PATH 3 + @ 1- ,
1 1 SERIAL 3 + @ 1- ,
1 1 SAMPLES 3 + @ 1- ,

: PATH.TO      ( waist_rotn - )
    [ ROBOT ]
    PATH.POSITIONS 0          ( Address of positions ar  counter )
    BEGIN
      3 PICK 3 PICK @         ( Find which of the four positions above is closest )
      <                       ( Compare positions )
      SWAP 1+ SWAP OVER       ( Increment counter and copy new value to TOS )
      4 > OR                  ( Check for overrun )
    WHILE
      SWAP 2+ SWAP            ( Increment pointer )
    REPEAT
    SWAP DROP SWAP DROP       ( Clean up stack )
    BEGIN-CASE                ( Having found closest position leave a tie point )
      1 CASE-OF
          STOCK.PATH
        ELSE
      2 CASE-OF
          PROBE.PATH
        ELSE
      3 CASE-OF
          SERIAL.PATH
        ELSE
      4 CASE-OF
          SAMPLE.PATH
        ELSE
      DROP
    END-CASE
    [ FORTH ]
;

: PATH.FIND    ( desired_posn - )
( This routine is used only for moving between probe stand, stock solution )
( tray, serial tray and samples tray )
    [ ROBOT ]
    CURRENT.POSITION @        ( Get waist rotation of current position )
    PATH.TO                   ( Get first tie point )
    OVER 3 + @                ( Get waist rotation of desired position )
    PATH.TO                   ( Get second tie point )
    OVER OVER                 ( Make copies )
    =                         ( Are they the same? )
    IF
      DROP DROP               ( If the tie points are the same ignore them )
    ELSE
      SWAP
      MOVE.TO                 ( Move to first tie point )
      MOVE.TO                 ( Goto the second tie point )
    THEN
    MOVE.TO                   ( Move to desired position )
    [ FORTH ]
;

: START.UP     ( - )
( Get the syringe pump ready for use )
   [ ROBOT ]
   CR
   ." Good Morning, Sir!" CR          ( Display greeting )
   INNER.CIRCLE MOVE.TO               ( Move to start position )
   FRWRD TO.PROBE.PATH                ( Move to probe stand )
   FRWRD HAND.PROBE                   ( Pick up probe )
   FRWRD PROBE.TO.WASTE.PATH          ( Move to waste disposal area )
   FRWRD PROBE.WASTE                  ( Put probe into waste disposal unit )
   SYRINGE.PUMP SELECT.DEVICE         ( Select syringe pump for output )
```

```
    [ PUMP.COMMANDS ]                   ( Select pump commands dictionary )
    INIT.SYRINGE.PUMP                   ( Initialise the syringe pump )
    5 SYRINGE.PRIME.PURGE               ( Prime the pump with 5 strokes of the syringe )
    SYRINGE.COMPLETION.MSG              ( Get completion message )
    SYRINGE.ERROR.CHECK DROP            ( Check for errors )
    ROBOT.PORT SELECT.DEVICE            ( Select robot for output )
    [ ROBOT ]                           ( Select robot dictionary )
    BCKWRD PROBE.WASTE                  ( Remove probe from waste disposal )
    BCKWRD PROBE.TO.WASTE.PATH          ( Return probe to stand area )
    BCKWRD HAND.PROBE                   ( Return probe to stand )
    [ ARL ]
    WIZ.PUMP SELECT.DEVICE              ( Start the WIZ pump )
    NORMAL.FLOW @
    [ PUMP.COMMANDS ]
    PUMP.SPEED
    PUMP.RESPONSE DROP
    [ FORTH ]
    ." Reporting for Duty." CR          ( Display completion message )
;

: SHUT.DOWN    ( - )
( Empty syringe of fluid and put robot to bed )
    [ ROBOT ]
    FRWRD HAND.PROBE                    ( Pick up probe )
    FRWRD PROBE.TO.WASTE.PATH           ( Take probe to waste disposal area )
    FRWRD PROBE.WASTE                   ( Put probe into waste disposal )
    WAIT.ROBOT                          ( Wait for robot to complete moves )
    SYRINGE.PUMP SELECT.DEVICE          ( Select syringe pump for output )
    [ PUMP.COMMANDS ]
    3 SYRINGE.PRIME.PURGE               ( Flush lines with 3 syringfuls of solvent )
    SYRINGE.COMPLETION.MSG              ( Get completion message )
    SYRINGE.ERROR.CHECK DROP            ( Check for errors )
    5 SYRINGE.DEPRIME                   ( Empty syringe )
    ROBOT.PORT SELECT.DEVICE            ( Select robot for output )
    [ ROBOT ]
    BCKWRD PROBE.WASTE                  ( Remove probe from waste disposal )
    BCKWRD PROBE.TO.WASTE.PATH          ( Return probe to stand area )
    BCKWRD HAND.PROBE                   ( Return probe to stand )
    BCKWRD TO.PROBE.PATH                ( Return to starting area )
    GOODNIGHT                           ( Execute robot shut down procedure )
    [ FORTH ]
;

: EMPTY.SYRINGE.CONTENTS     ( Solution_record - )
( Takes probe to receiving vessel and empties solution into it then )
( continues with the rest of the solution making procedure )
   [ ROBOT ]
   3 PICK 2+ @
   PATH.FIND                            ( Find path to receiving vessel )
   % 5 MOVE.FORWARD
   % 40 MOVE.DOWN                       ( Move down into it )
   WAIT.ROBOT                           ( Make sure robot has arrived before proceding )
   SYRINGE.PUMP SELECT.DEVICE           ( Select syringe pump for output )
   [ PUMP.COMMANDS ]
   31 SET.SYRINGE.SPEED                 ( Set syringe pump to maximum speed )
   SYRINGE.COMPLETION.MSG
   SYRINGE.ERROR.CHECK DROP
   SYRINGE.HOME                         ( Move syringe to top of stroke so expelling )
   SYRINGE.COMPLETION.MSG               ( contents )
   SYRINGE.ERROR.CHECK DROP
   5 SET.SYRINGE.SPEED                  ( Set syringe to low speed to prevent break up)
   SYRINGE.COMPLETION.MSG               ( of solution when being drawn into the probe )
   SYRINGE.ERROR.CHECK DROP
   ROBOT.PORT SELECT.DEVICE             ( Select robot for output )
   [ ROBOT ]
```

```
  % 40 MOVE.UP                      ( Bring probe out of vessel )
;

: COMP.VOL.CHECK      ( volume - volume )
    ( ." Checking Component Volume " )      ( Diagnostic statement )
    [ PUMP.COMMANDS ]
    DUP SYRINGE.SIZE @              ( Get syringe size )
    >                               ( Check if volume exceeds syringe capacity )
    IF
      SYRINGE.DISPLACEMENT @        ( If it does then transfer will require more )
      SYRINGE.SIZE @                ( than one trip )
      = NOT                         ( Check for anything in the probe )
      IF
        EMPTY.SYRINGE.CONTENTS      ( If there is then empty it )
      THEN
      SYRINGE.SIZE @
      /MOD                          ( Calculate how trips are needed and the amount )
      0                             ( left over )
      DO
        [ ROBOT ]
        OVER @ PATH.FIND            ( Move to stock solution )
        SYRINGE.PUMP SELECT.DEVICE
        [ PUMP.COMMANDS ]
        AIR.GAP @ SYRINGE.DOWN      ( Take up an air gap )
        SYRINGE.COMPLETION.MSG
        SYRINGE.ERROR.CHECK DROP
        ROBOT.PORT SELECT.DEVICE
        [ ROBOT ]
        % 90 MOVE.DOWN              ( Move down into solution )
        WAIT.ROBOT
        SYRINGE.PUMP SELECT.DEVICE
        [ PUMP.COMMANDS ]
        SYRINGE.SIZE @ SYRINGE.DOWN ( Draw up one full syringe of solution )
        SYRINGE.COMPLETION.MSG
        SYRINGE.ERROR.CHECK DROP
        ROBOT.PORT SELECT.DEVICE
        [ ROBOT ]
        % 90 MOVE.UP
        EMPTY.SYRINGE.CONTENTS      ( Empty into receiving vessel )
      LOOP
    ELSE
      [ PUMP.COMMANDS ]
      DUP
      SYRINGE.DISPLACEMENT @ AIR.GAP @ +
      - 0>                          ( Check if sufficient capacity remains )
      IF
        ( CR ." Volume Exceeds Capacity" ) ( Diagnostic statement )
        EMPTY.SYRINGE.CONTENTS      ( If not make some room )
      THEN
    THEN
    [ FORTH ]
;

: COMP.UPTAKE ( volume - )
    ( CR )
    ( ." Taking up Component " )
    ROBOT.PORT SELECT.DEVICE
    [ ROBOT ]
    % 90 MOVE.DOWN
    WAIT.ROBOT
    SYRINGE.PUMP SELECT.DEVICE      ( Select syringe for output )
    [ PUMP.COMMANDS ]
    DUP SYRINGE.DOWN                ( Pick up solution )
    SYRINGE.COMPLETION.MSG
    SYRINGE.ERROR.CHECK DROP
    -1 *
    SYRINGE.DISPLACEMENT +!          ( Update internal variable )
    ROBOT.PORT SELECT.DEVICE
    [ ROBOT ]
    % 90 MOVE.UP                     ( Remove probe from solution )
    WAIT.ROBOT
    4+                               ( Update pointer into solution record )
    [ PUMP.COMMANDS ]
    SYRINGE.PUMP SELECT.DEVICE
    AIR.GAP @ DUP SYRINGE.DOWN       ( Draw in an air gap )
    SYRINGE.COMPLETION.MSG
    SYRINGE.ERROR.CHECK DROP
    -1 *
    SYRINGE.DISPLACEMENT +!
    [ FORTH ]
;

: DUMP.SYRINGE.CONTENTS ( - )
( When things screw up this routine empties the probe into waste disposal )
   ROBOT.PORT SELECT.DEVICE
   [ ROBOT ]
   HAND.PROBE.REMOVE PATH.FIND      ( Find path to probe stand )
   FRWRD PROBE.TO.WASTE.PATH        ( Move to waste disposal area )
   FRWRD PROBE.WASTE                ( Put probe into waste disposal )
   SYRINGE.PUMP SELECT.DEVICE
   [ PUMP.COMMANDS ]
   SYRINGE.HOME                     ( Move syringe to top of stroke )
   SYRINGE.COMPLETION.MSG
   SYRINGE.ERROR.CHECK DROP
   ROBOT.PORT SELECT.DEVICE
   [ ROBOT ]
   BCKWRD PROBE.WASTE               ( Remove probe from waste disposal )
   BCKWRD PROBE.TO.WASTE.PATH       ( Return to probe stand )
   [ FORTH ]
;

: TIP.WASH    ( - )
 ( Rinse tip of probe in distilled water )
    [ ROBOT ]
    CURRENT.POSITION @              ( Get waist rotation of current position )
    PATH.POSITIONS @                ( Get waist rotation of stock tie position )
    <                               ( Are we at the stock tray? )
    IF
      2 4 STOCK.SOLUTIONS PATH.FIND ( Not at stock tray so use path find )
    ELSE
      2 4 STOCK.SOLUTIONS MOVE.TO   ( At stock tray so move directly there )
    THEN
    % 90 FDUP MOVE.DOWN             ( Lower probe into water )
    % -2 PITCH.BY
    5 0
    DO
      % 5 PITCH.BY                  ( Rock probe back and forth )
      % -5 PITCH.BY
    LOOP
    % 3 PITCH.BY
    MOVE.UP                         ( Raise probe from water )
    [ FORTH ]
;
```

```
: PICK.UP.SOLN        ( - )
 ( Pick up solution container )
   [ ROBOT ]
   % 260 MOVE.DOWN                 ( Move down to container )
   % 10 MOVE.BACK                  ( Center hand around container )
   GRIP.CLOSE                      ( Close hand )
   % 80 MOVE.UP                    ( Pick up container )
   [ FORTH ]
;

: PUT.DOWN.SOLN       ( - )
 ( Put down container reverse of the above )
   [ ROBOT ]
   % 80 MOVE.DOWN
   GRIP.OPEN
   % 10 MOVE.FORWARD
   % 260 MOVE.UP
   [ FORTH ]
;

: MIX                 ( posn - )
 ( Homogenise solution by a slow hand shake )
   [ ROBOT ]
   GRIP.ACTIVE @
   0 GRIP.ACTIVE !                 ( De-activate grip )
   SWAP
   PATH.FIND                       ( Move to position of solution )
   PICK.UP.SOLN                    ( Pick it up )
   % -60 ROLL.BY                   ( Rotate to start position )
   10 0
   DO
     % 120 ROLL.BY                 ( Shake solution 10 times )
     % -120 ROLL.BY
   LOOP
   % 60 ROLL.BY                    ( Rotate back to vertical )
   PUT.DOWN.SOLN                   ( Put solution back in position )
   GRIP.ACTIVE !                   ( Re-activate grip )
   [ FORTH ]
;

: MAKE.SOLUTION       ( solution_record - )
( Given the definition of a solution make it up )
    [ PUMP.COMMANDS ]
    SYRINGE.PUMP SELECT.DEVICE     ( Select syringe for output )
    5 SET.SYRINGE.SPEED            ( Set syringe speed to low value )
    SYRINGE.COMPLETION.MSG         ( Wait for completion message to be sent )
    SYRINGE.ERROR.CHECK DROP       ( Check for error codes, drop address of message )
    ROBOT.PORT SELECT.DEVICE       ( Select robot for output )
    [ ROBOT ]
    FRWRD HAND.PROBE               ( Pick up probe )
    SYRINGE.PUMP SELECT.DEVICE
    AIR.GAP @ DUP SYRINGE.DOWN     ( Draw up an air gap )
    SYRINGE.COMPLETION.MSG
    SYRINGE.ERROR.CHECK DROP
    -1 *                           ( Negate air gap size )
    SYRINGE.DISPLACEMENT +!        ( Update position of syringe )
    ROBOT.PORT SELECT.DEVICE
    WAIT.ROBOT                     ( Wait for robot to complete movement )
    DUP DUP 6+                     ( Make copies of record )
    SWAP @                         ( Get number of components )
    0
    DO
      DUP 2+ @                     ( Get volume of component )
      COMP.VOL.CHECK               ( Check if it fits into syringe )
      ?DUP                             ( Check if volume is non-zero )
      IF
        COMP.UPTAKE                    ( Take up the solution )
        TIP.WASH                       ( Rinse probe tip )
        WAIT.ROBOT                     ( Wait for robot before moving on )
      ELSE
        4+                             ( If volume zero do nothing )
      THEN
    LOOP
    DROP 2+ DUP @ UNDER                ( get position of receiving vessel )
    PATH.FIND                          ( Find path to receiving vessel )
    % 5 MOVE.FORWARD
    % 40 MOVE.DOWN                     ( Move down into it )
    WAIT.ROBOT                         ( Wait for robot )
    SYRINGE.PUMP SELECT.DEVICE         ( Select syringe for output )
    [ PUMP.COMMANDS ]
    31 SET.SYRINGE.SPEED               ( Set speed to maximum )
    SYRINGE.COMPLETION.MSG
    SYRINGE.ERROR.CHECK DROP
    SYRINGE.HOME                       ( Expel contents into vessel )
    SYRINGE.COMPLETION.MSG
    SYRINGE.ERROR.CHECK DROP
    2+ @ SOLVENT.DISPENSE              ( Get volume of solvent and dispense )
    ROBOT.PORT SELECT.DEVICE           ( Select robot for output )
    [ ROBOT ]
    % 40 MOVE.UP                       ( Remove probe for vessel )
    HAND.PROBE.REMOVE
    PATH.FIND                          ( Find path back to probe stand )
    BCKWRD HAND.PROBE
    MIX                                ( Homogenise solution )
    [ FORTH ]
;

: SOLUTION     ( Solution_defn - )
( Creates a named solution record )
    CREATE
      DUP , 1+       ( Store number of components in solution )
      0
      DO             ( Loop for each component + solvent volume and receiver posn )
       , ,           ( Store solution definition )
      LOOP
    DOES>            ( When name used leave pointer to solution on stack )
;

ARL DEFINITIONS
Cd1 CONSTANT Cd                        ( Equate Cd with Cd1 channel )
FORTH DEFINITIONS FORTH
CREATE STOCKS 48 ALLOT                 ( Create space for information about stock
solutions )

: DEFINE.STOCK.SOLUTIONS     ( Stocks info. number of stocks - )
( Store information about stock solutions, elements, conc. and position )
    STOCKS SWAP 3 *  ( There are three pieces of information for each solution )
    0
    DO
      DUP -ROT ! 2+  ( Store data in STOCKS )
    LOOP
    0 SWAP !         ( List is NULL terminated )
;
```

```
: FILL.SAMPLES.POSN   ( - )
( Store positions of vials in Samples tray going along rows )
   3 1                            ( Loop through the samples tray )
   DO
     5 1
     DO
       J I SAMPLES ,              ( Get position and store )
     LOOP
   LOOP
   0 ,                            ( NULL terminated list )
;

CREATE SAMPLES.POSN               ( Create space for list of vial positions )
FILL.SAMPLES.POSN                 ( Fill the list with the positions )

: FILL.SERIAL.POSN   ( - )
( Routine to fill list of positions for vials for intermediate solutions )
   3 1
   DO
     5 1
     DO
       J I SERIAL ,
     LOOP
   LOOP
   0 ,
;

CREATE SERIAL.POSN                ( Create space for list )
FILL.SERIAL.POSN                  ( Fill list of positions for intermediate solns. )

: GET.ELEMENT  ( - Channel_number )
( Get channel number for element to be included in soln from user )
  CR
  ." Please enter the next element for the solution "
  IN$ DROP                        ( Get element symbol )
  CONTEXT ,@                      ( Get current context, dictionary )
  ARL-SEG 14 + CONTEXT ,!         ( Switch context to use ARL dictionary )
  PLOAD SWAP CONTEXT ,!           ( Interpret symbol and restore context )
  STOCKS
  BEGIN                           ( Start search for stock solution for element )
    OVER OVER @ UNDER
    = SWAP 0= OR
    SWAP 6+ SWAP
  UNTIL
  6 - @ 0=                        ( Check to see if stock solution was found )
  IF                              ( If not abort with error message )
    ." No stock solution for " PAD $.
    ABORT
  THEN
;

CREATE AT.WEIGHTS 136 ALLOT       ( Create space to store atomic weights )
AT.WEIGHTS                        ( Store atomic weights for each channel )
% 91.22 DUP R32! 4+
% 87.62 DUP R32! 4+
% 137.34 DUP R32! 4+
% 58.71 DUP R32! 4+
% 26.9815 DUP R32! 4+
% 10.81 DUP R32! 4+
% 54.938 DUP R32! 4+
% 55.847 DUP R32! 4+
% 14.0067 DUP R32! 4+
% 30.9738 DUP R32! 4+
% 32.06 DUP R32! 4+
% 200.59 DUP R32! 4+
% 24.305 DUP R32! 4+
% 74.9216 DUP R32! 4+
% 118.69 DUP R32! 4+
% 28.086 DUP R32! 4+
% 12.011 DUP R32! 4+
% 50.9414 DUP R32! 4+
% 22.9898 DUP R32! 4+
% 95.94 DUP R32! 4+
% 51.996 DUP R32! 4+
% 121.75 DUP R32! 4+
% 72.59 DUP R32! 4+
% 40.08 DUP R32! 4+
% 65.37 DUP R32! 4+
% 63.546 DUP R32! 4+
% 107.868 DUP R32! 4+
% 207.2 DUP R32! 4+
% 6.941 DUP R32! 4+
% 47.90 DUP R32! 4+
% 112.4 DUP R32! 4+
% 112.4 DUP R32! 4+
% 114.82 DUP R32! 4+
% 39.102 DUP R32! 4+

( Constants used for concentration units )
1 CONSTANT pc                      ( Per cent concentration )
2 CONSTANT ppm
3 CONSTANT ppb
4 CONSTANT M
5 CONSTANT mM
6 CONSTANT uM                      ( microMolar )

( Variables to keep track of solution definition )
CREATE TOTAL.VOLUME 4 ALLOT
CREATE TOTAL.SERIAL.VOLUME 4 ALLOT
CREATE DEFAULT.VOLUME 4 ALLOT
CREATE CUMULATIVE.VOLUME 4 ALLOT
CREATE CUMULATIVE.SERIAL.VOLUME [illegible] ALLOT
CREATE DEFAULT.SAMPLE.SOLUTION [illegible] ALLOT     ( Create space for )
CREATE DEFAULT.SERIAL.SOL[illegible] ALLOT     ( two solution records )
CREATE SERIAL.DIL.RATIO 4 ALLOT
VARIABLE MINIMUM.VOLUME
VARIABLE NO.IN.SAMPLE
VARIABLE NO.IN.SERIAL
VARIABLE NO.ELEMENTS

: FIND.STOCK.SOLN     ( Element - Element Stock_posn )
 ( From channel number find position of stock solution )
    STOCKS                         ( Record of stock solution )
    BEGIN
      DUP @                        ( Get channel number of stock )
      3 PICK                       ( Copy element channel number )
      = NOT                        ( Compare channel numbers )
    WHILE
      6+                           ( While not equal goto next record )
    REPEAT
;

: GET.CONC     ( Element - Element :: - conc/ppm )
    CR
    ." Please enter the concentration of "
    [ ARL ]
    ELEMENT.SYMBOLS
    [ FORTH ]
    OVER
    0
    DO
      COUNT +
    LOOP
    $. SPACE                          ( Print element symbol )
    FIN# DROP                         ( Get concentration )
    CR
    ." Please enter the units" CR
    ." (% as pc, ppm, ppb, M, mM, or " 230 EMIT ." M as uM) "
    IN$ DROP                          ( Get units of concentration )
    PLOAD                             ( Interpret user response )
    BEGIN-CASE                        ( Convert to ppm )
      1 CASE-OF
         % 10000 F*                   ( Convert from % to ppm )
        ELSE
      2 CASE-OF
        ELSE                          ( Already ppm )
      3 CASE-OF
         % 1000 F/                    ( Convert from ppb to ppm )
        ELSE
      4 CASE-OF                       ( Convert from molarity )
         DUP 1- 4* AT.WEIGHTS + R32@ F*
         % 1000 F*                            ( molar )
        ELSE
      5 CASE-OF
         DUP 1- 4* AT.WEIGHTS + R32@ F*   ( millimolar )
        ELSE
      6 CASE-OF
         DUP 1- 4* AT.WEIGHTS + R32@ F*
         % 1000 F/                            ( micromolar )
        ELSE
      DROP
   END-CASE
;

: CALC.VOLUME  ( - Volume :: dilution_ratio - )
 ( From a dilution factor calculate volume of stock solution needed )
    TOTAL.VOLUME R32@ F* % 1E3 F*    ( Total volume in ml )
    F->                               ( Volume calculated in microlitres )
;

: GET.CONCENTRATION   ( Element - Element :: - Conc )
    GET.CONC DUP                      ( Get concentration )
    FIND.STOCK.SOLN DUP               ( Get record of stock solution )
    BEGIN
      2+ I16@ F/                      ( Calculate dilution ratio )
      FDUP MINIMUM.VOLUME I16@        ( Get minimum allowed volume )
      TOTAL.VOLUME R32@ % 1E3 F* F/  ( Calculate maximum dilution factor )
      FDUP F* F-                      ( Increase for one serial dilution )
      F0<                             ( Check if dilution required too much )
    WHILE
      FDROP OVER GET.CONC DROP        ( If it is re-enter concentration )
    REPEAT
;

VARIABLE NEXT.SAMPLES                 ( Keep count of how many vials have been used )
VARIABLE NEXT.SERIAL                  ( Ditto )
0 NEXT.SAMPLES !                      ( Intialise counter )
0 NEXT.SERIAL !                       ( Ditto )
```

```
: GET.NEXT.SAMPLES    ( - samples.posn )
   NEXT.SAMPLES @
   2* SAMPLES.POSN SWAP I@         ( Get position of next vial to use )
   DUP
   NOT
   IF                              ( If position NULL need to refill tray )
     OUTPUT @
     1 OUTPUT !                    ( Switch output to screen )
     7 EMIT 7 EMIT                 ( Ring bell twice )
     ." Samples tray empty please refill" CR
     ." Press any key when done" CR ( Print message )
     KEY DROP 1 NEXT.SAMPLES ! SAMPLES.POSN @
                                   ( Wait for keypress, reset samples counter )
     SWAP OUTPUT !                 ( Reset output )
   ELSE
     NEXT.SAMPLES 1+!              ( Increment vial counter )
   THEN
;

: GET.NEXT.SERIAL     ( - samples.posn )
( Similar routine to above but used for intermediate solutions )
   NEXT.SERIAL @
   2* SERIAL.POSN SWAP I@
   DUP
   NOT
   IF
     OUTPUT @
     1 OUTPUT !
     7 EMIT 7 EMIT
     ." Serial tray empty please refill" CR
     ." Press any key when done" CR
     KEY DROP 1 NEXT.SERIAL ! SERIAL.POSN @
     SWAP OUTPUT !
   ELSE
     NEXT.SERIAL 1+!
   THEN
;

% 10 TOTAL.SERIAL.VOLUME R32!

: CHECK.CUMULATIVE.VOLUME    ( volume/microlitres - volume )
   DUP
   ->F % 1E3 F/                    ( Transfer volume to f.p. stack and convert ml )
   CUMULATIVE.VOLUME DUP R32@ F+ FDUP      ( Add to cum. vol. )
   TOTAL.VOLUME R32@ F-
   F0>                             ( Check if volume exceeds total volume )
   IF
       ( If volume of stock solutions has exceeded total volume )
       ( abort with an error message )
     FDROP
     ." This solution cannot be prepared from the stock solutions present" CR
     ." as the total volume of stock solutions will exceed the total volume "
     ." required"
     ABORT
   THEN
   R32!
;
```

```
: CHECK.CUMULATIVE.SERIAL.VOLUME     ( volume/microlitres - volume )
( Same as above for intermediate solution )
   DUP
   ->F % 1E3 F/
   CUMULATIVE.SERIAL.VOLUME DUP R32@ F+ FDUP
   TOTAL.SERIAL.VOLUME R32@ F-
   F0>
   IF
     FDROP CR
     ." This solution cannot be prepared from the stock solutions present" CR
     ." as the total volume of stock solutions will exceed the total volume "
     ." required"
     ABORT
   THEN
   R32!
;

: ENTER.ON.SAMPLE.SOLUTION   ( Stock record - :: dilution_ratio - )
( From the record of position and concentration of stock solution )
( Enter new component on the current solution record )
   TOTAL.VOLUME R32@ F*             ( Convert dil. ratio to volume )
   % 1E3 F*                         ( Convert ml to ul )
   F->                              ( Transfer ul from f.p. stack to data stack )
   CHECK.CUMULATIVE.VOLUME          ( Check volumes )
   SWAP 4+ @                        ( Get position of stock solution )
   DEFAULT.SAMPLE.SOLUTION 6+
   NO.IN.SAMPLE @ 4* I!             ( Store position in solution record )
   DEFAULT.SAMPLE.SOLUTION 8 +
   NO.IN.SAMPLE @ 4* I!             ( Store volume in solution record )
   NO.IN.SAMPLE 1+!                 ( Increment number of components in solution )
;

( Only certain volumes of solution will be used )
( in the making of the intermediate solutions )
CREATE SERIAL.VOLUMES
100 , 200 , 500 , 1000 , 2000 , 5000 ,

: VOLUME.FOR.TWO.STEP.DIL    ( - Vol. :: Dil. Ratio - Dil. Ratio )
 ( Calculate optimum volume for a two step dilution )
 ( I.e. maximise both volumes )

    FDUP FSQRT                      ( Root the overall dilution factor )
    TOTAL.VOLUME R32@ F* % 1E3 F*   ( Calculate volume of intermediate )
    F->
;

: FIND.SYRINGE.VOLUME        ( Vol. - Vol. Addr. of Syr. Vol. )
 ( Find the address of the nearest volume to the ideal )
   SERIAL.VOLUMES
   BEGIN
     DUP @                          ( Compare ideal to allowed values )
     3 PICK -                       ( Search until find first that is )
     0<                             ( larger )
   WHILE
     2+
   REPEAT
;

: CHECK.END.VALUES    ( Addr. of Syr. Vol. - Addr. of Syr. Vol. Flag )
 ( Having found the volume check to see if it is at either end of the range )
    DUP SERIAL.VOLUMES =
    OVER SERIAL.VOLUMES 10 + =
    OR NOT
;
```

```
: FIND.NEAREST.VOLUME ( Vol. Addr. of Syr. Vol. Flag - Syr. Vol. )
 ( Find the volume which is nearer the ideal )
    IF                              ( If not an end value )
      DUP @                         ( Get larger volume )
      SWAP 2- @                     ( Get smaller volume )
      3 PICK OVER -                 ( Subtract smaller )
      3 PICK 5 PICK -               ( Subtract larger )
      -                             ( Take difference of differences )
      0>
      IF
        DROP                        ( Drop smaller and leave larger )
      ELSE
        PLUCK                       ( Drop larger and leave smaller )
      THEN
        PLUCK                       ( Remove ideal volume )
    ELSE
      PLUCK @                       ( If an end value just leave it )
    THEN
;

: FIND.BEST.SYRINGE.VOLUME   ( Vol. - Syr. Vol. )
 ( Find volume closest to the ideal )
    FIND.SYRINGE.VOLUME
    CHECK.END.VALUES
    FIND.NEAREST.VOLUME
;

: BACK.CALC.SER.DIL.RATIO    ( Syr. Vol. - :: - Dil. Ratio Ser. Dil. Rat )
 ( Calculate dilution factor from the volume used )
    ->F TOTAL.VOLUME R32@ % 1E3 F* F/
;

: CALC.VOL.OF.STD.IN.SERIAL  ( - Vol :: Dil. Rat. Ser. Rat. - Dil. Ser. Dil )
 ( From dilution factor of the intermediate solution calculate volume of )
 ( stock solution require in the intermediate )
    FOVER FOVER F/                  ( Divide overall dil. ratio by inter. dil. rat. )
    TOTAL.SERIAL.VOLUME R32@ % 1E3 F*
    F* F->                          ( Calculate volume of stock solution needed )
;

: CHECK.VOLUME.FOR.SERIAL    ( - Vol. Flag :: Dil. Ratio - Dil. R. Ser. Dil. R. )
 ( Check if using serial dilution if the amount of stock solution is less )
 ( than minimum allowed )
    VOLUME.FOR.TWO.STEP.DIL         ( Get ideal volume )
    FIND.BEST.SYRINGE.VOLUME        ( Find closest allowed volume )
    BACK.CALC.SER.DIL.RATIO         ( Calculate dilution ration used )
    CALC.VOL.OF.STD.IN.SERIAL       ( Calculate volume of standard required )
    DUP MINIMUM.VOLUME @ <          ( Check if less than minimum allowed )
    NOT
;

: ENTER.ON.SERIAL.DEFN       ( Stock Soln. Rec., Vol. - :: Dil. Ratio - )
    CHECK.CUMULATIVE.SERIAL.VOLUME
    SWAP 4+ @                       ( Get posn. of stock soln. )
    DEFAULT.SERIAL.SOLUTION 6+
    NO.IN.SERIAL @ 4* I!            ( Store position. )
    DEFAULT.SERIAL.SOLUTION 8 +
    NO.IN.SERIAL @ 4* I!            ( Store volume )
    NO.IN.SERIAL 1+!                ( Increment count )
    FDROP                           ( Drop Dil. Ratio )
;
```

```
: COMPARE.OLD.SER.DIL.RATIO  ( - Flag :: Ser. Dil. Rat. - Ser. Dil. Rat. )
 ( Check to see if existing dilution factor can be used )
   FDUP SERIAL.DIL.RATIO R32@
   F- F0< NOT
;

: USE.OLD.SER.DIL.RATIO      ( Std Vol. - :: Dil Rat. Ser. Dil. Rat. - Dil. Rat. )
   DROP FDROP FDUP              ( Drop new values )
   SERIAL.DIL.RATIO R32@ F/     ( Calculate dilution ratio required )
   TOTAL.SERIAL.VOLUME R32@ F*  ( Calculate new value of std. vol. )
   % 1E3 F* F->                 ( Convert to microlitres, move to integer stack )
   ENTER.ON.SERIAL.DEFN         ( Enter on definition record of inter. solution )
;

: UPDATE.FOR.NEW.SER.RATIO   ( - No. in defn. :: dil. rat. ser. dil. rat. - d. r )
   FDUP SERIAL.DIL.RATIO R32@
   FSWAP F/                            ( Calculate updating factor )
   DEFAULT.SERIAL.SOLUTION 8 + SWAP    ( Get definition record )
   F=0 CUMULATIVE.SERIAL.VOLUME R32!   ( Zero volume )
   0
   DO
     FDUP DUP DUP I16@                 ( Get Old Value )
     F* F->                            ( Calculate New Value )
     CHECK.CUMULATIVE.SERIAL.VOLUME    ( Check for overflow )
     SWAP ! 4+                         ( Store new value and goto next entry )
   LOOP
   DROP FDROP                          ( Clean up stack )
;

: START.OF.SERIAL.DEFN       ( - )
 ( Start new definition of an intermediate solution )
   F=0 CUMULATIVE.SERIAL.VOLUME R32!   ( Reset cumulated volume )
   GET.NEXT.SERIAL                     ( Get position of next vial )
   DEFAULT.SERIAL.SOLUTION 2+ !        ( Store position in definition )
;

: USE.NEW.SER.DIL.RATIO      ( Std. Vol. - :: Dil. Rat. Ser. Dil. Rat. - Dil. Rat. )
 ( Cannot use existing value )
  NO.IN.SERIAL @ ?DUP          ( Check for components already entered )
  IF
    UPDATE.FOR.NEW.SER.RATIO   ( Update volumes of existing comps. )
  ELSE
    START.OF.SERIAL.DEFN       ( First component so start new defn. )
  THEN
  SERIAL.DIL.RATIO R32!        ( Store new value )
  ENTER.ON.SERIAL.DEFN         ( Enter component into defn. )
;

: ENTER.ON.SERIAL.AUX        ( Std. Vol. - :: dil. ratio, ser. dil. ratio - )
   COMPARE.OLD.SER.DIL.RATIO   ( Check if existing value can be used )
   IF
     USE.OLD.SER.DIL.RATIO     ( Yes, use it )
   ELSE
     USE.NEW.SER.DIL.RATIO     ( No, use new value )
   THEN
;
```

```
: ENTER.ZERO   ( Std. Vol. - :: dil. ratio, ser. dil. ratio - )
 ( Enter zero volume on definition )
    DROP FDROP
    NO.IN.SERIAL @ 0=              ( New intermediate solution )
    IF
      START.OF.SERIAL.DEFN         ( Need to start new defn. )
    THEN
    0 ENTER.ON.SERIAL.DEFN         ( Enter zero volume )
;

: ENTER.ON.SERIAL.SOLUTION   ( Std. Vol. - :: dil. ratio, ser. dil. ratio - )
    CHECK.VOLUME.FOR.SERIAL
    IF
      ENTER.ON.SERIAL.AUX
    ELSE
      ENTER.ZERO
    THEN
;

( Create space for a record of the contents of the solution )
CREATE SOLUTION.CONTENTS 33 ALLOT

: FIND.ENTRY.ON.SAMPLE       ( Posn_addr - component_recd. )
 ( Find the entry for a component on the defn. of the sample solution )
    DEFAULT.SAMPLE.SOLUTION 6+
    BEGIN
      DUP @                        ( Compare addr. of component with that of )
      3 PICK                       ( the sought for component )
      = NOT
    WHILE
      4+                           ( Goto next component while not equal )
    REPEAT
;

: FIND.ENTRY.ON.SERIAL       (posn_addr - component_recd )
 ( Search for component on defn. of serial solution )
    DEFAULT.SERIAL.SOLUTION 6+
    BEGIN
      DUP @
      3 PICK
      = NOT
    WHILE
      4+
    REPEAT
;

: PRINT.SUMMARY       ( - )
   PRINT                                  ( Send output to printer )
   SOLUTION.CONTENTS
   BEGIN
     DUP @                                ( While channel number not 0 )
   WHILE
     DUP @                                ( Get channel number )
     [ ARL ]
     ELEMENT.SYMBOLS
     [ FORTH ]
     SWAP
     0
     DO
       COUNT +                            ( Get number of characters and skip over )
     LOOP
     $. SPACE                             ( Print element symbol )
     2+ DUP @                             ( Get route of component to solution )
     BEGIN-CASE
```

```
      1 CASE-OF                           ( Direct dilution )
         DUP 2- @                         ( Get channel number )
         FIND.STOCK.SOLN
         DUP 2+ I16@ 4+ @ PLUCK           ( Put conc. on f.p. stack and get posn. )
         FIND.ENTRY.ON.SAMPLE
         2+ I16@ DROP F*                  ( Get volume of soln. and mult. conc. )
         TOTAL.VOLUME R32@ % 1E3 F*       ( Convert total vol. to ul )
         F/ FDUP 4 OUT ,@ - SPACES        ( Calculate conc. and TAB to col. 4 )
         7 6 F.R ." ppm"                  ( Print conc. )
         DUP 2- @ 1- 4* AT.WEIGHTS + R32@ F/      ( Calculate mM )
         14 OUT ,@ - SPACES 7 6 F.R ." mM" CR     ( Print mM conc. )
      ELSE
      2 CASE-OF
         ( Serial dilution )
         DUP 2- @
         FIND.STOCK.SOLN
         DUP 2+ I16@ 4+ @ PLUCK           ( Put conc. on f.p. stack )
         FIND.ENTRY.ON.SERIAL
         2+ I16@ DROP F*
         TOTAL.SERIAL.VOLUME R32@ % 1E3 F* F/     ( Conc. in inter. soln. )
         DEFAULT.SAMPLE.SOLUTION 4+
         NO.IN.SAMPLE @ 4* I@ ->F F*              ( Get volume of inter. soln. )
         TOTAL.VOLUME R32@          ( Don't need factor 1000 since conc. in ppb )
         F/ FDUP 4 OUT ,@ - SPACES 7 6 F.R ." ppb"      ( Print conc. in ppb )
         DUP 2- @ 1- 4* AT.WEIGHTS + R32@ F/            ( Calculate uM conc. )
         14 OUT ,@ - SPACES 7 6 F.R 230 EMIT . " M" CR
      ELSE
    END-CASE
    2+                                    ( Increment pointer to next component )
  REPEAT
  DROP
;

: SOLUTION.PRINT       ( solution_record - )
( Print position and volumes of components in a solution )
   CR
   DUP 2+ @ OVER 4+ @         ( Get position of solution vial and volume of solvent )
   ->F % 1E3 F/               ( Transfer to f.p. stack and convert to ml )
   ." Solvent volume is " 7 6 F.R ." ml" CR
   ." Position Address of receiving vessel is " . CR
   DUP 4+ SWAP @              ( Get number of components )
   0
   DO
     2+ DUP @                 ( Get position of next component )
     ." Position of component " I 1+ . ." is " . CR
     2+ DUP @                 ( Get volume of stock solution )
     ." Volume of component " I 1+ . ." is " . 230 EMIT ." l" CR
   LOOP
   DROP
;
```

```
: CHECK.GRIP   ( - )
 ( Check if robot is holding anything if it is give the user the opportunity )
 ( to put it down )
    [ ROBOT ]
    BEGIN
      G.FLAG @
      3 =                          ( Check if grip is closed )
    WHILE
      CRT
      ." The Robot is holding something " CR
      ." Please put object back from where you got it " CR
      IN$ DROP                     ( Get commands from user )
      PLOAD                        ( Interpret them )
    REPEAT
    [ FORTH ]
;

: DUMP.SOLUTION       ( posn - )
 ( Empty contents of a solution )
    [ ROBOT ]
    CHECK.GRIP                     ( Check if robot is holding anything )
    GRIP.ACTIVE @ SWAP
    0 GRIP.ACTIVE !                ( De-activate grip )
    DUP PATH.FIND                  ( Move to solution to be emptied )
    PICK.UP.SOLN                   ( Pick it up )
    3 + @ PATH.TO MOVE.TO          ( Move to posn again )
    SAMPLE.TO.ARL.PATH.2 MOVE.TO   ( Move to solution disposal area )
    GRIP.ACTIVE !                  ( Re-activate grip )
    WAIT.ROBOT                     ( Wait for robot to arrive )
    FRWRD EMPTY.CONTAINER          ( Empty solution )
    WAIT.ROBOT                     ( Wait for robot to finish )
    [ ARL ]
    DRAIN @ WAIT                   ( Wait for solution to drain )
    [ ROBOT ]
    FRWRD DIRTY.GLASSWARE          ( Drop container into garbage )
    FRWRD TO.PROBE.PATH            ( Return to probe position )
    [ FORTH ]
;

: REMAKE.SERIAL       ( - )
 ( Remaking solution with new intermediate solution )
    DEFAULT.SERIAL.SOLUTION 2+ @
    DUMP.SOLUTION                          ( Get rid of old intermediate )
    GET.NEXT.SERIAL DUP                    ( Get position of next vial )
    DEFAULT.SERIAL.SOLUTION UNDER 2+ !     ( Enter onto defn. )
    SWAP DEFAULT.SAMPLE.SOLUTION 4+ NO.IN.SAMPLE @
    2* + !                                 ( Update sample soln. defn. )
    MAKE.SOLUTION                          ( Make new intermediate )
    ( SOLUTION.PRINT )
;

: REMAKE.SAMPLE ( - )
 ( Remake sample solution )
    GET.NEXT.SAMPLES                       ( Get position of next vial )
    DEFAULT.SAMPLE.SOLUTION UNDER 2+ !     ( Store position in defn. )
    MAKE.SOLUTION                          ( Make new solution )
    ( SOLUTION.PRINT )
;
```

```
: GET.ID      ( - )
 ( Get identification for the solution )
    CR
    ." Please enter solution i.d. "
    IN$
    PRINT                           ( Print name of solution )
    $. CR
    CRT CR
;

: INIT.PREP   ( - )
 (  nitialise variables for defn. of new solution )
     FP.STACK.CLEAR                        ( Clear floating point stack )
     F=1 SERIAL.DIL.RATIO R32!             ( Set dilution factor to 1 )
     NO.IN.SAMPLE 0!                       ( Zero component counters )
     NO.IN.SERIAL 0!
     F=0 FDUP CUMULATIVE.VOLUME R32!       ( Reset cumulated volumes )
     CUMULATIVE.SERIAL.VOLUME R32!
;

: GET.NO.OF.COMP      ( - n )
 ( Get number of components in solution )
    ." How many elements to be included in the solution "
    IN#
    BEGIN
      DUP 8 >                       ( Only 8 elements can be used )
    WHILE
      CR
      DROP ." This is too many elements. I can only deal with upto 8" CR
      ." Please try again " IN#
    REPEAT
    CR
;

: GET.VOLUME   ( - )
 ( Get total volume of solution to be made )
    ." The total volume of solution normally prepared is "
    TOTAL.VOLUME R32@ 5 6 F.R ."  ml" CR
    ." Is this OK " ?Y/N
    NOT
    IF
      TOTAL.VOLUME DUP R32@
      DEFAULT.VOLUME R32! CR
      ." Please enter the required volume "
      FIN# DROP
      BEGIN
        FDUP % 18 F-                ( Maximum of 18 mls is used to )
        F0>                         ( prevent possible overflow )
      WHILE
        CR FDROP
        ." The volume entered exceeds the bottle capacity of 18ml " CR
        ." Please try again "
        FIN# DROP
      REPEAT
      R32!
    THEN DUP
;
```

```
: FINISH.SERIAL       ( - )
 ( Add final details to definition )
    NO.IN.SERIAL @
    DEFAULT.SERIAL.SOLUTION !          ( Enter number of components )
    TOTAL.SERIAL.VOLUME R32@
    CUMULATIVE.SERIAL.VOLUME R32@
    F- % 1E3 F*                        ( Calculate volume of solvent )
    DEFAULT.SERIAL.SOLUTION 4+ I16! ( Enter volume of solvent )
    SERIAL.DIL.RATIO R32@              ( Get dilution factor of interm. soln. )
    DEFAULT.SERIAL.SOLUTION 2-
    ENTER.ON.SAMPLE.SOLUTION           ( Enter interm. soln. on defn. )
;

: FINISH.SAMPLE       ( - )
 ( Add final details to definition )
    NO.IN.SAMPLE @
    DEFAULT.SAMPLE.SOLUTION !          ( Number of components )
    GET.NEXT.SAMPLES                   ( Get position of bottle )
    DEFAULT.SAMPLE.SOLUTION 2+ !       ( Store position in definition )
    TOTAL.VOLUME R32@                  ( Get total volume required )
    CUMULATIVE.VOLUME R32@             ( Get total volume of components )
    F- % 1E3 F*                        ( Calculate volume of solvent )
    DEFAULT.SAMPLE.SOLUTION 4+ I16! ( Enter volume of solvent )
;

NO.IN.SAMPLE 0!
NO.IN.SERIAL 0!

: PREPARE.SOLUTION    ( - )
( Prepare a solution as defined by the user )
    GET.ID
    NO.IN.SAMPLE @ DUP
    IF
      DROP
      ." Remake solution " ?Y/N
      CR
    THEN
    IF
      NO.IN.SERIAL @
      IF
        ." Remake intermediate solution " ?Y/N
        IF
          REMAKE.SERIAL
        THEN
      THEN
      REMAKE.SAMPLE
    ELSE
      NO.IN.SERIAL @
      IF
        DEFAULT.SERIAL.SOLUTION 2+ @
        DUMP.SOLUTION
      THEN
      INIT.PREP
      GET.NO.OF.COMP
      GET.VOLUME
      DUP
      0
      DO                                      ( Loop to get identity and conc. of )
        GET.ELEMENT                           ( each component )
        [ ARL ]
        DUP ELEMENT.ARRAY I + C!
        DUP SOLUTION.CONTENTS I 4* I!         ( Put channel number on contents list )
        GET.CONCENTRATION FDUP                ( Get concentration )
        CALC.VOLUME
        MINIMUM.VOLUME @
        < NOT                               ( Check if volume required less than )
        IF                                  ( minimum allowed )
          ENTER.ON.SAMPLE.SOLUTION          ( If not do dilution directly )
          1 SOLUTION.CONTENTS 2+ I 4* I!    ( Store direct dil. on contents )
          0 SOLUTION.CONTENTS I 1+ 4* I!    ( List NULL terminated )
        ELSE
          ENTER.ON.SERIAL.SOLUTION          ( Otherwise do serial dilution )
          2 SOLUTION.CONTENTS 2+ I 4* I!    ( Store serial dilution on contents )
          0 SOLUTION.CONTENTS I 1+ 4* I!
        THEN
        DDROP
      LOOP
      0 SWAP ELEMENT.ARRAY + C!
      NO.IN.SERIAL @                        ( Check if intermediate soln. req. )
      IF
        FINISH.SERIAL
        MAKE.SOLUTION                       ( Make intermediate solution )
        ( SOLUTION.PRINT CR )               ( Diagnostic print out )
      THEN
      FINISH.SAMPLE
      MAKE.SOLUTION                         ( Make final solution )
      ( SOLUTION.PRINT )                    ( Diagnostic print out )
    THEN
    PRINT
    PRINT.SUMMARY                           ( Print summary of solution contents )
    CRT
;

: SAMPLE.TO.ARL       ( sample_posn. - )
( Take sample solution from samples tray to ARL sample tray )
    [ ROBOT ]
    CHECK.GRIP                       ( Check robot hand is empty )
    GRIP.ACTIVE @ SWAP               ( Get current state )
    0 GRIP.ACTIVE !                  ( Keep grip open during all moves )
    MOVE.TO                          ( Find path to sample )
    PICK.UP.SOLN                     ( Pick up solution )
    FRWRD SAMPLE.TO.ARL.PATH         ( Take to ARL tray )
    ARL.SAMPLE MOVE.TO               ( Move to sample posn. in ARL tray )
    % 35 MOVE.DOWN                   ( Put sample down into tray )
    GRIP.OPEN                        ( Let go )
    % 35 MOVE.UP
    % [illegible] MOVE.BACK
    SAMPLE.TO.ARL.PATH.2 MOVE.TO     ( Move away from sample )
    GRIP.ACTIVE !                    ( Restore grip activity )
    [ FORTH ]
;

VARIABLE VALVE.RESPONSE              ( Variable to store valve response time )
VARIABLE SAMPLE.PURGE                ( Time to remove sample from uptake line )

: ARL.TIP.WASH        ( - )
( Rinse tip of ARL probe )
   [ ROBOT ]
   2 4 ARL.SAMPLES MOVE.TO           ( Move to rinse position )
   % 90 FDUP MOVE.DOWN               ( Dip into rinse )
   5 0
   DO
     % 1 PITCH.BY                    ( Wash tip of probe )
     % -1 PITCH.BY
   LOOP
   [ ARL ]
   WIZ.PUMP SELECT.DEVICE
   FORCE.FLOW @                      ( Get high speed setting )
   [ PUMP.COMMANDS ]
```

```
    PUMP.SPEED                      ( Set pump to high speed )
    [ ARL ]
    ASPIRATE.SAMPLE                 ( Draw distilled water to wash out )
    SAMPLE.PURGE @ WAIT             ( previous solution )
    [ ROBOT ]
    MOVE.UP                         ( Take probe out of wash solution )
    [ ARL ]
    WIZ.PUMP SELECT.DEVICE
    NORMAL.FLOW @                   ( Get normal flow rate )
    [ PUMP.COMMANDS ]
    PUMP.SPEED                      ( Set to normal flow rate )
    [ ARL ]
    ASPIRATE.WATER                  ( Switch valve to aspirate blank solution )
    [ FORTH ]
;

: START.MEASURE.SAMPLE       ( ARL_sample_posn. - )
     [ ROBOT ]
     MOVE.TO                        ( Move sample )
     WAIT.ROBOT                     ( Wait for robot to arrive )
     WIZ.PUMP SELECT.DEVICE         ( Select Wiz pump for output )
     [ ARL ]
     ASPIRATE.SAMPLE                ( Switch valve to draw from probe )
     VALVE.RESPONSE @               ( Get valve response time )
     BUBBLE @ + WAIT                ( Get bubble time and wait )
     ROBOT.PORT SELECT.DEVICE       ( Select robot for output )
     [ ROBOT ]
     % 90 MOVE.DOWN                 ( Move down into solution )
     [ FORTH ]
;

: STOP.MEASURE.SAMPLE        ( - )
     [ ROBOT ]
     % 90 MOVE.UP                   ( Move up out of solution )
     [ ARL ]
     BUBBLE @ WAIT                  ( Allow bubble to form )
     WIZ.PUMP SELECT.DEVICE         ( Select Wiz pump for output )
     ASPIRATE.WATER                 ( Switch valve to draw from water )
     ROBOT.PORT SELECT.DEVICE       ( Select robot for output )
     [ ROBOT ]
     ARL.TIP.WASH                   ( Rinse tip of probe )
     [ FORTH ]
;

: EMPTY.SAMPLE        ( - )
    [ ROBOT ]
    CHECK.GRIP                      ( Check if robot is holding anything )
    GRIP.ACTIVE @ SWAP              ( Get state of grip activity )
    SAMPLE.TO.ARL.PATH.2 MOVE.TO    ( Move to area of sample )
    ARL.SAMPLE MOVE.TO              ( Move to above sample )
    % 40 MOVE.DOWN
    GRIP.CLOSE
    % 40 MOVE.UP                    ( Pick up sample )
    WAIT.ROBOT                      ( Wait for robot to complete move )
    FRWRD EMPTY.CONTAINER           ( Take tip vial contents into waste )
    WAIT.ROBOT                      ( Wait for robot to complete move )
    [ ARL ]
    DRAIN @ WAIT                    ( Delay to allow to drain )
    [ ROBOT ]
    FRWRD DIRTY.GLASSWARE           ( Drop vial into dirty glassware box )
    [ FORTH ]
;
```

```
: FORCE.FLUSH ( - )
 ( Run peristaltic pump at a higher flow rate during pre-flush )
   WIZ.PUMP SELECT.DEVICE
   [ ARL ]
   FORCE.FLOW @                     ( Get high speed setting )
   [ PUMP.COMMANDS ]
   PUMP.SPEED                       ( Set higher speed )
   13 CEMIT
   [ ARL ]
   PRE.FLUSH.TIME @                 ( Get pre-flush time )
   2* 3 / 0                         ( Calculate 2/3 of pre-flush time )
   DO
     1000 WAIT                      ( 1 sec delay )
   LOOP
   NORMAL.FLOW @                    ( Run at normal speed for remainder )
   [ PUMP.COMMANDS ]
   PUMP.SPEED
   13 CEMIT
   OLD.DEVICE SELECT.DEVICE         ( Return comm. to previous device )
;
```

Appendix C

Listing of Automated Analysis Program

This appendix contains the full listing of the procedures used to perform the automated analysis using feedback control of the preparation of the standards. The last word defined is used to perform the analysis. The samples are placed in the measurement tray in front of the plasma box. The positions 1,1 and 2,4 are used by the standard solutions and the wash solution respectively. Typing the command "AUTO.ANALYSIS" will start the analysis. The computer will ask for the names and positions of the samples, will ask about the elements to be analysed and the number of replicate integrations to be used. After the filename has been entered the analysis will start.

The constant AUTO.ANAL can be used with the command ARL.PROG.RUN to start the ARL part of the program when the ARL is running the COMMAND program.

Most of the program listing concerns the new solution preparation routine PREP.SOLN. One routine is, as yet, untested (SWITCH.NEXT.SOLN). This routine was designed to deal with the situation that may arise when a change in the dilution ratio required for the most dilute intermediate solutions (3 or 4) cause them to be switched from being included in solutions 1 to 2 or from 2 to 1. This situation never arose during the extensive testing of the program and so this code may be redundant. It was included simply for completeness.

```
500 CONSTANT AUTO.ANAL              ( Constant used to run AR1 part of program )
CREATE SAMPLE.ID 140 ALLOT          ( Space for sample names )
CREATE SAMPLE.POS 12 ALLOT          ( Space to store position of samples in tray )

VARIABLE SOLN.COMP 35 ALLOT         ( Definition of solution as returned from ARL )
VARIABLE NO.SAMPLES                 ( Number of samples to be analysed )
VARIABLE NO.REPL                    ( Number of replicate integrations )
VARIABLE MAX.POINTS                 ( Maximum number of points to be included )
VARIABLE CAL.SOLN.NO                ( Point number in calibration )

CREATE INTERMEDIATE.SOLUTIONS 152 ALLOT    ( Space for definition of intermendiates )
CREATE CUMULATED.INT.VOLUMES 16 ALLOT      ( Volume of stds in int. solutions )
CREATE INT.DIL.RATIOS 16 ALLOT             ( Dilution ratios for int. solutions )
CREATE NO.IN.INT 8 ALLOT                   ( Number of components in solution )
CREATE POSN.ON.NEXT 8 ALLOT                ( Position of int on next solution )
VARIABLE INT.3.ON                          ( Next intermediate solution for 3 )
VARIABLE INT.4.ON                          ( Next intermediate solution for 4 )

5 NO.REPL !                         ( Set default values )
10 MAX.POINTS !
2 INT.3.ON !
2 INT.4.ON !

VARIABLE MAX.RSD 2 ALLOT            ( Desired precision for analysis )
% 0.01 MAX.RSD R32!                 ( Default set to 1% )

: GET.SAMPLE.ID       ( - )
  ( Get name of sample form user and store )
   CR
   ." Please enter sample i.d. "
   IN$                              ( Get name )
   SAMPLE.ID
   NO.SAMPLES @ ?DUP                ( Is this first sample? )
   0>
   IF
     0                              ( If not skip over names of previous samples )
     DO
       COUNT +
     LOOP
   THEN
   OVER C@ 2+                       ( Get length and add two bytes for the Count )
   CMOVE                            ( Move name into string array )
;

: GET.SAMPLE.POS      ( - )
  ( Get position of sample in tray )
    BEGIN
      CR
      ." Please enter position of "
      SAMPLE.ID                     ( Find sample name )
      NO.SAMPLES @ ?DUP
      0>
      IF
        0
        DO
          COUNT +
        LOOP
      THEN
      $.                            ( Print sample name )
      IN$                           ( Get user response )
      PLOAD                          ( Interpret user response )
      ARL.SAMPLES                    ( Position in tray )
      DUP
      1 1 ARL.SAMPLES                ( Check not in 1,1 position as this is used by )
      = DUP                          ( the standards )
      IF
        CR
        ." Sorry, but that position is to be used for standards." CR
        ." Please move the sample to another position"
        PLUCK                        ( Print message, remove position from under flag)
      THEN
      NOT                            ( Loop until position not 1,1 )
    UNTIL
    SAMPLE.POS
    NO.SAMPLES @ 2* + !              ( Store position in array )
;

: ANOTHER.SAMPLE?     ( - flag )
    NO.SAMPLES DUP 1+! @             ( Get number of samples )
    6 = NOT DUP                      ( Cannot analyse more than 6 samples, so no more )
    IF
      DROP CR                        ( Less than 6 sample so ask user if any more )
      ." Another sample " ?Y/N
    THEN
;

: GET.ELEMENT.AUX     ( - element )
   CR
   ." Please enter the next element for the solution "
   IN$ DROP                          ( Get user input )
   CONTEXT ,@                        ( Get current vocabulary )
   ARL-SEG 14 + CONTEXT ,!           ( Switch to ARL vocabulary )
   PLOAD SWAP CONTEXT ,!             ( Interpret user response and restore vocabulary )
;

: ANOTHER.ELEMENT?    ( - flag )
    NO.ELEMENTS DUP 1+! @
    7 = NOT DUP                      ( More than 7 elements will exceed capacity )
    IF
      DROP CR
      ." Another element " ?Y/N      ( Ask the question )
    THEN
;

: GET.ELEMENTS        ( - )
    [ ARL ]
    CR
    LIST.ELEMENTS CR                        ( Print current list of elements )
    ." Use these elements " ?Y/N            ( Are these to be used? )
    NOT
    IF
      NO.ELEMENTS 0!                        ( If not, zero counter )
      BEGIN
        [ ARL ]
        GET.ELEMENT                         ( Get element from user )
        ELEMENT.ARRAY NO.ELEMENTS @ + C!    ( Store in list )
        ANOTHER.ELEMENT?                    ( Ask if there are any more )
        NOT                                 ( Repeat until no more )
      UNTIL
      0 ELEMENT.ARRAY NO.ELEMENTS @ + C!    ( Element list is NUL terminated )
    THEN
    NO.ELEMENTS @ SOLN.COMP !               ( Store number of components in SOLN.COMP )
;
```

```
: GET.NO.REPL  ( - )
   CR
   ." Use " NO.REPL @ . ." integrations per sample "
   ?Y/N                          ( Check if current value can be used )
   NOT
   IF
     CR                          ( If not get new value )
     ." How integrations per sample "
     IN# NO.REPL !
   THEN
;

: GET.SAMPLES.INFO    ( - )
   NO.SAMPLES 0!                 ( Zero counter )
   CAL.SOLN.NO 0!                ( Initialise counter )
   BEGIN
     GET.SAMPLE.ID               ( Get information on sample )
     GET.SAMPLE.POS
     ANOTHER.SAMPLE?             ( Repeat until no more samples )
     NOT
   UNTIL
   GET.ELEMENTS                  ( Get elements to be analysed )
   GET.NO.REPL                   ( Number of replicate integrations )
   NO.ELEMENTS @
   SOLN.COMP !
;

: GET.FILE    ( - )
   CR
   ." Please enter filename "
   IN$ MAKE-OUTPUT CR            ( Get filename and open for output )
   >FILE                         ( Put header in file )
   ." Automated Analysis Using Calibration Curves " CR
   PRINT                         ( Print header on printer )
   ." Automated Analysis Using Calibration Curves " CR
   >FILE                         ( Send output to file )
   DATE. 9 EMIT TIME. CR         ( Date and time stamp )
   PRINT
   DATE. 9 EMIT TIME. CR
   >FILE
;

: SEND.NO.SAMPLES     ( - )
   [ ARL ]
   NO.SAMPLES @                  ( Get number of samples to be analysed )
   ASCII S                       ( Synchronisation characters )
   ASCII N
   N.SEND.ARL                    ( Send integer to ARL )
;

: SEND.MAX.POINTS     ( - )
  [ ARL ]
  MAX.POINTS @                   ( Get maximum number of points for calibration )
  ASCII N
  ASCII M
  N.SEND.ARL
;
```

```
: SEND.MAX.ERROR      ( - )
  [ ARL ]
  MAX.RSD R32@                         ( Get value of desired precision )
  4                                    ( Send value to 4 sig. figs. )
  ASCII E                              ( Synchronisation characters )
  ASCII M
  F.SEND.ARL                           ( Send floating point number )
;

: INIT.ARL             ( - )
   [ ARL ]
   ARL.PORT SELECT.DEVICE              ( Select ARL for output )
   RESTART.ARL                         ( Restart program running on ARL )
   PAD COMPORT @ GET.RETURN DROP       ( Get program name and ignore )
   CKEY DROP                           ( Get LF character )
   13 CEMIT                            ( Send CR to ARL )
   SEND.ELEMENTS                       ( Send elements to be analysed )
   SEND.PREFLUSH                       ( Send preflush time )
   SEND.INTEGRATION                    ( Send integration time )
   NO.REPL @ SEND.REPLICATES           ( Send number of replicate integrations )
   SEND.NO.SAMPLES                     ( Send number of samples to be analysed )
   SEND.MAX.POINTS                     ( Send maximum number of points for calibration )
   SEND.MAX.ERROR                      ( Send desired precision )
;

: GET.DATA     ( Integration number - )
    [ ARL ]
    ASCII I
    WAIT.ARL                           ( Wait till integration complete )
    ASCII R
    ASCII E
    N.RECEIVE.ARL                      ( Get error code )
    0= NOT                             ( Check if error occured )
    IF
      ABORT" Integration Error"
    THEN
    1+ . 9 EMIT                        ( Print integration number and TAB )
    ELEMENT.ARRAY
    BEGIN
      DUP C@                           ( For each element get a value )
    WHILE
      ASCII D                          ( Synchronisation characters )
      ASCII R
      F.RECEIVE.ARL                    ( Get emission intensity )
      4 6 F.R 9 EMIT 1+                ( Print value to 4 sig figs. in 6 char field )
    REPEAT
    DROP CR
;

: GET.MEAN     ( - )
    [ ARL ]
    ." Mean" 9 EMIT
    ELEMENT.ARRAY
    BEGIN
      DUP C@
    WHILE
      ASCII D
      ASCII M
      F.RECEIVE.ARL                    ( Get average emission intensity )
      4 6 F.R 9 EMIT 1+
    REPEAT
    DROP CR
;
```

```
: START.BLANK.MEASURE          ( - )
   ." Blank " CR                        ( Print header )
   ARL.PORT SELECT.DEVICE
   13 CEMIT                             ( Send CR to ARL )
   [ ARL ]
   ASCII M
   ASCII B
   SYNC.ARL                             ( Send start signal )
;

: GET.BLANK.DATA       ( - )
   NO.REPL @
   0
   DO
     I GET.DATA                         ( Get individual intensities )
   LOOP
   GET.MEAN                             ( Get average intensity )
;

: MEASURE.BLANK        ( - )
   START.BLANK.MEASURE
   GET.BLANK.DATA
;

: GET.REGRESSION.DATA          ( - )
    [ ARL ]
    CR
    ." Regression Data " CR             ( Print table header )
    9 EMIT
    ." Slope " 9 EMIT
    ." Intercept " 9 EMIT
    ." Sy/x " 9 EMIT
    ." Sb " 9 EMIT
    ." Sa " CR
    ARL.PORT SELECT.DEVICE
    BEGIN
      ASCII L
      ASCII E
      N.RECEIVE.ARL                     ( Get channel number )
      ?DUP                              ( While not zero )
    WHILE
      ELEMENT.SYMBOLS
      SWAP
      0
      DO
        COUNT +
      LOOP
      $. 9 EMIT                         ( Print element symbol )
      ASCII L
      ASCII S
      F.RECEIVE.ARL                     ( Get slope )
      7 6 F.R 9 EMIT                    ( Print slope and TAB )
      ASCII N
      ASCII I
      F.RECEIVE.ARL                     ( Get intercept )
      7 6 F.R 9 EMIT
      ASCII Y
      ASCII S
      F.RECEIVE.ARL                     ( Get sy/x )
      7 6 F.R 9 EMIT
      ASCII B
      ASCII S
      F.RECEIVE.ARL                     ( Get std. dev. in slope )
      7 6 F.R 9 EMIT
      ASCII A
      ASCII S
      F.RECEIVE.ARL                 ( Get std. dev. in intercept )
      7 6 F.R CR
    REPEAT                          ( Repeat for each channel )
;

: GET.SAMPLES.RESULTS ( - )
   [ ARL ]
   ." Sample Results " TIME. CR     ( Print header and time stamp )
   CAL.SOLN.NO @                    ( Get calibration point number )
   0=                               ( If initial estimate then want a hard copy )
   IF
     PRINT                          ( Send to printer )
     ." Sample Results  " TIME. CR
   THEN
   >FILE
   9 EMIT LIST.ELEMENTS CR          ( Send element symbols to file )
   CRT                              ( Send to screen )
   9 EMIT LIST.ELEMENTS CR
   CAL.SOLN.NO @
   0=
   IF
     PRINT                          ( Send to printer )
     9 EMIT LIST.ELEMENTS CR
   THEN
   >FILE                            ( Off to file we go )
   ." Concentrations " CR
   CRT                              ( Hello, screen )
   ." Concentrations " CR
   CAL.SOLN.NO @
   0=
   IF
     PRINT                          ( Hi there, printer )
     ." Concentrations " CR
   THEN
   >FILE
   NO.SAMPLES @                     ( Get number of samples )
   0
   DO
     I                              ( Get sample number )
     SAMPLE.ID
     SWAP ?DUP                      ( Look for name of sample )
     0>
     IF
       0
       DO
         COUNT +
       LOOP
     THEN
     DUP
     CRT
     $.                             ( Print name of sample on screen )
     CAL.SOLN.NO @
     0=
     IF
       DUP
       PRINT                        ( and on the printer )
       $.
     THEN
     >FILE
     $.                             ( and finally in the file )
     ARL.PORT SELECT.DEVICE
     13 CEMIT                       ( Send CR to ARL )
   BEGIN
     ASCII L
     ASCII E
     N.RECEIVE.ARL                ( Get channel number from ARL )
   WHILE
     ASCII O
     ASCII C
     F.RECEIVE.ARL                ( Get concentration )
     FDUP
     CRT
     9 EMIT 7 6 F.R               ( Print concentration on screen )
     CAL.SOLN.NO @
     0=
     IF
       FDUP
       PRINT                      ( on printer )
       9 EMIT 7 6 F.R
     THEN
     >FILE                        ( and in file )
     9 EMIT 7 6 F.R
   REPEAT                         ( Repeat for each element )
   CRT CR >FILE                   ( Terminate line )
   CR
   CAL.SOLN.NO @
   0=
   IF
     PRINT
     CR
   THEN
 LOOP                             ( Loop for each sample )
 CRT ." Rel. Std. Dev. " CR >FILE
 ." Rel. Std. Dev. " CR           ( Header )
 NO.SAMPLES @
 0
 DO
   [ ARL ]
   I
   SAMPLE.ID                      ( Find sample name )
   SWAP ?DUP
   0>
   IF
     0
     DO
       COUNT +
     LOOP
   THEN
   DUP
   CRT
   $.                             ( Print sample name on screen )
   >FILE
   $.                             ( in file )
   ARL.PORT SELECT.DEVICE
   BEGIN
     ASCII L
     ASCII E
     N.RECEIVE.ARL                ( Get channel number )
   WHILE
     ASCII D
     ASCII S
     F.RECEIVE.ARL                ( Get %RSD )
     FDUP
     CRT 9 EMIT 7 6 F.R >FILE     ( Print RSD and TAB )
     9 EMIT 7 6 F.R
   REPEAT                         ( Repeat for each channel )
```

```
     CRT
     CR                             ( Terminate line )
     >FILE
     CR
   LOOP                             ( Loop for each sample )
;

: MEASURE.SAMPLES     ( - )
   [ ROBOT ]
   ." Sample Measurement " CR       ( Header )
   TIME. CR                         ( Time stamp )
   FRWRD ARL.PROBE                  ( Pick up sampling probe )
   [ ARL ]
   9 EMIT LIST.ELEMENTS CR          ( Print element symbols )
   MEASURE.BLANK                    ( Measure blank solution )
   NO.SAMPLES @
   0
   DO                               ( Loop for each sample )
     I
     SAMPLE.ID
     SWAP ?DUP                      ( Find sample name )
     0>
     IF
       0
       DO
         COUNT +
       LOOP
     THEN
     $. CR I                        ( Print sample name )
     2* SAMPLE.POS + @              ( Find position of sample )
     START.MEASURE.SAMPLE           ( Start sampling )
     [ ROBOT ]
     WAIT.ROBOT                     ( Wait for robot to complete move )
     ARL.PORT SELECT.DEVICE
     [ ARL ]
     13 CEMIT
     ASCII S
     ASCII I
     SYNC.ARL                       ( Send start signal to ARL )
     FORCE.FLUSH                    ( Use force flushing during preflush )
     NO.REPL @
     0
     DO
       I GET.DATA                   ( Get emission intensities )
     LOOP
     GET.MEAN                       ( Get average intensity )
     [ ROBOT ]
     STOP.MEASURE.SAMPLE            ( Stop sampling and wash probe )
     WAIT.ROBOT                     ( Wait for robot to complete operation )
   LOOP                             ( Loop to next sample )
   BCKWRD ARL.PROBE                 ( Replace probe )
   GET.SAMPLES.RESULTS              ( Get initial estimates )
;
```

```
: UPDATE.ELEMENT.LIST          ( - )
    [ ARL ]                    ( Rearrange the element list from SOLN.COMP )
    SOLN.COMP 2+
    ELEMENT.ARRAY
    BEGIN
      OVER C@                  ( Fetch element from SOLN.COMP )
      OVER C! 1+               ( Store in ELEMENT.ARRAY )
      SWAP 5 + SWAP            ( Increment SOLN.COMP )
      DUP C@ NOT               ( Check if next element number is 0 )
    UNTIL                      ( Repeat while next element not 0 )
    DDROP                      ( Clean up stack )
    [ FORTH ]
;

: GET.CONCENTRATIONS  ( - )
    [ ARL ]                    ( Get concentration of standard solution )
    ARL.PORT SELECT.DEVICE
    13 CEMIT
    ASCII C WAIT.ARL           ( Wait till ARL ready to send concentrations )
    SOLN.COMP 2+               ( Solution definition to be stored in SOLN.COMP )
    BEGIN
      ASCII L
      ASCII E
      N.RECEIVE.ARL            ( Get channel number )
      DUP
    WHILE                      ( While channel number not 0 )
      OVER C! 1+               ( Store channel number )
      ASCII O
      ASCII C
      F.RECEIVE.ARL            ( Get concentration )
      DUP R32! 4+              ( Store concentration and goto next entry )
    REPEAT
    DROP DROP                  ( Clean up stack )
    UPDATE.ELEMENT.LIST        ( Rearrange element list )
;

: GET.POSN.ON.NEXT    ( int no - Addr posn on next soln )
    1- 2*                      ( Calculate offset )
    POSN.ON.NEXT +             ( Calculate address )
;

: GET.NEXT.SOLN       ( Int no. - Addr next int soln )
    3 =
    IF
      INT.3.ON
    ELSE
      INT.4.ON
    THEN
;

: GET.NO.IN.INT.SOLN ( Int. No. - Addr of No. in Soln. )
    1- 2*                      ( Calculate offset )
    NO.IN.INT +                ( Calculate address )
;

: GET.INT.DEFN       ( Int. No. - Addr of Defn )
    1- 38 *                    ( Calculate offset )
    INTERMEDIATE.SOLUTIONS +   ( Calculate address )
;

: GET.INT.DIL.RATIO ( Int No. - :: - Dil. Rat. )
    1- 4*                      ( Calculate offset )
    INT.DIL.RATIOS + R32@      ( Calculate address and fetch )
;
```

```
: STORE.INT.DIL.RATIO ( Int No. - :: - Dil. Rat. )
    1- 4*                           ( Calculate offset )
    INT.DIL.RATIOS + R32!           ( Calculate address and store )
;

: GET.CUMULATED.VOLUME ( Int. No. - Addr of Vol. )
    1- 4*                           ( Calculate offset )
    CUMULATED.INT.VOLUMES +         ( Calculate address )
;

  GET.TOTAL.VOLUME          ( Int. No. - Int. No. :: - Total Vol. )
    DUP
    1 = OVER 2 =                    ( If intermediate 1 or 2 )
    OR
    IF
      TOTAL.VOLUME                  ( use TOTAL.VOLUME
    ELSE
      TOTAL.SERIAL.VOLUME           ( 3 or 4 use TOTAL.SERIAL.VOLUME )
    THEN
    R32@                            ( Get volume )
;

: FIND.ENTRY.ON.INT ( Posn Int. No. - Entry Recd. )
    GET.INT.DEFN 6+                 ( Get solution definition )
    BEGIN
      DUP @                         ( Get posn of component )
      3 PICK                        ( Copy of desired position )
      = NOT                         ( If not the same )
    WHILE
      4+                            ( Goto next entry )
    REPEAT
;

: FIND.STOCK.CONC     ( Recd. - Recd. :: - Conc. )
    DUP 2- @                        ( Get position of stock solution )
    FIND.STOCK.SOLN                 ( Find record of stock solution )
    DUP 2+ I16@ 4+ @ PLUCK          ( Get concentration )
;

: CALC.FINAL.CONC     ( :: Volume - Conc/ppm )
   F* TOTAL.VOLUME R32@ % 1E3 F* F/ ( Conc*volume/total volume )
;

: CALC.CONC.FROM.SERIAL      ( Int. No. - :: Volume - Conc )
    F*                              ( conc*volume )
    TOTAL.SERIAL.VOLUME R32@ % 1E3  ( convert ml to µl )
    F* F/                           ( divide by total volume, = conc, in int. soln. )
    GET.POSN.ON.NEXT @              ( Find int. soln. on next soln. defn. )
    DEFAULT.SAMPLE.SOLUTION 8 +     ( Get entry in sample definition )
    SWAP 4* I@ ->F                  ( Get volume of int. used, send to f.p. stack )
    CALC.FINAL.CONC                 ( Calculate concentration in final solution )
;

: CALC.CONC.FROM.START       ( Int. No. - :: Vol. - Conc. )
    F*
    TOTAL.SERIAL.VOLUME R32@
    % 1E3 F* F/ DUP                 ( Calculate conc. in int. soln. )
    GET.NEXT.SOLN @                 ( Find which int. soln. used for next step )
    SWAP GET.POSN.ON.NEXT @         ( Find record of this soln. on next )
    OVER GET.INT.DEFN 8 +           ( Get record )
    SWAP 4* I@ ->F                  ( Get volume of solution used )
    CALC.CONC.FROM.SERIAL           ( Continue calculation )
;
```

```
: CALC.PPM.CONC.2     ( Soln.Contents - Soln.Contents+ :: - Conc. )
   2+ DUP @                     ( Find solution at start of dilution )
   1 =
   IF                           ( Single step dilution )
     FIND.STOCK.CONC            ( Get concentration of stock solution )
     FIND.ENTRY.ON.SAMPLE       ( Find record for component )
     2+ I16@ DROP               ( Get volume of stock solution used )
     CALC.FINAL.CONC            ( Calculate concentration )
   ELSE                         ( More than one step dilution )
     FIND.STOCK.CONC            ( Get concentration of stock solution )
     OVER @ 1-                  ( Get intermediate solution number )
     FIND.ENTRY.ON.INT          ( Find record of component on soln. defn. )
     2+ I16@ DROP               ( Get volume of staock solution used )
     DUP @ 1-                   ( Get int. soln. no. )
     DUP 1 =
     OVER 2 =
     OR                         ( Was a two step dilution used )
     IF
       CALC.CONC.FROM.SERIAL    ( Calculate concentration for two step dil. )
     ELSE
       CALC.CONC.FROM.START     ( Calculate concentration for three step dil. )
     THEN
   THEN
;

: RETURN.CONCENTRATIONS     ( - )
   [ ARL ]                      ( Send actual concentration in standard solution )
   SOLUTION.CONTENTS
   BEGIN
     DUP @
   WHILE                        ( While channel number not 0 )
     CALC.PPM.CONC.2            ( Calculate concentration )
     ARL.PORT SELECT.DEVICE
     4                          ( Number of sig. figs. )
     ASCII O                    ( Synchronisation characters )
     ASCII C
     F.SEND.ARL                 ( Send concentration )
     2+                         ( Goto next entry )
  REPEAT
  DROP                          ( Clean up stack )
;

: FCBRT       ( :: x - x^1/3 )
   FABS FLOG % 3 F/ F10**       ( Calculate cube root of dilution factor )
;

: FROOT       ( Int. No. :: x - x^1/n )
   DUP                          ( Calculate appropriate root of dilution factor )
   1 = OVER 2 =
   OR
   IF
     FSQRT                      ( Square root for two step dilution )
   ELSE
     FCBRT                      ( Cube root for three step dilution )
   THEN
;

: TOO.DILUTE.ERROR    ( - )
   ABORT" Component to dilute to be made" ( Print message, return control to user )
;
```

```
: CHECK.MINIMUM.VOLUME        ( Vol. - Vol. )
    DUP MINIMUM.VOLUME @
    <                             ( Compare volume to minimum allowed )
    IF
      CRT                         ( If too small, switch output to screen )
      TOO.DILUTE.ERROR            ( Abort with error message )
    THEN
;

: CHECK.OVERFLOW      ( Vol. Int. No. - Vol. Int. No. flag )
    OVER ->F % 1E3 F/             ( Send volume to f.p. stack and convert ml )
    DUP GET.CUMULATED.VOLUME DUP  ( Get cumulated volume )
    R32@ F+ FDUP OVER             ( Fetch and add new volume )
    GET.TOTAL.VOLUME DROP         ( Get total required )
    F-                            ( Take difference )
    F0>                           ( Check for overflow )
;

: GET.OVERALL.DIL.RATIO      ( Int. No. - :: - Dil. Rat. )
    BEGIN-CASE
      1 CASE-OF
          1 GET.INT.DIL.RATIO     ( Get dilution ratio )
      ELSE
      2 CASE-OF
          2 GET.INT.DIL.RATIO
      ELSE
      3 CASE-OF
          3 GET.INT.DIL.RATIO     ( Get first dilution ratio )
          INT.3.ON @
          GET.INT.DIL.RATIO       ( Get second dilution ratio )
          F*                      ( Calculate overall dilution ratio )
      ELSE
      4 CASE-OF
          4 GET.INT.DIL.RATIO
          INT.4.ON @
          GET.INT.DIL.RATIO
          F*
      ELSE
      DROP
    END-CASE
;

: CHECK.OLD.RATIO     ( Int. No. - Vol. Flag :: Dil. Rat. - Dil. Rat. Old. D. R. )
    DUP                           ( Check if existing dilution ratio can be used )
    GET.OVERALL.DIL.RATIO         ( Get existing dilution ratio )
    CALC.VOL.CF.STD.IN.SERIAL     ( Calculate volume of standard required )
    DUP MINIMUM.VOLUME @ <        ( Check if too small )
    NOT                           ( If not then can use existing values )
;

: FIND.NEXT.SYRINGE.VOLUME   ( Vol - Vol' )
    FIND.SYRINGE.VOLUME @ PLUCK     ( Find closest allowed volume )
;

: BACK.CALC.SYRINGE.VOLUME   ( Int. No. - Int. No. Vol :: Dil. Rat. - )
    GET.TOTAL.VOLUME           ( From dilution ratio calculate volume to be used )
    F* % 1E3 F* F->
;
```

```
: SYRINGE.VOLUME.AVAILABLE   ( Int. No. - flag :: Dil. Rat. - Dil. Rat. )
    FDUP
    BACK.CALC.SYRINGE.VOLUME        ( Calculate volume from dilution ratio )
    PLUCK
    FIND.SYRINGE.VOLUME PLUCK       ( Find allowed volume )
    SERIAL.VOLUMES 10 + >           ( Check for overrun )
    NOT                             ( If no overrun then a volume is available )
;

: BACK.CALC.DIL.RATIO        ( Int. No. Vol. - Int. No. :: - Dil. Rat. )
    ->F % 1E3 F/                    ( From volume calculate dilution ratio )
    GET.TOTAL.VOLUME
    F/
;

: CALC.IDEAL.VOLUME   ( Int. No. - Int. No. Vol. :: Dil. Rat. - Dil. Rat. )
( From overall dilution ratio calculate ideal volume )
    FDUP FROOT                      ( Take nth root )
    GET.TOTAL.VOLUME                ( Multiply into total volume )
    F* % 1E3 F*                     ( Convert to µl )
    F->                             ( Send to integer stack )
;

: CALC.NEW.RATIO ( Int. No. - Vol. Int. No. :: Dil. Rat. - Dil. Rat. N.D.R. )
    CALC.IDEAL.VOLUME               ( Calculate ideal volume )
    FIND.BEST.SYRINGE.VOLUME        ( Find closest allowed volume )
    BACK.CALC.DIL.RATIO             ( Calculate actual dilution ratio )
    CALC.VOL.OF.STD.IN.SERIAL       ( Calculate volume of stock required )
    SWAP                            ( Rearrange the stack )
;

: CALC.NEW.TWO.STEP.RATIO
 ( Int. No. - Vol. Int. No. :: Dil. Rat. - Dil. Rat. N.D.R. )
    CALC.IDEAL.VOLUME
    FIND.BEST.SYRINGE.VOLUME
    BACK.CALC.DIL.RATIO       ( Calculate dilution ratio of last step )
    FOVER FOVER F/            ( Calculate dilution ratio required of first two steps )
    DUP 2- CALC.IDEAL.VOLUME        ( Calculate ideal volume for first two steps )
    FDROP
    FIND.BEST.SYRINGE.VOLUME
    BACK.CALC.DIL.RATIO DROP        ( Calculate dilution ratio of second step )
    FOVER F*                        ( Calculate dilution ratio of last two steps )
    FROT FSWAP
    CALC.VOL.OF.STD.IN.SERIAL       ( Calculate volume of stock solution required )
    SWAP FDROP FSWAP                ( Clean up stack )
;

: OVERFLOW.ERROR      ( - )
    CRT                             ( Send output to screen )
    FDROP CR
    ." This solution cannot be prepared from the stock solutions present" CR
    ." as the total volume of stock solutions will exceed the total volume "
    ." required"
    ABORT                           ( Print message and return control to user )
;

: CHECK.CUMULATED.VOLUME     ( Vol. Int. No. - Vol Int. No. )
    CHECK.OVERFLOW                  ( Check if overflow can occur )
    IF
      OVERFLOW.ERROR                ( If so abort )
    THEN
    %32!                            ( Store new cumulated volume )
;
```

```
: UPDATE.FOR.NEW.RATIO
 ( Int. No. No. in Soln. - Int. No.:: Dil. Rat. - Dil. Rat. )
   FDUP OVER                      ( Copy dilution ratio and soln. no. )
   GET.INT.DIL.RATIO              ( Get old value of dilution ratio )
   FSWAP F/ OVER                  ( Calculate update factor )
   GET.INT.DEFN 8 + SWAP          ( Get definition record )
   3 PICK GET.CUMULATED.VOLUME    ( Get old cumulated volume )
   F=0 R32!                       ( Zero volume )
   0
   DO                             ( Loop for each entry )
     FDUP DUP DUP I16@            ( Copy update factor, get old volume )
     F* F->                       ( Calculate new volume )
     CHECK.MINIMUM.VOLUME         ( Check not too small )
     4 PICK
     CHECK.CUMULATED.VOLUME DROP  ( Check does not overflow )
     SWAP ! 4+                    ( Store new volume and goto next entry )
   LOOP
   DROP FDROP                     ( Clean up stacks )
;

: UPDATE.VOLUME ( Vol. Int. No. - Vol' Int. No. :: Dil. Rat. - Dil. Rat. )
   FDUP DUP
   GET.INT.DIL.RATIO
   FSWAP F/                       ( Calculate update factor )
   SWAP
   ->F F* F->                     ( Calculate new volume )
   CHECK.MINIMUM.VOLUME           ( Check not too small )
   SWAP                           ( Tidy up )
;

: INCREASE.VOLUME.OF.INT.SOLN     ( Vol Int. No. - Vol' Int. No. :: D. R. - D. R.' )
   BACK.CALC.SYRINGE.VOLUME 1+    ( Calculate current value of syringe vol. )
   FIND.NEXT.SYRINGE.VOLUME       ( increase value to go to next vol. )
   BACK.CALC.DIL.RATIO            ( Calculate new dilution factor )
   DUP GET.NO.IN.INT.SOLN @
   UPDATE.FOR.NEW.RATIO           ( Update vols. of all entries in defn. )
   UPDATE.VOLUME                  ( Update vol. of this component )
;

: START.NEW.INT.DEFN  ( Int. No. - Int. No. )
   DUP GET.CUMULATED.VOLUME
   F=0 R32!                       ( Zero total volume of components )
   GET.NEXT.SERIAL                ( Get posn. of next empty vial )
   OVER
   GET.INT.DEFN 2+ !              ( Store posn. of vial )
;

: ENTER.ON.INT.DEFN   ( Stock Recd. Vol Int. No. - :: Dil. Rat. - )
   CHECK.CUMULATED.VOLUME         ( Check that volume won't cause overflow )
   DUP                            ( also update cumulated volume )
   GET.INT.DEFN                   ( Get solution defn. )
   4 ROLL 4+ @                    ( Get posn. of stock solution )
   OVER 6+                        ( Calculate base offset for posns. )
   4 PICK
   GET.NO.IN.INT.SOLN @ 4* I!     ( Calculate entry offset and store posn. )
   8 + ROT SWAP                   ( Calculate base offset for volumes )
   3 PICK
   GET.NO.IN.INT.SOLN @ 4* I!     ( Calculate entry offset and store volume )
   DUP GET.NO.IN.INT.SOLN 1+!     ( Increment number of components in solution )
   FDROP                          ( Drop dilution ratio )
;
```

```
: USE.OLD.RATIO       ( Vol. Int. No. - Int. No. :: Dil. Rat. - )

       ( Some entries are already in solution defn. and we can use the existing )
       ( dilution factor )

    SWAP
    BEGIN
      CHECK.OVERFLOW          ( Check to see if adding this volume will cause )
      PLUCK FDROP             ( an overflow, while yes and while a syringe volume )
      OVER                    ( is available reduce volume of std. )
      SYRINGE.VOLUME.AVAILABLE
      AND                     ( and increase volume of intermediate on next soln. )
    WHILE
      INCREASE.VOLUME.OF.INT.SOLN
    REPEAT                    ( We will assume some changes have been made )
    DUP STORE.INT.DIL.RATIO   ( Store new dilution factor )
    ENTER.ON.INT.DEFN         ( Enter this component of def[illegible] )
;

: USE.NEW.RATIO       ( Vol. Int. No. - Int. No. :: Dil. Rat. - )
    DROP FDROP                      ( Can't use existing values of dilution ratios )
    CALC.NEW.RATIO                  ( Get rid of old values and calculate new ones )
    DUP
    GET.NO.IN.INT.SOLN @ ?DUP       ( C[illegible] to see if anything already on defn. )
    IF
      UPDATE.FOR.NEW.RATIO          ( Yes, update existing entries for new ratio )
    ELSE
      START.NEW.INT.DEFN            ( No, Start a new defn. )
    THEN
    DUP STORE.INT.DIL.RATIO         ( Store new value of dilution factor )
    ENTER.ON.INT.DEFN               ( Put this component into defn. )
;

VARIABLE COMPONENT.NO

: FIND.INTERMEDIATE.TO.USE    ( - Int. No. :: Dil. Rat. - Dil. Rat. )
    F=1 MINIMUM.VOLUME I16@
    TOTAL.SERIAL.VOLUME R32@
    % 1E3 F* F/ FSQRT FSWAP         ( Calculate maximum range of dilutions for )
    1                               ( for each intermediate solution )
    BEGIN
      FOVER F* F3PICK FSWAP         ( Calculate maximum dilution factor for int. )
      CALC.VOL.OF.STD.IN.SERIAL     ( Calculate volume of std. that would be )
      FSWAP FDROP                   ( required )
      MINIMUM.VOLUME @ <            ( Is it less than minimum allowed )
    WHILE
      1+                            ( While it is increment count and loop )
    REPEAT
    FDROP FDROP                     ( Drop maximum dil. rat. and dil. range )
;

: USE.INT.SOLN.1.OR.2         ( Int. No. - :: Dil. Rat. - )
    CHECK.OLD.RATIO                 ( Check if existing dilution factors can be used )
    IF
      USE.OLD.RATIO                 ( Yes, use them )
    ELSE
      USE.NEW.RATIO                 ( No, try find new ones )
    THEN
       ( Enter intermediate solution used to dilute this component )
       ( This is used to calculate the actual concentration )
       ( in the final solution )
    1+ SOLUTION.CONTENTS 2+ COMPONENT.NO @ 4* I!
    0 SOLUTION.CONTENTS COMPONENT.NO @ 1+ 4* I!
;
```

```
: REMOVE.FROM.NEXT.CUMULATED.VOLUME
  ( Int. No. Int. Vol. Next. Soln. - Int. No. Int. Vol. Next. Soln. )
    3 PICK GET.POSN.ON.NEXT          ( Find position on next solution )
    OVER GET.INT.DEFN 8 +            ( Get defn. of next solution )
    SWAP 4* I@ ->F % 1E3 F/          ( Get volume of intermediate on next soln. )
    DUP GET.CUMULATED.VOLUME DUP     ( Get cumulated volume of next soln. )
    R32@ FSWAP F- R32!               ( Remove volume of intermediate )
;

: VOLUME.WITH.EXISTING.RATIO         ( Int. No. - Int. No. Vol. :: D.r. R.D.r. N.D.r. )
    DUP GET.NEXT.SOLN @       ( Get number of next solution )
    GET.INT.DIL.RATIO         ( Get dilution factor of next soln. )
    F/                        ( Divide total dilution ratio by dil. rat. of next )
    DUP GET.NEXT.SOLN @
       ( Calc vol. of std and dil. rat. for remaining dilution )
    CALC.NEW.RATIO DROP
;

: CHECK.EXISTING.RATIO
 ( Vol. Int. no. - Vol. Int. No. :: D.r. N.D.r - D. R. N.D.r )
    FOVER                            ( Copy overall dilution factor )
    VOLUME.WITH.EXISTING.RATIO       ( Calculate volume of std. needed if using )
    FDROP FDROP                      ( existing values of dil. ratios )
    MINIMUM.VOLUME @ <               ( Compare with minimum allowed )
    NOT
;

: GET.OTHER.NEXT      ( next soln. - other next soln. )
    DUP 2 MOD 2* 1- +
       ( 1 or 2, rem /2 1 or 0, 2 or 0, 1 or -1, 2 or 1 )
;

: USE.EXISTING.RATIO  ( Vol. Int. No. - Int. No. :: D.r. N.D.r - D.r O.D.r )
    PLUCK FDROP                              ( Get rid of old values )
    FDUP
    VOLUME.WITH.EXISTING.RATIO               ( Calculate volume and ratio required )
    SWAP FSWAP FDROP                         ( Tidy up stacks )
    DUP GET.NO.IN.INT.SOLN @                 ( Get number of entries in defn. )
    UPDATE.FOR.NEW.RATIO                     ( Update previous entries )
    DUP GET.NEXT.SOLN @                      ( Get number of next solution )
    GET.TOTAL.VOLUME                         ( Calculate volume of intermediate to be )
    F* % 1E3 F* F-> SWAP                     ( included in next solution )
    REMOVE.FROM.NEXT.CUMULATED.VOLUME        ( Remove old value )
    CHECK.CUMULATED.VOLUME        ( Check that we won't overflow next solution )
    GET.INT.DEFN 8 +
    3 PICK GET.POSN.ON.NEXT @
    4* I!                                    ( Put new value into defn. )
    DUP GET.INT.DIL.RATIO                    ( Get existing value of dil. ratio )
;

: UPDATE.FIRST.STEP
  ( Int. No. Vol. Next. Soln No. in.Int. -  Int. No. Vol. Next. Soln )

      ( When changing dilution ratio of both intermediate solution the volumes )
      ( of all previous standards need to be updated to reflect change in both )
      ( dilution ratio. This routine updates volumes to reflect the change in )
      ( the dilution ratio of the second intermediate solution )

    FDUP OVER
    GET.INT.DIL.RATIO                ( Get current value of dil. Rat. for next soln. )
    FSWAP F/ 4 PICK                  ( Calculate relative change )
    GET.INT.DEFN 8 + SWAP            ( Get defn. of this intermediate soln. )
    5 PICK GET.CUMULATED.VOLUME
    F=0 R32!                         ( Clear cumulated volume )
    0
    DO
      FDUP DUP DUP I16@               ( Get existing volume )
      F* F->                          ( Calculate new volume )
      CHECK.MINIMUM.VOLUME            ( Check not too small )
      6 PICK
      CHECK.CUMULATED.VOLUME DROP     ( Check it won't overflow )
      SWAP ! 4+                       ( Store volume and goto next entry )
    LOOP
    DROP FDROP                        ( Drop used defn. and update factor )
;

: KEEP.NEXT.SOLN      ( Vol. Int. No. Next Soln. - Vol. Int. No. :: )
    CALC.NEW.RATIO                    ( Calculate new ratio for next soln. )
    DUP GET.NO.IN.INT.SOLN @
    UPDATE.FOR.NEW.RATIO              ( Update next soln. for the new ratio )
    3 PICK GET.NO.IN.INT.SOLN @
    UPDATE.FIRST.STEP                 ( Update this intermediate for change as well )
    DUP STORE.INT.DIL.RATIO                  ( Store new ratio )
    REMOVE.FROM.NEXT.CUMULATED.VOLUME        ( Remove volume from C. V. )
    CHECK.CUMULATED VOLUME            ( Check adding the new volume won't overflow )
    GET.INT.DEFN 8 +
    3 PICK GET.POSN.ON.NEXT @
    4* I!                               Put new volume on defn. of next soln. )
    FDROP
    DUP GET.NO.IN.INT.SOLN @          ( Update this intermediate for new dilution )
    UPDATE.FOR.NEW.RATIO              ( factor of this intermediate )
;

: REMOVE.FROM.NEXT    ( Int. No Next Soln. - Int. No. )

        ( If switching to other intermediate for the next step in the dilution )
        ( need to remove the intermediate from the one it is already on )
        ( this is done by entering a zero for the volume )

    0 OVER
    REMOVE.FROM.NEXT.CUMULATED.VOLUME        ( Remove current volume from C. V. )
    GET.INT.DEFN 8 +
    4 PICK GET.POSN.ON.NEXT @
    4* I!
;

: SWITCH.NEXT.SOLN    ( Int. No. Next Soln. Curr. Soln. - Int. No. )

        ( WARNING: This code is untested !!! )
    ." Switching "
    REMOVE.FROM.NEXT                  ( Remove from defn. of current next soln. )
    DUP 3 PICK GET.NEXT.SOLN !        ( Store new next )
    DUP GET.NO.IN.INT.SOLN @
    3 PICK GET.POSN.ON.NEXT !         ( Get position on next next )
    DUP GET.NO.IN.INT.SOLN @          ( Check if anything already in new next )
    IF
      CALC.NEW.RATIO                  ( If yes, calculate new ratio for it )
      DUP GET.NO.IN.INT.SOLN @
      UPDATE.FOR.NEW.RATIO            ( Update to new ratio )
      GET.OTHER.NEXT
      3 PICK GET.NO.IN.INT.SOLN @
      UPDATE.FIRST.STEP               ( Update this intermediate for change )
      GET.OTHER.NEXT                  ( of next soln. )
      CHECK.CUMULATED.VOLUME          ( Check we can add to next )
      GET.INT.DEFN 8 +
      3 PICK GET.POSN.ON.NEXT @
      4* I!
    ELSE
      DUP GET.OTHER.NEXT
      UPDATE.FIRST.STEP
      DROP
      OVER GET.INT.DEFN 2-
      SWAP USE.INT.SOLN.1.OR.2
    THEN
;

: RESTART.TWO.STEP.DIL       ( Vol. Int. No. - Int. No. :: Dil. Rat. - )
    OVER
    BACK.CALC.DIL.RATIO           ( Calculate dilution factor in first step )
    F3PICK FSWAP F/               ( Calculate dilution still to go )
    FIND.INTERMEDIATE.TO.USE      ( Find what the next soln. required )
    DUP
    3 PICK GET.NEXT.SOLN @        ( Compare with the current soln. )
    =
    IF
      KEEP.NEXT.SOLN              ( They are the same )
    ELSE
      OVER GET.NEXT.SOLN @        ( The[illegible]e different )
      SWITCH.NEXT.SOLN
    THEN
;

: START.TWO.STEP.DIL  ( Vol. Int. No. - :: Dil. Rat. New Dil. Rat. - )
    START.NEW.INT.DEFN            ( Start a new defn. )
    OVER
    BACK.CALC.DIL.RATIO           ( Calculate dilution factor of first step )
    F3PICK FSWAP F/               ( Calculate what remains )
    FIND.INTERMEDIATE.TO.USE      ( Find which intermediate to use )
    DUP
    3 PICK GET.NEXT.SOLN !        ( Store value )
    DUP GET.NO.IN.INT.SOLN @
    3 PICK GET.POSN.ON.NEXT !     ( Store position in the defn. )
    OVER GET.INT.DEFN 2-          ( Enter this solution on defn. as if it were )
    SWAP USE.INT.SOLN.1.OR.2      ( like any other standard )
;

: UPDATE.TWO.STEP.DIL
  ( Vol. Int. No. - Vol. Int. No. :: D.r. N.D.r - D.r [illegible] )
    CHECK.EXISTING.RATIO          ( See if current ratio of next soln. can be used )
    IF
      USE.EXISTING.RATIO          ( Use it only change dil. ratio of first step )
    ELSE
      RESTART.TWO.STEP.DIL        ( No, try find a set that can be )
    THEN
;

: USE.NEW.TWO.S[illegible].RA[illegible]    ( Int. No. Vol. - Int. No. :: D.r N.D.r - Dil. Rat. )
    DROP FDROP                    ( Get rid off old values )
    CALC.NEW.TWO.STEP.RATIO       ( Calculate new values )
    DUP
    GET.NO.IN.INT.SOLN @          ( Check if anything already on defn. )
    IF
      UPDATE.TWO.[illegible]EP.DIL        ( Yes, then make necessary changes to existing )
    ELSE                          ( entries )
      START.TWO.STEP.DIL          ( No, then it is a new defn. )
    THEN
    DUP STORE.INT.DIL.RATIO       ( Store new value of dilution factor )
    ENTER.ON.INT.DEFN             ( Enter defn. )
;
```

```
: UPDATE.TWO.STEP.RATIO      ( Int. No. - Int. No. )

       ( Something entered on next soln. may have changed volume of this solution )
       ( and hence the dilution factor of this solution, using old values will )
       ( cause an error since the product of the two dilution factor will be )
       ( erroneous. Hence we recalculate the dilution factor of this solution. )

    DUP GET.NO.IN.INT.SOLN @        ( Check if there are entries in the defn. )
    IF
      DUP GET.NEXT.SOLN @           ( Yes, get next solution )
      GET.INT.DEFN 8 +
      OVER GET.POSN.ON.NEXT @
      4* I@ ->F % 1E3 F/            ( Get volume of intermediate )
      GET.TOTAL.VOLUME F/           ( Get total volume of next solution )
      DUP STORE.INT.DIL.RATIO       ( Store result )
    THEN
;

: USE.INT.SOLN.3.OR.4        ( Int. No. - :: Dil. Rat. - )
    UPDATE.TWO.STEP.RATIO           ( Get current dilution factor )
    CHECK.OLD.RATIO                 ( See if old values can be used )
    IF
      USE.OLD.RATIO                 ( Use old values )
    ELSE
      USE.NEW.TWO.STEP.RATIO        ( Try use new values )
    THEN
    1+ SOLUTION.CONTENTS 2+ COMPONENT.NO @ 4* I!
    0 SOLUTION.CONTENTS COMPONENT.NO @ 1+ 4* I!
;

: USE.INTERMEDIATES    ( - :: Dil. Rat. - )
    FIND.INTERMEDIATE.TO.USE        ( Find which solution to use first )
    DUP
    BEGIN-CASE
      1 CASE-OF
          USE.INT.SOLN.1.OR.2
      ELSE
      2 CASE-OF
          USE.INT.SOLN.1.OR.2
      ELSE
      3 CASE-OF
          USE.INT.SOLN.3.OR.4
      ELSE
      4 CASE-OF
          USE.INT.SOLN.3.OR.4
      ELSE
      TOO.DILUTE.ERROR              ( Force too dilute error )
       ( Concentration desired too dilute to be done in a three step dilution )
       ( and I'm NOT going to even think about how to use more stages )
       ( TOUGH CHUCK CHUM )
    END-CASE
;

: FINISH.INTERMEDIATES        ( - )
       ( Finish definition of intermediate solutions )
    5 1
    DO
      I GET.NO.IN.INT.SOLN @ ?DUP   ( Check to see if any entries are in it )
      IF
        I GET.INT.DEFN !            ( Store number of components )
        I GET.TOTAL.VOLUME
        I GET.CUMULATED.VOLUME
        R32@ F- % 1E3 F*            ( Calculate volume of solvent needed )
        I GET.INT.DEFN 4+ I16!      ( Store it )
        I 1 =
        I 2 =
        OR                            ( Check if these are second stage ints. )
        IF
          I GET.INT.DIL.RATIO         ( Get dilution factor )
          I GET.INT.DEFN 2-
          NO.IN.SAMPLE @
          I GET.POSN.ON.NEXT !        ( Store position of intermediate )
          ENTER.ON.SAMPLE.SOLUTION    ( Enter intermediate on final soln. defn. )
        THEN
      THEN
    LOOP
;

: MAKE.INTERMEDIATES  ( - )
    1 4                               ( Make intermediates in reverse order )
    DO
      I GET.NO.IN.INT.SOLN @          ( Check if soln. defined )
      IF
        I GET.INT.DEFN                ( Get Defn. )
        MAKE.SOLUTION
        ( Debugging code )
          ." Intermediate soln. #" I . CR )
        ( SOLUTION.PRINT )
        ( KEY DROP )
      THEN
    -1
    +LOOP
;

VARIABLE COUNTER

: FIND.MIN     ( 0 n1 n2 n3 .... - 0 ni nj ... min(n) )
    2 COUNTER !
    BEGIN
      COUNTER @ DUP 1+
      PICK                            ( Get next number on the stack, while not zero )
    WHILE
      ROLL                            ( Get the number )
      OVER OVER <                     ( Make copies of top two numbers and compare )
      IF
        SWAP                          ( If new low swap )
      THEN
      COUNTER 1+!                     ( Increment counter )
    REPEAT                            ( Repeat until zero is reached )
    DROP                              ( Drop zero )
;
```

```
: EMPTY.INTERMEDIATES         ( - )

       ( Empty intermediate solutions, empty vials in first row first to avoid )
       ( collisions when emptying vials in back row )

    0                                ( Mark end of list with zero )
    5 1
    DO
      I GET.NO.IN.INT.SOLN @         ( Check if intermediate was used )
      IF
        I GET.INT.DEFN 2+ @          ( If so get position of bottle )
      THEN
    LOOP
    BEGIN
      DUP                            ( Check if reached marker )
    WHILE
      FIND.MIN                       ( Find next bottle to empty )
      DUMP.SOLUTION                  ( Empty it )
    REPEAT
;

: INIT.INTERMEDIATES  ( - )

       ( Initialise variables used in definition of intermediates )

    5 1
    DO
      F=0                            ( Zero volumes )
      I GET.CUMULATED.VOLUME R32!
      I GET.NO.IN.INT.SOLN 0!        ( Zero counter )
      F=1 I STORE.INT.DIL.RATIO      ( No dilution )
    LOOP
;

: PRINT.PPM    ( :: conc/ppm - conc/ppm )
  ( Print concentration in appropriate w/v units )
    FDUP F=1 F-
    F0<
    IF
      FDUP % 1E3 F*
      7 6 F.R ." ppb"                ( if conc < 1ppm print in ppb )
      9 EMIT
    ELSE
      FDUP % 1E3 F-
      F0>
      IF
        FDUP % 1E4 F/
          6 F.R ." %"                ( if conc > 1000ppm print in % )
        9 EMIT
      ELSE
        FDUP
        7 6 F.R ." ppm"              ( print in ppm )
        9 EMIT
      THEN
    THEN
;

: PRINT.MOLARITY      ( soln. conts. - soln. conts. :: conc/ppm - )
    DUP 2- @ 1- 4*
    AT.WEIGHTS + R32@ F/             ( Calculate milliMolarity )
    FDUP F=1 F- F0<
    IF
      % 1E3 F*
      7 6 F.R 230 EMIT ." M"         ( if conc < 1mM print microMolarity )
      CR
    ELSE
      FDUP % 1E3 F- F0>
      IF
        % 1E3 F/
        7 6 F.R ." M"              ( if conc > 1000mM print M )
        CR
      ELSE
        7 6 F.R ." mM"             ( print mM )
        CR
      THEN
    THEN
;

: PRINT.CONC   ( :: conc/ppm - )
    PRINT.PPM
    PRINT.MOLARITY
;

: PRINT.SUMMARY.2
    SOLUTION.CONTENTS          ( Get contents of solution, tells what was in it )
    BEGIN                      ( and what solution the original std. was pipetted )
      DUP @                    ( into. Get element, while not zero )
    WHILE
      DUP @                    ( Get channel number )
      [ ARL ]
      ELEMENT.SYMBOLS
      [ FORTH ]
      SWAP
      0
      DO
        COUNT +
      LOOP
      $. 9 EMIT                ( Print element symbol )
      CALC.PPM.CONC.2          ( Calculate concentration in ppm )
      PRINT.CONC               ( print conc w/v and molarity )
      2+                       ( Goto next entry )
    REPEAT
    DROP
;

: PRINT.DESIRED.SOLN  ( - )
    ." Standard asked for " CR
    SOLN.COMP DUP @
    SWAP 2+ SWAP
    0
    DO
      DUP C@
      [ ARL ]
      ELEMENT.SYMBOLS
      [ FORTH ]
      SWAP
      0
      DO
        COUNT +
      LOOP
      $. 9 EMIT                      ( Print element symbol )
      1+ DUP  R32@                   ( Get concentration )
      PRINT.PPM                      ( Print concentration in w/v units )
      4+ FDROP CR                    ( Goto next entry and drop concentration )
    LOOP
    DROP
;
```

```
: PREP.SOLN    ( - )

      ( Prepare the solution defined by the ARL )

  CAL.SOLN.NO @
  ." Calibaration Solution #" . CR  ( Print calibration number )
  INIT.PREP                         ( Initialise variables )
  INIT.INTERMEDIATES                ( Initialise variable used for intermediates )
  SOLN.COMP DUP @ SWAP 2+ SWAP DUP  ( Get number of components )
  0
  DO
    OVER C@ ROT 1+ -ROT             ( Get channel number, element )
    [ ARL ]
    DUP SOLUTION.CONTENTS I 4* I!   ( Store in Soln. conts. )
    3 PICK R32@ FDUP
    ROT 4+ -ROT DUP                 ( Get concentration required )
    FIND.STOCK.SOLN                 ( Find record of stock solution )
    DUP
    2+ I16@ F/                      ( Get conc. of stock )
    FDUP                            ( Calculate dilution factor )
    CALC.VOLUME                     ( Calculate volume required )
    FSWAP F0=
    IF
      DROP                          ( If volume is zero enter anyway )
      ENTER.ON.SAMPLE.SOLUTION      ( so that concentration will still be calculated )
      1 SOLUTION.CONTENTS 2+ I 4* I!
      0 SOLUTION.CONTENTS I 1+ 4* I!
    ELSE
      MINIMUM.VOLUME @              ( Compare with minimum allowed )
      < NOT
      IF
        ENTER.ON.SAMPLE.SOLUTION    ( Can be prepared in one step )
        1 SOLUTION.CONTENTS 2+ I 4* I!
        0 SOLUTION.CONTENTS I 1+ 4* I!
      ELSE
        I COMPONENT.NO !            ( Needs more than one step )
        USE.INTERMEDIATES
      THEN
    THEN
    DDROP
  LOOP
  DROP
  0 SWAP ELEMENT.ARRAY + C!
  FINISH.INTERMEDIATES              ( Finish definition of intermediates )
  FINISH.SAMPLE                     ( Finish definition of sample )
  MAKE.INTERMEDIATES                ( Make intermediate solutions )
  DEFAULT.SAMPLE.SOLUTION           ( Make sample solution )
  MAKE.SOLUTION
  ( ." Sample Solution Composition " CR )   ( Debugging code )
  ( SOLUTION.PRINT )
  EMPTY.INTERMEDIATES               ( Empty intermadiate solutions )
  PRINT.SUMMARY.2                   ( Print actual concentrations )
  [ FORTH ]
;

: TAKE.STANDARD.TO.ARL       ( - )
    DEFAULT.SAMPLE.SOLUTION 2+ @    ( Get position of solution )
    SAMPLE.TO.ARL                   ( Carry it over to ARL )
;
```

```
: DO.SOLUTION.MEASURE        ( - )
    [ ROBOT ]
    FRWRD ARL.PROBE                ( Pick up probe )
    1 1 ARL.SAMPLES                ( Get position of standard )
    START.MEASURE.SAMPLE           ( Start sampling )
    WAIT.ROBOT                     ( Wait for robot to complete move )
    ARL.PORT SELECT.DEVICE
    [ ARL ]
    13 CEMIT
    ASCII S
    ASCII I                        ( Send start signal )
    SYNC.ARL
    FORCE.FLUSH                    ( Forced preflush )
    NO.REPL @
    0
    DO
      I GET.DATA                   ( Get emission intensities )
    LOOP
    GET.MEAN                       ( Get average intensities )
    [ ROBOT ]
    STOP.MEASURE.SAMPLE            ( Stop sampling )
    WAIT.ROBOT                     ( Wait for the slow coach )
    BCKWRD ARL.PROBE               ( Return sampling probe )
    EMPTY.SAMPLE                   ( Empty the bottle )
    WAIT.ROBOT                     ( Waiting again )
;

: MAKE.STANDARD       ( - )
   CAL.SOLN.NO 1+!                 ( Increment calibration point number )
   [ ROBOT ]
   FRWRD TO.PROBE.PATH             ( Goto sample preparation area )
   WAIT.ROBOT                      ( Wait for him to arrive )
   GET.CONCENTRATIONS              ( Get solution composition from ARL )
   TIME. CR                        ( Time stamp )
   PRINT.DESIRED.SOLN              ( Print solution composition required )
   PREP.SOLN                       ( Make solution )
   TIME. CR                        ( Time stamp, can calculate how long it takes )
   RETURN.CONCENTRATIONS           ( Send actual concentrations to ARL )
   [ FORTH ]
;

: MEASURE.STANDARD    ( - )
    [ ARL ]
    9 EMIT LIST.ELEMENTS CR        ( Print element list )
    START.BLANK.MEASURE            ( Start measuring blank )
    TAKE.STANDARD.TO.ARL           ( Move standard over to measurement area )
    ARL.PORT SELECT.DEVICE
    GET.BLANK.DATA                 ( Get intensity data on blank )
    ." Calibration Solution #" CAL.SOLN.NO @ . CR
    DO.SOLUTION.MEASURE            ( Measure standard )
;
```

```
: ?DONE         ( - flag )
    [ ARL ]
    ARL.PORT SELECT.DEVICE
    ASCII O
    ASCII D
    N.RECEIVE.ARL                     ( Get completion flag from ARL )
    DUP
    NOT                               ( If ARL not ready to give up )
    IF
      DROP                            ( Check if user is )
      CRT                             ( Need to use the screen )
      ." Do you wish to continue "    ( Ask the question )
      ?Y/N CR                         ( Get the reply )
      >FILE                           ( Return output to file )
      NOT
    THEN
;

: AUTO.ANALYSIS       ( - )
    GET.SAMPLES.INFO                  ( Get user supplied information on samples )
    GET.FILE                          ( Get filerame and open )
    INIT.ARL                          ( Initialise the ARL )
    MEASURE.SAMPLES                   ( Measure the samples and get initial estimates )
    BEGIN                             ( Start calibration )
      MAKE.STANDARD
      MEASURE.STANDARD
      GET.REGRESSION.DATA
      GET.SAMPLES.RESULTS
      ?DONE                           ( Check if done )
    UNTIL                             ( Repeat until done )
;
```

Appendix D

Listing of Matrix Study Program

This appendix contains the program listings used for performing automated matrix studies. The bulk of program is similar to that used for automated analysis as listed in Appendix C. The main difference is that some routines have been modified for the inclusion of the matrix component in the solutions prepared.

The matrix component uses the channel number 35 to distinguish it from the other elements. Data about the matrix is stored in separate records defined at the start of the program listing. The concentrations of matrix to be included in the solutions prepared are store in an array (MATRIX.CONCENTRATIONS). The definition of the solutions prepared are made from this array and the variable ANALYTE.CONCENTRATION.

The study starts by typing in "MATRIX.STUDY" the program will ask about the elements to be studied, the number of replicate integrations and the name of the file in which the data is to be stored. Once this has been done a solution containing just the analyte elements will be made. While this solution is taken to the measurement area the distilled water blank will be measured and then the analyte solution. After this for each matrix concentration two solutions are made, one containing just the matrix, the matrix blank, and one containing matrix and analytes, the full solution. These are then measured and the relative sensitivities are sent back and stored in the data file.

```
CREATE MATRIX.SOLN 6 ALLOT          ( Record for matrix stock solution )

CREATE MATRIX.MOL.WEIGHT 4 ALLOT    ( Molecular weight of matrix )
CREATE MATRIX.NAME 8 ALLOT          ( Name for matrix )

CREATE MATRIX.CONCENTRATIONS 20 ALLOT       ( Array of matrix conc. to be used )
VARIABLE ANALYTES.CONCENTRATION             ( Analyte concentration )
VARIABLE DATA.POINT.NO                      ( Point number )
VARIABLE NO.OF.DATA.POINTS                  ( Number of points to be collected )
VARIABLE SOLN.COMP 35 ALLOT                 ( Definition of solution )
VARIABLE NO.REPL                            ( Number of replicate integrations )
CREATE INTERMEDIATE.SOLUTIONS 152 ALLOT     ( Space for definition of intermendiates )
CREATE CUMULATED.INT.VOLUMES 16 ALLOT       ( Volume of stds in int. solutions )
CREATE INT.DIL.RATIOS 16 ALLOT              ( Dilution ratios for int. solutions )
CREATE NO.IN.INT 8 ALLOT                    ( Number of components in solution )
CREATE POSN.ON.NEXT 8 ALLOT                 ( Position of int on next solution )
VARIABLE INT.3.ON                           ( Next intermediate solution for 3 )
VARIABLE INT.4.ON                           ( Next intermediate solution for 4 )

5 NO.REPL !                         ( Set default values )
10 MAX.POINTS !
2 INT.3.ON !
2 INT.4.ON !

: GET.ELEMENT.AUX     ( - element )
   CR
   ." Please enter the next element for the solution "
   IN$ DROP                         ( Get user input )
   CONTEXT ,@                       ( Get current vocabulary )
   ARL-SEG 14 + CONTEXT ,!          ( Switch to ARL vocabulary )
   PLOAD SWAP CONTEXT ,!            ( Interpret user response and restore vocabulary )
;

: ANOTHER.ELEMENT?    ( - flag )
    NO.ELEMENTS DUP 1+! @
    7 = NOT DUP                     ( More than 7 elements will exceed capacity )
    IF
      DROP CR
      ." Another element " ?Y/N     ( Ask the question )
    THEN
;

: GET.ELEMENTS        ( - )
    [ ARL ]
    CR
    LIST.ELEMENTS CR                        ( Print current list of elements )
    ." Use these elements " ?Y/N            ( Are these to be used? )
    NOT
    IF
      NO.ELEMENTS 0!                        ( If not, zero counter )
      BEGIN
        [ ARL ]
        GET.ELEMENT                         ( Get element from user )
        ELEMENT.ARRAY NO.ELEMENTS @ + C!    ( Store in list )
        ANOTHER.ELEMENT?                    ( Ask if there are any more )
        NOT                                 ( Repeat until no more )
      UNTIL
      0 ELEMENT.ARRAY NO.ELEMENTS @ + C!    ( Element list is NUL terminated )
    THEN
    NO.ELEMENTS @ SOLN.COMP !              ( Store number of components in SOLN.COMP )
 ;

 : GET.NO.REPL  ( - )
    CR
    ." Use " NO.REPL @ . ." integrations per sample "
    ?Y/N                              ( Check if current value can be used )
    NOT
    IF
      CR                              ( If not get new value )
      ." How integrations per sample "
      IN# NO.REPL !
    THEN
 ;

 : GET.FILE     ( - )
    CR
    ." Please enter filename "        ( Ask the question )
    IN$ MAKE-OUTPUT CR                ( Get filename and open file )
    >FILE                             ( Transfer output to file )
 ;

 : GET.INFO     ( - )
    DATA.POINT.NO 0!                  ( Initialise counter )
    GET.ELEMENTS                      ( Get elements to be studied )
    GET.NO.REPL                       ( Get numeber of replicates )
    GET.FILE                          ( Get filename and open )
 ;

 : SEND.NO.OF.DATA.POINTS      ( - )
    [ ARL ]
    NO.OF.DATA.POINTS @               ( Get number )
    ASCII D                           ( Synchronisation characters )
    ASCII N
    N.SEND.ARL                        ( Send integer )
 ;

 : INIT.ARL                    ( - )
   [ ARL ]
   ARL.PORT SELECT.DEVICE
   RESTART.ARL                        ( Restart progam on ARL )
   PAD COMPORT @ GET.RETURN DROP      ( Get program name and ignore )
   CKEY DROP                          ( Get LF character )
   13 CEMIT                           ( Send CR to ARL )
   SEND.ELEMENTS                      ( Send channel numbers )
   SEND.PREFLUSH                      ( Send preflush time )
   SEND.INTEGRATION                   ( Send integration time )
   NO.REPL @ SEND.REPLICATES          ( Send number of replicate integrations )
   SEND.NO.OF.DATA.POINTS             ( Send number of data points to be collected )
 ;

 : GET.DATA     ( Integration number - )
    [ ARL ]
    ASCII I
    WAIT.ARL                          ( Wait till integration complete )
    ASCII R
    ASCII E
    N.RECEIVE.ARL                     ( Get error code )
    0= NOT                            ( Check if error occured )
    IF
      ABORT" Integration Error"
    THEN
    1+ . 9 EMIT                       ( Print integration number and TAB )
    ELEMENT.ARRAY
    BEGIN
```

```
      DUP C@                          ( For each element get a value )
    WHILE
      ASCII D                         ( Synchronisation characters )
      ASCII R
      F.RECEIVE.ARL                   ( Get emission intensity )
      4 6 F.R 9 EMIT 1+               ( Print value to 4 sig figs. in 6 char field )
    REPEAT
    DROP CR
;

: GET.MEAN     ( - )
    [ ARL ]
    ." Mean" 9 EMIT
    ELEMENT.ARRAY
    BEGIN
      DUP C@
    WHILE
      ASCII D
      ASCII M
      F.RECEIVE.ARL                   ( Get average emission intensity )
      4 6 F.R 9 EMIT 1+
    REPEAT
    DROP CR
;

: START.BLANK.MEASURE          ( - )
    ." Blank " CR                     ( Print header )
    ARL.PORT SELECT.DEVICE
    13 CEMIT                          ( Send CR to ARL )
    [ ARL ]
    ASCII M
    ASCII B
    SYNC.ARL                          ( Send start signal )
;

: GET.BLANK.DATA       ( - )
    NO.REPL @
    0
    DO
      I GET.DATA                      ( Get individual intensities )
    LOOP
    GET.MEAN                          ( Get average intensity )
;

: MEASURE.BLANK        ( - )
    START.BLANK.MEASURE
    GET.BLANK.DATA
;

: GET.RESPONSE         ( - )
    [ ARL ]
    ARL.PORT SELECT.DEVICE
    ASCII O
    ASCII C
    F.RECEIVE.ARL                     ( Get matrix concentration )
    FDUP CRT 7 6 F.R >FILE            ( Print on screen and in file )
    7 6 F.R
    BEGIN
      ASCII L
      ASCII E
      N.RECEIVE.ARL                   ( Get channel number )
    WHILE
      ASCII S
      ASCII R
      F.RECEIVE.ARL                   ( Get relative sensitivity )
      FDUP CRT 9 EMIT 7 6 F.R >FILE ( Print Tab and then value )
      9 EMIT 7 6 F.R
    REPEAT                          ( Repeat until channel zero received )
    CR CRT CR >FILE                 ( Terminate line )
;

: GET.RESPONSE.MATRIX        ( - )
    [ ARL ]                         ( Get all the data )
    ." Response Matrix" CR
    9 EMIT LIST.ELEMENTS CR         ( Print element symbols )
    NO.OF.DATA.POINTS @ 1+          ( Loop for each data point )
    0
    DO
      GET.RESPONSE                  ( Get relative sensitivities )
    LOOP
;

: GET.POSN.ON.NEXT    ( int no - Addr posn on next soln )
    1- 2*                           ( Calculate offset )
    POSN.ON.NEXT +                  ( Calculate address )
;

: GET.NEXT.SOLN       ( Int no. - Addr next int soln )
    3 =
    IF
      INT.3.ON
    ELSE
      INT.4.ON
    THEN
;

: GET.NO.IN.INT.SOLN  ( Int. No. - Addr of No. in Soln. )
    1- 2*                           ( Calculate offset )
    NO.IN.INT +                     ( Calculate address )
;

: GET.INT.DEFN        ( Int. No. - Addr of Defn )
    1- 38 *                         ( Calculate offset )
    INTERMEDIATE.SOLUTIONS +        ( Calculate address )
;

: GET.INT.DIL.RATIO   ( Int No. - :: - Dil. Rat. )
    1- 4*                           ( Calculate offset )
    INT.DIL.RATIOS + R32@           ( Calculate address and fetch )
;

: STORE.INT.DIL.RATIO        ( Int No. - :: - Dil. Rat. )
    1- 4*                           ( Calculate offset )
    INT.DIL.RATIOS + R32!           ( Calculate address and store )
;

: GET.CUMULATED.VOLUME       ( Int. No. - Addr of Vol. )
    1- 4*                           ( Calculate offset )
    CUMULATED.INT.VOLUMES +         ( Calculate address )
;
```

```
: GET.TOTAL.VOLUME    ( Int. No. - Int. No. :: - Total Vol. )
   DUP
   1 = OVER 2 =                ( If intermediate 1 or 2 )
   OR
   IF
     TOTAL.VOLUME              ( use TOTAL.VOLUME )
   ELSE
     TOTAL.SERIAL.VOLUME       ( 3 or 4 use TOTAL.SERIAL.VOLUME )
   THEN
   R32@                        ( Get volume )
;

: FIND.ENTRY.ON.INT   ( Posn Int. No. - Entry Recd. )
   GET.INT.DEFN 6+             ( Get solution definition )
   BEGIN
     DUP @                     ( Get posn of component )
     3 PICK                    ( Copy of desired position )
     = NOT                     ( If not the same )
   WHILE
     4+                        ( Goto next entry )
   REPEAT
;

: FIND.STOCK.CONC     ( Recd. - Recd. :: - Conc. )
   DUP 2- @                    ( Get position of stock solution )
   DUP
   35 =                        ( Check if matrix )
   IF
     MATRIX.SOLN               ( Get record for matrix solution )
   ELSE
     FIND.STOCK.SOLN           ( Find record of stock solution )
   THEN
   DUP 2+ I16@ 4+ @ PLUCK      ( Get concentration )
;

: CALC.FINAL.CONC     ( :: Volume - Conc/ppm )
  F* TOTAL.VOLUME R32@ % 1E3 F* F/  ( Conc*volume/total volume )
;

: CALC.CONC.FROM.SERIAL      ( Int. No. - :: Volume - Conc )
   F*                          ( conc*volume )
   TOTAL.SERIAL.VOLUME R32@ % 1E3  ( convert ml to µl )
   F* F/                       ( divide by total volume, = conc, in int. soln. )
   GET.POSN.ON.NEXT @          ( Find int. soln. on next soln. defn. )
   DEFAULT.SAMPLE.SOLUTION 8 + ( Get entry in sample definition )
   SWAP 4* I@ ->F              ( Get volume of int. used, send to f.p. stack )
   CALC.FINAL.CONC             ( Calculate concentration in final solution )
;

: CALC.CONC.FROM.START       ( Int. No. - :: Vol. - Conc. )
   F*
   TOTAL.SERIAL.VOLUME R32@
   % 1E3 F* F/ DUP             ( Calculate conc. in int. soln. )
   GET.NEXT.SOLN @             ( Find which int. soln. used for next step )
   SWAP GET.POSN.ON.NEXT @     ( Find record of this soln. on next )
   OVER GET.INT.DEFN 8 +       ( Get record )
   SWAP 4* I@ ->F              ( Get volume of solution used )
   CALC.CONC.FROM.SERIAL       ( Continue calculation )
;

: CALC.PPM.CONC.2     ( Soln.Contents - Soln.Contents+ :: - Conc. )
   2+ DUP @                    ( Find solution at start of dilution )
   1 =
   IF                          ( Single step dilution )
```

```
      FIND.STOCK.CONC                  ( Get concentration of stock solution )
      FIND.ENTRY.ON.SAMPLE             ( Find record for component )
      2+ I16@ DROP                     ( Get volume of stock solution used )
      CALC.FINAL.CONC                  ( Calculate concentration )
    ELSE                               ( More than one step dilution )
      FIND.STOCK.CONC                  ( Get concentration of stock solution )
      OVER @ 1-                        ( Get intermediate solution number )
      FIND.ENTRY.ON.INT                ( Find record of component on soln. defn. )
      2+ I16@ DROP                     ( Get volume of staock solution used )
      DUP @ 1-                         ( Get int. soln. no. )
      DUP 1 =
      OVER 2 =
      OR                               ( Was a two step dilution used )
      IF
        CALC.CONC.FROM.SERIAL          ( Calculate concentration for two step dil. )
      ELSE
        CALC.CONC.FROM.START           ( Calculate concentration for three step dil. )
      THEN
    THEN
;

: RETURN.CONCENTRATION        ( - )
    [ ARL ]
    SOLUTION.CONTENTS
    BEGIN
      DUP @
      35 = NOT                         ( Find matrix on record )
    WHILE
      4+
    REPEAT
    CALC.PPM.CONC.2                    ( Calculate concentration of matrix )
    ARL.PORT SELECT.DEVICE
    4                                  ( Number of places )
    ASCII O
    ASCII C
    F.SEND.ARL                         ( Send concentration )
;

: FCBRT          ( :: x - x^1/3 )
    FABS FLOG % 3 F/ F10**             ( Calculate cube root of dilution factor )
;

: FROOT          ( Int. No. :: x - x^1/n )
    DUP                                ( Calculate appropriate root of dilution factor )
    1 = OVER 2 =
    OR
    IF
      FSQRT                            . Square root for two step dilution )
    ELSE
      FCBRT                            ( Cube root for three step dilution )
    THEN
;

: TOO.DILUTE.ERROR     ( - )
    ABORT" Component to dilute to be made" ( Print message, return control to user )
;

: CHECK.MINIMUM.VOLUME        ( Vol. - Vol. )
    DUP MINIMUM.VOLUME @
    <                                  ( Compare volume to minimum allowed )
    IF
      CRT                              ( If too small, switch output to screen )
      TOO.DILUTE.ERROR                 ( Abort with error message )
    THEN
;
```

```
: CHECK.OVERFLOW       ( Vol. Int. No. - Vol. Int. No. flag )
   OVER ->F % 1E3 F/               ( Send volume to f.p. stack and convert ml )
   DUP GET.CUMULATED.VOLUME DUP    ( Get cumulated volume )
   R32@ F+ FDUP OVER               ( Fetch and add new volume )
   GET.TOTAL.VOLUME DROP           ( Get total required )
   F-                              ( Take difference )
   F0>                             ( Check for overflow )
;

: GET.OVERALL.DIL.RATIO       ( Int. No. - :: - Dil. Rat. )
   BEGIN-CASE
     1 CASE-OF
         1 GET.INT.DIL.RATIO         ( Get dilution ratio )
     ELSE
     2 CASE-OF
         2 GET.INT.DIL.RATIO
     ELSE
     3 CASE-OF
         3 GET.INT.DIL.RATIO         ( Get first dilution ratio )
         INT.3.ON @
         GET.INT.DIL.RATIO           ( Get second dilution ratio )
         F*                          ( Calculate overall dilution ratio )
     ELSE
     4 CASE-OF
         4 GET.INT.DIL.RATIO
         INT.4.ON @
         GET.INT.DIL.RATIO
         F*
     ELSE
     DROP
   END-CASE
;

: CHECK.OLD.RATIO     ( Int. No. - Vol. Flag :: Dil. Rat. - Dil. Rat. Old. D. R. )
   DUP                             ( Check if existing dilution ratio can be used )
   GET.OVERALL.DIL.RATIO           ( Get existing dilution ratio )
   CALC.VOL.OF.STD.IN.SERIAL       ( Calculate volume of standard required )
   DUP MINIMUM.VOLUME @ <          ( Check if too small )
   NOT                             ( If not then can use existing values )
;

: FIND.NEXT.SYRINGE.VOLUME   ( Vol - Vol' )
   FIND.SYRINGE.VOLUME @ PLUCK     ( Find closest allowed volume )
;

: BACK.CALC.SYRINGE.VOLUME   ( Int. No. - Int. No. Vol :: Dil. Rat. - )
   GET.TOTAL.VOLUME          ( From dilution ratio calculate volume to be used )
   F* % 1E3 F* F->
;

: SYRINGE.VOLUME.AVAILABLE   ( Int. No. - flag :: Dil. Rat. - Dil. Rat. )
   FDUP
   BACK.CALC.SYRINGE.VOLUME        ( Calculate volume from dilution ratio )
   PLUCK
   FIND.SYRINGE.VOLUME PLUCK       ( Find allowed volume )
   SERIAL.VOLUMES 10 + >           ( Check for overrun )
   NOT                             ( If no overrun then a volume is available )
;

: BACK.CALC.DIL.RATIO        ( Int. No. Vol. - Int. No. :: - Dil. Rat. )
   ->F % 1E3 F/                    ( From volume calculate dilution ratio )
   GET.TOTAL.VOLUME
   F/
;
```

```
: CALC.IDEAL.VOLUME   ( Int. No. - Int. No. Vol. :: Dil. Rat. - Dil. Rat. )
( From overall dilution ratio calculate ideal volume )
    FDUP FROOT                        ( Take nth root )
    GET.TOTAL.VOLUME                  ( Multiply into total volume )
    F* % 1E3 F*                       ( Convert to µl )
    F->                               ( Send to integer stack )
;

: CALC.NEW.RATIO      ( Int. No. - Vol. Int. No. :: Dil. Rat. - Dil. Rat. N.D.R. )
    CALC.IDEAL.VOLUME                 ( Calculate ideal volume )
    FIND.BEST.SYRINGE.VOLUME          ( Find closest allowed volume )
    BACK.CALC.DIL.RATIO               ( Calculate actual dilution ratio )
    CALC.VOL.OF.STD.IN.SERIAL         ( Calculate volume of stock required )
    SWAP                              ( Rearrange the stack )
;

: CALC.NEW.TWO.STEP.RATIO
 ( Int. No. - Vol. Int. No. :: Dil. Rat. - Dil. Rat. N.D.R. )
    CALC.IDEAL.VOLUME
    FIND.BEST.SYRINGE.VOLUME
    BACK.CALC.DIL.RATIO         ( Calculate dilution ratio of last step )
    FOVER FOVER F/              ( Calculate dilution ratio required of first two steps )
    DUP 2- CALC.IDEAL.VOLUME          ( Calculate ideal volume for first two steps )
    FDROP
    FIND.BEST.SYRINGE.VOLUME
    BACK.CALC.DIL.RATIO DROP          ( Calculate dilution ratio of second step )
    FOVER F*                          ( Calculate dilution ratio of last two steps )
    FROT FSWAP
    CALC.VOL.OF.STD.IN.SERIAL         ( Calculate volume of stock solution required )
    SWAP FDROP FSWAP                  ( Clean up stack )
;

: OVERFLOW.ERROR      ( - )
    CRT                               ( Send output to screen )
    FDROP CR
    ." This solution cannot be prepared from the stock solutions present"  H
    ." as the total volume of stock solutions will exceed the total volume "
    ." required"
    ABORT                             ( Print message and return control to user )
;

: CHECK.CUMULATED.VOLUME     ( Vol. Int. No. - Vol Int. No. )
    CHECK.OVERFLOW                    ( Check if overflow can occur )
    IF
      OVERFLOW.ERROR                  ( If so abort )
    THEN
    R32'                              ( Store new cumulated volume )
;

: UPDATE.FOR.NEW.RATIO
 ( Int. No. No. in Soln. - Int. No.:: Dil. Rat. - Dil. Rat. )
    FDUP OVER                         ( Copy dilution ratio and entry no. )
    GET.INT.DIL.RATIO                 ( Get old value of dilution ratio )
    FSWAP F/ OVER                     ( Calculate update factor )
    GET.INT.DEFN 8 + SWAP             ( Get definition record )
    3 PICK GET.CUMULATED.VOLUME       ( Get old cumulated volume )
    F=0 R32'                          ( Zero volume )
    0
    DO                                ( Loop for each entry )
      FDUP DUP DUP I6@                ( Copy update factor, get old volume )
      F* F->                          ( Calculate new volume )
      CHECK.MINIMUM.VOLUME            ( Check not too small )
      4 PICK
      CHECK.CUMULATED.VOLUME DROP     ( Check does not overflow )
      SWAP ! 4+                         ( Store new volume and goto next entry )
    LOOP
    DROP FDROP                          ( Clean up stacks )
  ;

: UPDATE.VOLUME        ( Vol. Int. No. - Vol' Int. No. :: Dil. Rat. - Dil. Rat. )
    FDUP DUP
    GET.INT.DIL.RATIO
    FSWAP F/                            ( Calculate update factor )
    SWAP
    ->F F* F->                          ( Calculate new volume )
    CHECK.MINIMUM.VOLUME                ( Check not too small )
    SWAP                                ( Tidy up )
  ;

: INCREASE.VOLUME.OF.INT.SOLN           ( Vol Int. No. - Vol' Int. No. :: D. R. - D. R.' )
    BACK.CALC.SYRINGE.VOLUME 1+         ( Calculate current value of syringe vol. )
    FIND.NEXT.SYRINGE.VOLUME            ( increase value to go to next vol. )
    BACK.CALC.DIL.RATIO                 ( Calculate new dilution factor )
    DUP GET.NO.IN.INT.SOLN @
    UPDATE.FOR.NEW.RATIO                ( Update vols. of all entries in defn. )
    UPDATE.VOLUME                       ( Update vol. of this component )
  ;

: START.NEW.INT.DEFN  ( Int. No. - Int. No. )
    DUP GET.CUMULATED.VOLUME
    F=0 R32!                            ( Zero total volume of components )
    GET.NEXT.SERIAL                     ( Get posn. of next empty vial )
    OVER
    GET.INT.DEFN 2+ !                   ( Store posn. of vial )
  ;

: ENTER.ON.INT.DEFN   ( Stock Recd. Vol Int. No. - :: Dil. Rat. - )
    CHECK.CUMULATED.VOLUME              ( Check that volume won't cause overflow )
    DUP                                 ( also update cumulated volume )
    GET.INT.DEFN                        ( Get solution defn. )
    4 ROLL 4+ @                         ( Get posn. of stock solution )
    OVER 6+                             ( Calculate base offset for posns. )
    4 PICK
    GET.NO.IN.INT.SOLN @ 4* I!          ( Calculate entry offset and store posn. )
    8 + ROT SWAP                        ( Calculate base offset for volumes )
    3 PICK
    GET.NO.IN.INT.SOLN @ 4* I!          ( Calculate entry offset and store volume )
    DUP GET.NO.IN.INT.SOLN 1+!          ( Increment number of components in solution )
    FDROP                               ( Drop dilution ratio )
  ;

: USE.OLD.RATIO       ( Vol. Int. No. - Int. No. :: Dil. Rat. - )

      ( Some entries are already in solution defn. and we can use the existing )
      ( dilution factor )

    SWAP
    BEGIN
      CHECK.OVERFLOW            ( Check to see if adding this volume will cause )
      PLUCK FDROP               ( an overflow, while yes and while a syringe volume )
      OVER                      ( is available reduce volume of std. )
      SYRINGE.VOLUME.AVAILABLE
      AND                       ( and increase volume of intermediate on next soln. )
    WHILE
      INCREASE.VOLUME.OF.INT.SOLN
    REPEAT                      ( We will assume some changes have been made )
    DUP STORE.INT.DIL.RATIO     ( Store new dilution factor )
    ENTER.ON.INT.DEFN           ( Enter this component of defn. )
;
```

```
: USE.NEW.RATIO         ( Vol. Int. No. - Int. No. :: Dil. Rat. - )
    DROP FDROP                    ( Can't use existing values of dilution ratios )
    CALC.NEW.RATIO                ( Get rid of old values and calculate new ones )
    DUP
    GET.NO.IN.INT.SOLN @ ?DUP     ( Check to see if anything already on defn. )
    IF
      UPDATE.FOR.NEW.RATIO        ( Yes, update existing entries for new ratio )
    ELSE
      START.NEW.INT.DEFN          ( No, Start a new defn. )
    THEN
    DUP STORE.INT.DIL.RATIO       ( Store new value of dilution factor )
    ENTER.ON.INT.DEFN             ( Put this component into defn. )
;

VARIABLE COMPONENT.NO

: FIND.INTERMEDIATE.TO.USE   ( - Int. No. :: Dil. Rat. - Dil. Rat. )
    F=1 MINIMUM.VOLUME I16@
    TOTAL.SERIAL.VOLUME R32@
    @ 1E3 F* F/ FSQRT FSWAP        ( Calculate maximum range of dilutions for )
    1                             ( for each intermediate solution )
    BEGIN
      FOVER F* F3PICK FSWAP       ( Calculate maximum dilution factor for int. )
      CALC.VOL.OF.STD.IN.SERIAL   ( Calculate volume of std. that would be )
      FSWAP FDROP                 ( required )
      MINIMUM.VOLUME @ <          ( Is it less than minimum allowed )
    WHILE
      1+                          ( While it is increment count and loop )
    REPEAT
    FDROP FDROP                   ( Drop maximum dil. rat. and dil. range )
;

: USE.INT.SOLN.1.OR.2          ( Int. No. - :: Dil. Rat. - )
    CHECK.OLD.RATIO               ( Check if existing dilution factors can be used )
    IF
      USE.OLD.RATIO               ( Yes, use them )
    ELSE
      USE.NEW.RATIO               ( No, try find new ones )
    THEN
       ( Enter intermediate solution used to dilute this component )
       ( This is used to calculate the actual concentration )
       ( in the final solution )
    1+ SOLUTION.CONTENTS 2+ COMPONENT.NO @ 4* +!
    0 SOLUTION.CONTENTS COMPONENT.NO @ 1+ 4* +!
;

: REMOVE.FROM.NEXT.CUMULATED.VOLUME
  ( Int. No. Int. Vol. Next. Soln. - Int. No. Int. Vol. Next. Soln. )
    3 PICK GET.POSN.ON.NEXT          ( Find position on next solution )
    OVER GET.INT.DEFN @ +            ( Get defn. of next solution )
    SWAP 4* I@ ->F @ 1E3 F/          ( Get volume of intermediate on next soln. )
    DUP GET.CUMULATED.VOLUME DUP     ( Get cumulated volume of next soln. )
    R32@ FSWAP F- R32!               ( Remove volume of intermediate )
;

: VOLUME.WITH.EXISTING.RATIO         ( Int. No. - Int. No. Vol. [illegible] )
    DUP GET.NEXT.SOLN @          ( Get number of next solution )
    GET.INT.DIL.RATIO            ( Get dilution factor of next soln. )
    F/                           ( Divide total dilution ratio by dil. rat. of next )
    DUP GET.NEXT.SOLN @
        ( Calc vol. of std and dil. rat. for remaining dilution )
    CALC.NEW.RATIO DROP
;
```

```
: CHECK.EXISTING.RATIO
 ( Vol. Int. no. - Vol. Int. No. :: D.r. N.D.r - D. R. N.D.r )
   FOVER                          ( Copy overall dilution factor )
   VOLUME.WITH.EXISTING.RATIO     ( Calculate volume of std. needed if using )
   FDROP FDROP                    ( existing values of dil. ratios )
   MINIMUM.VOLUME @ <             ( Compare with minimum allowed )
   NOT
;

: GET.OTHER.NEXT      ( next soln. - other next soln. )
   DUP 2 MOD 2* 1- +
      ( 1 or 2, rem /2 1 or 0, 2 or 0, 1 or -1, 2 or 1 )
;

: USE.EXISTING.RATIO  ( Vol. Int. No. - Int. No. :: D.r. N.D.r - D.r O.D.r )
   PLUCK FDROP                          ( Get rid of old values )
   FDUP
   VOLUME.WITH.EXISTING.RATIO           ( Calculate volume and ratio required )
   SWAP FSWAP FDROP                     ( Tidy up stacks )
   DUP GET.NO.IN.INT.SOLN @             ( Get number of entries in defn. )
   UPDATE.FOR.NEW.RATIO                 ( Update previous entries )
   DUP GET.NEXT.SOLN @                  ( Get number of next solution )
   GET.TOTAL.VOLUME                     ( Calculate volume of intermediate to be )
   F* % 1E3 F* F-> SWAP                 ( included in next solution )
   REMOVE.FROM.NEXT.CUMULATED.VOLUME    ( Remove old value )
   CHECK.CUMULATED.VOLUME          ( Check that we won't overflow next solution )
   GET.INT.DEFN 8 +
   3 PICK GET.POSN.ON.NEXT @
   4* I!                                ( Put new value into defn. )
   DUP GET.INT.DIL.RATIO                ( Get existing value of dil. ratio )
;

: UPDATE.FIRST.STEP
  ( Int. No. Vol. Next. Soln No. in.Int. -  Int. No. Vol. Next. Soln )

      ( When changing dilution ratio of both intermediate solution the volumes )
      ( of all previous standards need to be updated to reflect change in both )
      ( dilution ratio. This routine updates volumes to reflect the change in )
      ( the dilution ratio of the second intermediate solution )

  FDUP OVER
  GET.INT.DIL.RATIO              ( Get current value of dil. Rat. for next soln. )
  FSWAP F/ 4 PICK                ( Calculate relative change )
  GET.INT.DEFN 8 + SWAP          ( Get defn. of this intermediate soln. )
  5 PICK GET.CUMULATED.VOLUME
  F=0 R32!                       ( Clear cumulated volume )
  0
  DO
    FDUP DUP DUP I16@            ( Get existing volume )
    F* F->                       ( Calculate new volume )
    CHECK.MINIMUM.VOLUME         ( Check not too small )
    6 PICK
    CHECK.CUMULATED.VOLUME DROP  ( Check it won't overflow )
    SWAP ! 4+                    ( Store volume and goto next entry )
  LOOP
  DROP FDROP                     ( Drop used defn. and update factor )
;
```

```
: KEEP.NEXT.SOLN       ( Vol. Int. No. Next Soln. - Vol. Int. No. !! )
    CALC.NEW.RATIO                      ( Calculate new ratio for next soln. )
    DUP GET.NO.IN.INT.SOLN @
    UPDATE.FOR.NEW.RATIO                ( Update next soln. for the new ratio )
    3 PICK GET.NO.IN.INT.SOLN @
    UPDATE.FIRST.STEP                   ( Update this intermediate for change as well )
    DUP STORE.INT.DIL.RATIO                    ( Store new ratio )
    REMOVE.FROM.NEXT.CUMUL .ED.VOLUME          ( Remove volume from C. V. )
    CHECK.CUMULATED.VOLUME              ( Check adding the new volume won't overflow )
    GET.INT.DEFN @ +
    3 PICK GET.POSN.ON.NEXT @
    4* I!                               ( Put new volume or defn. of next soln. )
    FDROP
    DUP GET.NO.IN.INT.SOLN @            ( Update this intermediate for new dilution )
    UPDATE.FOR.NEW.RATIO                ( factor of this intermediate )
;

: REMOVE.FROM.NEXT    ( Int. No Next Soln. - Int. No. )

      ( If switching to other intermediate for the next step in the dilution )
      ( need to remove the intermediate from the one it is already on )
      ( this is done by entering a zero for the volume )

   0 OVER
   REMOVE.FROM.NEXT.CUMULATED.VOLUME        ( Remove current volume from C. V. )
   GET.INT.DEFN @ +
   4 PICK GET.POSN.ON.NEXT @
   4* I!
;

: SWITCH.NEXT.SOLN    ( Int. No. Next Soln. Curr. Soln. - Int. No. )

      ( WARNING: This code is untested !!! )
   ." Switching "
   REMOVE.FROM.NEXT                    ( Remove from defn. of current next soln. )
   DUP 3 PICK GET.NEXT.SOLN !          ( Store new next )
   DUP GET.NO.IN.INT.SOLN @
   3 PICK GET.POSN.ON.NEXT !           ( Get position on next next )
   DUP GET.NO.IN.INT.SOLN @            ( Check if anything already in new next )
   IF
     CALC.NEW.RATIO                    ( If yes, calculate new ratio for it )
     DUP GET.NO.IN.INT.SOLN @
     UPDATE.FOR.NEW.RATIO              ( Update to new ratio )
     GET.OTHER.NEXT
     3 PICK GET.NO.IN.INT.SOLN @
     UPDATE.FIRST.STEP                 ( Update this intermediate for change )
     GET.OTHER.NEXT                    ( of next soln. )
     CHECK.CUMULATED.VOLUME            ( Check we can add to next )
     GET.INT.DEFN @ +
     3 PICK GET.POSN.ON.NEXT @
     4* I!
   ELSE
     DUP GET.OTHER.NEXT
     UPDATE.FIRST.STEP
     DROP
     OVER GET.INT.DEFN 2-
     SWAP USE.INT.SOLN.1.OR.2
   THEN
;
```

```
: RESTART.TWO [illegible]IL        ( Vol. Int. No. - Int. No. :: Dil. Rat. - )
    OVER
    BACK.CALC.DIL.RAT[illegible]          ( Calculate dilution factor in first step )
    F3PICK FSWAP F/                ( Calculate dilution still to go )
    FIND.INTERMEDIATE.TO.USE       ( Find what the next soln. required )
    DUP
    3 PICK GET.NEXT.SOLN @         ( Compare with the current soln. )
    =
    IF
      KEEP.NEXT.SOLN               ( They are the same )
    ELSE
      OVER GET.NEXT.SOLN @         ( They are different )
      SWITCH.NEXT.SOLN
    THEN
;

: START.TWO.STEP.DIL  ( Vol. Int. No. - :: Dil. Rat. New Dil. Rat. - )
    START.NEW.INT.DEFN             ( Start a new defn. )
    OVER
    BACK.CALC.DIL.RATIO            ( Calculate dilution factor of first step )
    F3PICK FSWAP F/                ( Calculate what remains )
    FIND.INTERMEDIATE.TO.USE       ( Find which intermediate to use )
    DUP
    3 PICK GET.NEXT.SOLN !         ( Store value )
    DUP GET.NO.IN.INT.SOLN @
    3 PICK GET.POSN.ON.NEXT !      ( Store position in the defn. )
    OVER GET.INT.DEFN 2-           ( Enter this solution on defn. as if it were )
    SWAP USE.INT.SOLN.1.OR.2       ( like any other standard )
;

: UPDATE.TWO.STEP.DIL
  ( Vol. Int. No. - Vol. Int. No. :: D.r. N.D.r - D.r N.D.r )
    CHECK.EXISTING.RATIO           ( See if current ratio of next soln. can be used )
    IF
      USE.EXISTING.RATIO           ( Use it only change dil. ratio of first step )
    ELSE
      RESTART.TWO.STEP.DIL         ( No, try find a set that can be )
    THEN
;

: USE.NEW.TWO.STEP.RATIO    ( Int. No. Vol. - Int. No. :: D.r N.D.r - Dil. Rat. )
    DROP FDROP                     ( Get rid off old values )
    CALC.NEW.TWO.STEP.RATIO        ( Calculate new values )
    DUP
    GET.NO.IN.INT.SOLN @           ( Check if anything already on defn. )
    IF
      UPDATE.TWO.STEP.DIL          ( Yes, then make necessary changes to existing )
    ELSE                           ( entries )
      START.TWO.STEP.DIL           ( No, then it is a new defn. )
    THEN
    DUP STORE.INT.DIL.RATIO        ( Store new value of dilution factor )
    ENTER.ON.INT.DEFN              ( Enter defn. )
;
```

```
: UPDATE.TWO.STE..RATIO      ( Int. No. - Int. No. )

        ( Something entered on next soln. may have changed volume of this solution )
        ( and hence the dilution factor of this solution, using old values will )
        ( cause an error since the product of the two dilution factors will be )
        ( erroneous. Hence we recalculate the dilution factor of this solution. )

    DUP GET.NO.IN.INT.SOLN @          ( Check if there are entries in the defn. )
    IF
      DUP GET.NEXT.SOLN @             ( Yes, get next solution )
      GET.INT.DEFN 8 +
      OVER GET.POSN.ON.NEXT @
      4* I@ ->F % 1E3 F/              ( Get volume of intermediate )
      GET.TOTAL.VOLUME F/             ( Get total volume of next solution )
      DUP STORE.INT.DIL.RATIO         ( Store result )
    THEN
;

: USE.INT.SOLN.3.OR.4        ( Int. No. - :: Dil. Rat. - )
    UPDATE.TWO.STEP.RATIO             ( Get current dilution factor )
    CHECK.OLD.RATIO                   ( See if old values can be used )
    IF
      USE.OLD.RATIO                   ( Use old values )
    ELSE
      USE.NEW.TWO.STEP.RATIO          ( Try use new values )
    THEN
    1+ SOLUTION.CONTENTS 2+ COMPONENT.NO @ 4* I!
    0 SOLUTION.CONTENTS COMPONENT.NO @ 1+ 4* I!
;

: USE.INTERMEDIATES    ( - :: Dil. Rat. - )
    FIND.INTERMEDIATE.TO.USE          ( Find which solution to use first )
    DUP
    BEGIN-CASE
      1 CASE-OF
          USE.INT.SOLN.1.OR.2
      ELSE
      2 CASE-OF
          USE.INT.SOLN.1.OR.2
      ELSE
      3 CASE-OF
          USE.INT.SOLN.3.OR.4
      ELSE
      4 CASE-OF
          USE.INT.SOLN.3.OR.4
      ELSE
      TOO.DILUTE.ERROR                ( Force too dilute error )
        ( Concentration desired too dilute to be done in a three step dilution )
        ( and I'm NOT going to even think about how to use more stages )
        ( TOUGH CHUCK CHUM )
    END-CASE
;

: FINISH.INTERMEDIATES       ( - )
        ( Finish definition of intermediate solutions )
    5 1
    DO
      I GET.NO.IN.INT.SOLN @ ?DUP     ( Check to see if any entries are in it )
      IF
        I GET.INT.DEFN !              ( Store number of components )
        I GET.TOTAL.VOLUME
        I GET.CUMULATED.VOLUME
        R32@ F- % 1E3 F*              ( Calculate volume of solvent needed )
        I GET.INT.DEFN 4+ I16!        ( Store it )
        I 1 =
        I 2 =
        OR                          ( Check if these are second stage ints. )
        IF
          ' GET.INT.DIL.RATIO       ( Get dilution factor )
          ' GET.INT.DEFN 2-
          NO.IN.SAMPLE @
          I GET.POSN.ON.NEXT !      ( Store position of intermediate )
          ENTER.ON.SAMPLE.SOLUTION  ( Enter intermediate on final soln. defn. )
        THEN
      THEN
    LOOP
;

: MAKE.INTERMEDIATES  ( - )
    1 4                             ( Make intermediates in reverse order )
    DO
      I GET.NO.IN.INT.SOLN @        ( Check if soln. defined )
      IF
        I GET.INT.DEFN              ( Get Defn. )
        MAKE.SOLUTION
        ( Debugging code )
        ( ." Intermediate soln. #" I . CR )
        ( SOLUTION.PRINT )
        ( KEY DROP )
      THEN
    -1
    +LOOP
;

VARIABLE COUNTER

: FIND.MIN     ( 0 n1 n2 n3 .... - 0 ni nj ... min(n) )
    2 COUNTER !
    BEGIN
      COUNTER @ DUP 1+
      PICK                          ( Get next number on the stack, while not zero )
    WHILE
      ROLL                          ( Get the number )
      OVER OVER <                   ( Make copies of top two numbers and compare )
      IF
        SWAP                        ( If new low swap )
      THEN
      COUNTER 1+!                   ( Increment counter )
    REPEAT                          ( Repeat until zero is reached )
    DROP                            ( Drop zero )
;

: EMPTY.INTERMEDIATES        ( - )

       ( Empty intermediate solutions, empty vials in first row first to avoid )
       ( collisions when emptying vials in back row )

    0                               ( Mark end of list with zero )
    5 1
    DO
      I GET.NO.IN.INT.SOLN @        ( Check if intermediate was used )
      IF
        I GET.INT.DEFN 2+ @         ( If so get position of bottle )
      THEN
    LOOP
    BEGIN
      DUP                           ( Check if reached marker )
    WHILE
      FIND.MIN                      ( Find next bottle to empty )
      DUMP.SOLUTION                   ( Empty it )
    REPEAT
  ;

  : INIT.INTERMEDIATES  ( - )

        ( Initialise variables used in definition of intermediates )

    5 1
    DO
      F=0                             ( Zero volumes )
      I GET.CUMULATED.VOLUME R32!
      I GET.NO.IN.INT.SOLN 0!         ( Zero counter )
      F=1 I STORE.INT.DIL.RATIO       ( No dilution )
    LOOP
  ;

  : PRINT.PPM    ( :: conc/ppm - conc/ppm )
    ( Print concentration in appropriate w/v units )
     FDUP F=1 F-
     F0<
     IF
       FDUP % 1E3 F*
       7 6 F.R ." ppb"                ( if conc < 1ppm print in ppb )
       9 EMIT
     ELSE
       FDUP % 1E3 F-
       F0>
       IF
         FDUP % 1E4 F/
         7 6 F.R ." %"                ( if conc > 1000ppm print in % )
         9 EMIT
       ELSE
         FDUP
         7 6 F.R ." ppm"              ( print in ppm )
         9 EMIT
       THEN
     THEN
  ;

  : PRINT.MOLARITY      ( soln. conts. - soln. conts. :: conc/ppm - )
     DUP 2- @
     DUP
     35 =                             ( Check if matrix )
     IF
       DROP
       MATRIX.MOL.WEIGHT              ( Use molecular weight of matrix )
     ELSE
       1- 4*
       AT.WEIGHTS +                   ( Otherwise use atomic weight of element )
     THEN
     R32@ F/                          ( Calculate milliMolarity )
     FDUP F=1 F- F0<
     IF
       % 1E3 F*
       7 6 F.R 230 EMIT ." M"         ( if conc < 1mM print microMolarity )
       CR
     ELSE
       FDUP % 1E3 F- F0>
       IF
         % 1E3 F/
         7 6 F.R ." M"                ( if conc > 1000mM print M )
         CR
       ELSE
         7 6 F.R ." mM"               ( print mM )
        CR
      THEN
    THEN
;

: PRINT.CONC   ( :: conc/ppm - )
    PRINT.PPM
    PRINT.MOLARITY
;

: PRINT.SUMMARY.2     ( - )
    SOLUTION.CONTENTS          ( Get contents of solution, tells what was in it )
    BEGIN                      ( and what solution the original std. was pipetted )
      DUP @                    ( into. Get element, while not zero )
    WHILE
      DUP @
      DUP
      35 =                     ( Check if matrix )
      IF
        DROP
        MATRIX.NAME            ( Use matrix name )
      ELSE
        [ ARL ]
        ELEMENT.SYMBOLS        ( Otherwise use element symbol )
        [ FORTH ]
        SWAP
        0
        DO
          COUNT +
        LOOP
      THEN
      $. 9 EMIT                ( Print element symbol or matrix name )
      CALC.PPM.CONC.2          ( Calculate concentration in ppm )
      PRINT.CONC               ( print conc w/v and molarity )
      2+                       ( Goto next entry )
    REPEAT
    DROP
;

: PRINT.DESIRED.SOLN  ( - )
    ." Standard asked for " CR
    SOLN.COMP DUP @
    SWAP 2+ SWAP
    0
    DO
      DUP C@
      [ ARL ]
      ELEMENT.SYMBOLS
      [ FORTH ]
      SWAP
      0
      DO
        COUNT +
      LOOP
      $. 9 EMIT                      ( Print element symbol )
      1+ DUP  R32@
      PRINT.PPM
      4+ FDROP CR
    LOOP
    DROP
;
```

```
: PREP.SOLN    ( - )

      ( Prepare the solution defined )

  INIT.PREP                          ( Initialise variables )
  INIT.INTERMEDIATES                 ( Initialise variable used for intermediates )
  SOLN.COMP DUP @ SWAP 2+ SWAP DUP   ( Get number of components )
  0
  DO
    OVER C@ ROT 1+ -ROT              ( Get channel number, element )
    [ ARL ]
    DUP SOLUTION.CONTENTS I 4* I!    ( Store in Soln. conts. )
    3 PICK R32@ FDUP
    ROT 4+ -ROT DUP                  ( Get concentration required )
    DUP
    35 =                             ( Check if this is matrix )
    IF
      MATRIX.SOLN                    ( Get matrix stock solution record )
    ELSE
      FIND.STOCK.SOLN                ( Find record of stock solution )
    THEN
    DUP
    2+ I16@ F/                       ( Get conc. of stock )
    FDUP                             ( Calculate dilution factor )
    CALC.VOLUME                      ( Calculate volume required )
    FSWAP F0=
    IF
      DROP                           ( Zero concentration )
      ENTER.ON.SAMPLE.SOLUTION
      1 SOLUTION.CONTENTS 2+ I 4* I!
      0 SOLUTION.CONTENTS I 1+ 4* I!
    ELSE
      MINIMUM.VOLUME @               ( Compare with minimum allowed )
      < NOT
      IF
        ENTER.ON.SAMPLE.SOLUTION     ( Can be prepared in one step )
        1 SOLUTION.CONTENTS 2+ I 4* I!
        0 SOLUTION.CONTENTS I 1+ 4* I!
      ELSE
        I COMPONENT.NO !             ( Needs more than one step )
        USE.INTERMEDIATES
      THEN
    THEN
    DDROP
  LOOP
  DROP
  0 SWAP ELEMENT.ARRAY + C!
  FINISH.INTERMEDIATES               ( Finish definition of intermediates )
  FINISH.SAMPLE                      ( Finish definition of sample )
  MAKE.INTERMEDIATES                 ( Make intermediate solutions )
  DEFAULT.SAMPLE.SOLUTION            ( Make sample solution )
  MAKE.SOLUTION
  ( ." Sample Solution Composition " CR )   ( Debugging code )
  ( SOLUTION.PRINT )
  EMPTY.INTERMEDIATES                ( Empty intermadiate solutions )
  PRINT.SUMMARY.2                    ( Print actual concentrations )
  [ FORTH ]
;

: TAKE.SAMPLE.TO.ARL  ( - )
  DEFAULT.SAMPLE.SOLUTION 2+ @
  SAMPLE.TO.ARL                      ( Take solution to 1,1 position )
;
```

```
: DO.SOLUTION.MEASURE          ( - )
    [ ROBOT ]
    1 1 ARL.SAMPLES              ( Measure full matrix solution )
    START.MEASURE.SAMPLE         ( Start sampling solution )
    WAIT.ROBOT                   ( Wait for robot )
    ARL.PORT SELECT.DEVICE
    [ ARL ]
    13 CEMIT
    ASCII S
    ASCII I
    SYNC.ARL                     ( Send start signal )
    FORCE.FLUSH                  ( Forced preflush )
    NO.REPL @
    0
    DO
      I GET.DATA                 ( Get emission data )
    LOOP
    GET.MEAN                     ( Get average intensity )
    [ ROBOT ]
    STOP.MEASURE.SAMPLE          ( Stop sampling )
    WAIT.ROBOT                   ( Wait for robot )
    BCKWRD ARL.PROBE             ( Replace sampling probe )
;

: DO.MATRIX.BLANK.MEASURE      ( - )
    [ ROBOT ]
    FRWRD ARL.PROBE              ( Pick up sampling probe )
    1 2 ARL.SAMPLES
    START.MEASURE.SAMPLE         ( Start sampling matrix blank )
    WAIT.ROBOT
    ARL.PORT SELECT.DEVICE
    [ ARL ]
    13 CEMIT
    ASCII S
    ASCII I
    SYNC.ARL                     ( Send start signal )
    FORCE.FLUSH                  ( Forced preflush )
    NO.REPL @
    0
    DO
      I GET.DATA                 ( Get emission intensities )
    LOOP
    GET.MEAN                     ( Get avarage intensity )
    [ ROBOT ]
    STOP.MEASURE.SAMPLE          ( Stop sampling )
    WAIT.ROBOT                   ( Wait for Marvin )
;

: DEFINE.MATRIX.BLANK          ( - )
    1 SOLN.COMP !                ( Only matrix included )
    35 SOLN.COMP 2+ C!           ( The matrix uses channel number 35 )
    MATRIX.CONCENTRATIONS DATA.POINT.NO @ 2* +
    I16@ SOLN.COMP 3 + R32!      ( Fetch concentration and store in defn. )
;
```

```
: DEFINE.MATRIX.SOLUTION      ( - )
    [ ARL ]
    NO.ELEMENTS @ 1+ SOLN.COMP !     ( Matrix + analytes )
    ANALYTES.CONCENTRATION I16@      ( Get analyte concentration )
    SOLN.COMP 2+                     ( Start of solution defn. )
    ELEMENT.ARRAY                    ( Analytes )
    BEGIN
      DUP C@                         ( Get analyte channel number )
      ?DUP                           ( While not zero )
    WHILE
      [ ARL ]
      DUP
      Cd2 = NOT                      ( Check if Cd2 channel being used )
      IF
        3 PICK C!                    ( If not store channel number )
        SWAP 1+ DUP FDUP R32!        ( Store concentration )
        4+ SWAP                      ( Goto next entry )
      ELSE
        DROP                         ( If Cd2 ignore, Cd will entered as Cd1 )
      THEN
      1+
    REPEAT
    DROP FDROP                       ( Clean up stacks )
    DUP 35 SWAP C!                   ( Store matrix channel number )
    1+ MATRIX.CONCENTRATIONS
    DATA.POINT.NO @ 2* + I16@        ( Get matrix concentration )
    R32!                             ( Store matrix concentration )
;

: STORE.MATRIX.CONCS  ( 0 c1 c2 c3 .... cn - )
    MATRIX.CONCENTRATIONS
    NO.OF.DATA.POINTS 0!             ( Zero counter )
    BEGIN
      SWAP
      ?DUP                           ( Check if end of list reached )
    WHILE
      OVER ! 2+                      ( Store concentration )
      NO.OF.DATA.POINTS 1+!          ( Increment counter )
    REPEAT                           ( Repeat until zero reached )
;

: STORE.MATRIX.NAME   ( addr$ - )
    MATRIX.NAME                      ( String variable address )
    OVER C@ 2+                       ( Get character count )
    CMOVE                            ( Move string to MATRIX.NAME )
;
```

```
: TAKE.BLANK.TO.ARL   ( - )
    [ ROBOT ]
    DEFAULT.SAMPLE.SOLUTION 2+ @      ( Get position of bottle )
    CHECK.GRIP
    GRIP.ACTIVE @ SWAP
    0 GRIP.ACTIVE !                   ( Deactivate grip )
    MOVE.TO                           ( Move to solution position )
    PICK.UP.SOLN                      ( Pick it up )
    FRWRD SAMPLE.TO.ARL.PATH          ( Move to measurement area )
    1 2 ARL.SAMPLES MOVE.TO           ( Move to 1,2 position )
    % 5 MOVE.BACK
    % 235 MOVE.DOWN
    % 40 MOVE.DOWN                    ( Put solution down )
    GRIP.OPEN                         ( Let go )
    % 35 MOVE.UP
    % 10 MOVE.BACK                    ( Move away )
    SAMPLE.TO.ARL.PATH.2 MOVE.TO
    GRIP.ACTIVE !                     ( Reactivate grip )
;

: EMPTY.BLANK  ( - )
    [ ROBOT ]
    CHECK.GRIP
    GRIP.ACTIVE @
    0 GRIP.ACTIVE !                   ( Deactivate grip )
    SAMPLE.TO.ARL.PATH.2 MOVE.TO
    1 2 ARL.SAMPLES MOVE.TO           ( Move to blank solution )
    % 5 MOVE.BACK
    % 275 MOVE.DOWN
    GRIP.CLOSE
    GRIP.ACTIVE !
    % 40 MOVE.UP                      ( Pick up blank solution )
    WAIT.ROBOT                        ( Wait for slow coach )
    FRWRD EMPTY.CONTAINER             ( Empty used solution )
    WAIT.ROBOT                        ( Ho Hum )
    [ ARL ]
    DRAIN @ WAIT                      ( Wait for solution to drain )
    [ ROBOT ]
    FRWRD DIRTY.GLASSWARE             ( Dump used glass in bucket )
    [ FORTH ]
;

: MAKE.MATRIX.BLANK   ( - )
    DEFINE.MATRIX.BLANK               ( Define solution )
    PREP.SOLN                         ( Make it )
    TAKE.BLANK.TO.ARL                 ( Take it to measurement area )
;

: MAKE.MATRIX.SOLUTION       ( - )
    DEFINE.MATRIX.SOLUTION            ( Define full solution )
    PREP.SOLN                         ( Make it )
    TAKE.SAMPLE.TO.ARL                ( Take it to be measured )
;

: MAKE.ANALYTE.SOLUTION      ( - )
    DEFINE.MATRIX.SOLUTION            ( Define solution with matrix )
    [ ARL ]
    NO.ELEMENTS @ SOLN.COMP !         ( Force PREP.SOLN to ignore the matrix by )
    PREP.SOLN                         ( reducing the component count to just analytes )
;
```

```
: MATRIX.STUDY          ( - )
    GET.INFO
    ." Matrix Study Program " CR      ( Print header )
    DATE. 9 EMIT TIME. CR             ( Date and time stamp )
    INIT.ARL                          ( Initialise ARL )
    MAKE.ANALYTE.SOLUTION             ( Make solution with analytes only )
    START.BLANK.MEASURE               ( Start measurement of distilled water blank )
    TAKE.SAMPLE.TO.ARL                ( Bring analyte solution to be measured )
    GET.BLANK.DATA                    ( Get water blank data )
    [ ROBOT ]
    FRWRD ARL.PROBE                   ( Pick up sampling probe )
    ." Analytes " CR
    DO.SOLUTION.MEASURE               ( Start measuring analyte solution )
    EMPTY.SAMPLE                      ( Empty it )
    CRT
    [ ARL ]
    9 EMIT LIST.ELEMENTS CR           ( List element symbols )
    >FILE
    NO.OF.DATA.POINTS @
    0
    DO                                ( Start matrix study )
      I DATA.POINT.NO !
      MAKE.MATRIX.BLANK
      MAKE.MATRIX.SOLUTION
      RETURN.CONCENTRATION            ( Send matrix concentration to ARL )
      ." Matrix Blank" CR
      [ ARL ]
      9 EMIT LIST.ELEMENTS CR
      DO.MATRIX.BLANK.MEASURE         ( Measure matrix blank )
      ." Matrix Solution " I 1+ . CR
      DO.SOLUTION.MEASURE             ( Measure full solution )
      EMPTY.SAMPLE                    ( Empty full solution )
      EMPTY.BLANK                     ( Empty matrix blank )
      ." Relative Sensitivity " CR
      GET.RESPONSE                    ( Get relative sensitivities )
    LOOP                              ( Loop for each data point )
    GET.RESPONSE.MATRIX               ( Get full data table )
    TIME. CR                          ( Time stamp )
    CLOSE-OUTPUT                      ( Close file )
;
```