EPICARDIAL POTENTIAL DISTRIBUTIONS CALCULATED FROM BODY SURFACE MEASUREMENTS USING MULTIPLE TORSO MODELS

Joseph
Stephen J. Walker
(B.Sc. Hons Sydney University)

A thesis submitted for the degree of

Doctor of Philosophy

Department of Medicine,
University of Tasmania,

July 1985

CONTENTS

ABSTRACT

This thesis describes the development of a system for the calculation of epicardial potentials from measured body surface electrocardiographic data, using highly detailed computerised models of the torso. To provide data so that the system can be evaluated in a clinical situation, a body surface mapping system which can be used easily on sick patients has been developed. The equipment is located in a trolley which can be wheeled to a patient's bedside. A 'jacket' of electrodes is used to allow simple and rapid electrode placement.

In order to calculate the forward and inverse transfer matrices required to calculate epicardial potentials, an automated system for modelling three dimensional volume conductors has been developed. This system has been validated by comparison of the modelled potential distribution due to a dipole located in a sphere with the known analytical solution. Initial calculations performed on a highly detailed inhomogeneous model of the human torso show that, by using a regularisation method, physiologically plausible epicardial potentials can be calculated from measured body surface potentials.

Four torso models with identical torso shape but varying internal conductivities, and seven other models with varying torso shape were used to explore the effects of varying conductivity and geometry on the forward and inverse calculations. It was found that omission of the lungs or bony structures caused variations in both the forward and inverse calculations, with the lungs having a major effect. Variations in external torso geometry also caused sizeable changes in both the forward and inverse calculations.

STATEMENT REGARDING ORIGINALITY

The investigations described within this thesis constitute my own work. This thesis contains no material which has been accepted for the award of any other degree in any University or College. To the best of my knowledge and belief this thesis contains no copy or paraphrase of material previously published or written by another person except where due reference is made in the text.

Signed: S J. Walker

Date: July 1985

ACKNOWLEDGEMENTS

I would like to express sincere appreciation to my supervisor, Dr. David Kilpatrick, for his constant helpful suggestions, support and encouragement throughout the course of this work.

I am most grateful to the following people whose help and advice were sought at various stages of the work :-

Dr. M. Loughhead, Sr. M. Murray and the staff of the Intensive Care Unit at the Royal Hobart Hospital for the generous provision of facilities and time for the collection of body surface map data,

Mr. D. Lees and the staff of the photographic unit for help with circuit board artwork reductions and general illustrations,

The staff of the Radiology Department at the Royal Hobart Hospital for the provision of copies of CT scan images.

Dr. P. Hamilton of the Physics Department, Mr R. Bilson of the Computer Centre, and Mr N. Williams, a student in the Electrical Engineering Department, for helpful suggestions regarding the computer hardware, software, and modifications to the UNIX operating system,

Dr. P. Lavercombe for advice and encouragement,

Miss J. Heath for assistance with the digitising of the CT scan images,

Mrs. M. Kernot for Library facilities.

Thanks are extended to Professor G. W. Boyd for providing the opportunity to undertake this work in the Department of Medicine, and also to my many colleagues both at the Clinical School and

throughout the University who provided a congenial and friendly atmosphere in which to work.

The work described in this thesis was carried out partly under a National Health and Medical Research Council project grant and partly under a Commonwealth of Australia Postgraduate Research Award. Financial assistance for equipment and travel was also obtained from the National Heart Foundation and from the Royal Hobart Hospital Research Trust Fund.

TO MY PARENTS

CHAPTER ONE

INTRODUCTION

Introduction

Electrocardiology is the study of the electrical signals produced by the heart. Although the electrocardiogram is an important clinical tool in the assessment of heart disease, conventional electrocardiographic techniques provide very little quantitative information about the heart. It is hoped that a much more accurate picture of the function of the heart can be obtained by using the extra data which are contained in the entire body surface electrocardiogram.

In myocardial infarction, for example, it is generally not possible, using conventional electrocardiographic techniques, to accurately measure the amount of cardiac muscle involved. It is hoped that calculation of cardiac source distributions from measured body surface potentials will allow the extent of the infarcted region to be determined. This ability would then allow the effects of various interventions designed to save myocardium to be assessed much more accurately than is possible at present. The great attraction of using electrocardiographic techniques, when compared with most other means of examining the heart's function, is that it is non-invasive, needs relatively inexpensive equipment and provides immediate results at the bedside.

To enable such ideas to be tested clinically, it is necessary

to provide a means of measuring body surface potential distributions and a method which can be used to calculate the corresponding cardiac sources. This thesis describes the development of a system for the calculation of epicardial potentials from measured body surface electrocardiographic data, using highly detailed computerised models of the torso.

Background

The electrocardiogram is the manifestation on the body surface of the electrical activity of the heart. The ionic currents associated with contraction and relaxation of the cardiac muscle cells manifest themselves as differences of electrical potential between points on the surface of the body. These potential differences can be recorded and are the data on which electrocardiology is based. The signals to be measured are small, ranging from tens of microvolts or less to at most several millivolts. The dominant frequency present is the heart rate, which is of the order of 1Hz. There are significant higher frequency components present, notably in the QRS complex. The QRS complex arises from the ionic currents associated with a sudden change in the cellular trans-membrane potential which propagates rapidly through the heart, initiating muscular contraction (Noble 1979).

For many years, 12 standard measurements derived from six chest and three limb electrode positions have been used to specify the electrocardiogram. This '12 lead' electrocardiogram is a valuable clinical tool. An experienced observer can relate deviations from

the normal measured patterns to changes in electrical activity and function of the heart. For several decades (Taccardi et al. 1976), attempts have been made to obtain more information from the electrocardiogram by recording electrocardiographic potential distributions from large numbers of leads distributed over the surface of the body (located mainly on the torso). This procedure is generally known as body surface mapping. An excellent review of the procedures involved in constructing and interpreting body surface maps has recently been published (Barr and Spach 1983).

There are two distinct problem areas associated with body surface mapping. The first is in the collection of data. Low level signals with a bandwidth from one tenth of a hertz to a few hundred hertz must be recorded from many locations on the torso. This requires development of large numbers of high gain amplifiers and associated equipment to store the large amount of data produced. The skin impedance is quite high (ranging from hundreds of kilohms to a few megohms), so that either the electrodes must be buffered at the skin (active electrodes), or conductive jelly must be used to lower the skin impedance. The placement of many electrodes on the torsos of patients can also be a difficult task, especially when the patient is critically ill.

Early attempts at body surface mapping (for example Taccardi 1963) show up these difficulties well. Potentials were measured from one lead, or a small group of leads, at a time. The entire body surface data, measured from several hundred points distributed on the torso, were built up over many cardiac cycles. The pen

recordings or oscilloscope tracings obtained were magnified and measured by hand at various instants throughout the heart cycle. After possibly several weeks of effort, body surface potential distributions were obtained.

The development of modern electronics and high speed computers has allowed these procedures to become substantially more automated. A number of groups (for example Taccardi et al. 1976, Kilpatrick et al. 1979, Spach et al. 1979, Heringa et al. 1981 and Yajima et al. 1983) have developed systems which, although differing in detail, exploit digital techniques. Typically, many high gain amplifiers are connected via a multiplexer to one or several analog to digital converters. The digital data are then usually stored on floppy disk or magnetic tape.

Various systems of lead placement have been tried, the most common being strips of electrodes running vertically on the torso and attached to the skin with double sided adhesive tape. Other methods include belts of electrodes running around the torso (Yajima et al. 1983), electrodes mounted on vests (Liebman et al. 1981 and Walker et al. 1983) and electrodes held by suction (Reek et al. 1984). Some systems use active electrodes, which eliminate the need for conductive jelly when the electrodes are applied. Others use jelly, which tends to increase the time taken to apply the electrodes, but allows a simpler electrode construction.

The number of leads used in systems ranges from 24 to over 200. A typical system of, say, 64 leads, sampled at 1000Hz on each lead, will generate 64000 measurements per second. This high data rate

can pose major storage problems. Most present mapping systems store data on floppy disc or tape, so that it is generally not practical to store or compare many sets of data in a clinical situation. Most systems also use an 8 bit microprocessor, which limits the amount of data processing which can be performed at the bedside. These are not serious problems if the system is to be used solely for research, as processing and analysis of many sets of data can be done off-line on a large computer. In fact, a small, portable system has obvious advantages, as it can be taken to the subjects to be mapped, rather than them having to go to it. In a clinical situation it is desirable to be able to obtain results quickly, and to have immediate access to data collected in the past. Because of the large quantities of data involved, a computer with large amounts of mass storage and reasonable processing speed is desirable.

The second set of problems associated with body surface mapping lies in analysing the large amounts of data that mapping systems generate. A good first step is to make sure that large amounts of redundant data are not collected. Both Barr et al. (1971) and Lux et al. (1978) have shown that body surface potential distributions are adequately represented by measurements taken from 24 to 32 correctly positioned leads. In order to reconstruct the entire potential distribution from this number of measurements, it is first necessary to have calculated (or obtained from some other source) a transform matrix which relates the lead measurements to the potentials at a much larger number of points. Most systems use somewhat more than the minimum number of leads to allow some latitude for non-operational or noisy leads, and to allow reconstruction of the

entire distribution using simpler interpolation techniques. Several schemes have been developed to further reduce the amount of data which must be stored, using Karhunen-Loeve expansions (Lux et al. 1981, Lux et al. 1981a, Uijen et al. 1984) or discrete cosine expansions (Ahmed et al. 1975).

Two fairly distinct approaches are followed when analysing body surface map data. The first is to treat the data in much the same way as a 12 lead electrocardiogram is treated. One builds up sets of data on large numbers of patients with various conditions and attempts to classify the data patterns which correspond to the various clinical conditions. A study by Yamada et al. (1978) on the use of body surface maps in the diagnosis of various forms of myocardial infarction is an example of this type of approach.

A more complex but potentially more powerful approach is to obtain parameters from the data which are directly related to properties of the heart. Ideally, given the potential distribution on the body surface throughout a cardiac cycle, it is required to compute the distribution and strength of electrical sources within the heart throughout that cycle. This is the so called "inverse problem" of electrocardiology.

Unfortunately there is no solution to this problem in its most general form. This is because there can exist an infinite number of different arrangements of electrical sources within a volume which can give rise to the same potential distribution on the surface of that volume. However, the problem can be solved by using extra information to restrict the possible range of solutions.

Anatomically, the sources must be located in the heart. In the case of a normal heart, the order of depolarisation (a sudden decrease in cell trans-membrane potential) is known fairly well from various studies, such as those performed on isolated human hearts by Durrer et al. (1970) or on intact dogs by Spach and Barr (1975). Alternatively, the solution can be restricted to the region lying outside the heart (this involves the calculation of epicardial potential distributions or activation times) where the solution is, in theory, unique (Yamashita 1982).

The first step in solving the inverse problem is always to construct some model of the cardiac electrical sources and express the body surface potential distribution in terms of the source strengths. This is the electrocardiographic forward problem. Solving this problem results in the dependency of the body surface potential distribution on the cardiac sources, which can be expressed in a matrix equation:

$$Th = b \tag{1.1}$$

where h is a vector of cardiac source strengths and b is a vector of body surface potentials. T is called the forward transfer matrix, and reflects the physical properties of the torso. Solving the inverse problem then becomes the problem of solving the system of equations represented by equation 1.1, so that the vector h is expressed in terms of T and b. This system is usually ill-conditioned, meaning that small variations in b (such as measurement noise) can lead to very large variations in the solution for h.

This ill-conditioned behaviour tends to get worse as the size of the system is increased, so that for a given noise level, there is usually a limit on the number of parameters related to the heart which can be determined.

One direct method for solving the forward problem has been to build an actual physical model of the torso, simulating the differing resistivities of the various internal organs with various suitable materials. Real electrical sources may be placed in desired positions within the model and the corresponding potential distributions measured directly. Probably the most sophisticated such 'tank' model is a twice life size model built by Rush (1971) using grids of interlocking plastic rods of various sizes to control the resistivity throughout the model. Such models provide a direct, easily measurable solution to the forward problem. However they suffer from being tedious to construct accurately and from being difficult to adapt if, for example, it is desired to measure the effect of varying the geometry of the model.

Most attempts at the forward problem have been mathematical in nature. A number of assumptions are made. Firstly, only a simplified model of the torso is used. It is assumed (though this is rarely stated explicitly) that the problem is essentially unchanged if the head and limbs are neglected. The torso itself is assumed to be made up of a number of homogeneous regions. Completely homogeneous models (no internal structures) are often used, and simpler geometries (spheres, cylinders and unbounded regions) have also been used. Secondly, it is assumed that the electrical sources are

confined to the heart - this is reasonable as contributions from skeletal muscle and other sources are small by comparison with the electrocardiogram and are regarded as part of the unavoidable noise in such measurements. Lastly, the fields in the body are assumed to be quasi-static - that is , the potential distribution throughout the torso at a particular instant of time is assumed to depend only on the source strengths and distribution at that same instant of time. This last assumption has been shown to be reasonable by a number of workers, including Plonsey (1976) and Barnard et al.(1967) Both these groups also showed that electric fields induced by the time varying cardiac magnetic fields (the magnetocardiogram) can be ignored.

The equations determining the potential at a point in any of the homogeneous regions of the torso can be developed in a manner analogous to that used for the equations for static charge distributions (for example see Lorrain and Corson 1970). We have

$$\nabla^2\Phi = \frac{\rho}{\sigma} \tag{1.2}$$

where Φ is the potential, ρ is the source density, and σ the conductivity. This is Poisson's equation. Note that ρ is not the current density, but the divergence of the current density (Plonsey 1976). The current density is non-zero throughout the body, whereas the sources are at those locations where the divergence of the current density is non-zero. In any one region, σ is assumed constant, and if we assume the region is isotropic, it is a scalar constant. Because the torso is assumed to be surrounded by a non-

conducting medium (air), an important boundary condition is that the normal current density is zero on the surface of the torso. At internal interfaces between regions of different conductivity, the normal current density must be continuous. In regions containing no sources, ρ is zero, giving

$$\nabla^2\Phi = 0 \tag{1.3}$$

This is Laplace's equation.

Most previous work has proceeded from this point by deriving an integral equation for the potential (Barnard et al. 1967, Barr et al. 1966) or for the charge distribution on the torso surface Gelernter and Swihart 1964). In an infinite homogeneous medium, we have (Lorrain and Corson 1970)

$$\Phi = \frac{1}{4\pi\sigma} \int \frac{\rho}{r} dv \tag{1.4}$$

where r is the distance between the calculation point and the volume element dv. For a bounded homogeneous volume with surface S, we have

$$\Phi = \frac{1}{4\pi\sigma} \int \frac{\rho}{r} dv + \frac{1}{4\pi} \int \Phi \frac{\underset{\sim}{r}}{r^3} \cdot dS \tag{1.5}$$

by the application of Green's theorem (Barr et al. 1966). The formula can be extended to allow multiple regions of different conductivity (Geselowitz 1967). The first integral in 1.5 is the

potential of a source in an infinite medium, which can be calculated analytically for simple sources such as dipoles. The second integral can be approximated numerically be dividing the surface S into a number of area elements. The integral then becomes a sum of the potential on each area element times the solid angle subtended by that element at the calculation point. Triangular elements are normally used because analytical formulae exist for the solid angle subtended by a plane triangle at an arbitrary point (VanOosterom and Strackee 1983, Barnard et al. 1967a). A systematic method for triangulating a human torso has been published by VanOosterom (1978).

By taking the calculation point near each surface element in turn, a system of simultaneous linear equations for the potential on each area element can be obtained. This system is usually too large to be solved directly, so that iterative techniques are used.

More recently, the forward problem has been attacked using finite difference or finite element methods. In these methods, the partial differential equations for the potential are approximated by linear equations on a grid of points throughout the torso. Except at source points, Laplace's equation (equation 1.3) applies. In rectangular coordinates, this equation becomes:

$$\frac{\delta^2\Phi}{\delta x^2} + \frac{\delta^2\Phi}{\delta y^2} + \frac{\delta^2\Phi}{\delta z^2} = 0 \qquad (1.6)$$

In the finite difference method, the first term in 1.6 above is approximated at a point (x_1, y_1, z_1) as follows (Smith 1978)

$$\frac{\delta^2\Phi}{\delta x^2} = \frac{\Phi(x_1+h,y_1,z_1) - 2\Phi(x_1,y_1,z_1) + \Phi(x_1-h,y_1,z_1)}{h^2} \qquad (1.7)$$

Here, h is the grid spacing in the x direction. Similar approximations are made in the y and z directions. These approximations, when substituted into 1.6, give a linear equation for the potential at a point in terms of the potentials at the surrounding points. Repeating this procedure throughout the grid produces a system of linear equations for the potentials at the grid points. This system will in general be much larger than the system obtained in the integral equation approach because the torso is approximated by a three dimensional grid rather than by two dimensional surfaces. However, in the integral equation approach, the coefficient matrix of the system is full of non-zero entries, as every area element interacts with every other element. In the finite difference approach the coefficient matrix is sparse, with the potential at each grid point expressed only in terms of the potential at neighbouring grid points. This means that the system of equations is much more amenable to solution by iterative techniques. Also, such a model allows inhomogeneities to be included in detail right down to the grid size with no increase in the size of the system. Including inhomogeneous regions in the integral approach results in increasing the size of the system to be solved, as more area elements are needed to approximate the extra surfaces.

In both the integral equation approach and the finite difference approach, the cardiac electrical sources must be

specified. Single dipoles (as determined by vectorcardiography), and multipole expansions (Geselowitz 1976, Guardo et al. 1976) have been used in the past. These approaches suffer from the fact that no physiological or anatomical information is incorporated in the specification of the source, so that inverse calculations supply no direct physiological or anatomical information. In these cases, the information supplied by an inverse calculation must be treated in a similar manner to the 12 lead electrocardiogram.

The depolarisation wavefront which propagates through the heart is usually considered to be a uniform dipole layer, although work by Roberts et al. (1979) showed that this may not be a good approximation. A number of studies have been performed where multiple dipoles distributed throughout the heart are used to approximate the wavefront.

A model which uses 23 current dipole sources in a torso shaped homogeneous volume conductor was developed by Miller and Geselowitz (1978). Simulation of the electrocardiogram (the forward problem) using this model achieved good agreement with observed human data both for a normal heart and also for simulated ischaemia and infarction (Miller and Geselowitz 1978a). Inverse calculations have been performed with this model (Geselowitz and Thorsson 1983), but the results did not appear to be physiologically meaningful, even when constraints on dipole orientation were applied. This behaviour was attributed to the large degree of cancellation of the fields produced by the dipoles.

A 12 dipole model incorporating a non-homogeneous torso model

(lungs and intracardiac blood are included) has been successfully used to study the normal heart and left and right ventricular hypertrophy (Holt et al. 1969, Holt et al. 1969a, Holt et al. 1969b). Diagnosis of left ventricular hypertrophy has been successfully demonstrated using this model. Attempts have also been made to use this model in the diagnosis of myocardial infarction, with limited success (Barnard et al. 1976). In this model, the orientation of the dipoles was fixed, and the dipole strengths were constrained to be non-negative. These constraints tended to force the model to follow a normal activation sequence, which may not be valid after infarction.

Other multiple dipole solutions of the forward problem have been described by Horacek (1974) using one dipole at a time in a torso containing lungs and the intraventricular blood masses, and by Selvester et al. (1968) using 20 dipoles in a torso both with and without lungs.

In a somewhat different approach, Cuppen (1983) modelled the source by replacing the depolarisation wavefront, (assumed to be a uniform dipole layer), by that part of the heart's surface which, with the activation wavefront, forms a closed layer. This technique depends on the fact that zero field is produced outside a closed uniform double layer. Inverse calculations, based on surface potentials integrated over the entire activation time, then yield the activation sequence on the surface enclosing the heart. Studies of this technique, using both homogeneous torso models and models containing lungs and intraventricular blood, show that calculation

of the activation sequence on the surface of the heart is feasible (Cuppen and VanOosterom 1984). This technique depends on the assumption that the heart is bounded by a single continuous surface. It is thus not clear that it can be applied to the case of myocardial infarction or other situations where the heart may contain one or more electrically inactive areas.

In the past ten years, a number of models have been used where the cardiac sources are specified by the potential distribution on a surface surrounding the heart. The inverse problem then becomes the problem of determining the 'epicardial' potential distribution corresponding to a measured torso surface potential distribution. An advantage of this approach is that there are no sources in the volume between the epicardium and the torso surface, so that, at least in theory, this version of the inverse problem will have a unique solution (Yamashita 1982). Of course, the accuracy to which the epicardial potential distribution can be obtained is still limited by the ill-conditioned behaviour of the problem, which becomes worse as noise increases. Studies by Martin and Pilkington (1972) and Okamoto et al. (1983) both showed that in the presence of realistic noise levels, an unconstrained solution would reliably determine less than twenty independent parameters. This has been borne out in studies by Lo (1977) and Yamashita (1981). Lo modelled the torso with a computer simulation of a three dimensional resistive network (he showed this to be equivalent to implementing a finite difference model). Inverse calculations of the potentials on 26 epicardial regions showed large, oscillatory values. Yamashita modelled the torso, including heart and lungs, using the more

general finite element method (see Zienkiewicz and Morgan 1983). Unconstrained inverse calculations using this model also showed large oscillatory values.

Better results can be obtained by placing constraints on the epicardial potentials. Constraints usually take the form of limiting the magnitude of the potentials or limiting some spatial derivative of the potentials, in an attempt to keep the solutions 'smooth' in some sense. These so called regularisation methods are further discussed in chapter four. A study by Martin et al. (1975) used statistical constraints to determine the potential distribution on a sphere surrounding the heart in a homogeneous dog torso model. Both torso surface and 'epicardial' potential distributions were calculated using a multiple dipole model with known activation times. Inverse potentials were then calculated from the torso surface distributions, and were compared with the initial values. Sum-squared errors of the order of twenty-five percent were achieved. A similar study was performed by Barr and Spach (1978), except that in this study, both torso surface and epicardial potentials were directly measured in a live, intact dog. A homogeneous torso model was constructed using geometry data obtained after sacrificing the dog, and inverse epicardial potentials were calculated from the measured torso surface data using methods described in Barr et al.(1977). Although the root mean square errors between the calculated and measured epicardial potentials were substantial, the calculated epicardial potential distributions correctly reproduced the major features of depolarisation and repolarisation seen in the measured data. Similar results

(substantial numerical error but major feature agreement) were found in a study by Franzone (1980), using an isolated dog heart suspended in a tank.

Aims

The overall aim of this work was to develop solutions of the inverse problem which were suitable for clinical evaluation and which were based on models of the human torso substantially more detailed and realistic than those which had been used previously. Achieving this aim required a number of steps.

Firstly, it was necessary to develop a body surface mapping system which could be easily used in a clinical situation, on very sick patients, and operated by relatively untrained staff. The data collected must be available for immediate viewing at the bedside. Data collected with this system were to be used to calculate epicardial potential distributions.

Secondly, the solution of the inverse problem required a solution of the forward problem. It was thus desired to set up computerised models of the human torso which were substantially more detailed than those used by previous workers. The procedures used to develop these models were to be as automated as possible, so that many models with varying geometries could be set up with a minimum of effort.

Thirdly, because patients are not all the same size and shape, it was considered necessary to explore the effects of torso geometry

and conductivity on the solutions of the forward and inverse problem.

Organisation of thesis

The remainder of this thesis is divided into five chapters and several appendices. Chapter two describes the data collection system used to obtain body surface potential distribution data from patients. Chapter three describes an automated system for construction of highly detailed three dimensional resistive network models of the human torso. A torso model is constructed using this system, and the forward transfer matrix is calculated. Chapter four describes the methods used in solving the inverse problem, based on the forward transfer matrix developed using the model described in chapter three. Chapter five examines the effects of varying torso geometry and conductivity on the forward and inverse solutions developed in chapters 3 and 4. Chapter six is a concluding chapter, followed by the appendices and references.

CHAPTER TWO

SAMPLING EQUIPMENT

Introduction

This chapter describes a system for the collection of electrocardiographic body surface potential data. An earlier version of this system has been described previously (Walker et al. 1983). The system is designed to be easily used by untrained staff on sick patients at the bedside. Sampling and digitising of the data takes place in the intensive care unit of the Royal Hobart Hospital. The equipment is located on a trolley which can be wheeled to each bedside. The computer which controls the system and is used to store and process the data is located in the University of Tasmania Clinical School, about 200m distant. Figure 2.1 shows a general layout of the system. Figure 2.2 shows a photograph of the system in use. The following sections describe the major components of the system.

Electrode Jacket

It is considered essential to use a system of electrode placement which enables electrodes to be applied quickly and easily to all patients. This has been achieved by designing a flexible 'jacket' of electrodes which is wrapped around the patient's torso. The jacket is constructed of two layers of closed cell foam. The surface of the inner layer shows a regular grid of slightly raised

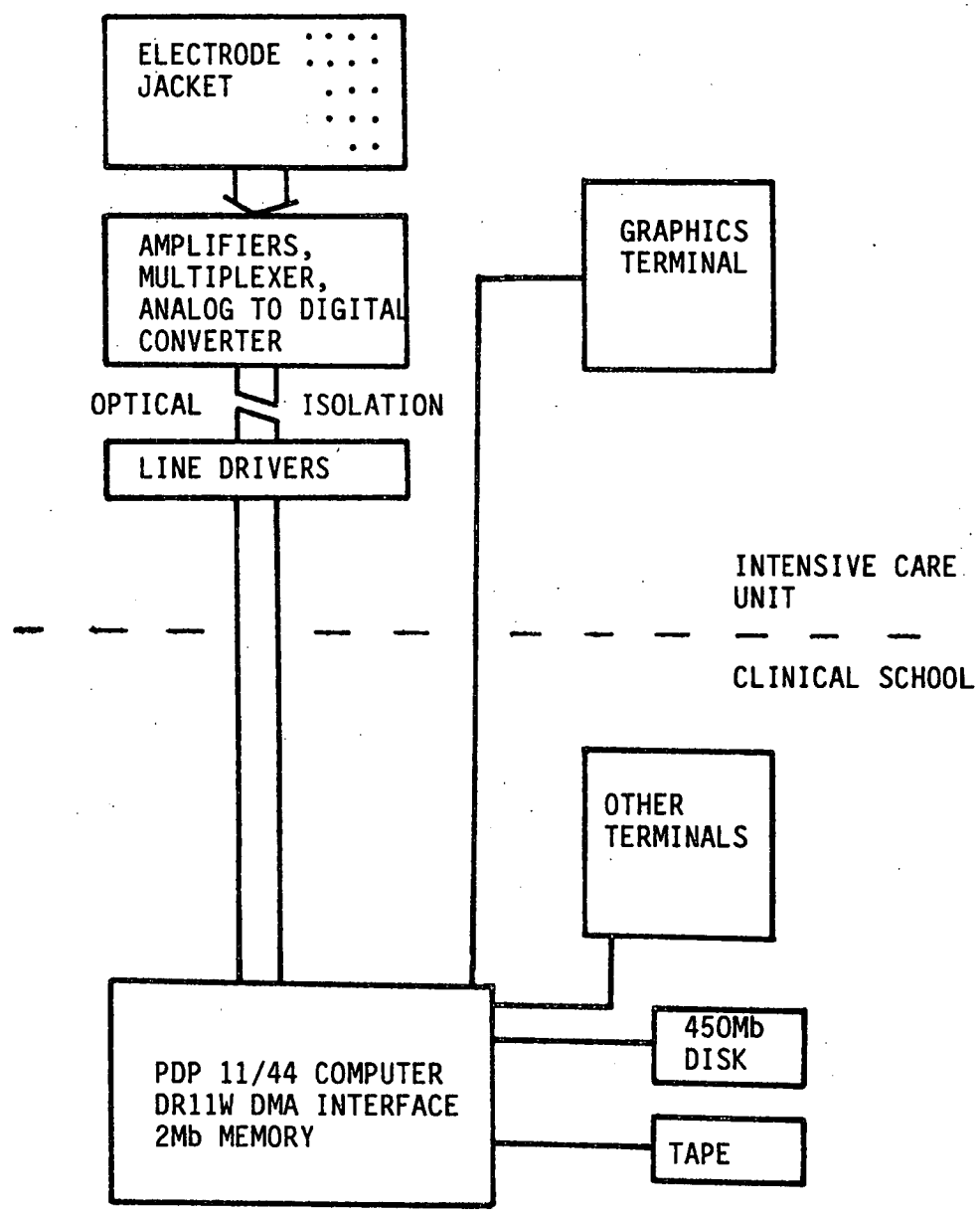


Figure 2.1 Block diagram of the body surface mapping system.

Figure 2.2 Photo of the mapping system in use. The electrode jacket is in place on the subject and raw data is being displayed on the graphics terminal (note that the terminal is a VT125, despite the VT100 badge shown).

electrodes each approximately 1 cm in diameter. There are 5 rows, each of 10 electrodes. The rows are spaced 7.5 cm apart in the vertical direction. Within a row, the electrodes are spaced horizontally 7.5 cm apart on the region which contacts the front and left side of the torso, and 15 cm apart elsewhere.

The jacket is wrapped around the patient with the first column of electrodes placed 7.5cm to the right of the sternum and the second column of electrodes placed over the sternum. The jacket is wrapped under the left arm, around the back, and under the right arm to return to the front of the chest. After sitting (or being supported) for jacket placement, the subject may lie down for the remainder of the procedure. The closed cell foam construction makes the jacket quite comfortable. Straps fastened with "velcro" hold the jacket in place. This method of fastening allows rapid removal of the jacket if necessary. Small patients will have less electrodes contacting the torso than large patients due to overlap of last columns of electrodes with the jacket. The position of the last column of electrodes which contacts the patient is recorded once the jacket has been fitted to the patient. This provides a convenient measure of the chest circumference of each patient, which is taken into account by all further processing software.

The slightly convex heads of nickel plated brass drawing pins (thumb tacks) are used to contact the skin. These have been found to produce low skin contact potentials (Kilpatrick et al. 1979) and are much less expensive than silver or silver/silver chloride electrodes which are often used. Each electrode is active, using an

Analog Devices AD544 wired as a buffer. The input impedance is of the order of 10^{11} ohms. The buffers are sandwiched between the foam layers of the jacket, immediately behind the electrodes contacting the skin. Wiring for the signals and power for the buffers also runs between the foam layers. The buffers save a substantial amount of time when placing the jacket by eliminating the need for electrode jelly.

Two active electrodes are attached by flexible leads to the jacket and are placed using standard adhesive electrode pads. One is used to obtain the signal from the back of the neck. The other is the reference electrode, placed near the right anterior superior iliac spine. The potentials on all other electrodes are measured with reference to this electrode. This reference is used in preference to the Wilson central terminal to avoid the need to place limb leads. (In fact, later processing allows the signals to be referenced to any point, including an artificial central terminal obtained by averaging all torso leads).

Electronics

The jacket is attached to a wheeled trolley by a cable which carries power to the jacket and signals from it. Within the trolley are amplifiers, multiplexers, an analog to digital converter, and isolation and line driving circuitry. A computer graphics terminal (Digital Equipment Corporation VT125) on the trolley is used to give commands to the controlling computer and to display data. A printer may be attached to this terminal to allow permanent copies

of raw data and body surface maps to be produced in the intensive care unit.

The signal from each electrode is passed through a differential amplifier with a gain of 1000. DC offsets and very low frequencies are removed with a single order high pass filter with 3dB point at 0.05Hz. The high frequency range is limited by a third order low pass filter with 3dB point at approximately 500 Hz. Although this implies that some noise at frequencies approaching the sampling rate (usually 1 kHz) will be passed by the filter, in practise it is found that the amount of noise with frequencies above 500 Hz present in the signals is very small. Electronic noise associated with the amplifiers (including the electrode buffers) is approximately 10μV peak to peak (with respect to amplifier input). Although at present only 50 torso leads plus the neck lead are used, 60 amplifiers are available, allowing extra leads to be added if required.

The amplifier outputs are connected through a 64 channel multiplexer (four Analog Devices AD7506 followed by half of an Analog Devices AD7501) to a high speed analog to digital converter (Burr Brown ADC60-10). A sample and hold on the output of each amplifier allows the signal at all electrodes to be sampled simultaneously. The analog to digital converter has a resolution of 10 bits and is set up to digitise a range of +5 to -5 volts, giving a resolution of 10 μV with respect to amplifier input. Multiplexer addressing and sample timing signals are generated from a quartz crystal oscillator, allowing precise sample timing. Software selected sampling rates of 1 kHz, 500 Hz, 250 Hz and 125 Hz are

available. Signals from the computer turn battery power on or off, select sampling rate and start or stop sampling. Control signals and data are passed in parallel form between the trolley and the computer via a multiple twisted-pair cable.

To comply with patient electrical isolation standards, the jacket, amplifiers, multiplexer and analog to digital converter are battery powered. Digital signals output from the analog to digital converter and all control signals from the computer are optically isolated. Low voltage (12 V) circuitry is used in the jacket and the input buffer circuits are individually potted in epoxy resin. With a jacket electrode connected to the mains supply (240 V rms, 50 Hz), leakage current to mains earth is less than 30 μA.

Two sets of sealed, recombination electrolyte, batteries are used, one being charged and one providing power at a given time. Battery voltages are monitored by the controlling software using spare multiplexer channels. This allows the operator to be warned before the batteries become discharged. The software also prevents power from being left on for an extended period if there is no operator activity. These features solve the problem of inconvenient discharge often associated with battery powered equipment.

Circuit diagrams of the electronic systems are contained in appendix A.

Computer System and Software

The system is controlled by a Digital Equipment Corporation PDP

11/44 computer, with 2 megabytes (Mb) of memory. Permanent storage consists of 1 RA81 disk drive (456Mb), 1 RL02 disk drive (10Mb) and a Pertec 9 track tape system. This computer system is also used to develop the models and perform the calculations described in chapters three, four and five. The speed of floating point calculations is much increased by use of an FP11 hardware floating point processor. The computer runs the UNIX* version 7 operating system. Software to run the mapping system has also been developed for the RT11** single user operating system.

Being a time-sharing system, UNIX V7 is not well suited to the collection of real time data at high rates. Also, the data addressing range for a single program running under UNIX on a PDP 11/44 is restricted to 64 kilobytes, whereas a set of body surface map data can be much larger. To overcome these problems and in order to allow uninterrupted use of the computer by other users as well as by the mapping system, a number of minor modifications to the operating system were made. The UNIX system was restricted to run in the first 1 Mb of memory, leaving the second 1 Mb of memory free for use as a buffer to store and process data from the mapping system. This modification entailed minor changes to the UNIX startup code. A system call was added to UNIX to allow some programs to map part of their data space to areas in the 1 Mb memory buffer. This allowed programs to access and process the sampled mapping data. Finally, a device driver for a high speed direct memory access interface (DEC DR11W) was added to the system. This

* UNIX is a trademark of Bell Laboratories
** RT11 is a trademark of Digital Equipment Corporation

allowed data from the mapping system to be stored in the memory buffer using direct memory accesses, with minimal impact on the rest of the system.

A program called sample controls the mapping system hardware by setting appropriate signals on the DR11W control lines. When sample is run from the mapping trolley terminal, the program turns on the battery power, checks the battery voltages and allows a short time for the amplifiers to warm up and settle. The program prompts the operator to enter the position of the last column of electrodes which contacts the patient (this information is used to allow for different sized patients) and then prompts the operator to select the sampling rate. Sampling starts as soon as the sampling rate has been entered. While sampling continues, data are transmitted along the multiple pair cable and stored in the computer memory by the DR11W. A sampling rate of 1 kHz for each channel gives 8 seconds of sampling time per megabyte of memory. Slower sampling rates give proportionally longer times. Because other programs (described in chapter three) also use some of the second megabyte of memory as a work area, the entire megabyte is not always available for storing mapping data. The system has been set up so that at least 1 second of sampling time at 1 kHz is always available.

When sampling is finished, the program de-multiplexes the stored data and plots them on the graphics terminal (Figure 2.3). Saturated or excessively noisy leads are automatically marked and the operator can remove or restore leads manually if desired. Rejected leads are reconstructed by interpolation from the

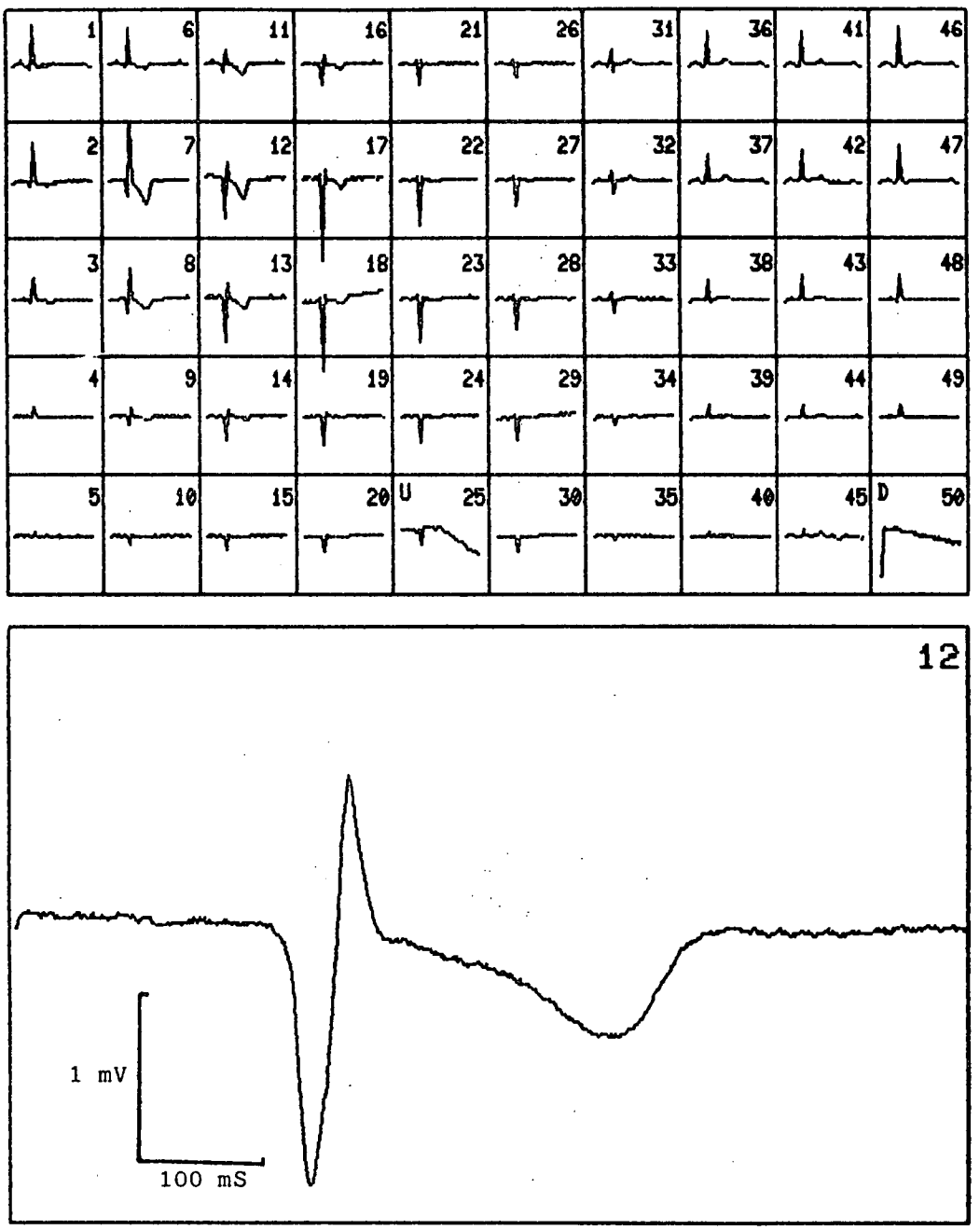


Figure 2.3 The upper box shows a plot of raw data as it is displayed on the graphics terminal. Channel 25 has been rejected by the operator, and channel 50 has been automatically rejected by the computer. The lower box shows channel 12 at a larger scale.

surrounding leads. If the operator is satisfied with the overall quality of the data, he selects a program option which saves the data and relevant patient information in permanent, numbered files. Otherwise the sampling process may be repeated, collecting a new set of data. Good data are usually obtained on the first run. Each run takes less than 1 minute (most of that time is taken up plotting the raw data on the graphics terminal). A simple database is maintained by sample to keep records of each patient's name, hospital record number, time and date of sampling and file name of stored ecg data.

The automatic detection of a saturated or noisy lead is done by building a table containing how many times each voltage value occurs in the lead over the sampling period. If either of the voltage values corresponding to the upper or lower conversion limit of the analog to digital converter occurs at all, the lead is considered saturated. Otherwise, the spread of values around the most common value is examined. A large spread implies that the lead is noisy or has an excessively sloping baseline, and such a lead will be marked accordingly. Occasionally a lead may not contact the patient and yet yield little noise and a reasonably flat baseline. Such a lead will not be rejected by the above automatic methods. In this case, the operator must reject the lead manually. It would be convenient if the software could detect such a lead from the 'context' of the surrounding leads. This type of detection has not yet been attempted.

Because many of the features of interest in the electrocardiogram are located with reference to the onset of the QRS

complex, it is convenient to calculate and save the QRS onset location with the data. For this calculation, the baseline of each lead is taken to be an average of a small range of voltage values around the most common value. The QRS onset location is then calculated from the sum of the absolute values of all leads (the comments with the program listing qrs in Appendix B give details of the algorithm used). The calculated QRS onset location is displayed and can be adjusted by the operator if desired.

Data from the stored files can be processed to produce various types of body surface maps. A collection of programs allows mapping at a single point in time, integrating over selected time intervals, subtracting or adding maps, scaling maps or interpolating maps. The inverse transforms described in chapters four and five can also be applied to the data in order to calculate epicardial potential distributions.

A minimum of processing is done on the data to produce body surface maps. Linear interpolation is used to create columns of data between the original electrode columns which were spaced 15 cm apart. The neck and reference electrode values are expanded out to form a row of identical values along the top and bottom of the map respectively. A copy of the first column of values is placed after the last column so that the maps appear to wrap horizontally. These procedures produce an array of voltage values with 7 rows and a variable number of columns depending on the size of the patient. Contours are then found and plotted by simple linear interpolation along horizontal and vertical lines which join adjacent electrode

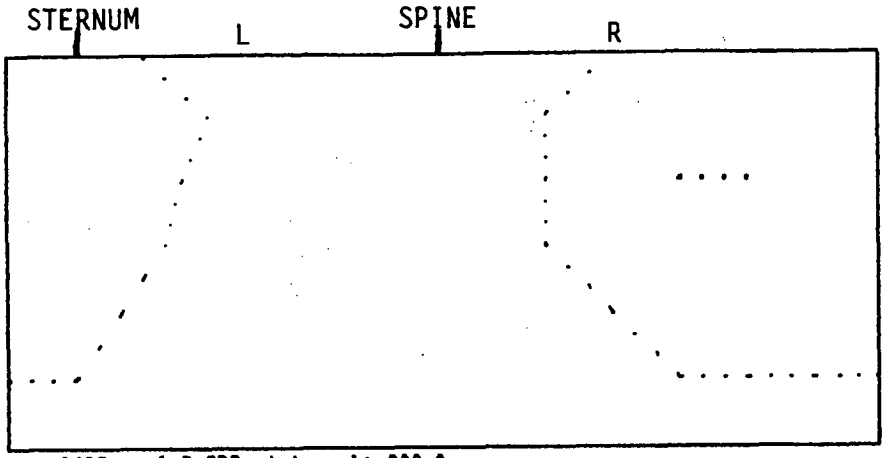
values. Normally the maps are constructed with all leads referenced to an artificial central terminal, which is calculated as the average of all the torso leads. Other reference points can be calculated if required. Figure 2.4 shows some examples of isopotential contour maps derived from data measured from a 38 year old normal male subject.

Storage of the data has so far not been a major problem. At the time of writing, over 1300 sets of data measured from over 600 distinct patients and normals have been collected by doctors, nurses and medical students. These data occupy approximately 140Mb of disc space, and are stored in a somewhat wasteful form (no compression of any kind has been done, and each 10 bit sample is contained in a 16 bit word). Quite simple compression techniques (such as storing 8 bit differences between values instead of 10 bit values) would result in a saving of at least 50% of this space, at the expense of somewhat increased processing time when storing data and producing maps.

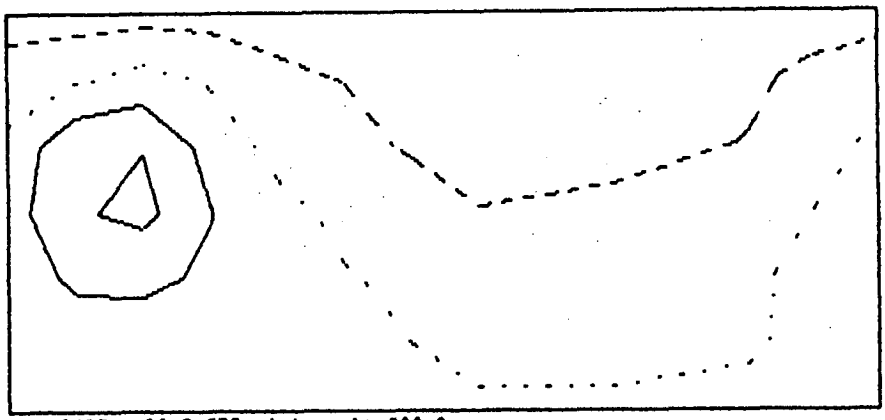
Listings of the alterations to the UNIX system, DR11W device driver and the main program used to run the sampling system are contained in Appendix B.

Conclusion

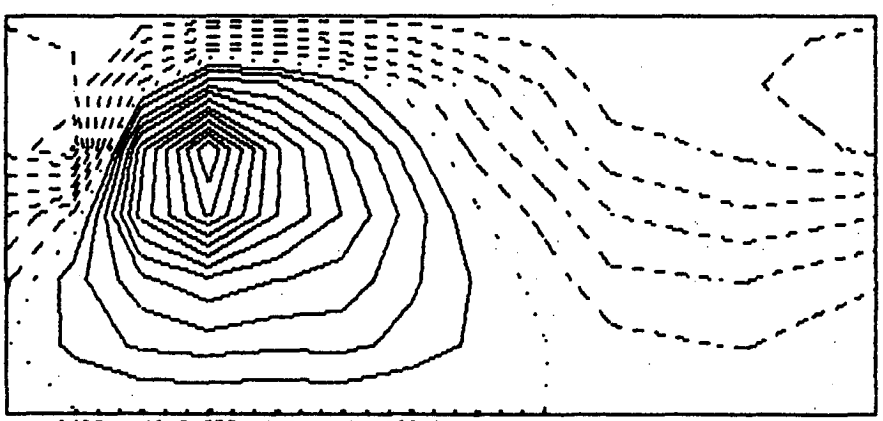
A system for sampling and processing electrocardiographic body surface potential data has been designed and built. The system enables rapid placement of electrodes on sick patients. It takes less than 2 minutes to put the jacket on a patient, and only a few



ecg.0428 0mS QRS interval: 200.0
Sample: 201 Integrated 1 samples
RMS: 18.0
Max 59.0
Min -29.0

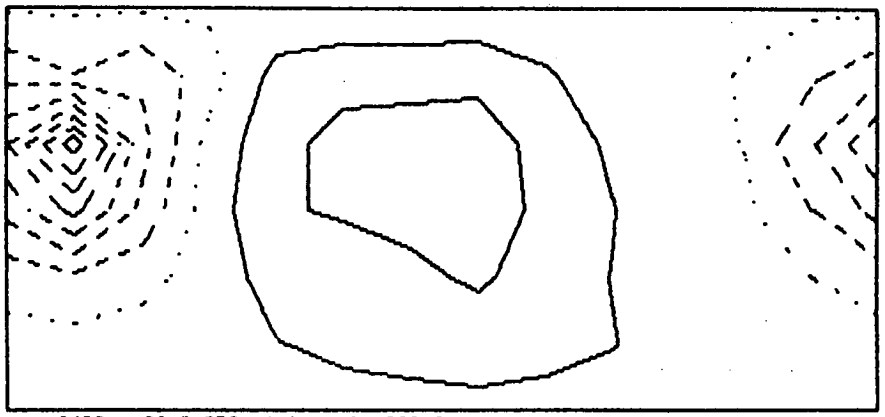


ecg.0428 20mS QRS interval: 200.0
Sample: 221 Integrated 1 samples
RMS: 176.2
Max 459.0
Min -342.0

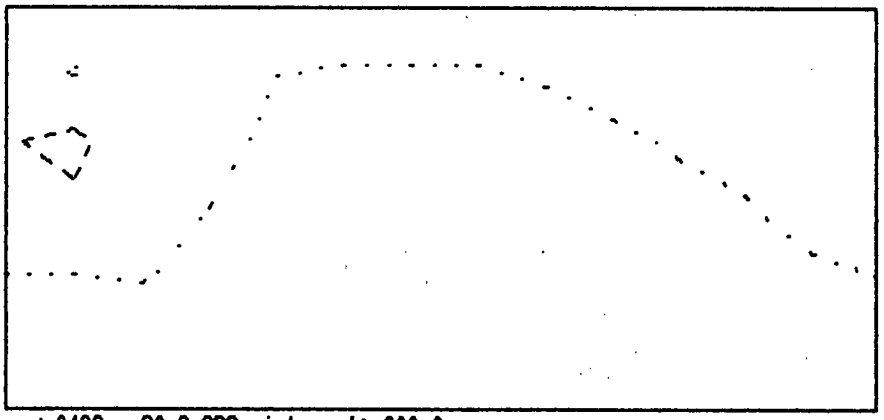


ecg.0428 40mS QRS interval: 200.0
Sample: 241 Integrated 1 samples
RMS: 866.0
Max 2588.0
Min -1611.0

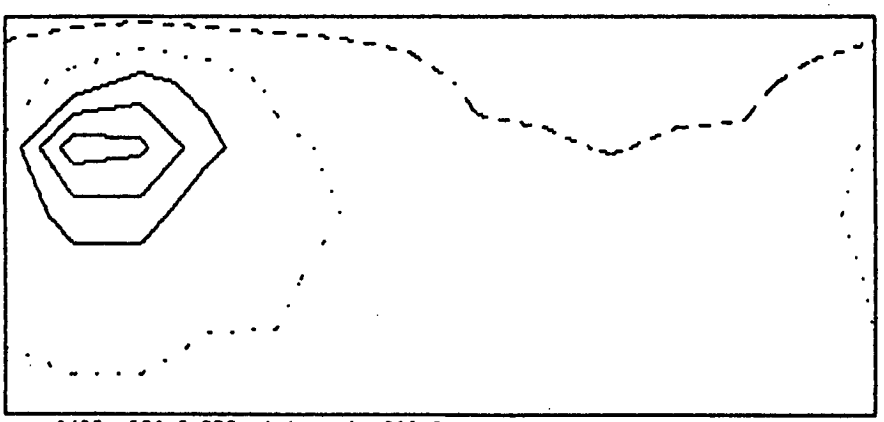
Figure 2.4 Example isopotential contour maps produced from the raw data shown in figure 2.3. The first 5 maps (see also next page) are spaced at 20mS intervals throughout the QRS complex. The last map is located in the T wave. Solid contour lines are positive, dashed lines are negative and the dotted line is the zero line.



ecg.0428 60mS QRS interval: 200.0
Sample: 261 Integrated 1 samples
RMS: 357.6
Max 508.0
Min -1514.0



ecg.0428 80mS QRS interval: 200.0
Sample: 281 Integrated 1 samples
RMS: 78.7
Max 146.0
Min -244.0



ecg.0428 280mS QRS interval: 200.0
Sample: 481 Integrated 1 samples
RMS: 171.5
Max 723.0
Min -264.0

Figure 2.4 continued.

more minutes to complete sampling. Both raw data and various types of body surface maps can be displayed within minutes at the bedside. This allows evaluation of the data while they are still clinically useful. Body surface potential maps may be stored for large numbers of patients and are immediately accessible.

CHAPTER THREE

THE FORWARD PROBLEM

Introduction

This chapter describes the implementation of a model used to solve a restricted version of the electrocardiographic forward problem. It is restricted in the sense that no attempt has been made to model physiologically reasonable electric sources within the heart muscle. Instead, the sources used are regions of the heart surface held at specified potentials. Thus, this solution gives the potential distribution on the surface of the torso produced from a given epicardial potential distribution The problem is approached in this way so that the inverse calculations described in the next chapter will yield epicardial potentials from measured torso surface potentials.

Suppose that these are n source regions on the surface of the heart. The epicardial potential distribution can be written as a column vector of length n :

$$h = (h_1 \ldots\ldots h_n)^t \qquad (3.1)$$

where h_j is the potential at the jth source region, and the superscript 't' signifies the transpose operation. Similarly, for the surface of the torso, we write :

$$b = (b_1 \ldots\ldots b_m)^t \qquad (3.2)$$

where we are interested in knowing the potential at m locations on the surface of the torso, and b_i is the potential value at the i'th location. Note that, in solving the forward problem, potentials may be calculated at a large number of points on the surface of the torso. The m locations mentioned above will be a subset of these points and may correspond to electrode locations or other sites of special interest.

Once an appropriate model has been constructed, the contribution from a unit source located at the j'th source region to the potential at each of the m torso surface locations can be calculated. Denote these quantities t_{1j}, t_{2j}, ... t_{mj}. This calculation can be performed for each of the n source regions in turn. Then for an arbitrary epicardial potential distribution h we can construct the following matrix equation:

$$Th = b \qquad (3.3)$$

where T is the m x n matrix made up of the elements t_{ij}. In effect, we have scaled each source region's contribution and then added the contributions from all the regions to find the torso surface potential distribution b. This is analogous to the principle of superposition in electrostatics (Lorrain and Corson 1970). The matrix T is called the forward transfer matrix. Solving the forward problem becomes the problem of setting up an appropriate model of the electrical properties of the human torso and calculating the

elements t_{ij} of the matrix T.

A relatively automatic, computerised system has been set up to perform these calculations. The system involves gathering data on the geometry and conductivity of regions of the torso, generating a system of equations which describe the potential distribution throughout the torso and solving the equations. A collection of computer programs form the system, each performing a particular task and providing output in a form suitable for the next stage. Both a homogeneous sphere (for validation and theoretical studies), and a human torso model with realistic geometry have been studied in this chapter.

The Model

A numerical model which approximates the electrical properties of the human torso has been developed. The model simulates a three dimensional resistor network which resembles the human torso both geometrically and electrically. The type of model is the same as used by Lo (1977), although the models produced are substantially more detailed. Also, the procedures used here to set up the model are substantially more automated than those used by Lo, allowing multiple models to be set up without an unrealistic amount of effort.

The first step in setting up the model is to record the geometry of the torso. Computerised Tomographic (CT) scan images of the torso are used as the input data. Each slice is traced on a computer digitising tablet, producing a list of co-ordinates for

points on the boundaries of the major organs and the boundary of the torso. The structure of the heart is not present in conventional CT scan images because the heart is moving while the scan is being done. However it is possible to define a surface which closely surrounds the heart from the images, and this is all that is required for the models described here. For the purposes of this thesis, all further references to the heart's surface (or epicardium) are actually referring to this surface which closely surrounds the heart.

A sample scan and a plot of the corresponding co-ordinates are shown in figure 3.1. Note that any other method for producing cross-sectional images of the torso, such as nuclear magnetic resonance scanning or even a cross-sectional anatomy text, could be used to provide the necessary geometrical data. CT scans were chosen because they were readily available from the Royal Hobart Hospital, and also in the hope that a range of torso shapes (with relatively normal internal geometry) could be recorded. It is possible that given appropriate edge detecting software, the CT scans could be directly transferred from the digital data recorded by the CT scanning system. A video display would be needed to allow some operator intervention, but the task of manually tracing the printed images would be eliminated. This procedure has not yet been attempted.

Once the co-ordinates have been recorded, each slice is approximated by a large number of rectangular elements. Each element is assigned a code corresponding to the conductivity of the

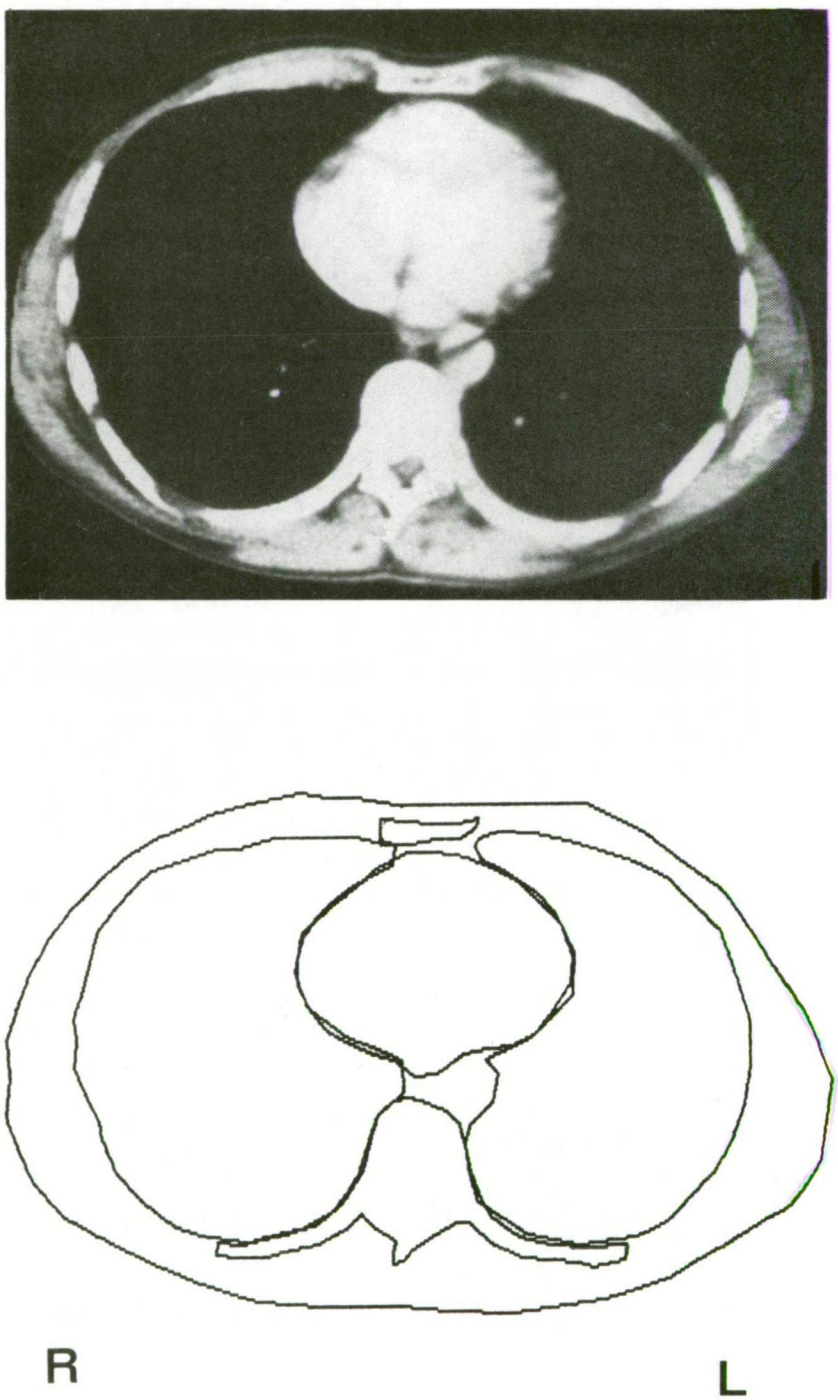


Figure 3.1 A sample CT scan image with the corresponding digitised boundaries plotted below.

tissue at the corresponding point in the original slice. For example, a coded slice which is triangular in shape, with a rectangular region inside, will look like this:

```
x
xxx
xxxxx
xxoooxx
xxxoooxxx
xxxxoooxxxx
xxxxxxxxxxxxx
xxxxxxxxxxxxxxx
```

This is more compactly stored in the computer, especially for large slices, by using run length coding:

```
7- 1x
6- 3x
5- 5x
4- 2x 3o 2x
3- 3x 3o 3x
2- 4x 3o 4x
1- 13x
15x
```

Here, the code '-' means an empty element.

The dividing of a slice up into a rectangular grid is performed in much the same way as a person would proceed using a ruled grid overlaid on a plotted slice. Elements which lie entirely within a single region are assigned the conductivity of that region. Elements which are intersected by boundaries are assigned the conductivity of the region which has most area inside the element. Two programs are used. The first program converts the lists of boundary coordinates into a very detailed approximation of the slice using small square elements. The second program takes this highly detailed approximation and produces the final model. Each

rectangular element is assigned a conductivity code by counting the number of small square elements of each type contained within it. This two stage procedure avoids the highly complex calculations required to directly determine the appropriate code for a rectangular element which lies across region boundaries. Appendix C contains details of the algorithms used, as well as program listings.

Each coded slice can be thought of as a horizontal slab cut from the torso at some point. The thickness of this slab is equal to the slice spacing. (If for some reason the slices are not evenly spaced, the thickness can be assigned the value of half the spacing to the previous slice plus half the spacing to the next slice). When all the slabs are put together, an approximation of the torso is produced which consists of a large number of coded rectangular prisms. The next step in developing the model is to replace the coded set of elements making up the torso volume by an equivalent resistor network.

The corners of the volume elements are a set of points distributed throughout the torso model. It is at these points, or 'nodes', that the potentials due to the epicardial sources will be calculated. The conducting volume between two adjacent nodes is replaced in the model by an equivalent resistor. The value of these resistors will depend on the dimensions and resistivity of the volume elements between adjacent nodes. Consider a small part of the coded volume shown in figure 3.2. Nodes N and A are joined directly by four volume elements with resistivities R_1, R_2, R_3 and R_4 Ωcm.

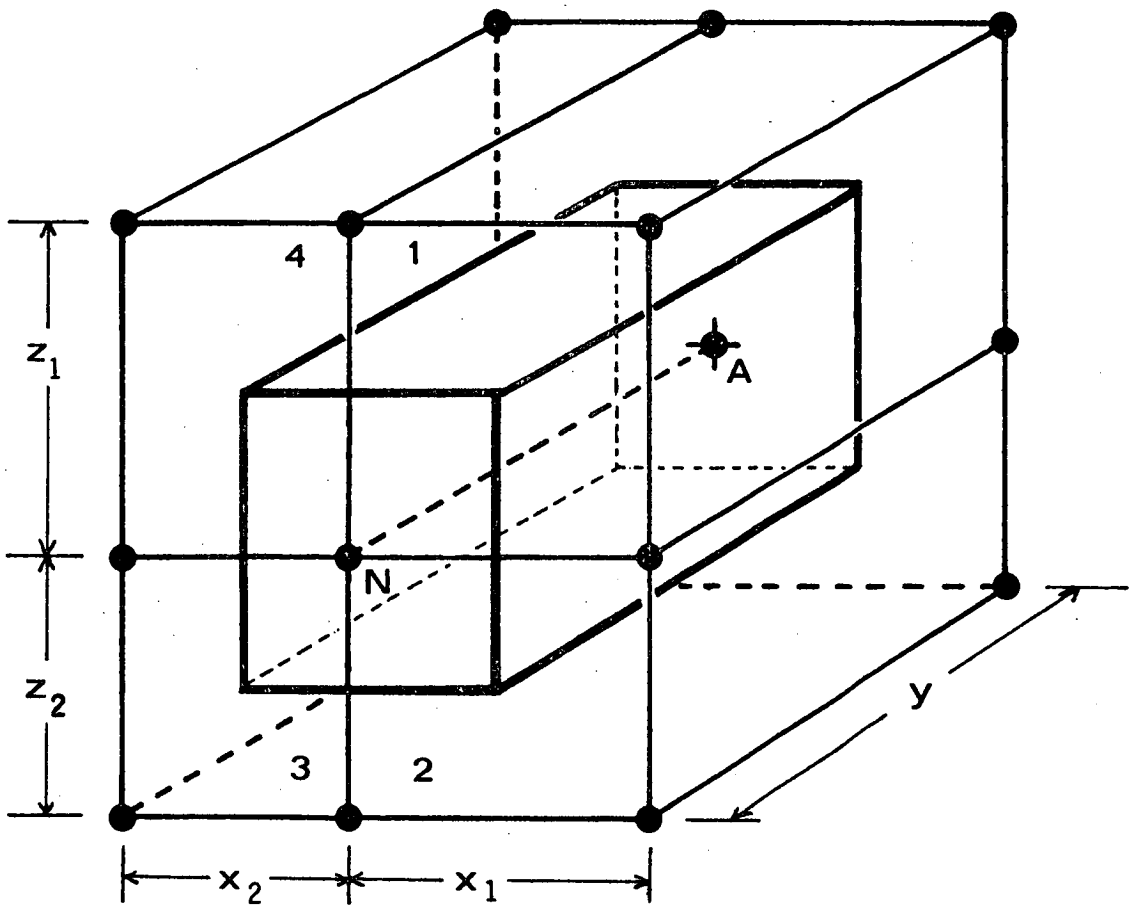


Figure 3.2 A small part of the coded volume. Nodes N and A are connected by elements 1, 2, 3 and 4. The equivalent resistance calculated between N and A is the resistance of the central volume drawn in thick lines.

The dimension of the elements are as shown by x_1, x_2, z_1, z_2 and y. The central volume is the conductive path between N and A which is to be replaced by an equivalent resistor. It contains the line joining N to A and extends to include one quarter of each of the surrounding four volume elements. The equivalent resistance value is calculated as the resistance of each of these quarter elements placed in parallel. The resistances r_1, r_2, r_3 and r_4 of the quarter elements are:

$$r_1 = \frac{4R_1 y}{x_1 z_1} \tag{3.4}$$

$$r_2 = \frac{4R_2 y}{x_1 z_2} \tag{3.5}$$

$$r_3 = \frac{4R_3 y}{x_2 z_2} \tag{3.6}$$

$$r_4 = \frac{4R_4 y}{x_2 z_1} \tag{3.7}$$

The resistance r_{NA} between N and A is then given by:

$$r_{NA} = 4\left(\frac{x_1 z_1}{r_1 y} + \frac{x_1 z_2}{r_2 y} + \frac{x_2 z_2}{r_3 y} + \frac{x_2 z_1}{r_4 y} \right)^{-1} \tag{3.8}$$

Note that if nodes N and A are on the surface of the torso, there will be less than four volume elements connecting them. This simply means that some terms will be missing in the right hand side of expression 3.8 for r_{NA}. Also, in this example, the resistance calculated lies in the y co-ordinate direction. Resistances lying

in the other two co-ordinate directions are calculated using expressions similar to those above, with x, y, and z appropriately permuted.

It is at this point in the procedure that anisotropic resistivities may be modelled. All that has to be done is to use the x, y, or z component value of resistivity when calculating the equivalent resistance in each of those directions. (Note that, in full generality, the conductivity (or resistivity) is a 3x3 tensor. The x, y and z components mentioned here are the diagonal elements of the tensor). The resistivities appearing in the equations would be r_1^x, r_1^y, r_1^z, r_2^x etc. The problem with including anisotropy into the model is that it is necessary to know the resistivity component values on an element by element basis. This level of detail is needed as the components of resistivity change from place to place even within an organ or single tissue type (this is especially true of the heart and skeletal muscle). Because of the difficulty of obtaining such information and incorporating it into the model, all regions of the torso have been assumed to be isotropic.

The values used for tissue resistivity in this model have been obtained from a paper published by Rush et al.(1963). They summarise results of measurements of body tissue resistivity by Kaufman and Johnston (1943), Burger and VanMilaan (1943), Schwan and Kay (1956) and Burger and VanDongen (1961), as well as performing their own measurements. Previous differences in measured values are explained and the values which Rush et. al. present seem to be the most reliable which are currently obtainable. Where the tissue is

anisotropic, the geometric mean of the high and low values stated by Rush et al. has been used. The tissue resistivities used, and their corresponding codings, are shown in table 3.1.

Once the resistances from a node to each of the adjacent nodes have been calculated, equations giving the potential at each node in terms of the potentials at the adjacent nodes are derived. Consider the situation shown in figure 3.3. A node N has 6 adjacent nodes labelled A to F. The resistances from node N to each of the nodes A to F are r_a, r_b, r_c, r_d, r_e and r_f respectively. The potential at node N is v_n and at nodes A to F the potentials are v_a, ... v_f. If node N is not a source node then the net current flowing into or out of it is zero (this is Kirchhoff's current law). Thus, applying Ohm's law to find the currents in the resistors and then summing them, we have:

$$\frac{(v_a-v_n)}{r_a} + \frac{(v_b-v_n)}{r_b} + ... + \frac{(v_f-v_n)}{r_f} = 0 \qquad (3.9)$$

Rearranging to express v_n in terms of the other potentials gives:

$$v_n = \left(\frac{1}{r_a} + \frac{1}{r_b} + ... + \frac{1}{r_f} \right)^{-1} \left(\frac{v_a}{r_a} + \frac{v_b}{r_b} + ... + \frac{v_f}{r_f} \right) \qquad (3.10)$$

At a source node the potential is assigned a fixed value s_n:

$$v_n = s_n \qquad (3.11)$$

Note here that the sources may be specified in terms of currents

TABLE 3.1

Tissue Codings and Resistivities

Tissue type	Code	Resistivity (Ωcm)
Blood	B	160
Bone	b	∞
Fat	F	2500
Heart	H	380
Liver	I	700
Lung	L	2100
Torso	T	460

These resistivity values have been taken from Rush et. al. (1963). The resistivity of the heart is the geometric mean of the high and low values quoted in that paper. Bone is assumed to be totally non-conducting. The value corresponding to the tissue type 'torso' is used for all regions in the models whose resistances are not otherwise explicitly specified.

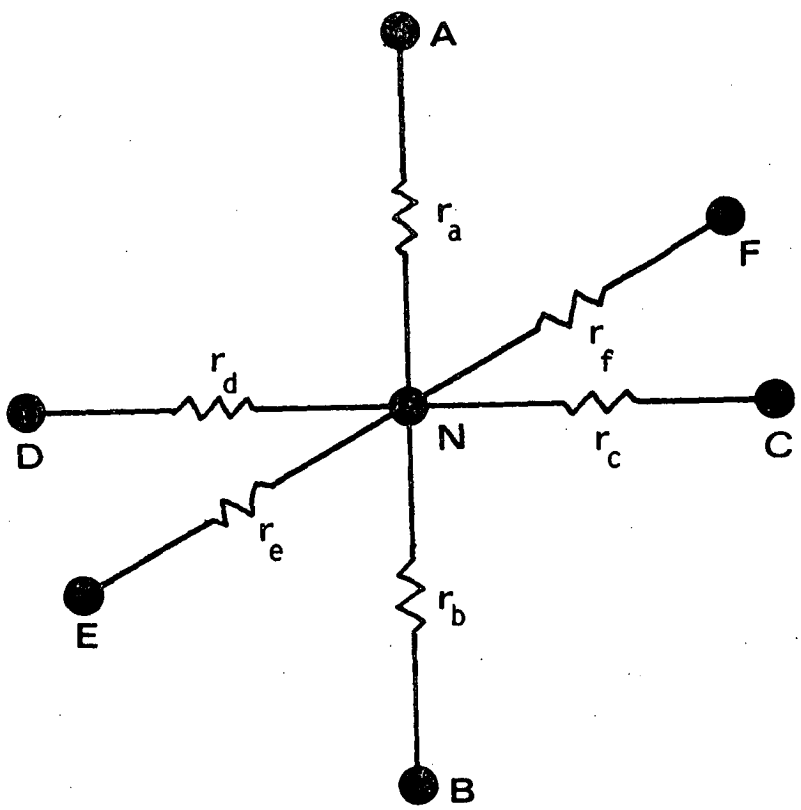


Figure 3.3 Nodal arrangement used to develop equation 3.9.

instead of potentials. Then the equation for a source node will have the form of equation 3.9 except that the right hand side will be the non-zero source current I_N. If this is done, the potential at one node must still be fixed for the equations to be solved (otherwise an arbitrary constant may be added to all the potentials without affecting the current distribution, thus giving an infinite number of solutions). Also, the sources must then be set up so that there is no net current flowing into or out of the torso. These two ways of specifying the sources are not independent. For a given model geometry, specifying the source potentials determines the source currents and vice versa. In the models developed here, the sources have been specified in terms of potentials with a view to obtaining a solution of the inverse problem which yields epicardial potential distributions.

Writing equation 3.9 for each node in the torso model gives a system of simultaneous linear equations for the potentials at every node. If a matrix A is constructed where the ij'th element a_{ij} is the coefficient of the j'th potential in the i'th equation, then the system of equations can be written in matrix form thus:

$$AV = k \tag{3.12}$$

Here V is a column vector of potentials and k is a column vector of constants (the right hand sides of equations 3.9 and 3.11).

The equations are generated by two programs. The first program takes the coded torso volume and produces a list of all the nodes in the volume, each with the resistance codes which surround it. The

second program takes this list and generates the equations, finding the neighbours of each node and calculating the corresponding coefficients. Because similar node configurations occur many times throughout the model, most coefficient values appear repeatedly. Storage is saved both when generating and solving the equations by building a hashed table of coefficient values, allowing each distinct value to be stored only once. The coefficients in each equation are then specified by pointers to the appropriate table values, the pointers requiring only half the storage of the actual values. Program listings are contained in Appendix D.

Solving the Equations

The system of equations represented by 3.12 is large - in some of the torsos modelled in chapter five there are over 20000 nodes and hence over 20000 equations. As a result, the system is not amenable to direct methods of solution such as matrix inversion or gaussian elimination. However, A is a sparse matrix - there are at most seven non-zero elements in any row since at most seven potentials appear in any one equation. This makes the system of equations amenable to solution by iterative methods. Each equation is re-written, so that for the i'th equation the potential at the i'th node is expressed in terms of all the other potentials. The i'th equation then becomes:

$$V_i = \frac{1}{a_{ii}} \left(k_i - \sum_{j=1}^{i-1} a_{ij} V_j - \sum_{j=i+1}^{m} a_{ij} V_j \right) \qquad (3.13)$$

where m is the total number of equations.

From this point, a number of iteration schemes can be used. (These schemes are described in more detail in Smith (1978) or Dahlquist and Bjorck 1974) The simplest is the Jacobi method, where at each iteration the new potential at each node is calculated by substituting the previous values for all the potentials into the right hand sides of the equations. That is:

$$V_i^{n+1} = \frac{1}{a_{ii}} \left(k_i - \sum_{j=1}^{i-1} a_{ij} V_j^n - \sum_{j=i+1}^{m} a_{ij} V_j^n \right) \qquad (3.14)$$

A disadvantage of this scheme is that storage must be allocated for both the old and new values of each variable. The Gauss-Seidel method overcomes this by using new values as soon as they are available. The iteration scheme becomes:

$$V_i^{n+1} = \frac{1}{a_{ii}} \left(k_i - \sum_{j=1}^{i-1} a_{ij} V_j^{n+1} - \sum_{j=i+1}^{m} a_{ij} V_j^n \right) \qquad (3.15)$$

The rate of convergence can often be improved by making the change in each variable larger than would occur in the above scheme. This leads to the accelerated Gauss-Seidel method, alternatively called Successive Over-Relaxation (SOR). In the SOR method the new potential becomes the old potential plus the change in the potential times a constant factor w, called the acceleration factor. The iteration scheme now becomes :

$$V_i^{n+1} = V_i^n + \frac{w}{a_{ii}} \left(k_i - \sum_{j=1}^{i-1} a_{ij} V_j^{n+1} - \sum_{j=i+1}^{m} a_{ij} V_j^n \right) - w V_i^n \quad (3.16)$$

When w=1 this method is identical to the Gauss-Seidel method. The rate of convergence of this iterative procedure is heavily dependent on w. The optimum acceleration factor has a value between 1 and 2 and can be found empirically from the following formula (Smith 1978):

$$w = \frac{2}{1 + \sqrt{1-\lambda}} \quad (3.17)$$

where λ is defined as follows:

$$\lambda = \lim_{n \to \infty} \left(\frac{\max | v_i^{n+1} - v_i^n |}{\max | v_i^n - v_i^{n-1} |} \right) \quad (3.18)$$

where the maxima are taken over all i (i=1,m) and the values v^n etc are successive iteration values for the variables calculated using the Gauss-Seidel method (SOR with w=1).

The SOR method has been used to solve the system of equations in all cases. The optimum acceleration factor is calculated by solving the system of equations once with the Gauss-Seidel method and determining the value for λ after a large number of iterations (typically several thousand). The time spent in calculating λ is saved many times over because each system of equations must be solved many times (once for each source region on the heart

surface). Studies described later in this chapter show that using an acceleration factor which is near optimal can reduce the number of iterations required by an order of magnitude or more

A problem with all iterative methods is deciding when to stop iteration; that is, deciding when you have approached a solution which is acceptable. For these models, iteration was stopped when the maximum change in any variable was less than or equal to 10^{-9} (the values of variables were mostly in the range 1 to 100). All calculations were done in double precision. Although this uses double the storage space of single precision and can be slightly slower (much slower if floating point calculations are not performed by hardware), it was found to be necessary, as single precision attempts showed erratic behaviour once the changes in the variable values decreased to around 10^{-5}.

A considerable amount of time was spent trying to make the process of solving the equations as efficient as possible. The modelling was performed on the PDP 11/44 computer system described in the previous chapter. To reduce the processing time required to solve the models, the second megabyte of memory was used as storage for the variables and equation coefficient pointers. This allowed models with up to 24500 nodes to be solved in memory with no disk I/O at all. The program which solves the equations was written in the C programming language (Kernighan and Ritchie 1978). The program was then compiled to assembly language, which was optimised by hand before final assembly. This optimisation resulted in a speed increase of almost 50%, mainly due to the fact that the C compiler

only uses two of the six available registers in the hardware floating point processor on the PDP 11. Time can also be saved by starting the iteration procedure with initial values for the variables which are as close as possible to the solution. In most of the models solved, the variables were initially set at an estimate of the average potential throughout the model. In cases where a model had previously been solved and was to be solved again with only a slight variation in the source values, the previous solution was used as an initial estimate.

The Forward Transfer Matrix

Once the equations have been solved, the potential at every node in the model is known. In order to construct the forward transfer matrix for a given torso model, the potential distributions arising from a unit potential placed on each source region in turn must be calculated. This involves solving the system of equations as many times as there are source regions. In the torsos studied, the surface of the heart was divided into 25 regions, as shown in figure 3.4. This procedure is automated by two programs; one which finds the set of nodes on the surface of a specified region, and another which orders these nodes and divides them in cylindrical co-ordinates based on the centre of mass of the region. The system of equations for the torso was then solved with a value of +100V on each source region in turn (with 0V on all the other source regions). This +100V source value is chosen purely for convenience - the resulting potential values can be scaled if required. The number of source regions was chosen to be 25 for reasons related to

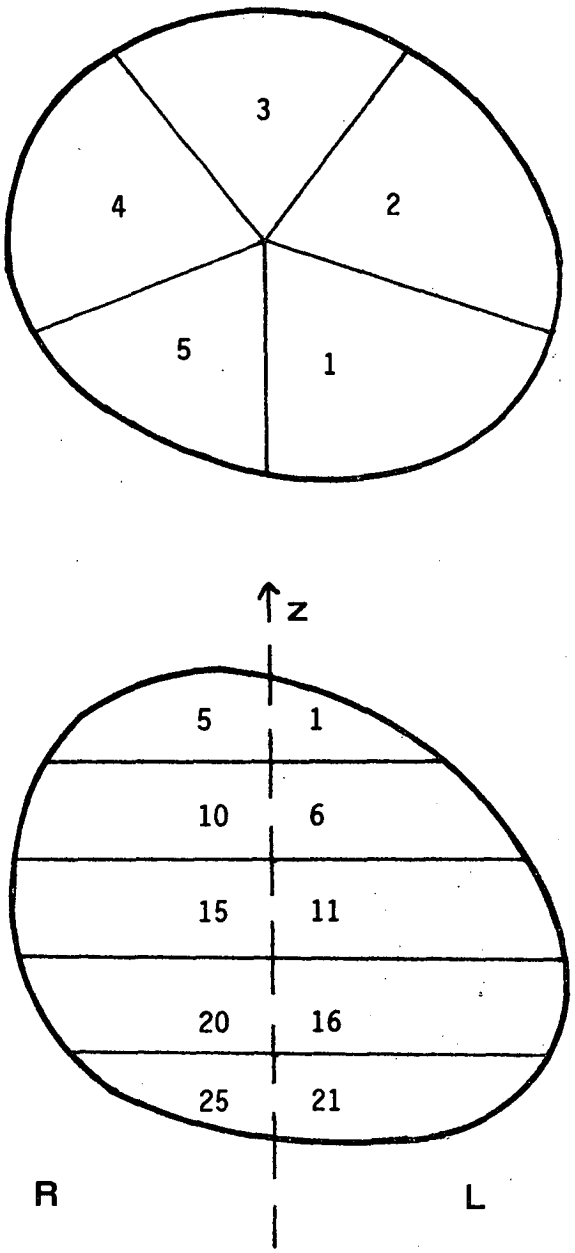


Figure 3.4 Division of the surface of the heart into source regions. Cylindrical co-ordinates are used, with origin at the centre of mass of the heart and z axis running parallel to the axis of the torso (foot to head). The upper drawing shows a top view of the heart, divided into 5 segments of equal angle around the z axis. The lower drawing shows an anterior view of the heart, divided into 5 sections of equal length along the z axis. These divisions combine to yield 25 regions on the surface of the heart. The regions are numbered as shown.

the number of parameters which are determined by an inverse transform based on the forward transfer matrices produced here. This point is discussed further in the next chapter.

Using the equipment described in chapter two, measurements of the electrocardiogram can be made at 40-60 locations on the surface of human subjects. It is of particular interest to obtain the potential values at corresponding locations on the surface of the model. This allows comparisons to be made between real human data and simulated cardiac signals. It also allows data collected in the intensive care unit to be directly used as input to the inverse transforms described in the next chapter. In order to find the set of nodes on the surface of the model which correspond to the mapping system electrode locations, a program has been developed which simulates the wrapping of the electrode jacket around the modelled torso. The program does this for each row of electrodes by finding the set of nodes on the surface of the model at the appropriate vertical level, ordering these nodes in sequence around the torso and then selecting those nodes at the correct spacing to match the electrode spacing on the jacket.

Once the selection of nodes on the surface of the body is done, the forward transfer matrix T is easy to construct. An element t_{ij} of the matrix is just the potential at the i'th electrode location arising from the j'th source region. As there are 40-60 electrode locations and 25 source regions, the matrix T will have 40-60 rows and 25 columns. It is convenient when generating T to scale the values t_{ij} so that in equation 3.3, if h is expressed in millivolts,

then b is expressed in microvolts. Note that forward transfer matrices based on other sets of nodes on the model surface can easily be constructed. This may be of value if it is desired to use body surface map data obtained by other workers as input data for the inverse transforms described in the next chapter.

The forward transfer matrix will only approximate the 'true' relationship between the epicardial potentials and body surface potentials in a real human torso. Obviously, a finite sized grid cannot exactly represent the continuous torso. Determining the error due to the finite grid size is in general very difficult. Some results exist for general finite element methods in one and two dimensions (Zienkiewicz and Morgan 1983) but they are not applicable to these models. The approach taken here has been to use as fine a mesh as it is practical to solve with the computing resources available. A non-uniform grid is used for the torso models, allowing the grid to be finer in regions where the potential is varying more rapidly (near the sources) without including an unnecessarily large number of nodes in other regions.

Other sources of error also occur in the modelling. The geometry may not be exact due to the finite resolution and finite spacing of the CT scan input data. This is unlikely to be a major problem, as these errors are smaller than errors of geometry resulting from the finite grid size. Another possible source of error is lack of convergence in the iterative procedure used to solve the equations. This can be due either to the finite precision arithmetic used or to a problem with the system of equations

themselves (there may be no solution, or the coefficients in the equations may not be accurate). Here, such problems were avoided by performing all calculations in double precision (approximately 17 significant digits). Certainly, no lack of convergence (resulting in unbounded solution times) was ever observed. A check on how closely the solutions converged to the limiting solution (which would be obtained after an infinite number of iterations) is described in the next section. Finally, it should be remembered that the resistivity values used for the various tissues are not completely accurate.

Validation and Preliminary studies

Having set up the framework necessary to generate computer models of the electrical properties of the human torso, it was necessary both to check the validity of the model and to perform a number of studies to 'learn' about the numerical behaviour of the models.

The validity of the model was tested by modelling a homogeneous spherical volume conductor with two point current sources and comparing the model solution with the known analytical solution . A computer program was used to slice a sphere of radius 15cm into slices each 1 cm thick. Co-ordinates of a large number of points on the boundaries of the slices were computed (this simulates the digitising process used for CT scans). The slices were then divided into a uniform 1cm square grid. The resulting model approximated the sphere by 1cm cubes and contained 16585 nodes. The nodes are

located on the corners of the cubes, some of which are necessarily just outside the boundary of the sphere. This results in a somewhat larger number of nodes than might be expected from consideration of the volume of the sphere ($14137cm^3$). Sources at potentials +100v and -100v were respectively located at +1cm and -1cm in the vertical direction from the centre of the sphere. The optimum acceleration factor for the sphere was calculated as 1.973.

Note that, because of the symmetry of the sphere model and sources, the problem could be considerably reduced in size by modelling only one half, one quarter, or even less of the sphere. The entire sphere has been modelled here to provide a test of the modelling procedure using a volume which is similar in size and number of nodes to the later torso models.

The formula used to obtain the analytical solution is the result obtained by Frank (1952):

$$V = \frac{I}{4\pi s}\left(\frac{1}{r_b} - \frac{1}{r_a} + \frac{R}{br_{bi}} - \frac{R}{ar_{ai}} + \frac{1}{R}\ln\frac{ar_{ai}+R^2-ra\cos\beta}{br_{bi}+R^2-rb\cos\theta}\right) \qquad (3.19)$$

where

$$r_a = (r^2 + a^2 - 2ra\cos\beta)^{0.5} \qquad (3.20)$$

$$r_b = (r^2 + b^2 - 2ra\cos\theta)^{0.5} \qquad (3.21)$$

$$ar_{ai} = (R^4 + r^2a^2 - 2R^2ra\cos\beta)^{0.5} \qquad (3.22)$$

$$br_{bi} = (R^4 + r^2b^2 - 2R^2rbCos\theta)^{0.5} \qquad (3.23)$$

and R, r, a, b, θ and β are as shown in figure 3.5. I is the source current, s is the conductivity of the sphere, and V is calculated at the point P. In the model considered here, a=b=1cm and the angle α subtended by the sources at the centre of the sphere is π radians so that $Cos\beta = -Cos\theta$. The value of source current used was obtained by calculating the currents flowing at the source points in the model solution. This was done so that the model and analytical solutions could be compared directly, with no scaling of potentials. In both solutions, the resistivity of the sphere was 360 Ωcm.

The resulting potential distributions for both the model and the analytical result are shown in figure 3.6. It can been seen that the isopotential contour maps in various sections through the sphere for both the model and the analytical result are almost identical. As a further check, the percentage difference d between the two solutions was calculated at positions in the sphere corresponding to the model nodes. At a given node position, d was calculated as follows:

$$d = \left| \frac{v_a - v_m}{v_a} \right| * 100 \qquad (3.24)$$

where v_a is the analytic value and v_m the model value of the potential at that node. This quantity cannot be calculated on the horizontal plane midway between the sources as v_a is zero on this plane (v_m was also zero to within the resolution of the computer). Contour plots of the percentage difference on various sections

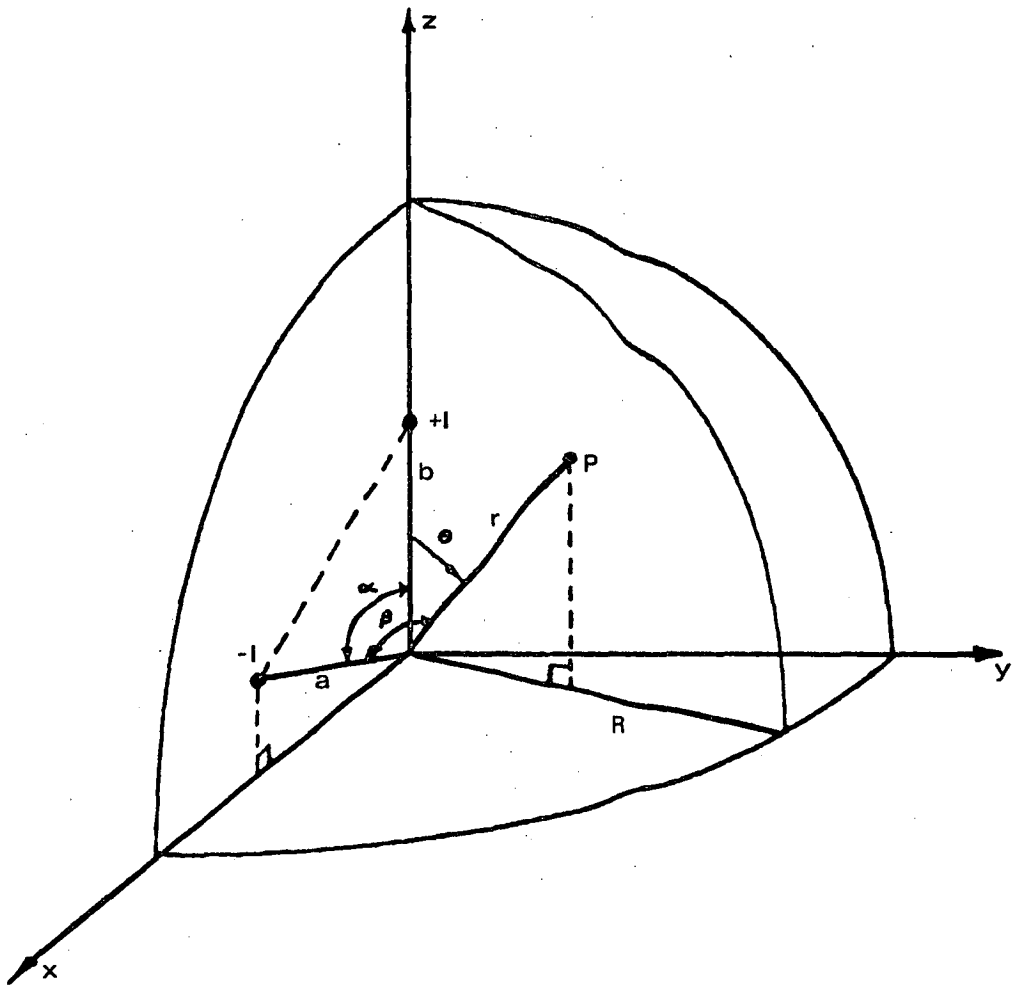
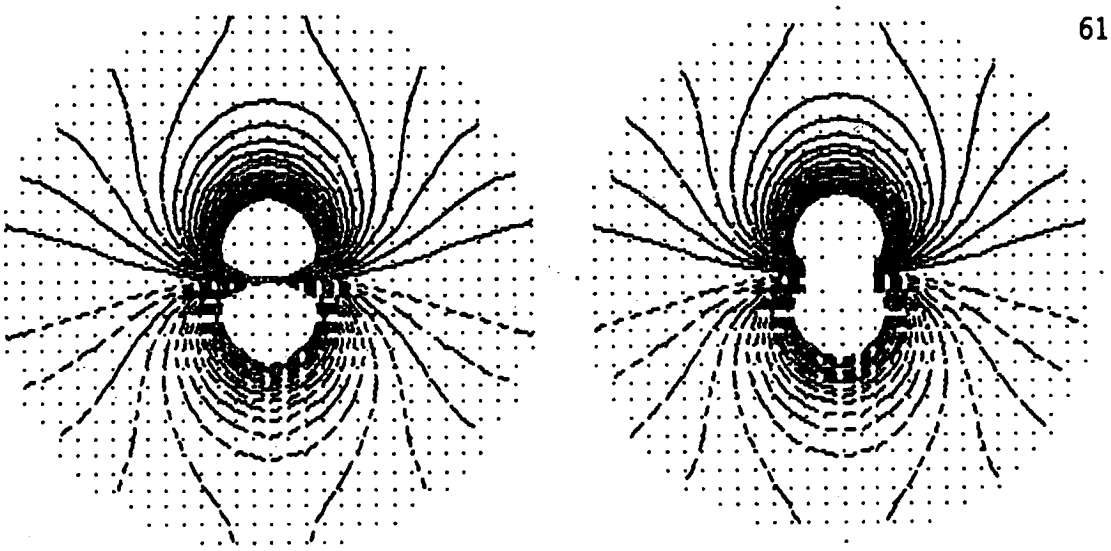
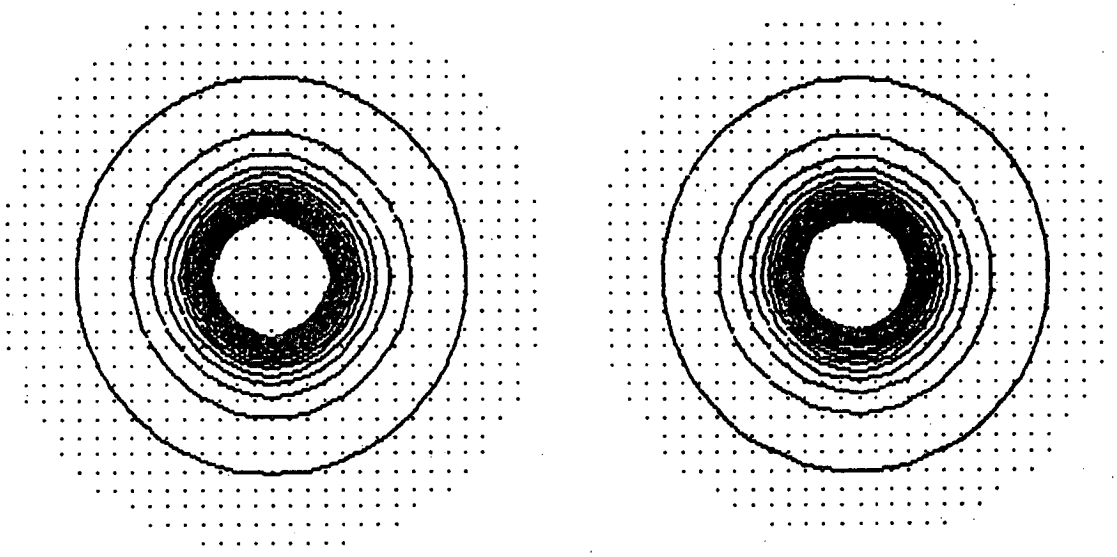


Figure 3.5 Two point current sources located in a conducting sphere, one octant of which is shown. Reproduced (with minor alterations) from Frank (1952).



MODEL ANALYTIC

CONTOUR INTERVAL 0.2V



MODEL ANALYTIC

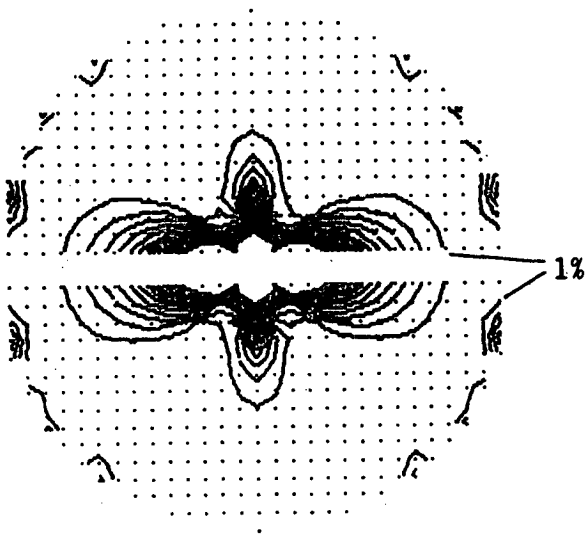
CONTOUR INTERVAL 0.1V

Figure 3.6 Potential distributions in the conducting sphere described in the text. The upper pair show the potential distributions in a vertical plane containing both sources. The lower pair show the potential distributions in the horizontal plane containing the positive source. Solid lines are positive and dashed are negative. Contour lines are omitted where they would become too close. Some nodes are omitted in the analytic distributions where the analytic formula does not apply (near the centre of the sphere and just outside the surface of the sphere).

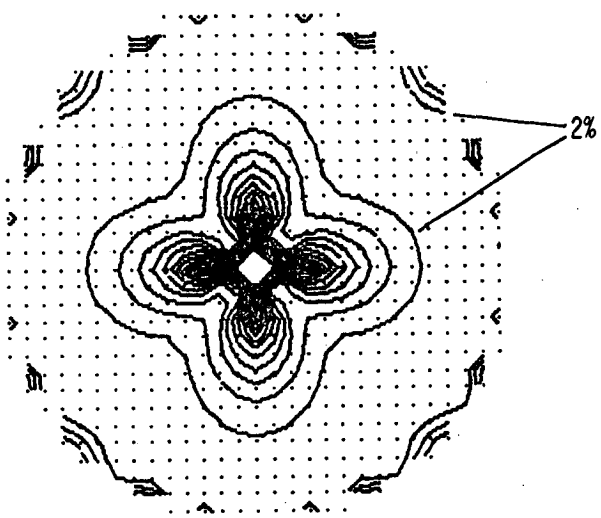
through the sphere are shown in figure 3.7. Through the outer half of the sphere the difference is of the order of 1% - 2%. At the surface of the sphere, the difference is somewhat more, 4% - 5% , but still small. This is probably due to the fact that the outer surface of the model cannot exactly fit the surface of the sphere. The worst errors occur near the sources, as would be expected (the potential approaches infinity when nearing the sources in the analytical solution). At nodes adjacent to the sources, the highest percentage difference was 27%.

A number of studies were performed on a human torso model. This torso was digitised from the CT scans of a 40 year old male with normal torso geometry. A non-uniform grid was used; grid spacing ranged from 7.5mm to 15mm. A slice with grid superimposed is shown in figure 3.8. The model was made from 25 slices, ranging in thickness from 10mm in the vicinity of the heart to 50mm in the lower part of the torso. The resulting model is highly detailed, containing 18924 nodes, and including the heart, lungs, spine and sternum.

To measure the effect on solution time, the model was solved a number of times with a fixed source distribution but varying the acceleration factor. A graph of the number of iterations required for a solution versus acceleration factor is shown in figure 3.9. It can be seen that the number of iterations (and hence time) taken for a solution is very dependent on acceleration factor. At the optimum acceleration factor of 1.94, the number of iterations was 248, whereas with an acceleration factor of 1.0 (Gauss-Seidel), 6556



CONTOUR INTERVAL 1%



CONTOUR INTERVAL 2%

Figure 3.7 Contour plots of the percentage difference between the model and analytic potential distributions shown in figure 3.6. The upper plot shows the percentage difference in a vertical plane containing both sources. Note that the percentage difference cannot be calculated on the horizontal plane between the sources (see text). The lower plot shows the percentage difference in the horizontal plane containing the positive source.

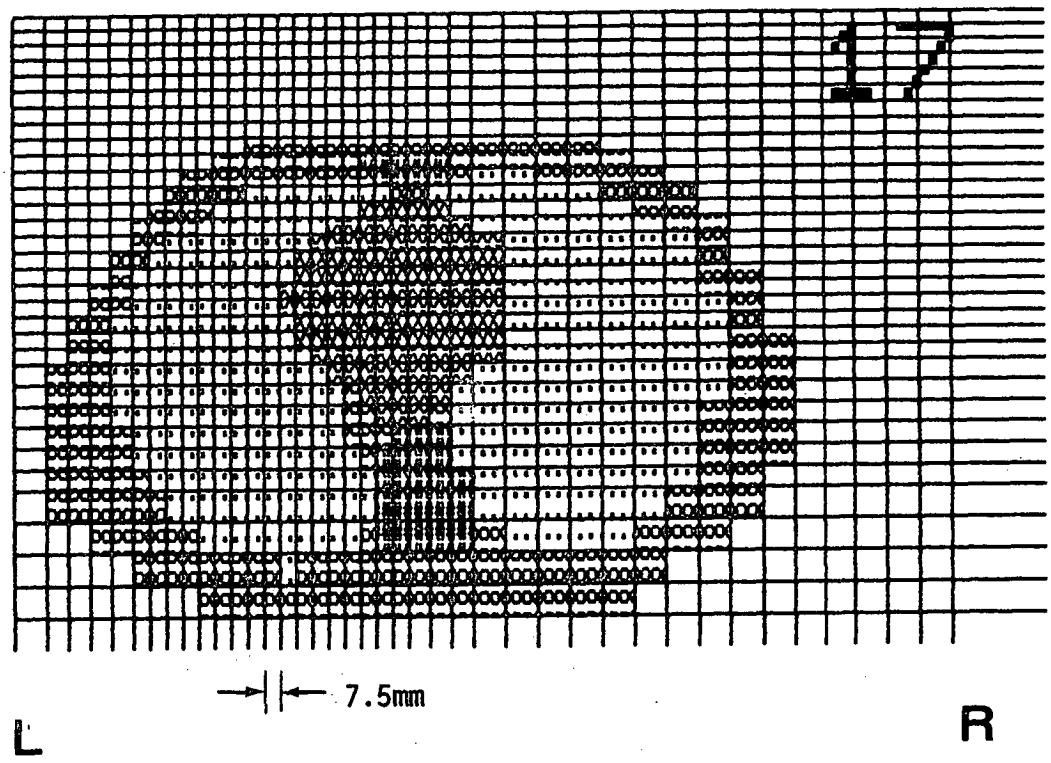


Figure 3.8 A computerised plot of slice 17 of the torso model with grid superimposed.

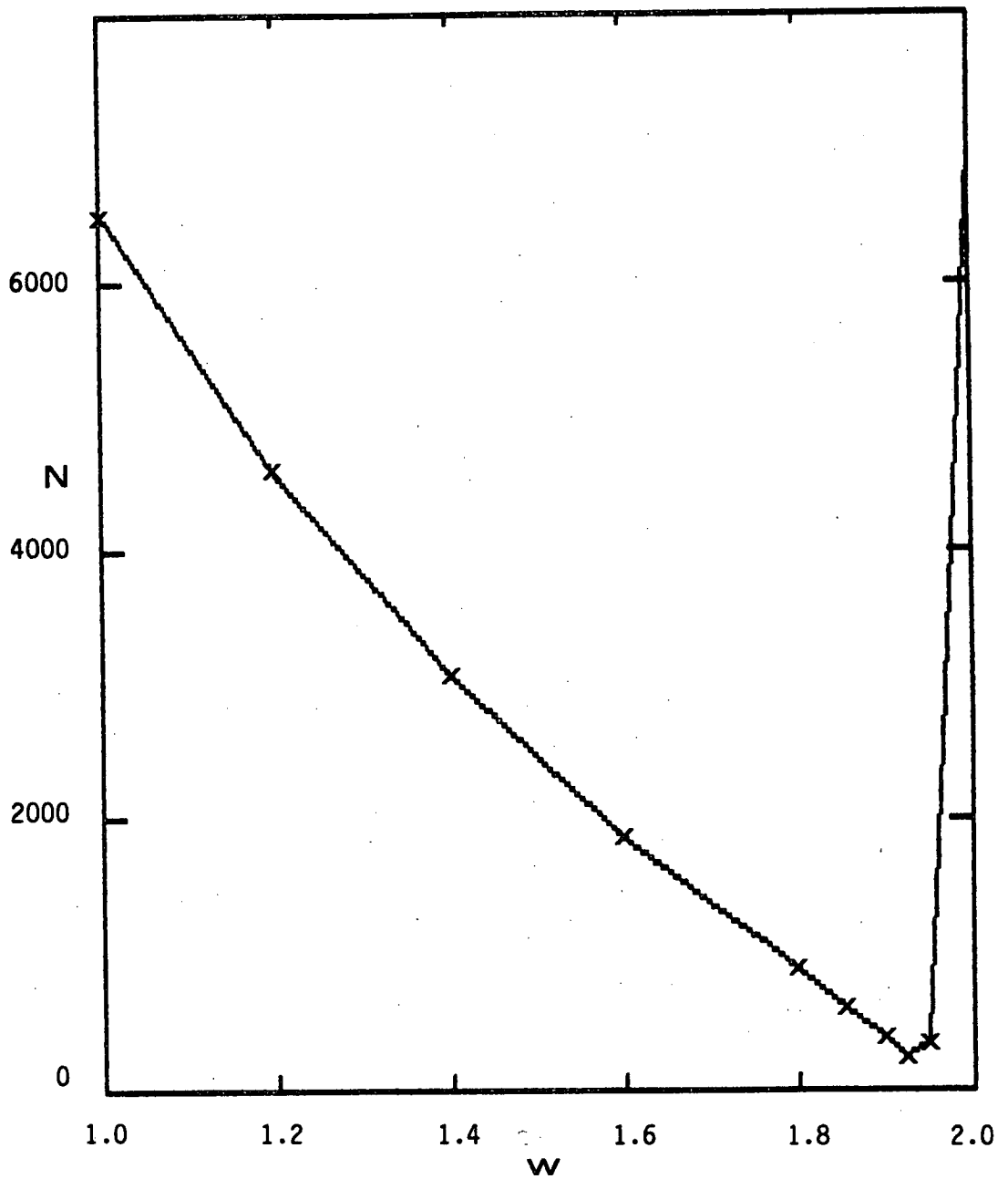


Figure 3.9 Graph of the number of iterations, N, taken to solve the equations for the torso model versus acceleration factor, w. The number of iterations approaches infinity as the acceleration factor approaches 2.0.

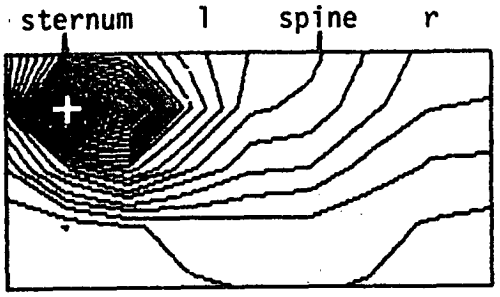
iterations were required. The rate of solution was about 7 iterations per minute, giving a solution time of approximately 35 minutes at the optimum acceleration factor.

The model was then solved 25 times to produce the surface potential distributions required to construct the forward transfer matrix. A simulated wrapping of the electrode jacket around the torso model located the 45 torso surface points at which the torso surface potentials were specified. Figure 3.10 shows the torso surface potential distributions resulting from each source region. Note that for regions 6 to 25, the position of maximum potential on the torso surface corresponds well with the position of the source region around the heart. Source regions 1 to 5 all produce a maximum located near the sternum. The forward transfer matrix obtained from these distributions is listed in table 3.2.

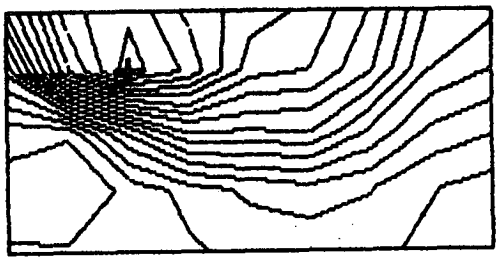
A useful check on the convergence of the solutions can be obtained by noticing that adding the solutions for all the source regions is equivalent to placing a potential of +100V on the entire heart surface. The exact solution in this situation is a uniform potential of +100V throughout the body*. When the 25 source solutions were added for this torso, and it was found that the maximum difference from 100v at any node in the model was less than $5*10^{-4}$. It is thus likely that the individual solutions of

---

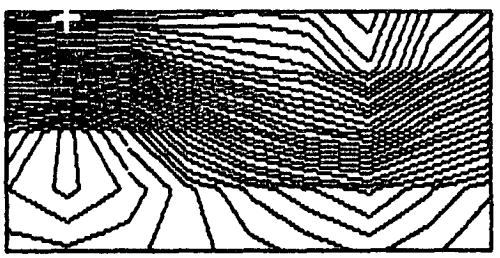
* Physically speaking, it is nonsense to specify the potential of an object without reference to some other location. The point here is that there will be no potential differences throughout the torso, and, numerically speaking, the solutions should add to +100V at all nodes.



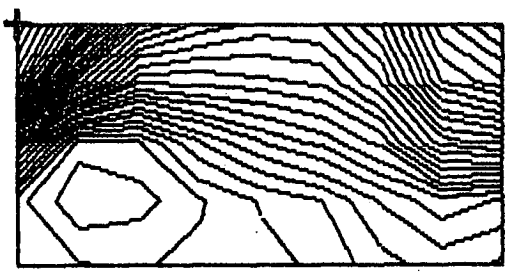
region 1 interval 0.5
Max 16.5 Min 0.49



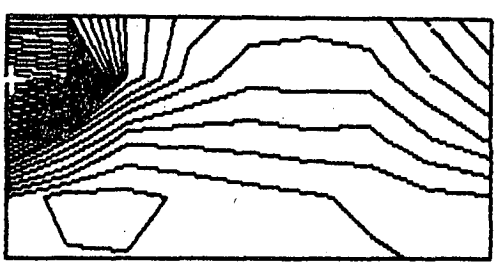
region 2 interval 0.5
Max 8.8 Min 0.65



region 3 interval 0.5
Max 18.2 Min 1.8

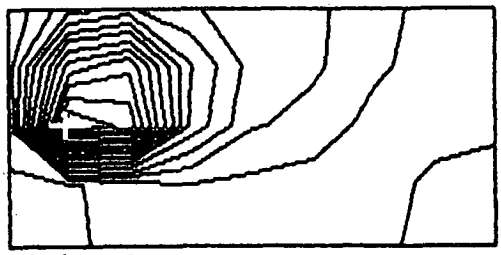


region 4 interval 0.5
Max 19.0 Min 1.6

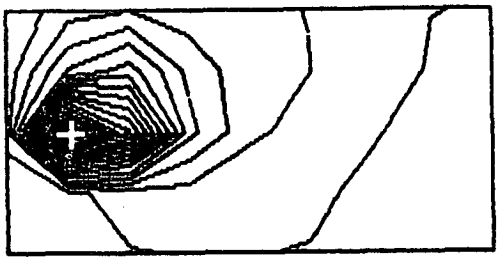


region 5 interval 0.5
Max 14.9 Min 0.77

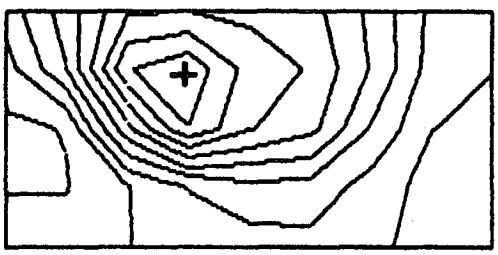
Figure 3.10 (3 pages). Contour plots of the body surface potentials generated by each of the source regions on the surface of the heart. These body surface maps are based only on the 45 locations used to produce the forward transfer matrix. They do not have rows at the top and bottom representing the neck or reference value. They also do not wrap horizontally. Positioning of the maps around the body is shown on the first map. The '+' signs mark the position of the maximum potential on each map.



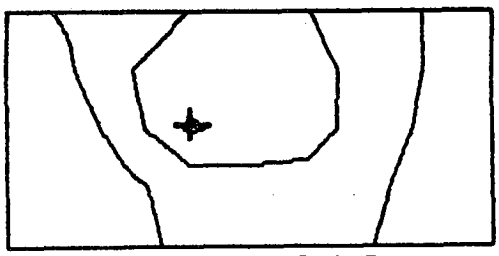
region 6 interval 0.5
Max 7.5 Min 0.36



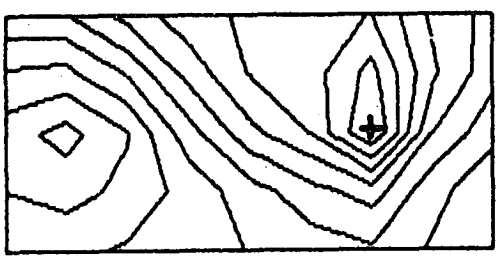
region 11 interval 0.5
Max 8.9 Min 0.29



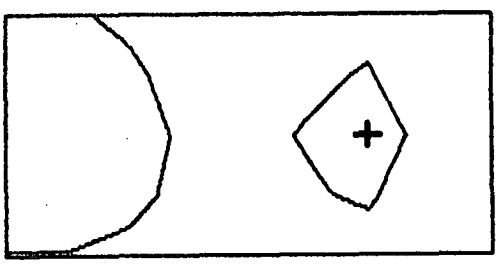
region 7 interval 0.5
Max 4.9 Min 0.38



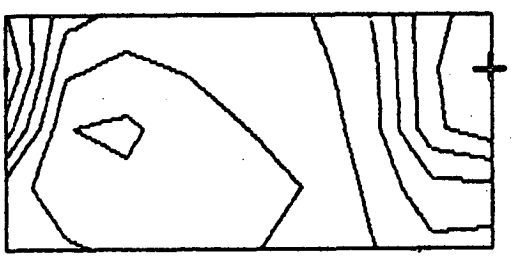
region 12 interval 0.5
Max 1.6 Min 0.13



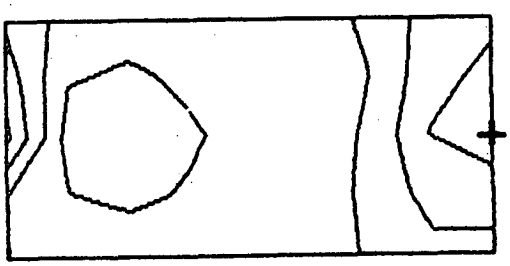
region 8 interval 0.5
Max 5.6 Min 0.32



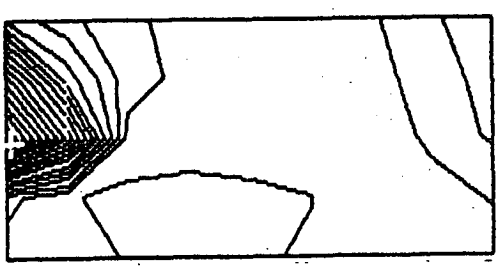
region 13 interval 0.5
Max 1.3 Min 0.08



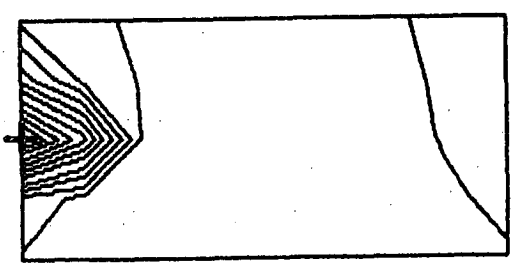
region 9 interval 0.5
Max 4.0 Min 0.40



region 14 interval 0.5
Max 2.3 Min 0.24

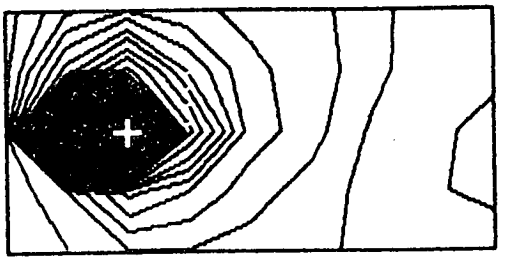


region 10 interval 0.5
Max 9.2 Min 0.37

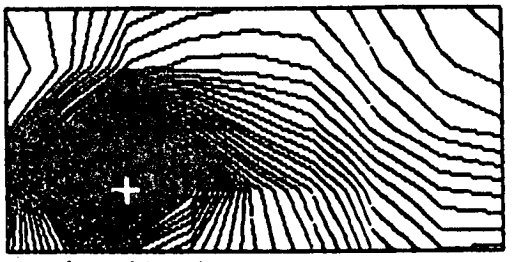


region 15 interval 0.5
Max 5.8 Min 0.22

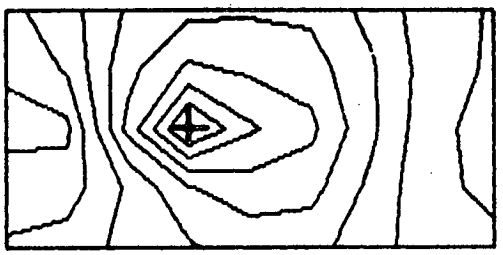
Figure 3.10 continued.



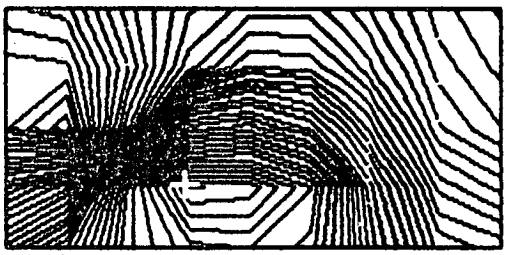
region 16 interval 0.5
Max 18.8 Min 0.96



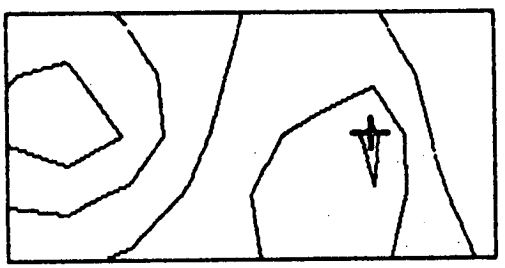
region 21 interval 0.5
Max 33.0 Min 3.5



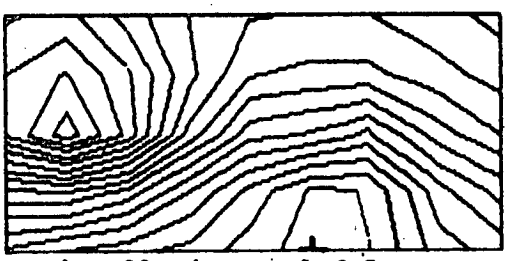
region 17 interval 0.5
Max 4.6 Min 0.36



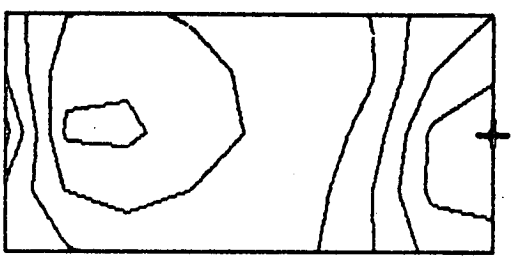
region 22 interval 0.5
Max 21.6 Min 1.6



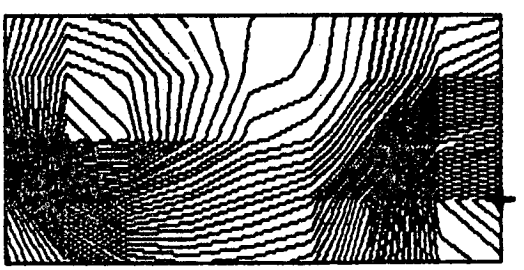
region 18 interval 0.5
Max 2.6 Min 0.16



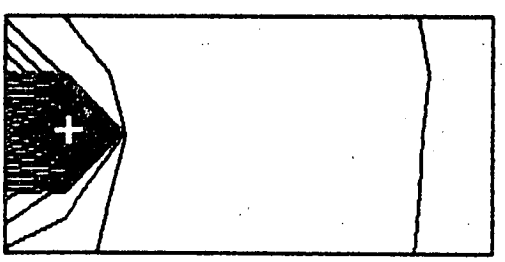
region 23 interval 0.5
Max 9.3 Min 0.71



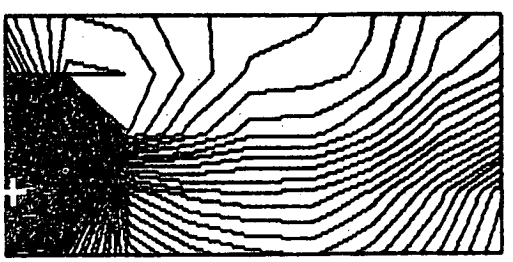
region 19 interval 0.5
Max 3.5 Min 0.38



region 24 interval 0.5
Max 29.2 Min 2.4



region 20 interval 0.5
Max 12.9 Min 0.63



region 25 interval 0.5
Max 31.5 Min 3.9

Figure 3.10 continued.

TABLE 3.2 (2 pages)

Forward Transfer Matrix

Columns 1 - 12

36.1	49.4	171.6	190.3	85.9	10.9	12.1	23.8	30.8	23.6	6.2	3.5
71.8	78.4	182.0	150.7	86.2	23.1	22.4	24.4	17.2	16.9	12.4	5.5
62.9	83.9	174.7	105.0	53.5	25.8	34.3	31.9	13.2	10.6	14.7	8.9
47.2	73.6	169.7	93.3	42.4	20.5	34.7	37.9	13.6	9.0	12.2	10.0
40.0	67.2	165.8	88.3	37.8	17.8	33.8	41.0	14.0	8.4	10.8	10.4
36.7	64.0	168.2	89.3	36.6	16.2	32.0	42.8	14.9	8.3	9.9	10.1
29.4	54.6	179.1	105.6	37.8	11.9	23.8	45.2	20.2	9.3	7.3	7.8
23.1	39.0	150.7	138.0	46.0	8.3	13.7	34.1	33.6	14.2	5.1	4.6
23.4	36.2	139.5	151.9	55.2	8.0	11.3	28.2	37.7	18.3	4.8	3.7
33.3	30.4	96.2	170.2	149.0	12.3	7.6	13.9	36.3	62.4	7.2	2.2
164.8	68.9	104.3	101.6	139.0	52.4	19.8	13.2	11.0	28.1	25.3	4.3
110.5	87.7	111.1	70.2	54.7	58.9	44.4	20.1	8.5	12.4	32.7	9.7
56.0	78.9	133.6	71.4	37.8	30.3	48.9	31.5	10.4	8.4	18.6	13.2
36.2	62.5	143.8	73.5	31.9	17.9	37.8	40.2	12.4	7.4	11.3	12.2
33.6	59.8	148.9	76.0	31.7	16.0	34.6	42.7	13.2	7.5	10.1	11.5
26.5	51.0	169.3	89.6	32.2	11.2	24.7	51.8	18.1	8.1	7.0	8.5
20.3	34.9	139.1	124.7	39.3	7.5	13.2	36.2	34.2	12.5	4.7	4.6
19.4	30.1	117.9	134.0	46.8	6.9	10.1	26.9	40.2	17.1	4.3	3.5
14.4	11.7	36.9	66.4	74.3	8.6	3.8	7.5	27.9	91.8	6.1	1.3
38.8	10.7	17.7	22.1	42.1	75.4	5.2	3.2	5.2	54.6	89.4	1.3
52.8	31.2	38.6	24.4	21.3	68.4	28.5	9.5	4.0	8.3	60.4	8.0
36.4	45.5	73.2	39.7	21.7	28.8	44.8	22.8	7.5	6.1	21.7	15.6
26.4	44.5	96.5	49.8	22.0	15.6	35.3	33.6	10.1	5.9	10.8	13.3
25.1	45.5	112.2	56.8	23.6	13.2	31.7	40.3	11.5	6.2	8.8	12.1
17.9	34.5	121.0	66.9	23.2	8.1	19.4	55.8	17.8	6.9	5.4	7.8
14.5	24.1	95.5	93.5	29.6	5.8	10.1	30.4	33.9	11.1	3.8	3.8
13.9	20.8	80.1	97.0	35.5	5.5	8.0	21.6	38.5	15.9	3.6	3.0
5.9	8.0	25.4	26.8	14.1	3.6	4.7	10.0	12.6	11.8	2.9	2.1
4.9	6.5	18.4	15.9	7.7	4.0	4.7	8.1	6.7	5.4	3.8	2.1
9.1	11.1	25.8	17.8	8.2	8.3	9.7	11.4	6.0	3.7	7.5	4.2
12.2	17.0	38.6	24.4	10.9	9.6	15.4	16.9	7.5	4.3	7.9	6.8
13.3	21.2	51.1	30.5	12.9	8.8	18.3	22.9	8.9	4.7	6.7	8.3
13.2	22.7	59.7	34.8	14.1	7.8	18.1	28.3	10.2	5.0	5.8	8.3
11.5	20.3	66.6	45.7	16.3	5.9	13.1	32.7	16.2	6.3	4.2	5.8
9.5	15.4	55.7	53.7	18.6	4.6	8.1	21.8	23.8	8.5	3.2	3.4
8.5	12.9	45.6	49.5	19.3	4.2	6.6	16.6	23.6	10.5	3.0	2.8
6.9	10.4	32.4	29.5	12.5	4.1	6.8	13.9	13.2	7.5	3.2	3.1
7.0	10.4	30.7	25.3	10.6	4.6	7.5	13.8	10.6	5.9	3.7	3.4
8.3	12.4	33.5	24.5	10.2	5.9	9.9	15.6	9.2	4.8	4.8	4.5
9.5	14.4	37.9	26.2	10.9	6.5	11.8	17.7	9.4	4.8	5.2	5.4
10.2	16.2	42.8	28.6	11.7	6.6	13.1	20.3	9.9	4.9	5.1	6.1
10.3	17.0	46.6	30.8	12.3	6.4	13.4	22.7	10.6	5.1	4.9	6.3
9.7	16.4	51.3	38.2	14.2	5.4	11.1	24.4	14.7	6.1	4.0	5.1
8.5	13.6	46.1	42.3	15.5	4.5	8.1	19.4	18.8	7.5	3.3	3.5
7.9	12.2	40.8	39.4	15.3	4.2	7.2	16.7	18.1	8.2	3.2	3.2

TABLE 3.2 continued

Forward Transfer Matrix

Columns 13 - 25

4.3	14.4	9.3	12.7	8.5	7.4	17.5	13.7	40.6	43.6	25.9	95.5	62.7
4.2	8.1	6.2	23.7	12.9	7.4	10.2	9.7	47.0	48.1	23.1	63.3	44.8
6.0	6.7	4.1	29.6	20.6	10.7	9.3	7.4	65.0	72.7	32.1	69.4	47.1
7.5	7.2	3.7	26.0	23.4	13.6	10.4	6.9	72.3	88.9	40.1	82.3	53.7
8.3	7.5	3.5	23.9	24.4	15.2	11.1	6.8	75.7	97.0	44.4	89.6	57.4
8.6	8.0	3.5	22.2	23.7	15.7	11.8	6.8	74.9	97.2	45.6	93.9	59.1
8.7	10.7	4.0	16.8	18.6	15.4	15.2	7.5	66.3	85.6	45.2	110.3	63.8
7.0	17.1	6.2	11.9	11.7	12.3	23.1	10.7	57.0	67.2	41.9	145.6	78.0
5.9	18.9	7.9	11.1	9.7	10.5	24.9	12.9	52.7	59.2	38.2	148.1	81.8
2.8	17.9	23.1	13.6	5.6	5.2	19.7	30.6	35.2	32.3	20.0	87.3	85.7
2.4	5.4	8.9	42.3	10.5	4.4	6.6	13.8	47.1	33.5	14.4	39.2	38.7
4.1	4.4	4.6	62.0	23.5	7.5	6.2	8.6	81.2	64.6	23.8	48.7	40.0
6.7	5.6	3.4	39.5	30.8	12.5	8.4	6.7	87.5	98.0	38.3	72.1	51.1
8.8	6.9	3.2	25.9	29.1	16.6	10.6	6.5	87.4	115.0	49.7	92.1	61.0
9.3	7.4	3.2	23.4	27.5	17.4	11.2	6.5	84.8	113.8	51.1	96.2	62.4
10.3	9.9	3.6	16.7	20.6	18.3	14.7	7.0	72.2	96.9	51.4	114.0	66.3
7.8	18.0	5.6	11.5	11.9	13.6	25.0	10.1	61.1	72.7	46.7	161.9	82.8
6.1	21.2	7.7	10.4	9.4	11.1	29.0	13.2	57.4	63.6	42.9	177.4	93.2
2.0	21.3	58.3	12.9	3.8	4.2	26.9	97.3	42.5	30.7	20.2	103.4	225.8
0.8	3.3	36.6	159.9	3.6	1.6	4.2	129.0	93.9	16.3	7.1	24.2	153.8
2.5	2.4	3.9	187.9	26.6	5.4	3.8	9.7	219.5	78.0	21.0	38.7	45.1
6.1	4.5	2.7	57.1	46.4	13.1	7.6	6.3	149.4	155.6	46.9	78.6	62.1
8.9	6.1	2.7	27.8	35.8	18.9	10.2	6.3	118.0	162.5	62.7	103.1	73.1
10.3	6.9	2.9	22.4	31.3	21.0	11.3	6.4	103.7	149.0	65.4	109.2	73.4
13.4	10.7	3.3	14.2	20.9	25.9	17.2	7.1	84.6	119.4	70.9	145.8	82.0
7.9	20.3	5.3	10.3	10.9	14.7	31.1	10.5	70.9	81.0	57.2	220.2	103.7
5.8	23.2	7.8	9.6	8.7	11.5	35.4	14.5	66.2	70.0	49.9	234.8	119.1
3.6	9.9	8.2	10.7	7.2	9.2	19.1	27.0	109.1	80.2	51.8	221.4	314.8
3.0	5.1	3.7	20.0	7.6	8.3	10.0	18.0	239.4	98.8	50.1	140.9	307.1
4.1	4.4	2.2	36.5	15.8	11.2	8.3	7.4	330.4	176.1	56.8	108.1	115.8
5.9	5.3	2.3	30.0	24.7	15.5	9.9	6.7	218.5	215.5	69.9	121.6	102.6
7.9	6.1	2.5	22.0	28.1	20.0	11.3	6.7	158.9	213.5	81.5	133.2	100.4
9.6	6.9	2.6	17.9	27.0	24.2	12.7	6.8	134.0	195.2	90.8	143.6	100.7
10.5	11.1	3.3	13.1	18.6	24.9	20.0	8.0	106.6	143.4	89.8	195.4	110.5
7.1	16.9	4.5	10.2	10.8	15.7	31.0	10.5	89.9	98.4	71.6	275.1	132.0
5.5	17.4	5.9	9.6	9.1	12.7	32.9	13.7	87.7	88.5	62.9	292.3	158.9
5.1	10.1	4.6	12.1	10.7	13.3	19.8	13.9	130.8	116.2	71.7	239.1	209.0
5.2	8.0	3.6	15.0	12.3	14.0	15.6	11.5	170.8	139.9	75.7	200.8	194.2
5.8	6.8	2.8	19.4	16.4	15.7	13.1	8.6	200.7	178.5	78.8	165.5	143.9
6.5	6.8	2.7	19.7	19.5	17.4	13.0	7.9	184.0	193.3	82.1	159.6	127.8
7.4	7.1	2.7	18.2	21.5	19.6	13.5	7.7	161.3	195.2	87.4	161.6	121.3
8.2	7.5	2.8	16.5	21.7	21.8	14.2	7.7	145.4	188.3	92.6	167.5	119.4
8.5	10.6	3.3	13.3	17.0	21.5	19.8	8.6	120.3	149.5	90.1	210.0	126.8
6.8	13.9	4.2	11.0	11.8	16.2	26.5	10.5	105.3	113.4	77.6	266.2	145.6
5.9	13.6	4.7	10.8	10.7	14.4	26.4	12.1	106.4	108.0	72.7	272.9	164.9

potential distribution arising from any one source region are at least this accurate. Of course, it is possible that the individual solutions may be highly inaccurate but still chance on adding to 100v at every node, but such behaviour seems highly unlikely. In any case, this check provides some proof that the model is behaving as it should.

Conclusion

A computerised system has been set up which can model electric fields in non-homogeneous volume conductors. The amount of human effort required to set up a model has been minimised wherever possible, so that the system is suitable for the study of multiple torso models (such work is described in chapter five). Approximately two hours of human work are needed to specify a model, consisting mainly of the time taken to digitise the serial slices defining the geometry, plus a small amount of time selecting a suitable grid. This system has been validated by comparison of the modelled and analytical solutions of the potential distributions produced by two point sources located in a conducting sphere. The system has also been used to generate a highly detailed model of the human torso. Calculation of the forward transfer matrix for this model takes approximately 15 hours of processor time on a PDP 11/44 computer.

CHAPTER FOUR

THE INVERSE PROBLEM

Introduction

This chapter describes a solution of the inverse problem of electrocardiography. For a given torso geometry, using procedures such as those described in the previous chapter, a matrix T can be constructed such that the source distribution h and the torso surface potential distribution b are related by the following matrix equation:

$$Th = b \tag{4.1}$$

where h and b are n and m dimensional column vectors respectively (as in equations 3.1 and 3.2). Solving the inverse problem then becomes the problem of solving the system of linear equations represented by equation 4.1 for the components of h in terms of T and b. If a unique solution is to exist, then there must be at least as many equations (or components of b) as there are unknowns (or components of h). If there are more equations than unknowns then the matrix T is not square and hence not invertible. Even if T is square, it may still not be invertible. It is necessary, then, to find a way to solve equation 4.1 when an inverse of the matrix T may not exist.

Unfortunately, given a measured torso potential distribution b, it may well be the case that there does not exist any source

distribution h such that equation 4.1 is satisfied exactly. At first thought this may seem odd, because if b is measured from the surface of a human subject, then there must have been a corresponding source distribution at the heart which gave rise to the measured torso surface potentials. The problem is that the torso surface potentials cannot be measured exactly, so that the measured potential distribution on the surface of the torso will differ to some extent from the potential distribution which actually exists there. Also, it cannot be expected that the forward transfer matrix T will exactly represent the relationship between the source distribution and torso surface potential distribution which exists in reality.

A natural answer to this problem is to look for a solution h which minimises the difference between the calculated torso surface potential distribution, (Th), and the measured distribution, b. It can be shown (Dahlquist and Bjorck 1974) that a vector h which minimises the quantity

$$|Th - b|^2 \tag{4.2}$$

will satisfy the so called 'normal' equations, which are represented by the following matrix equation:

$$T^tTh = T^tb \tag{4.3}$$

where T^t is the transpose of the matrix T.

If the inverse of the matrix (T^tT) exists, then the normal

equations may be solved directly, giving:

$$h = (T^t T)^{-1} T^t b \tag{4.4}$$

Note that if the matrix T itself is non-singular, then this reduces to

$$h = T^{-1} b \tag{4.5}$$

as expected. It can be shown (Dahlquist and Bjorck 1974) that the inverse of $T^t T$ will exist if and only if the columns of T are linearly independent (that is, none is a linear combination of the others). Due to the way that the matrix T is constructed, the j'th column of T is the torso surface potential distribution caused by the j'th source with unit strength. Thus, the normal equations may be solved if and only if the torso surface potential distributions due to each of the sources are linearly independent.

A problem which can be encountered here is that some sources may have corresponding torso surface potential distributions which are very similar - hence 'nearly' linearly dependent. For example, in the torso model described in the previous chapter, the torso surface potential distributions due to source regions 14 and 19 are almost identical in shape (figure 3.10). The effect of this is to make the system of equations ill-conditioned - that is, small changes in b may cause very large changes in the solution for h given by equation 4.4. Looking at it the other way, there may exist source distributions h_1 and h_2 which differ greatly but which have corresponding calculated torso surface distributions (Th_1) and (Th_2)

which are almost identical. This means that any errors in the measured torso potential distribution may cause drastic errors in the solution.

A simple way to deal with the problem of an ill-conditioned system would be to reduce the number of sources (unknowns) in the hope that the corresponding torso surface potential distributions become 'more linearly independent'. This is easily done by using a model with less sources when creating the forward transfer matrix T. The hope here is that the new, smaller, system of normal equations is well-conditioned, so that errors in the measured values of b do not unduly affect the solution h. The obvious disadvantage of this scheme is that, when solving the inverse problem, less unknowns are determined and so less information is obtained about the heart.

It is often possible to use extra information about the nature of the solution to eliminate unreasonable solutions. An ill-conditioned system of equations often leads to solutions which have very large alternating positive and negative values on adjacent sources. (The contributions from these large adjacent values almost cancel at the torso surface). Solutions are desired with much smaller, physically reasonable values. This suggests trying to minimise h as well as minimising the difference $|Th - b|^2$. One way to achieve this is to minimise

$$|Th - b|^2 + \alpha|h|^2 \qquad (4.6)$$

where α is a scalar constant. The vector which minimises this expression is given (Tikhonov and Arsenin 1977) by

$$h = (T^tT + \alpha I)^{-1}T^tb \qquad (4.7)$$

where I is the appropriately sized identity matrix. Spatial derivatives or other suitable functions of h can also be minimised. In these cases, the identity matrix I in equation 4.7 is replaced by a term involving the corresponding matrix operator. These methods are known as regularisation methods and α is the regularisation parameter. The effect of such methods is to replace the original ill-conditioned system by a slightly different system which is well conditioned. As α is increased, the new system differs further from the original system, and the amount of smoothing present in the solution increases. It is thus necessary to choose a value for α which provides adequate smoothing in the solution without perturbing the system of equations too much.

Both Martin et al. (1975) and Barr and Spach (1978) describe theoretical methods for calculating the optimum value of α given appropriate statistical data about the epicardial signals and noise in the input data. Because such data are difficult to obtain on realistic signals and torso models, both groups use simplifying assumptions to find a value for α. Effectively, a value for α is chosen with regard to how much error exists in the measurement of b (strictly, errors in T should also be taken into account). The aim is to obtain reasonable values of h while still satisfying equation 4.1 to within the level of noise contained in b.

In this chapter, the regularisation method has been used to develop an inverse transfer matrix based on the forward transfer

matrix calculated using the model described in the previous chapter. Data obtained using the equipment described in chapter two have been used to calculate epicardial potential distributions.

Methods

The forward transfer matrix used to calculate the inverse transfer matrix was generated from a model with twenty-five source regions on the epicardium. This means that inverse calculations will yield twenty-five epicardial potentials. Twenty-five regions were chosen because several studies (Lux et al. 1979, Barr et al. 1971) have shown that observed body surface potential distributions can be approximated to within reasonable error levels using twenty to thirty-five parameters. This suggests that at most twenty to thirty-five independent parameters describing the electrical activity of the heart can be obtained from an observed body surface distribution.

Inverse matrices were calculated according to the right hand side of equation 4.7. The matrix inversion was performed using an inversion algorithm based on row operations on an augmented matrix (Kolman 1980). Calculation of an inverse transfer matrix of 25 rows by 45 columns, including inverting a 25 by 25 matrix, takes approximately 15 seconds on the PDP 11/44 computer system described in chapter 2.

Epicardial potential distributions are displayed as contour maps in a similar format to that used for body surface distributions. The data for each map consist of the potentials at

the 25 source regions, arranged in five rows, each of five potential values. No additional rows or columns of values are added, so that the epicardial maps do not appear to wrap either vertically or horizontally. Figure 4.1 shows the approximate locations of the ventricles and atria on the map.

Once an epicardial potential distribution, h, has been calculated, the discrepancy between the corresponding calculated torso surface distribution and the original measured distribution is checked. This is done by calculating both the rms difference and the correlation coefficient. For two n-dimensional vectors x and y, these quantities are calculated as follows:

$$\text{rms difference} = \left(\sum_{i=1}^{n}(x_i-y_i)^2/n\right)^{0.5} \tag{4.8}$$

$$\text{correlation cft.} = \frac{x.y}{|x|\ |y|} \tag{4.9}$$

When calculating epicardial potentials, it is obviously preferable to use data measured from the patient whose CT scans were used to generate the modelled torso geometry. A full torso CT scan requires a high radiation dose and it is thus necessary to obtain body surface potential data from patients who have had a full torso CT scan for some other reason. This has not been possible at the time of writing, as the number of patients with normal torso geometry (no gross abnormalities) who undergo a full torso CT scan is relatively small. As a result, the input data used when calculating epicardial potentials were measured from a normal 25

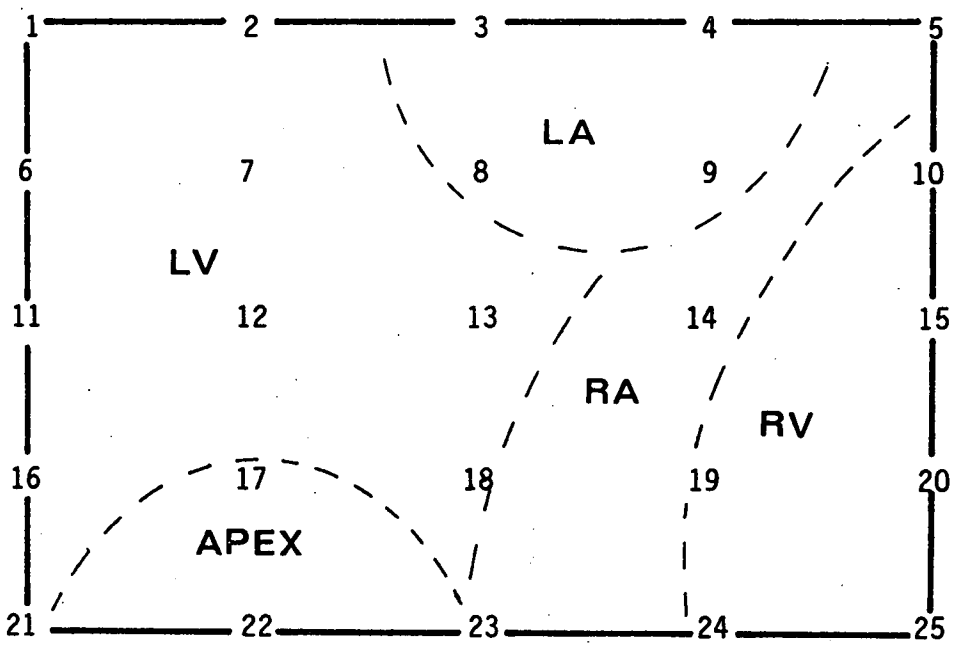


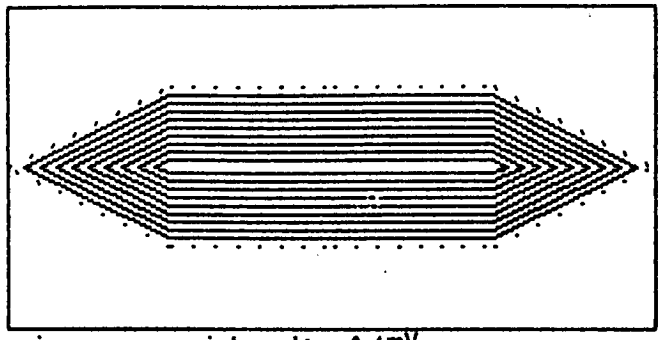
Figure 4.1 Approximate locations of the ventricles (LV and RV), atria (LA and RA) and apex of the heart on an epicardial map. Numbers show the locations of the source regions on the map. For the purposes of drawing epicardial maps the source regions are considered as single points.

year old male with external torso dimensions similar to the subject whose CT scans were used in the previous chapter. All input data were referenced to an artificial 'central terminal' calculated by averaging all torso leads.

Preliminary Calculations

Some initial calculations were performed to check the correctness of the method and to explore the regularisation method. A simple (non-physiological) source distribution was set up on the epicardium by setting regions 12, 13 and 14 to 1mV and all other regions to 0mV (Figure 4.2a). The corresponding torso surface potential distribution was calculated by equation 4.1 and is shown in figure 4.2b. Inverse epicardial distributions were then calculated from this torso distribution using equation 4.7 for four values of α (α = 0, 10^{-6}, 10^{-5} and 10^{-4}). These are shown in figures 4.2c,d,e and f. For α=0, the inverse matrix becomes $(T^tT)^{-1}T^t$, and so applying the forward transform T followed by this inverse should leave the epicardial distribution unchanged (as $(T^tT)^{-1}T^tT$ = I, the identity matrix). Figure 4.2c shows that this is indeed the case, confirming that the operations used in calculating the inverse matrices are accurate.

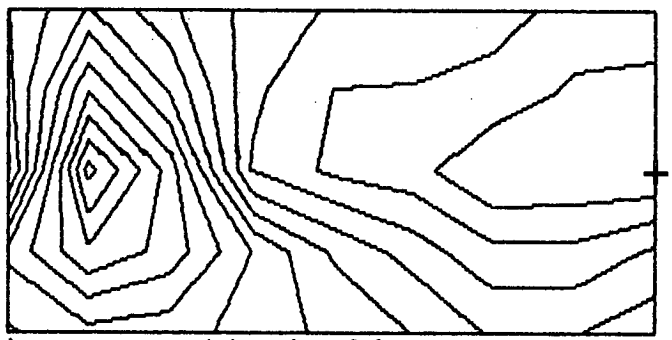
As α is increased, the resultant epicardial potential distributions become smaller in magnitude and spread over more of the epicardium. It is interesting to calculate the torso surface distributions corresponding to these epicardial distributions. These are shown in Figure 4.3. They are all almost identical to the



(a)

epi interval: 0.1mV

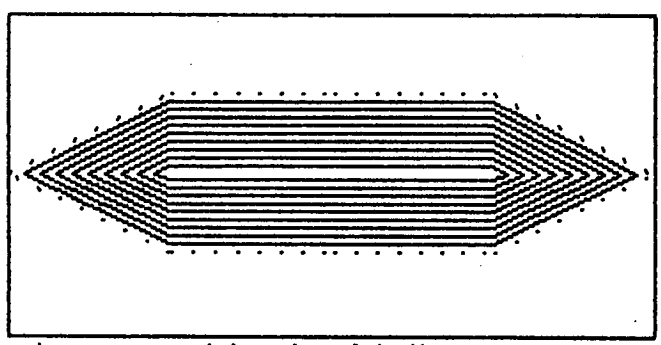
RMS: 0.3
Max 1.0
Min 0.0



(b)

bsm interval: 2.0μV

RMS: 23.9
Max 32.0
Min 5.3

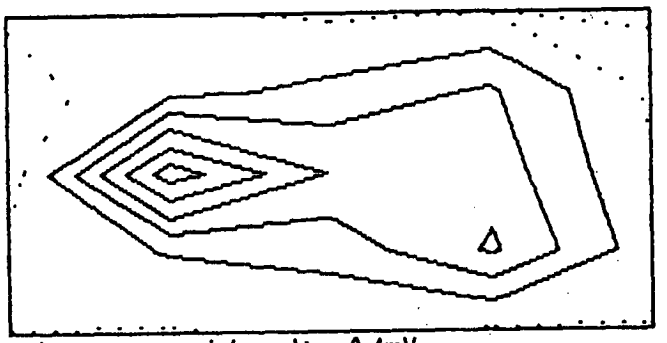


(c)

epi interval: 0.1 mV

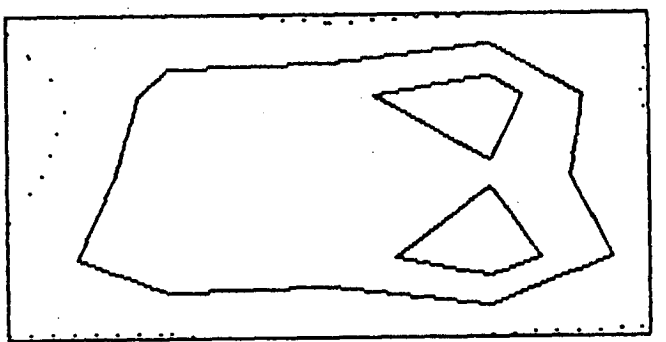
RMS: 0.3
Max 1.0
Min -0.0

Figure 4.2 Epicardial and body surface maps for a simple epicardial source (see also next page).
(a). Initial epicardial potential distribution.
(b). Body surface map corresponding to (a).
(c). epicardial map calculated from (b) with regularisation parameter $\alpha = 0$.



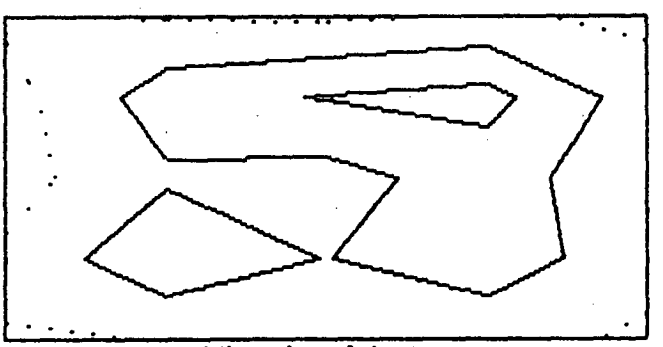
(d)

RMS: 0.2
Max 0.6
Min -0.1



(e)

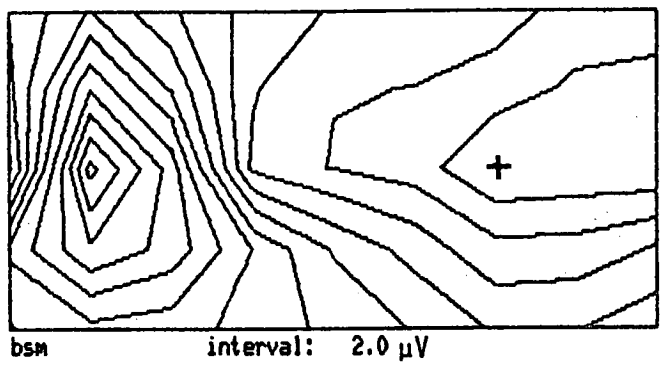
RMS: 0.1
Max 0.3
Min -0.1



(f)

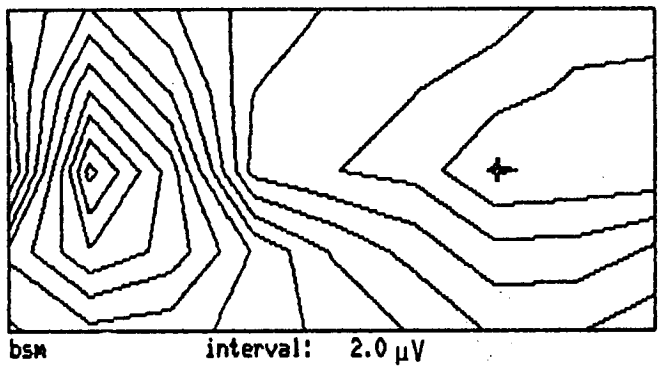
RMS: 0.1
Max 0.2
Min -0.0

Figure 4.2 continued.
(d), (e) and (f). Epicardial maps calculated from (b) with $\alpha = 10^{-6}$, 10^{-5} and 10^{-4} respectively.



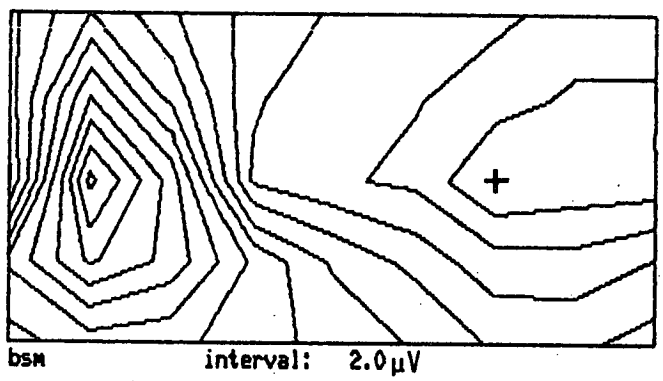
(a)

RMS: 23.9
Max 32.1
Min 5.3



(b)

RMS: 23.9
Max 32.2
Min 5.3



(c)

RMS: 23.9
Max 32.0
Min 5.4

Figure 4.3 Body surface maps which have been calculated from the epicardial maps shown in figure 4.2. Maps (a), (b) and (c) have been calculated from figure 4.2 (d), (e) and (f) respectively.

original torso surface distribution shown in figure 4.2b. The largest difference is for $\alpha=10^{-4}$ with rms difference 0.3μV and correlation coefficient 0.9999. This example demonstrates the behaviour described in the introduction: there can exist widely differing epicardial potential distributions with corresponding torso surface distributions which are almost identical. It also shows that the value of α must be chosen with some care. Here, $\alpha=0$ gives the correct epicardial distribution. It should be remembered, however, that this is a discontinuous, non-physiological epicardial distribution which is what the regularisation method is designed to eliminate. Because all the calculations have been performed in double precision, there is essentially no noise present in the torso surface data used to calculate the inverse epicardial distribution.

It is important to know what effect noise in the torso surface data will have on the calculated epicardial distributions. This has been determined by using noise (limited in amplitude to 200μV peak to peak) as input data and applying the inverse transform for various values of α. For each value of α, the noise amplification factor was calculated as the resultant rms epicardial noise divided by the rms input noise (where the rms is taken over all epicardial regions or all torso surface locations respectively). This factor was calculated one hundred times for each value of α. It is not sufficient to calculate this factor once only, due to the random nature of the input data. The average value and standard deviation of the noise amplification factor is shown for each value of α in table 4.1. When $\alpha = 0$, the amplification factor is very large, so that even small errors in the input data will cause large errors in

TABLE 4.1

Noise Amplification Factor

α	Amplification factor	Standard deviation
0	1907	945
10^{-6}	206	57.0
10^{-5}	63.3	15.7
10^{-4}	22.0	4.74
$2*10^{-4}$	16.3	3.43
10^{-3}	7.97	1.47

The noise amplification factor for an inverse transfer matrix based on a particular value of the regularisation parameter α is calculated by dividing the rms epicardial noise by the rms input body surface noise. The values above are averages and standard deviations obtained by performing the calculation 100 times using random input data.

the calculated epicardial potentials.

When calculating epicardial potential distributions from measured torso data, the 'correct' epicardial distribution is not known in advance. In order to obtain a reasonable value for the regularisation parameter α, epicardial potential distributions were calculated for various values of α. The input data correspond to the instant located 40mS after the QRS onset and are shown in figure 4.4. Figure 4.5 shows the calculated epicardial potential distributions and the resultant torso surface potential distributions. Choosing α=0 here clearly gives unreasonable results. There is a maximum of +495mV located on the right front of the heart, and a minimum of -350mV immediately above the maximum. These values are much higher than the voltages which are found on the epicardium in reality (for example see Barbato et al. 1958).

Increasing α leads to smoother and lower level epicardial distributions, but also a decrease in correspondence between the input and calculated torso surface distributions. It is hoped that up to a certain value of α (which will be the optimum value), the main effect will be to reduce unwanted oscillations in the epicardial potential distribution, without affecting its capability to represent the 'real' epicardial distribution. Beyond this value, the epicardial distribution is smoothed too much, so that it cannot closely represent the 'real' distribution. This is illustrated by plotting the rms difference and correlation coefficients for various values of log α in figure 4.6. Both curves show a distinct 'knee'. Before (to the left of) this knee, both the rms difference and

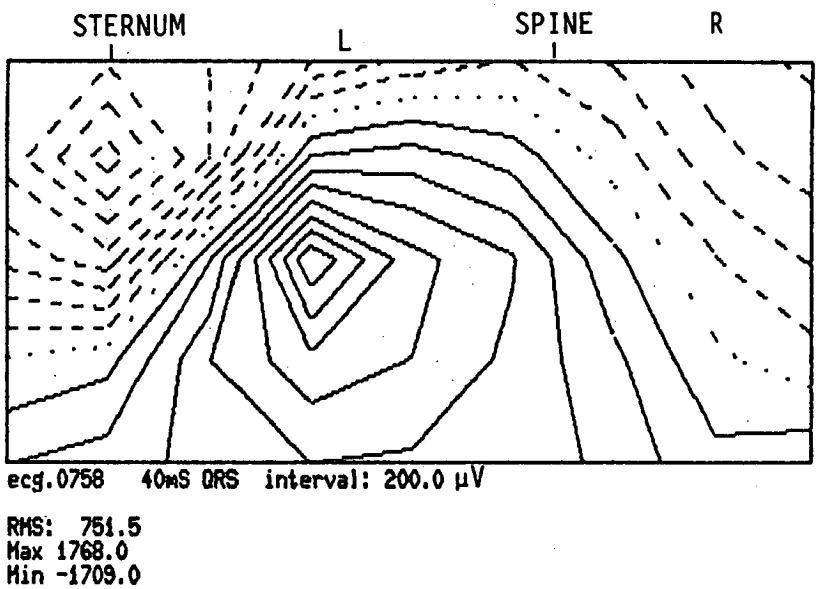
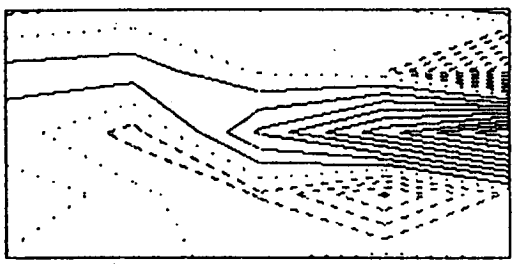
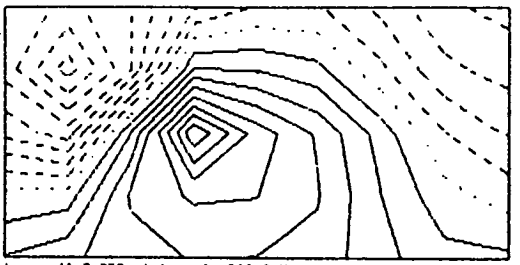
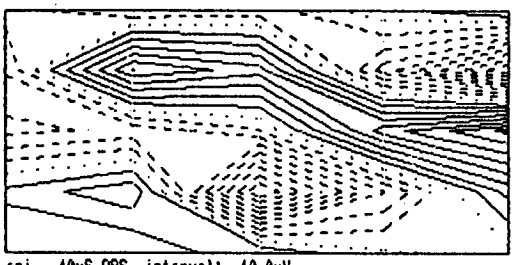


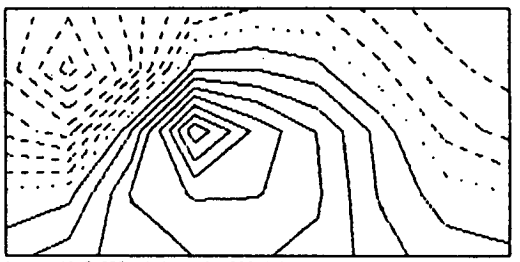
Figure 4.4 Contour plot of the data used to calculate the epicardial maps shown in figure 4.5. This map was recorded from a normal 25 year old male, at 40mS from the onset of the QRS complex. This map is in the same format as the maps shown in figure 3.10 (no rows of neck or reference potentials, and no horizontal wrap around).



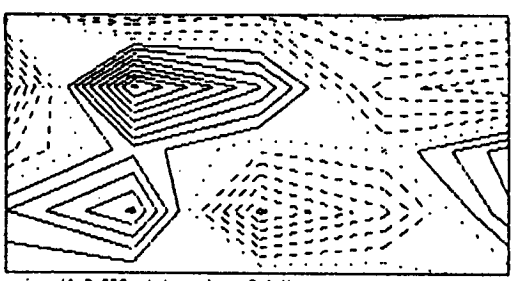


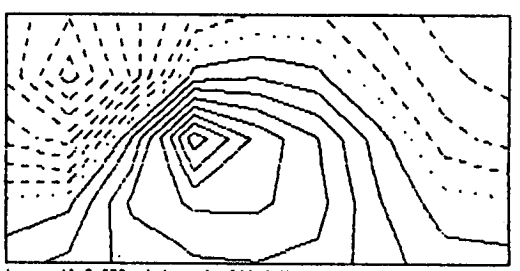
$\alpha = 0$ C 0.997 R 56.0





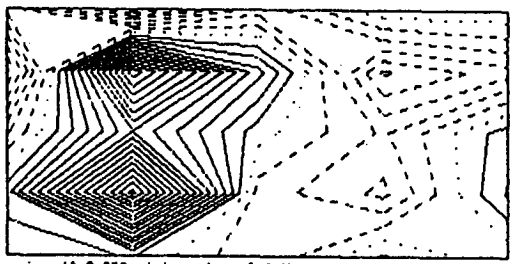
$\alpha = 10^{-6}$ C 0.996 R 68.3



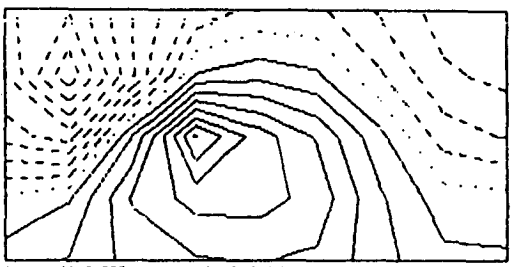


$\alpha = 10^{-5}$ C 0.994 R 81.0

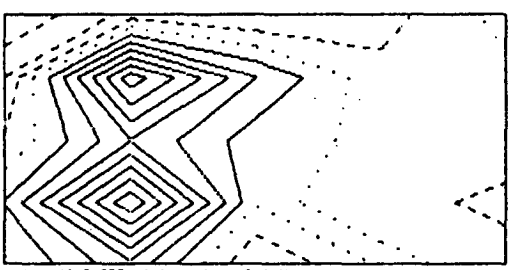
Figure 4.5 (2 pages). Epicardial maps and corresponding body surface maps calculated using various values of the regularisation parameter α. Note that the contour interval varies between epicardial maps. The values 'C' are the correlation coefficients between the input map (figure 4.4) and the resultant body surface maps. The values 'R' are the corresponding rms differences expressed is microvolts.



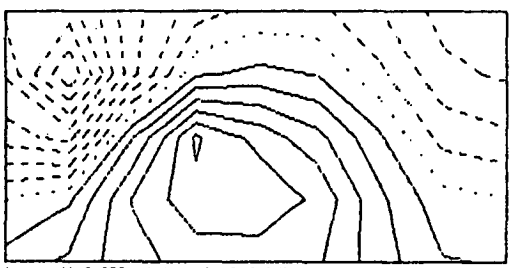
$\alpha = 10^{-4}$



C 0.991 R 101.3



$\alpha = 10^{-3}$



C 0.979 R 155.0

Figure 4.5 continued.

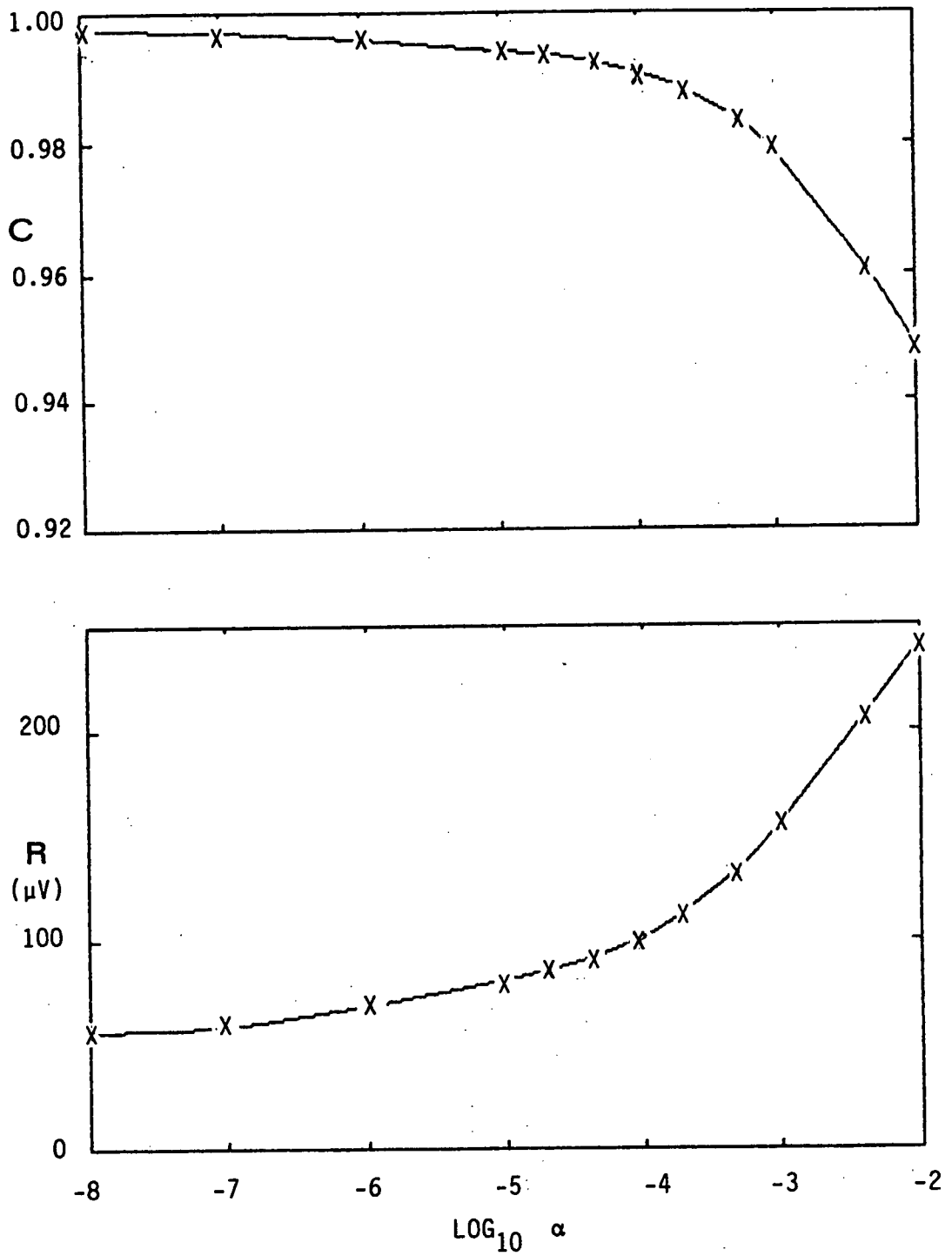


Figure 4.6 Correlation coefficient 'C' and RMS difference 'R' plotted for various values of the regularisation parameter α. The horizontal axis is logarithmic and is the same for both graphs. Points where C and R were calculated are marked with an 'X'.

correlation coefficient are relatively independent of α, while after the knee the rms difference increases, and the correlation coefficient decreases, with increasing α. This behaviour suggests that the optimum value of α is likely to be a value occurring in the neighbourhood of the knee.

When choosing the optimum value for α, a value should be chosen so that the rms difference remains within the noise level expected in the input data. The value should also be large enough so that unwanted features in the calculated epicardial potential distribution are removed. A reasonable estimate of the total noise in the input data is 100μV (this figure is also used by Barr and Spach 1978). Consideration of the graphs plotted in figure 4.6 then leads to a value of approximately 10^{-4} for α. Re-plotting these graphs using torso surface data measured at various times throughout the cardiac cycle (Figure 4.7) shows that the optimum value of α varies throughout the cardiac cycle. During the QRS complex (top graphs in figure 4.7), the optimum value of α ranges from approximately 10^{-4} (at 40mS from QRS onset) to approximately 10^{-3} (at 30mS from QRS onset). In the T wave (bottom graphs in figure 4.7), the value is more consistent at approximately $8*10^{-4}$. In the S-T segment (middle graphs in figure 4.7), the value is harder to specify because of the gentle change in the curves. It should be noted here, however, that the magnitude of the input data here is very low, approximately 65μV rms over all leads, and the rms difference is less than 25μV.

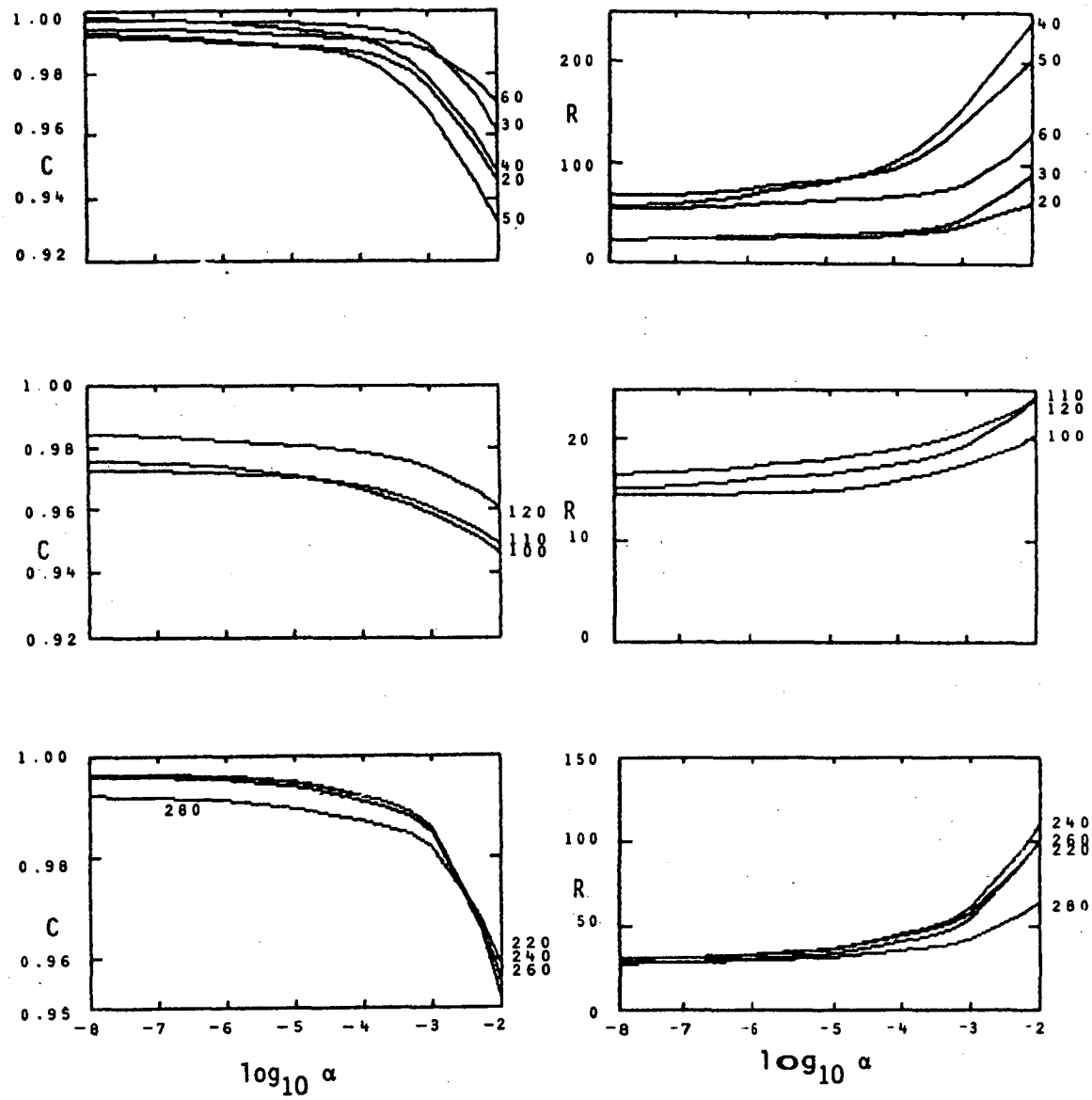


Figure 4.7 Graphs of correlation coefficient 'C' and RMS difference 'R' versus log α. The horizontal axis is the same for all graphs. Graphs have been plotted using input data measured at various instants throughout the cardiac cycle. The numbers at the right hand end of each curve give the time from the onset of the QRS complex in milliseconds. C and R were calculated at the same points as shown in figure 4.6.

Epicardial Potential Distributions

A series of epicardial potential distributions has been calculated at various instants throughout the cardiac cycle. These are shown in Figure 4.8 beside the corresponding torso surface input data. For all of these calculations, α was given the value of 10^{-4}. The rms differences and correlation coefficients between the input data and corresponding calculated torso surface data are shown under each pair of maps.

This sequence shows a positive area which starts near the top front of the left ventricle, proceeds downwards and reaches a maximum near the apex at about 45mS from the onset of the QRS complex. This area them moves upwards and towards the posterior of the heart and dies away at about 65-70mS. A negative area develops on the upper left ventricle during 35-50mS and a lower level negative area is located on the right ventricle to right atrium region during the same interval. Very little activity is seen during the S-T segment. During the T wave, a relatively stationary positive area develops on the left ventricle, reaching a peak at 240mS. A negative area is located higher and more towards the back of the heart.

Discussion

It would be of great value to be able to directly validate the results shown above. This would require measurements of epicardial potential distributions in a human subject using chronically implanted electrodes to minimise departures from a normal torso

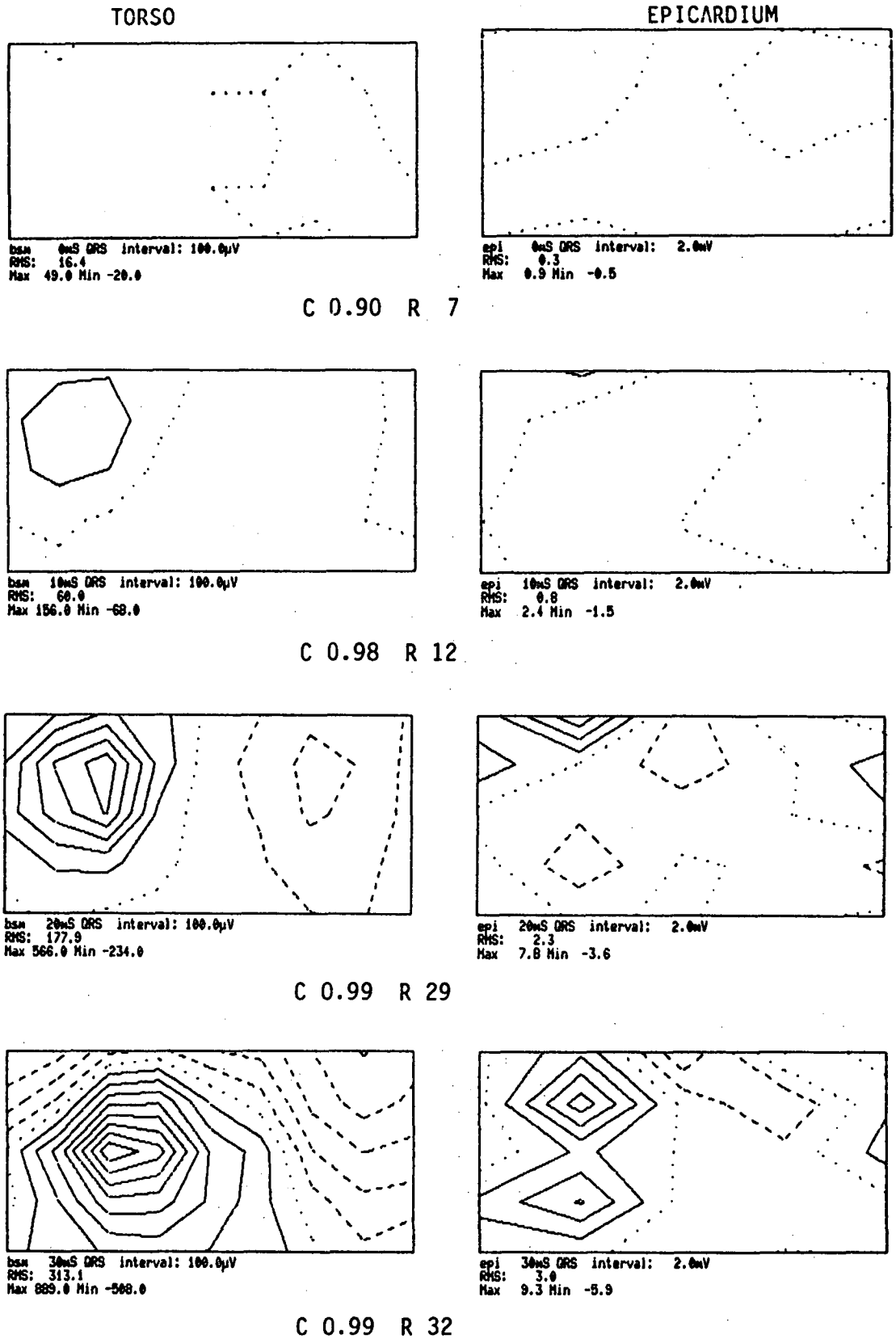
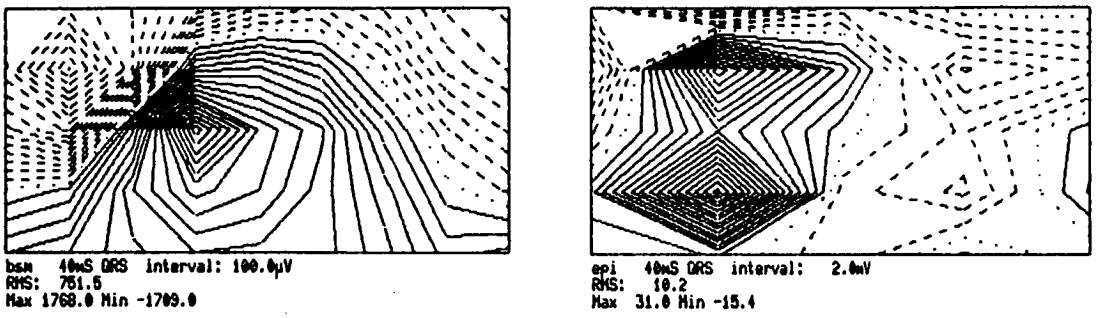
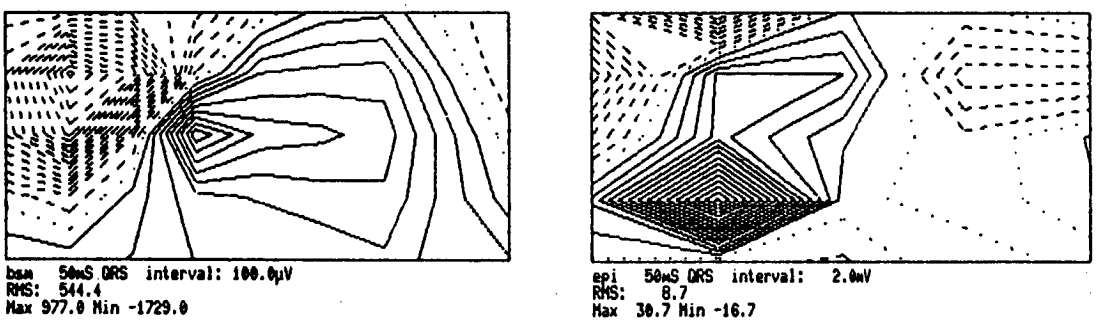


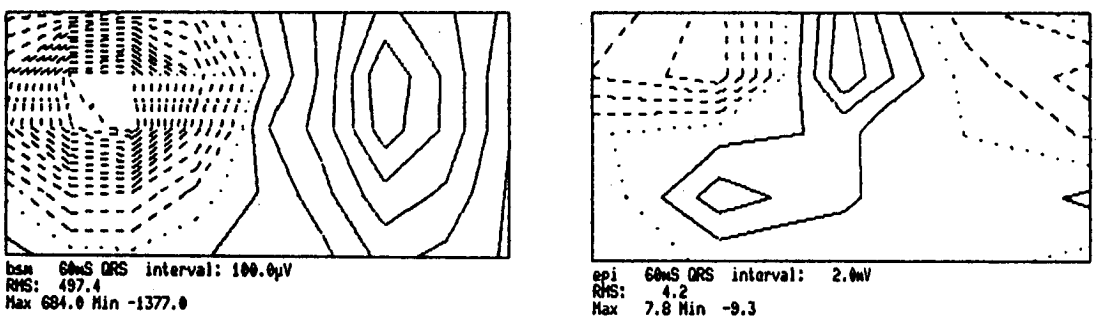
Figure 4.8 (3 pages). Contour plots of measured torso surface potentials and corresponding calculated epicardial potentials at 0, 10, 20, 30, 40, 50, 60, 70, 80, 160 and 240 milliseconds from the onset of the QRS complex. Epicardial maps are on the right. C is the correlation coefficient between the resultant torso surface potentials (not shown) and the measured data. R is the corresponding RMS difference, expressed in microvolts.



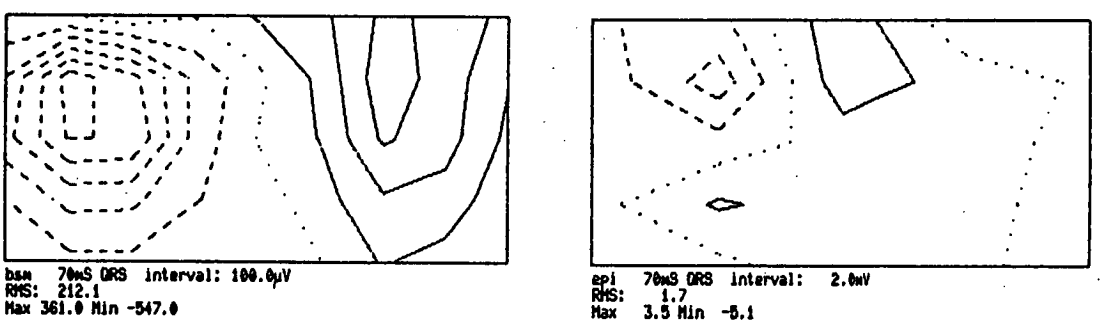
C 0.99 R 101



C 0.98 R 95

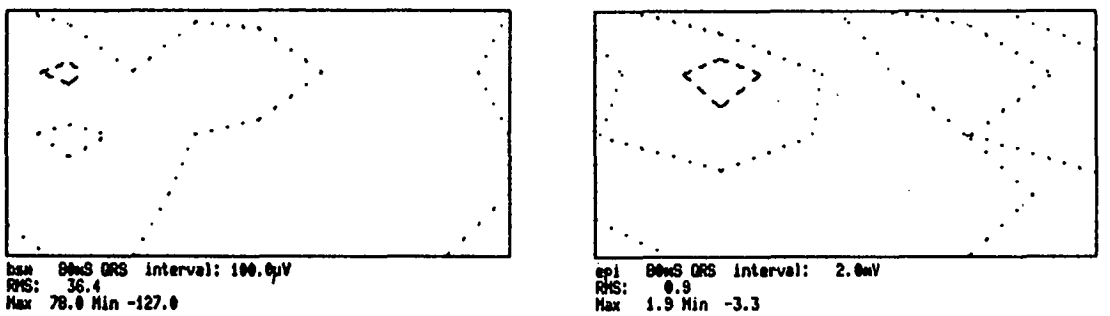


C 0.99 R 67

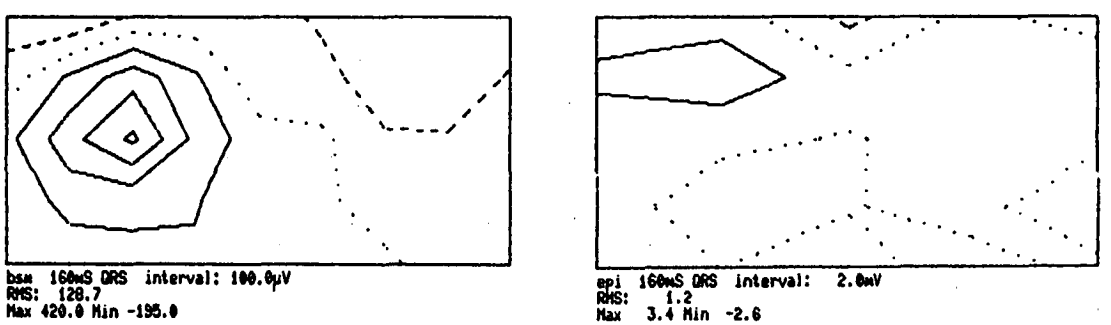


C 0.99 R 36

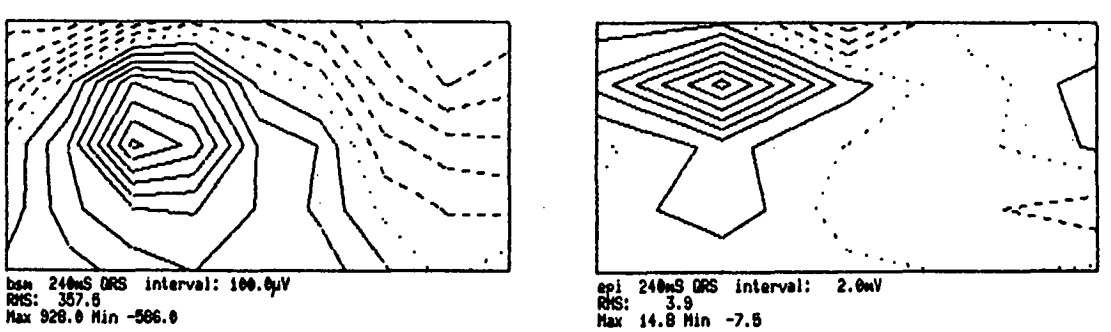
Figure 4.8 continued.



C 0.84 R 19



C 0.99 R 21



C 0.99 R 45

Figure 4.8 continued.

geometry. To the author's knowledge, this study has never been done - there are obvious practical and ethical problems. However, is is possible to measure human epicardial potentials during surgery (Barbato et al. 1958, Monro and Bourdillon 1978), or using an isolated live human heart (Durrer et al. 1970). Comparison with these studies shows that, as far as major features are concerned, the epicardial potential distributions calculated above are consistent with the electrical activity of a normal human heart.

Validation of calculated epicardial potentials has been performed by Barr and Spach (1978) using a dog model. In this study, potential measurements were taken from the torso surface of a dog and also from chronically implanted electrodes on the epicardium. Torso geometry was obtained afterwards by serially slicing the dog's frozen carcass. A forward transfer matrix was obtained assuming a homogeneous torso and using a method based on Green's second identity (Barr et al. 1977). An inverse transfer matrix was then derived using the regularisation method, as described above. The results of the study showed that calculated and measured epicardial potential distributions agreed in the major features and sequence of events. A similar study by Franzone (1980) using an isolated dog heart in a tank, and using various regularisation methods, showed similar results. This provides some encouragement that such methods are valid in human models.

The calculations of noise amplification factor described above show that when no regularisation is used ($\alpha=0$), noise in the input data is on average increased by a factor of almost 2000. This

implies that inverse transforms based solely on solutions of the normal equations are unlikely to produce satisfactory results. Lo (1977) found that epicardial potentials thus calculated were unreasonably large and oscillatory. He ascribed this behaviour to numerical errors which occur while inverting the forward matrix, and obtained reasonable epicardial potentials by other means (these are not clearly stated). Yamashita (1981) calculated the torso surface distribution from a theoretical source distribution and then used this torso data to re-calculate the epicardial potential distribution. He did not obtain the initial distribution, instead a sharp, oscillatory distribution was obtained. This result may be somewhat surprising, as the whole procedure is theoretically equivalent to multiplying by an identity matrix. The likely explanation is that the floating point precision of the computer used was only seven significant digits. The slight errors thus caused in the calculated torso surface data and inverse matrix would lead to large errors in the calculated epicardial distribution. Use of the regularisation method resulted in a calculated epicardial distribution much closer to the original.

These studies and the preliminary calculations above show that it is essential to use some form of smoothing (regularisation) when calculating the epicardial potentials. When using the regularisation method, both Barr and Spach (1978) and Franzone (1980) use a constant value for the regularisation parameter α throughout the cardiac cycle. The results described above suggest that the optimum value varies throughout the cardiac cycle. However, use of a constant value may be acceptable, as moderate variations in α

(less than an order of magnitude) lead to relatively minor changes in the calculated epicardial distributions. The advantage of using a constant value is that only one inverse matrix needs to be calculated for use throughout the cardiac cycle. The preliminary calculations show that when using a value of 10^{-4} for α, noise in the input data will on average be increased by a factor of 22 at the epicardium. However the calculated epicardial potentials are 10 to 15 times larger in magnitude than the measured torso surface potentials, so that the signal to noise ratio is not decreased by a large factor.

Because the regularisation method is designed to produce smooth solutions, calculated epicardial potential distributions may not be capable of showing sharp features which may exist in reality. For example, the change in potential at a point where the depolarisation wavefront intersects the epicardium may be quite rapid. This is a basic limitation imposed by the ill-conditioned nature of the problem. If it is desired to avoid unstable behaviour in the solutions, some loss of resolution is unavoidable. This problem may be bypassed to some extent by calculating inverse integral distributions, which are of their nature smoother than potential distributions at a particular instant. These are of some interest, as a study by Burgess et al. (1978) shows that in particular the epicardial QRST integral distribution reflects local changes in repolarisation properties.

In another study, Cuppen (1983) used a different formulation of the inverse problem which yielded the activation times on a surface

bounding the heart (both endocardium and epicardium were included in this surface). The activation time is more likely to be a smooth function of position on this surface, so that there is less chance that the regularisation method employed will remove wanted features. This study showed good results for a normal heart. However, the method depends strongly on two assumptions; firstly, that the active heart muscle is bounded by a single connected surface, and secondly, that the sources within the heart can be represented by a uniform dipole layer. A study by Roberts et al. (1979) on the influence of cardiac muscle fibre orientation in a dog heart suggested that the depolarisation wavefront is not a uniform layer. It is not clear how large the discrepancy is or what effect it has on Cuppen's method. More importantly, an abnormal heart (for example an infarcted heart) may well have more than one surface bounding the active muscle. In this case, the method is not applicable, as it depends on being able to close the depolarisation layer at any instant with a single portion of the surface which bounds the active heart muscle. In this context, an advantage of calculating epicardial potential distributions is that no assumptions need be made about the internal structure of the heart or the nature of the cardiac electrical sources.

Conclusion

A method has been developed for calculating epicardial potential distributions from measured torso surface potential distributions. Although the resulting epicardial distributions cannot be directly validated, they are physiologically plausible.

Furthermore, a number of similar studies performed with dogs or isolated hearts, where the calculated epicardial potential distributions can be directly checked, show that good results can be obtained. These results should encourage the evaluation of inverse epicardial potential distributions as a useful tool in a clinical situation.

CHAPTER FIVE

MULTIPLE TORSO GEOMETRIES

Introduction

In chapters three and four, the forward and inverse problems were respectively solved for a single torso geometry. The aim of this chapter is to explore the effects on these solutions caused by varying the torso geometry. It is important to know this in order to determine how much detail (and hence effort) is needed when applying these solutions to real patients. If a standard torso model can be used for all patients, then the effort involved is far less than if individual models must be used for each patient. Similarly, less effort is needed if only homogeneous models need to be used.

Methods

Torso geometries were digitised from CT scans of eight subjects using the methods described in chapter three. A sample CT scan slice from each subject is shown in Appendix E. Eleven torso models were produced, the first four being based on a single subject. Models 1, and 5 to 11, include heart, lungs, spine and sternum. Models 2, 3 and 4 have identical external geometry to model 1, but differ in the internal organs which have been included. Model 2 includes the heart, spine and sternum (no lungs). Model 3 includes the heart and lungs (no spine or sternum). Model 4 includes only

the heart (no lungs, spine or sternum). In all cases, organs which are not specifically included in the model are assumed to be homogeneous and assigned the average torso conductivity listed in table 3.1. An identical grid was used to approximate the slices in all models. Table 5.1 lists the age, sex, lung volume and chest circumference of each subject, as well as the number of nodes and calculated optimum acceleration factor (for the SOR method) for each corresponding model. Lung volumes and chest circumferences were calculated automatically from the modelled torsos. Unfortunately, not all of the torsos modelled have a completely normal torso geometry. The two major deviations are model 8, where the axis of the heart lies almost in a horizontal plane, and model 11, where there is a large left pleural effusion.

Intracardiac blood masses have not been modelled for two reasons. When studying the forward problem, the aim was to determine the effect of varying torso geometry external to the heart. Also, when solving the inverse problem using the methods described in chapter four, the internal structure of the heart does not affect the solution because the epicardial source regions form a surface completely enclosing the heart.

The forward problem has been solved for each model using the methods described in chapter three. For the purposes of examining the effects of geometrical differences on the forward solutions, a dipole-like source located in the heart was used in all models instead of epicardial source regions. This was done so that differences in the solutions would not be due to variations in the

TABLE 5.1

Torso Models

Model	age	sex	lung volume (l)	chest circumference (cm)	number of nodes	optimum acceleration factor
1	40	m	5.8	96	18924	1.943
2	40	m	0	96	18924	1.936
3	40	m	5.8	96	19233	1.943
4	40	m	0	96	19233	1.936
5	———	m	4.5	99	18326	1.947
6	50	m	4.3	100	20256	1.947
7	23	f	3.2	91	13668	1.944
8	74	f	3.3	90	15858	1.945
9	56	m	3.7	100	20173	1.951
10	72	m	3.1	95	17290	1.942
11	———	m	3.1	98	19606	1.945

Models 1 to 4 are constructed from data taken from a single subject, and differ only in the internal organs which have been included. Lung volumes and chest circumferences were calculated by computer from the digitised CT scan slices. The ages of subjects 5 and 11 is not known.

size or shape of the sources between models. The dipole source was placed in each model so that the negative pole was located at the node closest to the centre of mass of the heart. The positive pole was located 10mm downwards, 7.5mm left and 7.5mm in front of the negative pole. The poles were thus 14.58mm apart. This positioning approximately aligned the axis of the dipole along the axis of the heart in most models, with the positive pole nearer the apex of the heart. The positive and negative poles were held at potentials of +1mV and -1mV respectively.

In order to check that differences in the forward solutions are not due to variations in the source currents, the source currents were calculated for each model once the forward solution had been obtained. These values are listed in table 5.2. Despite the differences in geometry between models, the source currents are almost identical. This implies that the effective resistance seen by the source depends almost entirely on the conductivity in the neighbourhood of the source.

The inverse problem was solved for each model using the methods described in chapter four. Forward transfer matrices were calculated for each model based on 25 epicardial source regions and 45 torso surface locations. The same number of torso surface locations was used on all models so that identical input data could be used with every inverse calculation. These forward transfer matrices are listed in appendix F. Inverse transfer matrices were then calculated for each model according to equation 4.7. The value of the regularisation parameter α used in each case was 10^{-4}.

TABLE 5.2

Dipole Source Currents

model	current (μA)
1	10.79
2	10.80
3	10.79
4	10.80
5	10.77
6	10.78
7	10.79
8	10.78
9	10.77
10	10.78
11	10.78

Source currents were calculated after the potential distribution had been determined by calculating the net current flowing from the source nodes to adjacent nodes.

Epicardial potential distributions were calculated for each model using the input torso surface data as shown in figure 4.4 (40mS from QRS onset in a 25 year old normal male).

Results

Contour maps of the torso surface potential distributions due to the dipole-like sources in each model are shown in figure 5.1. The overall shapes of the distributions are similar in all models. A positive peak is located on the centre front of the torso, on or just left of the sternum. A negative peak is located just right of the sternum, higher than the positive peak. The magnitude of the positive peak is generally equal to or higher than that of the negative peak.

It is of particular interest to examine the differences between model 1 and models 2 to 4, where only the internal geometry has been altered. In model 1 (figure 5.1a), the magnitude of the positive peak is 10.1μV, positioned just to the left of the sternum in the centre of the chest. The magnitude of the negative peak is 10.9μV, positioned just to the right of the sternum, in the upper part of the torso.

Removing the lungs (model 2, figure 5.1b) causes the positive peak to be reduced by 2.7μV (27%). The positive peak is also shifted to a slightly higher location on the torso surface, and the area of positive potentials is reduced. The magnitude of the negative peak is reduced by 0.7μV (6%) and its position shifts slightly towards the sternum.

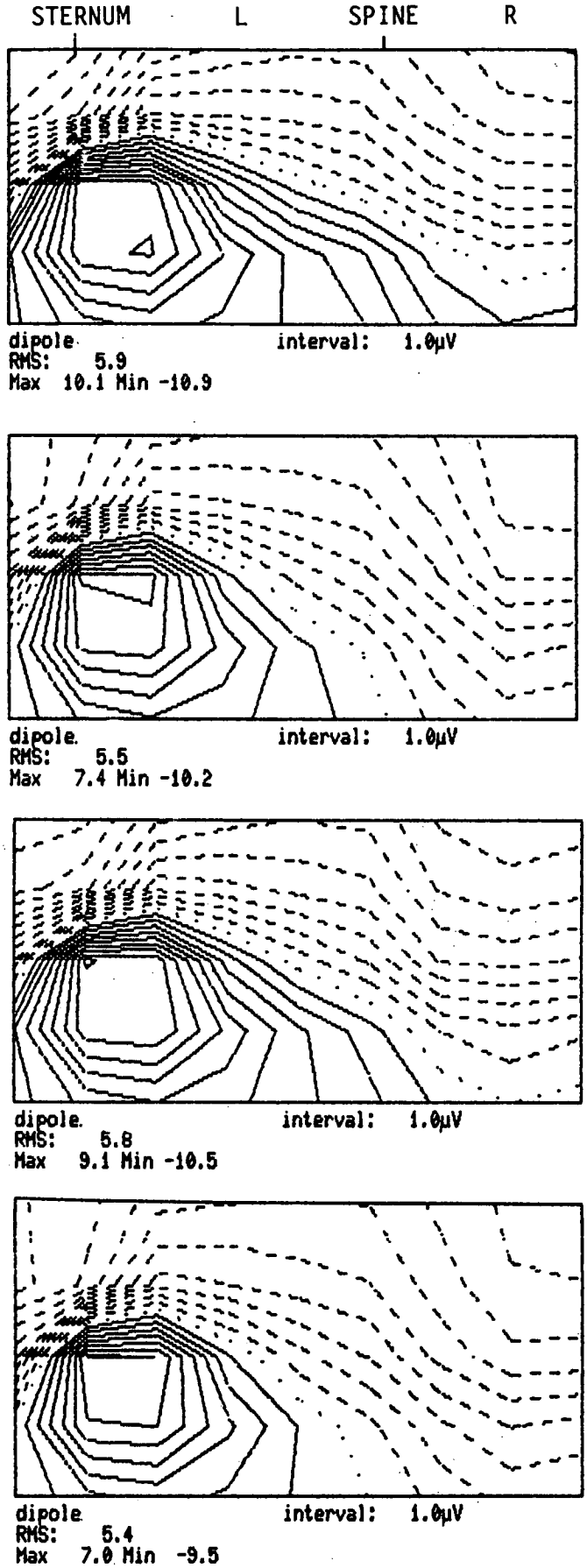
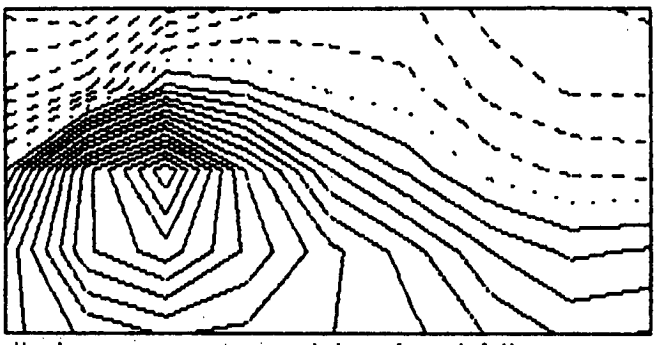
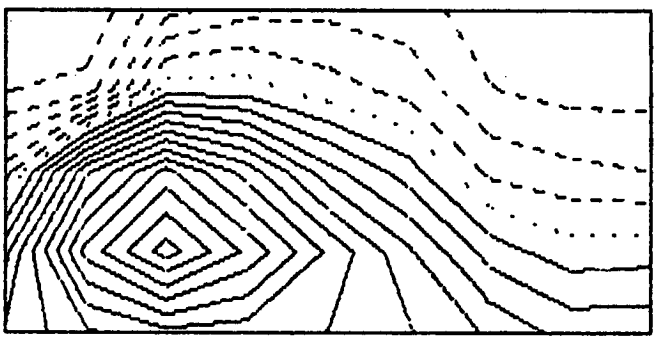


Figure 5.1 (3 pages). Body surface potential distributions due to a dipole-like source located in the heart. Maps (a) to (k) correspond to models 1 to 11 respectively. Format and positioning is as in figure 4.4.



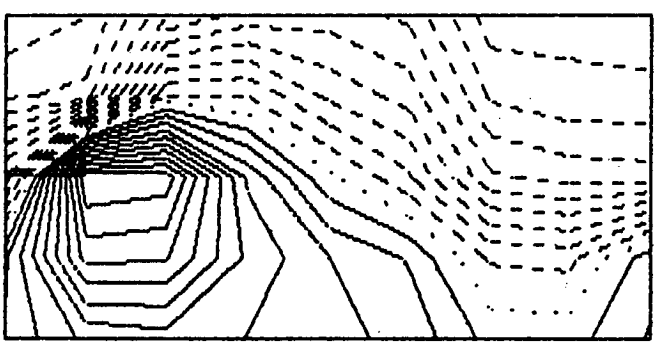
dipole interval: 1.0μV
RMS: 5.9
Max 15.8 Min -7.9

(e)



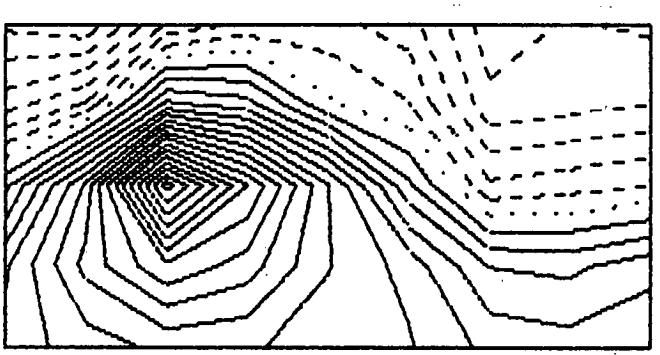
dipole interval: 1.0μV
RMS: 4.8
Max 12.5 Min -5.9

(f)



dipole interval: 1.0μV
RMS: 6.3
Max 11.0 Min -11.8

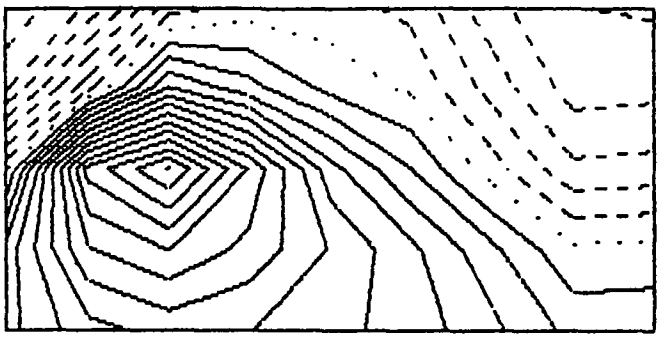
(g)



dipole interval: 1.0μV
RMS: 5.9
Max 17.5 Min -6.8

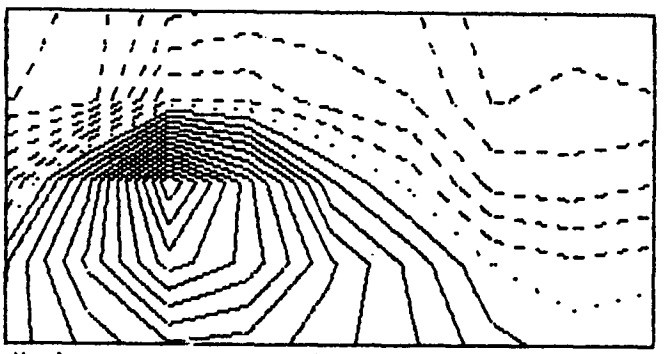
(h)

Figure 5.1 continued.



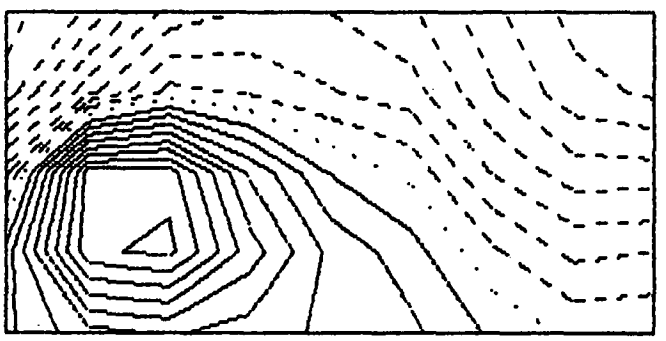
dipole interval: 1.0μV
RMS: 5.0
Max 13.1 Min -7.9

(i)



dipole interval: 1.0μV
RMS: 5.1
Max 13.8 Min -7.9

(j)



dipole interval: 1.0μV
RMS: 4.7
Max 9.3 Min -8.4

(k)

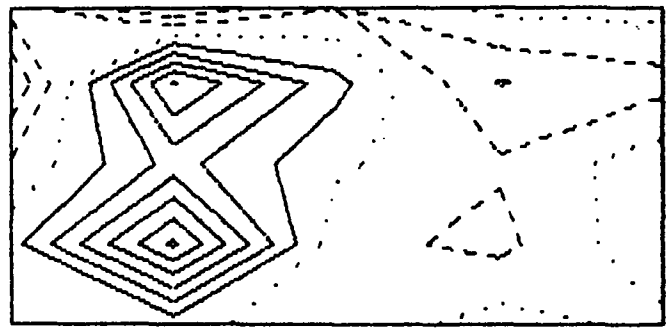
Figure 5.1 continued.

Removing the spine and sternum (model 3, figure 5.1c) has a smaller effect. The positive peak is reduced by 1μV (10%) and is shifted higher and over the sternum (or where the sternum was). The area of positive potentials is slightly reduced. The negative peak is reduced by 0.4μV (4%) and its position is not changed.

Combining the above changes, that is, removing lungs, spine and sternum (model 4, figure 5.1d), combines the effects of the above two models. The positive peak is substantially reduced in magnitude and area, and the negative peak is also reduced.

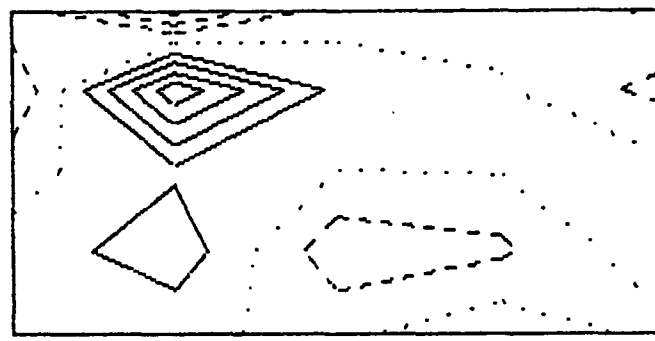
Examining models 5 to 11 (figure 5.1 e to k) shows that there is a significant variation in solutions with different torso shapes. The positive peak ranges in magnitude from 9.3μV (model 11) to 17.5μV (model 8). The magnitude of the negative peak ranges from 5.9μV (model 5) to 11.8μV (model 6). Positioning of the positive peak varied from slightly left of the sternum, mid chest, to slightly right of the sternum, lower chest.

The epicardial potential distributions obtained by solution of the inverse problem for each model are shown in figure 5.2. All models show a positive peak on the left side or posterior of the heart. There is usually a negative peak lying close to and above the positive peak. Examining models 1 to 4 (figure 5.2 a to d) shows once again that omitting lungs and/or spine and sternum from the models leads to large differences in the solutions. Omitting lungs has a larger effect than omitting bone, causing a 24% reduction in the magnitude of the positive peak and a 9% increase in the magnitude of the negative peak. The location of the positive



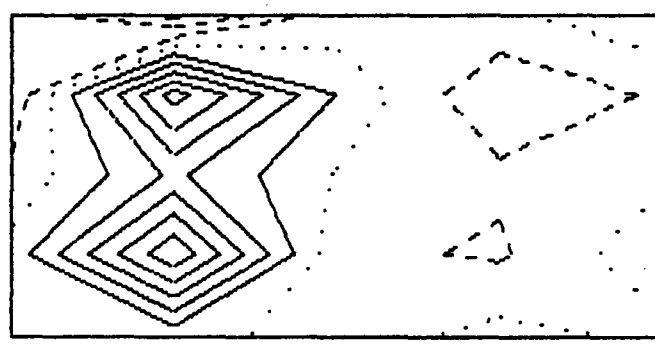
epi 40mS QRS interval: 5.0mV
RMS: 10.2
Max 31.0 Min -15.4

(a)



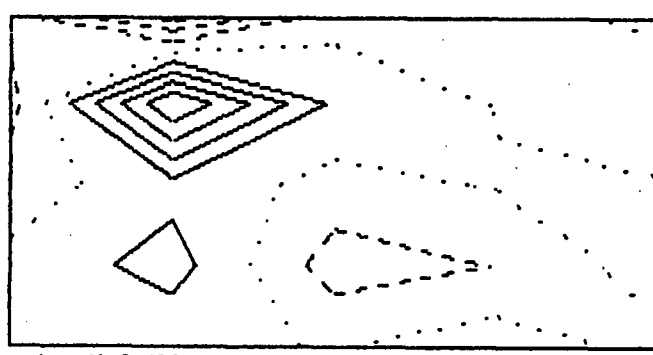
epi 40mS QRS interval: 5.0mV
RMS: 7.1
Max 23.6 Min -16.8

(b)



epi 40mS QRS interval: 5.0mV
RMS: 9.9
Max 29.3 Min -17.1

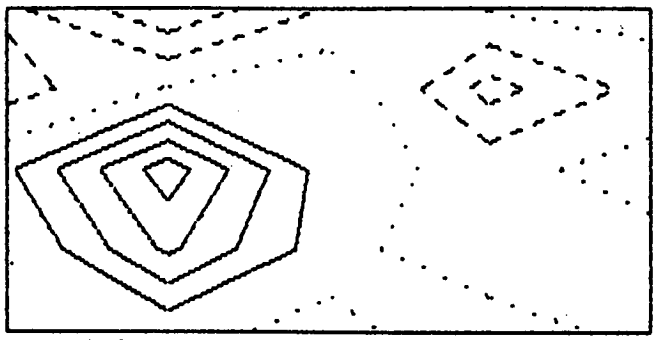
(c)



epi 40mS QRS interval: 5.0mV
RMS: 7.0
Max 25.0 Min -17.6

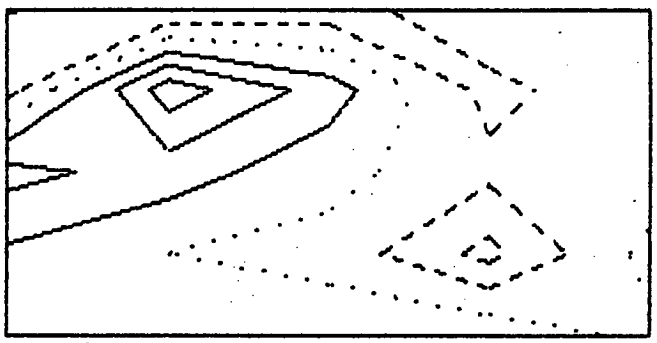
(d)

Figure 5.2 (3 pages). Epicardial potential distributions calculated from the measured torso surface potential distribution shown in figure 4.4. Maps (a) to (k) correspond to models 1 to 11 respectively.



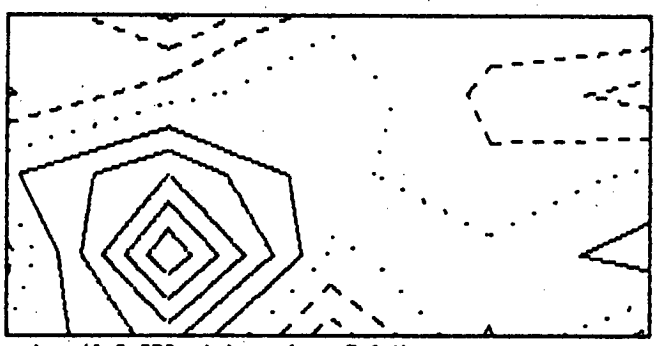
(e)

epi 40mS QRS interval: 5.0mV
RMS: 7.3
Max 22.7 Min -14.6



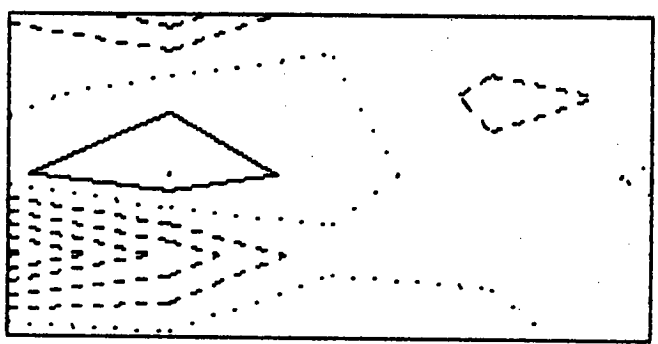
(f)

epi 40mS QRS interval: 5.0mV
RMS: 6.8
Max 17.8 Min -11.6



(g)

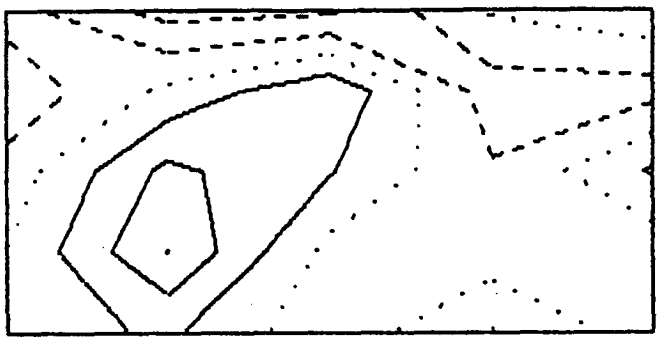
epi 40mS QRS interval: 5.0mV
RMS: 9.5
Max 29.4 Min -16.1



(h)

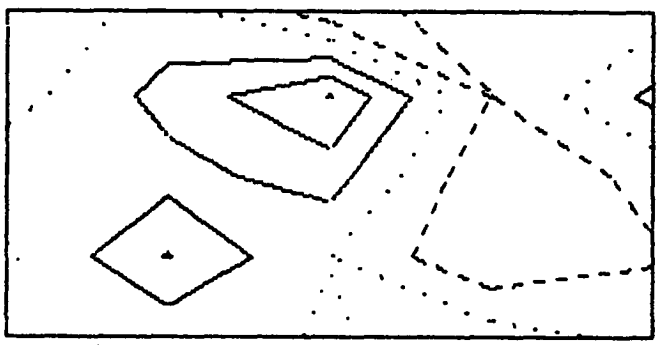
epi 40mS QRS interval: 5.0mV
RMS: 7.4
Max 10.0 Min -26.2

Figure 5.2 continued.



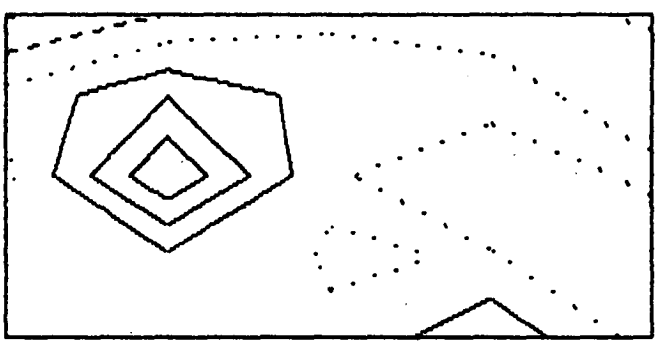
(i)

epi 40mS QRS interval: 5.0mV
RMS: 6.7
Max 15.0 Min -12.1



(j)

epi 40mS QRS interval: 5.0mV
RMS: 5.9
Max 15.2 Min -9.7



(k)

epi 40mS QRS interval: 5.0mV
RMS: 5.6
Max 19.9 Min -10.6

Figure 5.2 continued.

peak is then also shifted much higher on the left side of the heart.

Examining models 5 to 11 (figure 5.2 e to k) shows significant variations in the solutions. Model 8 is particularly different, with a large negative peak located near the apex of the heart, under the positive peak. The magnitude of the positive peak ranges from 10.0mV (model 8) to 29.4mV (model 6) and the magnitude of the negative peak ranges from 9.7mV (model 10) to 26.2mV (model 8).

Discussion

The main features shown in the solutions of the forward problem are consistent with the nature and orientation of the sources. In fact the torso surface potential distributions produced are quite similar in shape to those measured from a normal subject at mid QRS (compare, for example, figures 4.4 and 5.1a). This supports the idea underlying most conventional electrocardiography and vectorcardiography, that the electrical sources in the heart can be approximated quite well by a single dipolar source.

Models 1 to 4 differ only in the internal organs which have been included. When the lungs, spine or sternum are omitted from the model, the corresponding regions increase in conductivity, because they are assigned an average torso conductivity (460Ωcm) which is substantially higher than the conductivity of either lung (2100Ωcm) or bone (assumed 0Ωcm). This increase in conductivity will tend to 'short circuit' the current flowing through the torso, resulting in less current flowing near the skin and hence smaller potential differences on the surface of the torso. This decrease in

torso surface potentials with increasing lung conductivity is also seen in the eccentric spheres model described by Rudy et al. (1979).

The results obtained from models 1 to 4 described above show that inclusion of the lungs is essential if the forward problem is to be solved accurately. These results also show that the spine and sternum have a smaller effect on the forward solution. The effect of the lungs is greater than the effect of the bony regions probably because of the much greater volume which the lungs occupy. If the ribs were included as well, the effect of the bony regions may be much larger, because the bony regions are then greatly increased in extent around the torso. The models described in this chapter include only small portions of ribs where they are attached to the spine (for example see figure 3.1). It is hoped to construct a model which properly includes ribs to determine their effect. At the time of writing, the available computing resources have prevented the construction of a sufficiently detailed model.

The forward solutions for models 5 to 11 show that although major features are similar, there are substantial differences in detail. As expected, the largest positive potential (17.5μV) occurs on the smallest torso (model 8), although the next largest positive potential (15.8μV) occurs on one of the largest torsos (model 6). Perhaps in this case, the reduction in potentials which would be expected on a large torso is offset by the greater lung volume, which tends to increase torso surface potentials. The number of models presented here is insufficient to provide a detailed account of the effect that each variation in geometry has on the torso

surface potentials. Nevertheless, these solutions, together with the results from models 1 to 4, show that it is important to use accurately shaped, inhomogeneous models if accurate solutions of the forward problem are required.

The calculated epicardial potential distributions also show substantial differences between models. The distribution for model 8 is particularly different when compared to the other distributions, having a large negative area below the positive peak. This may be due to the different position of the heart in this model. It is not possible to say which model gives the closest approximation to the original epicardial potential distribution which gave rise to the measured input data, as the original epicardial distribution is not known. The point is that variations in both the internal conductivity and overall torso geometry cause substantial variations in the solutions. Thus, when calculating epicardial potential distributions from measured torso surface data, it is important to use an inverse transform based on a model which approximates the torso geometry of the subject as closely as possible.

Conclusion

This chapter shows that the shape and conductivity of the torso models used in solving the forward and inverse problems substantially affect the solutions. For the forward problem, this implies that if solutions are desired which are to be accurate beyond the level of major features, then accurate modelling of the

individual torso geometry is essential. For the inverse problem, the implication is that it is probably not sufficient to use a single torso model when calculating epicardial potential distributions for more than one subject. In practical terms this means that substantially more computing is required to produce clinically useful inverse solutions, as 'personalised' inverse transfer matrices may have to be computed for each patient. It may be possible to circumvent this problem to a large extent by building up a 'library' of inverse transfer matrices matched to various types of geometry (fat, thin, tall, short, male, female etc). In order to minimise the amount of work involved, it is necessary to use an automated system such as the one described in chapter three for the construction of the torso models. Work on building such a library and applying it clinically is beyond the scope of this thesis, but is proceeding.

CHAPTER SIX

CONCLUSION

Conclusion

This thesis has described the implementation of a system for calculating epicardial potential distributions from body surface potentials which is suitable for clinical evaluation.

A body surface mapping system which is suitable for use on very sick patients has been constructed. A 'jacket' of electrodes is used to enable rapid placement and removal of electrodes. In the two years that this system has been operating, over 1300 sets of data have been obtained from over 600 subjects by doctors, nurses and medical students.

An automated system for constructing computer models of the human torso has been developed. Because the system is capable of simulating the placement of the jacket of electrodes on a subject, models can be developed which are directly compatible with data collected using the surface mapping system. Initially, a single, highly detailed torso model was constructed and used to calculate both forward and inverse transfer matrices. It was found that using a suitable regularisation method, a sequence of physiologically plausible epicardial potential distributions could be calculated from body surface data recorded throughout a heart cycle. It is recognised that these epicardial potential distributions are not directly verifiable. Previous work using simpler models and with

dogs, where the results can be checked, encourages the belief that epicardial potential distributions calculated using the methods described in this thesis may be clinically useful. In the end, their value must be determined by the clinical results which can be obtained. Work on applying the transforms developed in this thesis to large sets of clinical data is continuing.

A number of highly detailed torso models were constructed to explore the effects of varying the conductivity and geometry of the torso. When inverse transfer matrices calculated from each of these models are applied to identical body surface data, substantial differences appear in the resulting epicardial potential distributions. Removal of the lungs or bony structures from a model also caused substantial variations. These results show that it is unlikely that a single torso model can be used to develop an inverse transform for a wide range of subjects. Because the removal of the major organs also caused substantial variations in the inverse solutions, it is likely that inhomogeneous torso models are necessary for accurate inverse transforms.

Future Work

It is likely that a completely self contained mapping system could be constructed with most of the capabilities of the present system. Such a system would use the extremely compact disk drives and high performance 16/32 bit microcomputers which are becoming available at the time of writing. This system would have the advantage of being completely mobile, while still having enough

storage space and processing power to both maintain large sets of data and perform the calculations required for the forward and inverse calculations.

It is unlikely that it will be possible to obtain accurate internal geometries for all patients on whom it is desired to calculate epicardial potential distributions. However, it is possible to develop a 'library' of inverse transfer matrices based on torso models of various geometries. The present mapping system already records a single approximate measure of the chest circumference of each subject mapped. This measurement is used to determine the number of electrodes which contact the patient and hence the number of rows in the forward transfer matrix. With minor changes and little extra work when mapping, adequate measurements could be recorded so that at least the overall size and shape of each patient can be matched to a suitable model. The automated system described in chapter three will allow such a library to be constructed with a minimum of effort. Setting up such a library would also eliminate almost all of the computation time which would otherwise be involved in producing individualised inverse transfer matrices. Calculations of epicardial potential distributions would simply require the application of an already calculated inverse matrix, a process requiring only a few seconds.

To evaluate the inverse transforms described in this thesis will require the comparison of measurements obtained from the epicardial potentials with other independent measures of the function of the heart. In the case of myocardial infarction, this

may involve the comparison of areas of abnormal epicardial potentials (such as ST segment changes) with other methods of determining the sizes of infarcts (such as cardiac enzyme changes or post mortem studies). Further calculations should also be applied to the epicardial potentials. It is a simple matter, for example, to calculate the distribution of current flowing into or out of the epicardium once the epicardial potentials in a model are known. This current distribution may provide a much better means of localising ST segment changes, as the current distribution does not depend on the choice of reference point for potentials.

APPENDIX A

CIRCUIT DIAGRAMS

This appendix contains circuit diagrams for the major units which make up the body surface mapping system described in chapter two.

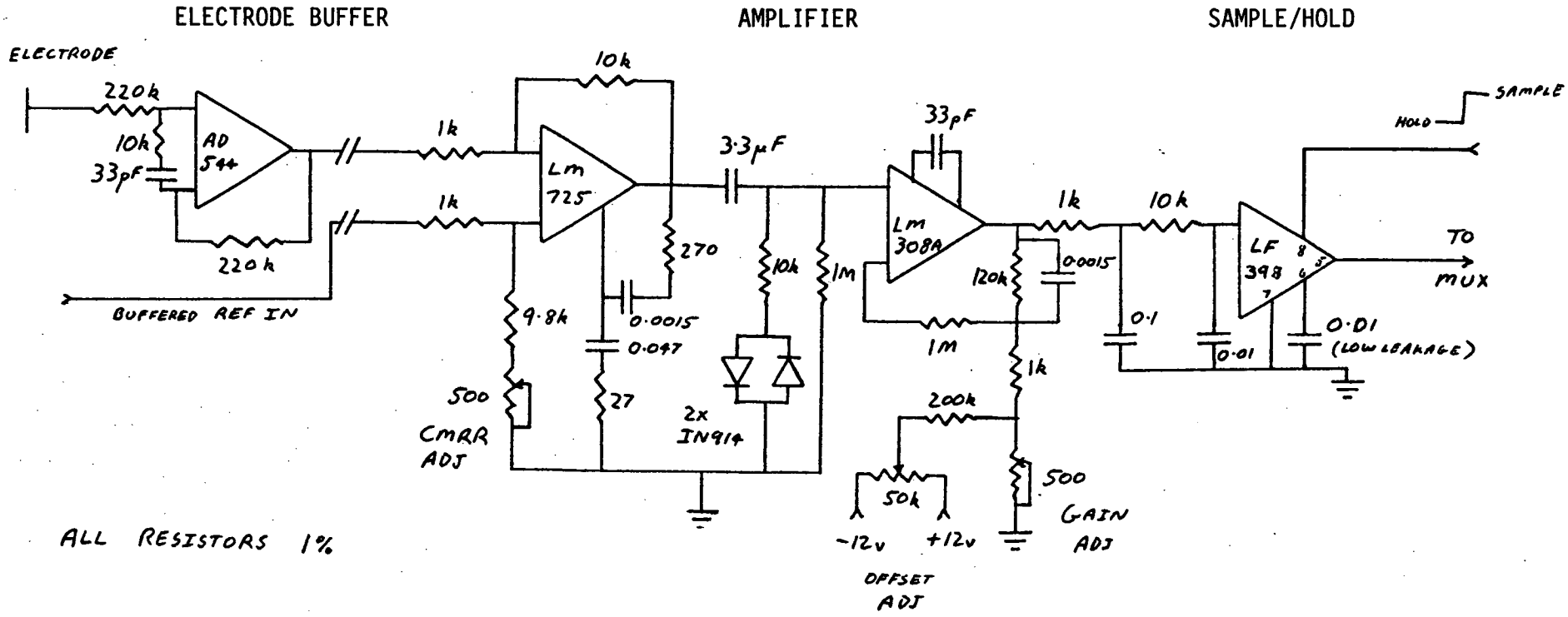


Figure A.1 Buffer, amplifier and sample/hold

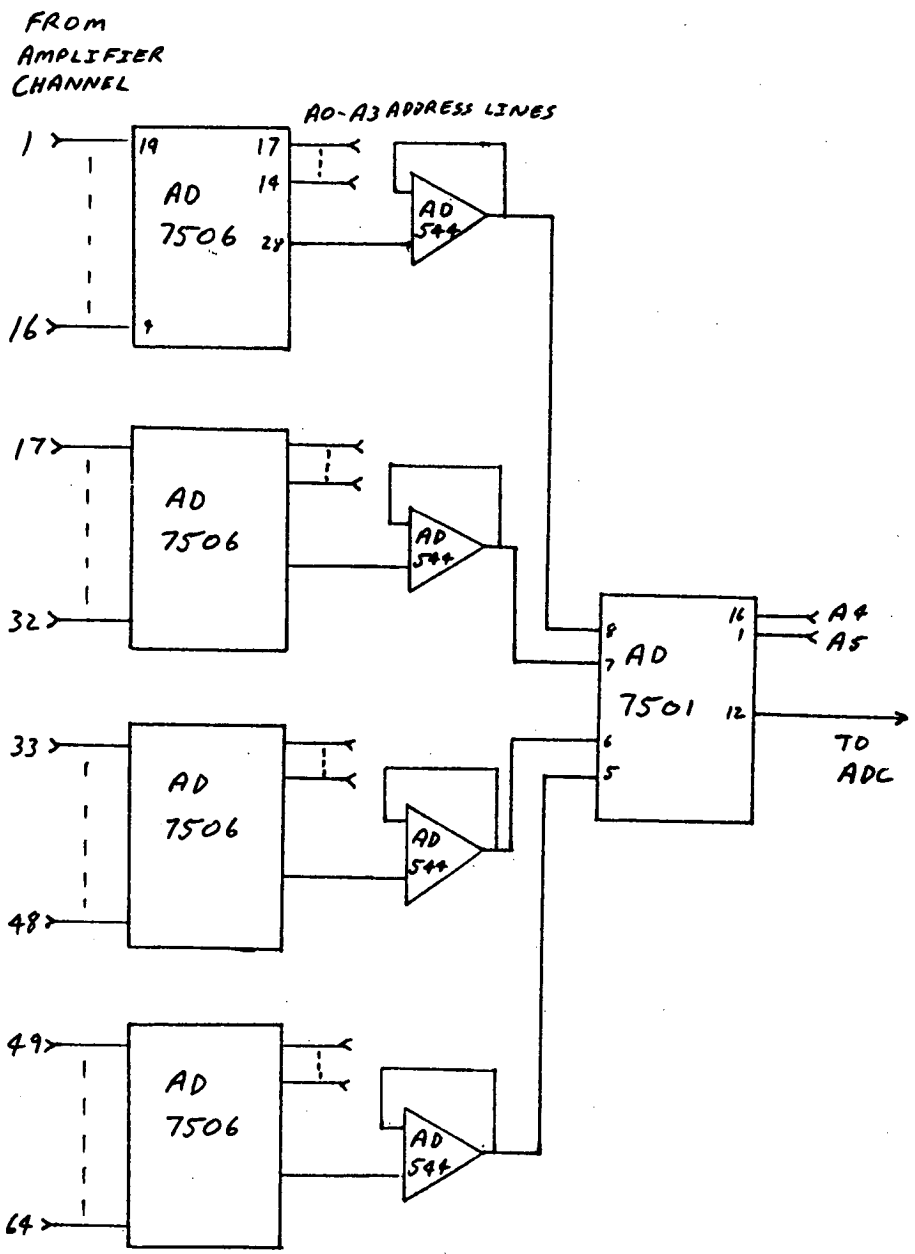


Figure A.2 Multiplexer.

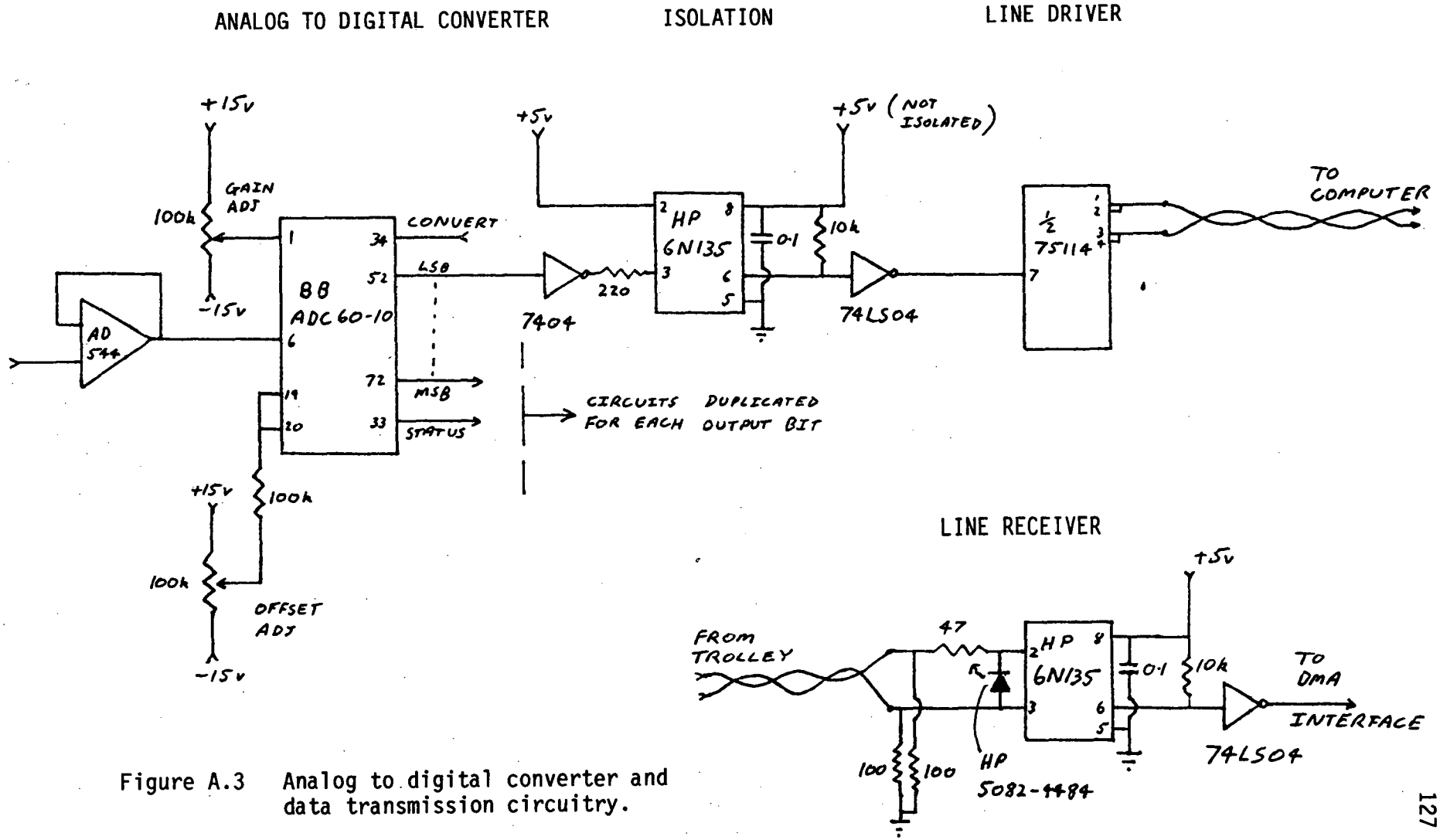


Figure A.3 Analog to digital converter and data transmission circuitry.

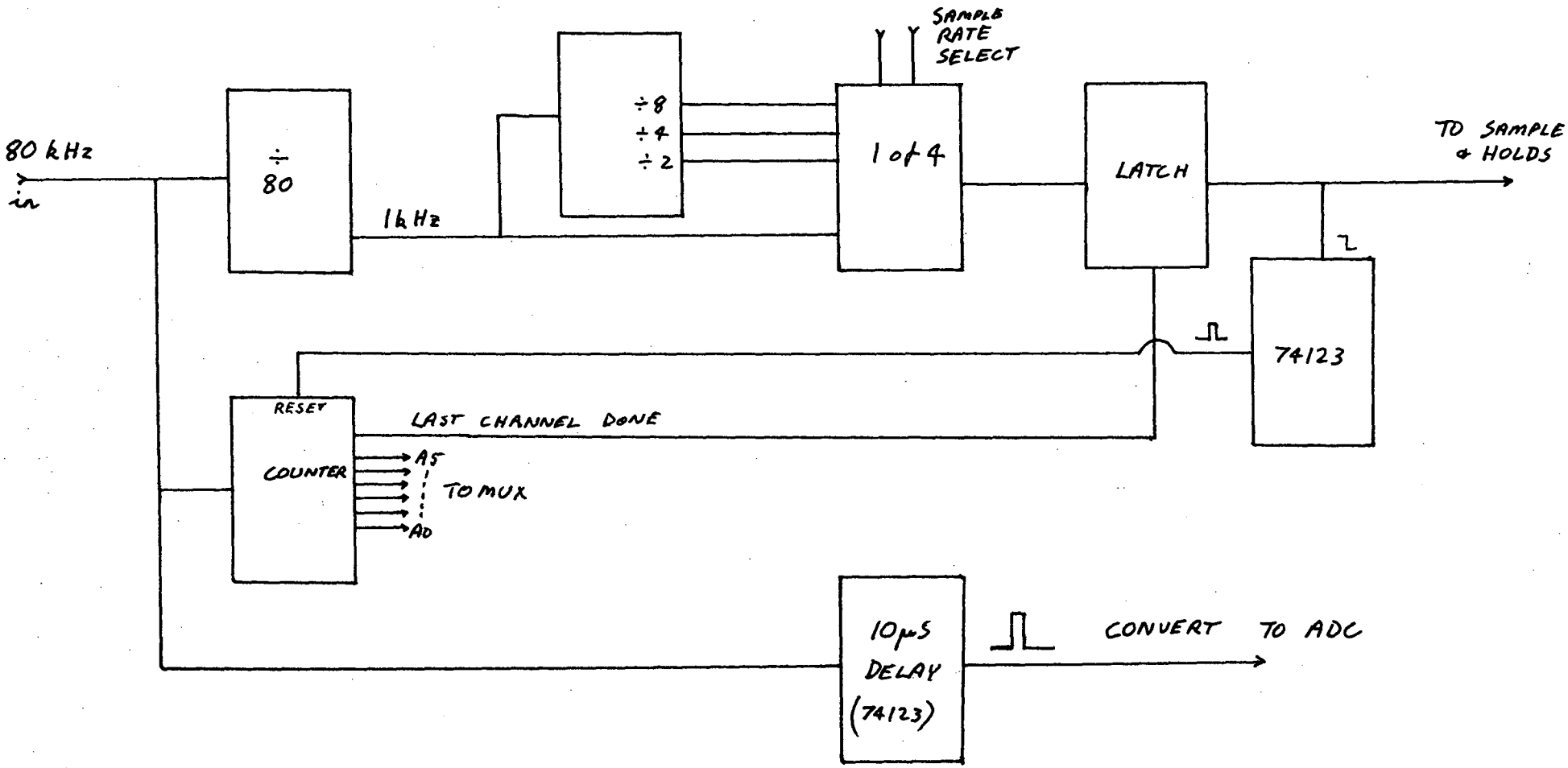


Figure A.4 Timing circuit block diagram.

APPENDIX B

SAMPLING SOFTWARE

This appendix contains listings of the main programs used to run the surface mapping system. The following programs are included:

UNIX — Modifications to the UNIX system to restrict the amount of memory used and allow programs to map their data address space.

dr.c — Device driver for the DR11W direct memory access interface

sample.c — The program which runs the mapping system

qrs.c — The QRS onset picking algorithm

All of these listings are in the C programming language. Due to the wide range of characters used in this language, it has been necessary to print most of the listings in this and following appendices on a dot matrix printer.

To restrict UNIX to less than the full amount of memory available, it is simply necessary to add the following statement after the system has performed the memory size test in the startup routine, and before mfree is called:

```
if( (SYSMAXMEM - (ka6->r[0] + USIZE)) < maxmem )
    maxmem = SYSMAXMEM - ( ka6->r[0] + USIZE );
```

Here, SYSMAXMEM is a constant which is the desired amount of memory which UNIX should use (in units of 64 bytes).

If the UNIX system is running on a PDP11 without separate I/D space, the statement should be

```
if( (SYSMAXMEM - (eu + USIZE)) < maxmem )
    maxmem = SYSMAXMEM - ( eu + USIZE );
```

To allow programs to access the spare memory above SYSMAXMEM, the system call listed on the following page was added:

```
/* phys system call
 * Allows a program to access memory above SYSMAXMEM (including I/O
 * page !! )
 *
 * VERY dangerous
 * use with care
 *
 *        Stephen Walker
 *        University of Tasmania
 *
 */

phys()
{
    struct a {
        int seg;
        int size;
        int physadr;
    } *uap;

    if( suser() )  /* check program has permission to use this call */
    {
        /* get arguments */
        uap = (struct a *)u.u_ap;
        /* check arguments within range */
        if(uap->seg <0 || uap->seg >7 || uap->size <0 || uap->size >127)
        {
            u.u_error = EINVAL;
            return;
        }
        /* check address is above SYSMAXMEM */
        if( uap->physadr < SYSMAXMEM )
        {
            u.u_error = EINVAL;
            return;
        }
#if     ID_SPACE
        /* if separate i/d  , add 8 to seg ( map D space ) */
        if( u.u_sep )
            uap->seg += 8;
#endif  ID_SPACE
        /* check segmentation register for availability */
        if( u.u_uisd[uap->seg] && (u.u_uisd[uap->seg]&ABS) == 0 )
        {
            u.u_error = ENOMEM;
            return;
        }
        u.u_uisa[uap->seg] = uap->physadr;
        u.u_uisd[uap->seg] = ((uap->size)<<8) | RW | ABS;
        /* set segmentation registers */
        sureg();
        return(0);
    }
}
```

```
/*      dr.c --- DR11W DRIVER
 *
 * This driver is a special driver written to sample data from an
 * external device via a DR11W DMA interface.
 *
 * It allows the process to set up the DR11W registers and
 * will also manipulate the UNIBUS MAP to allow data to be
 * stored in a large extra area of memory which UNIX does not use.
 *
 * The device is single user only (only open once).
 *
 * Routines:
 *              open    -   set open flag
 *
 *              close   -   reset open flag
 *
 *              ioctl   -   set up regs , sample to appropriate area
 *
 *
 *                              Stephen Walker
 *                              University of Tasmania
 *
 *                              June 1983
 */


#include <sys/param.h>
#include <sys/dir.h>
#include <sys/user.h>
#include <sys/buf.h>
#include <sys/dr.h>
#include <sys/seg.h>

/* start of extra memory */
#define XMEMSTART (512*32)
/* top of memory which DR11W is allowed to use */
#define XMEMTOP   (640*32)


unsigned drflags;
struct buf bp;

dropen(dev,flag)
{
        if ( minor(dev) != 0 )
        {
                u.u_error = ENXIO;
                return;
        }
        /* check if already open */
        if ( drflags & DROPEN )
        {
                u.u_error = EBUSY;
                return;
        }
        drflags = DROPEN;
```

```
}


drclose(dev,flag)
{
        drflags = 0;
}


/* This routine performs various user requests */
drioctl(dev,cmd,addr,flag)
caddr_t addr;
{
        register unsigned address;
        unsigned data;
        struct dr_ioparam ioparam;

        switch (cmd)
        {
        case SETDRWC:
        case SETDRBA:
        case SETDRCS:
        case SETDRIO:
                if ( copyin(addr, &data, sizeof(data) ) )
                {
                        u.u_error = EFAULT;
                        break;
                }
                switch(cmd)
                {
                case SETDRWC:  /* set word count reg */
                        DRADDR->drwc = data;
                        break;
                case SETDRBA:  /* set bus address reg */
                        DRADDR->drba = data;
                        break;
                case SETDRCS:  /* set control/status  reg */
                        DRADDR->drcs = data & ~GO;
                        break;
                case SETDRIO:  /* set I/O reg */
                        DRADDR->drio = data;
                        break;
                }
                break;

        case GETDRWC:  /* get value in word count reg */
                data = DRADDR->drwc;
                goto out;
        case GETDRBA:
                data = DRADDR->drba;
                goto out;
        case GETDRCS:
                data = DRADDR->drcs;
                goto out;
        case GETDRIO:
                data = DRADDR->drio;
```

```
            out:
                if ( copyout(&data, addr, sizeof(data)) )
                        u.u_error = EFAULT;
                break;

/* TRANSFER request samples data from the mapping equipment */
        case TRANSFER:
                /* get parameters */
                if ( copyin(addr, &ioparam, sizeof(ioparam)) )
                {
                        u.u_error = EFAULT;
                        break;
                }

                if (ioparam.ioflags & ( ECG ) )
                {
                    if( ((address=ioparam.addr)>XMEMTOP)
                          ||(address<XMEMSTART) )
                    {
                        u.u_error = EFAULT;
                        break;
                    }

                    drflags = DROPEN;

                    /* grab unibus map */
                    mapalloc(&bp);
                    /* map data area */
                    drmap(address);

                    /* start sampling */
                    DRADDR->drcs |= (GO|IE|XBA16);
                    DRADDR->drio |= START;

                    drmonitor(); /* wait till transfer complete */
                    drstop();

                    /* give back unibus map */
                    drunmap();
                    ioparam.ioflags |= drflags;
                    if ( copyout(&ioparam,addr,sizeof(ioparam)) )
                                u.u_error = EFAULT;
                    break;
                }

        }
}

drintr(d)    /* interrupt routine - start another transfer if necessary */
             /* ( yet to be implemented ) */
{
                drflags |= FINISHED;
}


drmap(addr)
```

```
register unsigned addr;    /* address to point to ( * 64 bytes ) */

{
        register int i;

        for (i=16; i<48; i += 2)
        {
                UBMAP->r[i] = ( addr<<6 ) + (i<<12);
                UBMAP->r[i+1] = ( (addr>>10) & 077) + ((i-16)>>4);
        }

}

drunmap()
{
        register int i;

        for (i=16; i<48; i += 2)
                UBMAP->r[i] = (i<<12);

        mapfree(&bp);
}

drmonitor()
{
        long i;

        i = 0;
        while( ((drflags&FINISHED)==0) && i < 500000)
                i++;
                /* loop until interrupt or timeout */

}

drstop()
{
        /* stop ecg sampling */
        DRADDR->drio &= ~START;

        /* reset dr11w if ready bit not set.
         * This should only occur if timeout occurs and
         * DR11W has not seen any error */
        if( (DRADDR->drcs&READY) == 0)
        {
                DRADDR->drcs = MAINT;
                DRADDR->drcs = READY;
                drflags |= DRTIMEOUT;
        }
}
```

```
/* sample.c -- Program to run the mapping system
 *
 * This program samples digitised ECG data simultaneously
 * from a large number of electrodes.
 * It uses a specially written device driver (dr11w)
 * Data is stored in an area of memory above that which UNIX uses
 *
 * It also demultiplexes the ecg data in extended memory
 * which has been (or should have been ) collected by the
 * sampling routine.
 *
 * Data is plotted for examination and may be saved if required.
 *
 * All graphics output assumes that the terminal interprets REGIS
 * graphics commands (eg DEC VT125, VT240);
 *
 * Program must be setuid to root to allow access to the special device
 * drivers and to the phys system call
 *
 *              Stephen Walker
 *              Uni of Tasmania
 *              December 1983
 */



#include <sys/param.h>
#include <sys/buf.h>
#include <sys/timeb.h>
#include <ctype.h>
#include  <stdio.h>
#include  <signal.h>
#include  <sys/dr.h>

#include  "ecg.h"
#include  "screen.h"

/* bits in DR11W CS reg to turn trolley power on */
#define   POWERON   02
/* word count reg value */
#define   WCVAL     0
/* bus address reg value (0 because data mapped by unibus map) */
#define   BAVAL     0

/* size of buffer for de-multiplexing */
#define   BUFSIZE  (16384)



#define XMEMSTART (512*32)          /* first No is Kw */

int *bigbuf;            /* big buffer for mapping to extended mem */
int chan[SPC];          /* buffer for 1 channel */
int memoffset;          /* offset in words from XMEMSTART of valid data */
```

```
char rate[2];              /* sampling rate (string, length 1)  */
int posn;                  /* jacket position */
int v1,v2,v3;              /* battery voltages */
int dr;                    /* dr11w device file descriptor */

main()
{

    int nasty(),timeout(), ok;


    /* ignore delete key           */
    signal(SIGINT,SIG_IGN);
    /* catch alarm to turn off power and exit */
    signal(SIGALRM,timeout);
    /* other signals trap to nasty routine */
    signal(SIGTERM,nasty);
    signal(SIGBUS,nasty);

    /* Increase process priority a bit so we dont keep patients
       waiting too long */
    nice(-10);

    /* clear text and graphics screens */
    clearts();
    cleargs();
    printf("ECG SAMPLING PROGRAM\n\n");

    if( (dr = open(DRDEV,0) ) == -1)    /* open dr11w   */
    {
        fflush(stdout);
        fprintf(stderr,"Sorry, extra memory is busy\n");
        exit(1);
    }

    ok = 0;
    poweron();
    printf("Power is on - wait for warm up (15 sec)\n\n");
    fflush(stdout);

    /* while they are waiting we set up regis
       graphics to save time later */
    enterregis(0);
    showplane(2);
    writeplane(1);
    border(11,5,"");
    writeplane(2);
    endregis();
    sleep(10);

    /* main sampling loop */
    while( ok == 0 )
    {
            /* set alarm so power goes off if no response
               after 10 minutes */
            alarm(1200);
```

```
            /* sample data */
            sample();
            /* demultiplex it */
            demux();
            /* inspect it and return whether ok or not */
            ok = inspect();
            /* reset alarm */
            alarm(0);
    }
    poweroff();  /* turn trolley power off */
    /* reset alarm */
    alarm(0);

    /* close dr11w */
    close(dr);

}


/*************************************************************************/
/*  routine to set up dr11w and  do actual sampling                      */
/*************************************************************************/

sample()

{
    unsigned data;
    struct dr_ioparam io;

    clearts();
    signal(SIGINT, SIG_IGN);

    data = WCVAL;
    if( ioctl( dr, SETDRWC, &data ) != 0) /* load word count reg */
        ohdear("setting word count register");

    data = BAVAL;
    if( ioctl( dr, SETDRBA, &data ) != 0) /* load bus addr reg   */
        ohdear("setting bus address register");

    data = 0;
    if( ioctl( dr, SETDRIO, &data ) != 0) /* clear io reg   */
        ohdear("clearing io register");

    do  /* read jacket position */
    {
        printf("\nInput jacket position (-1 to 3)>");
        if( scanf("%d",&posn) == 0)
                while( getchar() != '\n' );
    }
    while( posn<-1 || posn>3 );

    printf("\nInput sampling rate :\n\n"); /* read sampling rate */
    printf("    0 -->   1000 samples per second\n");
    printf("    1 -->   500  samples per second\n");
    printf("    2 -->   250  samples per second\n");
```

```
    printf("    3 -->   125  samples per second\n");
    printf(">");
    fflush(stdout);
    scanf("%ls",rate);

    data = READY | POWERON;
    /* set sampling rate bits */
    data |= (*rate & 03) << 2;
    if( ioctl( dr, SETDRCS, &data ) != 0) /* load cs reg */
        ohdear("setting sampling rate");

    io.ioflags = ECG;       /* ecg type transfer */
    io.addr    = XMEMSTART;    /* where to start transfer */

    /* now we do the sampling */
    if( ioctl( dr, TRANSFER, &io) != 0)
        ohdear("sampling (ioctl call error)");

    /* check cs reg for errors */
    if ( ioctl( dr, GETDRCS, &data) != 0)
        ohdear("reading CS register");

    if( (data&ERROR) || (io.ioflags&DRTIMEOUT) )
        /* print a message and exit if errors while sampling */
    {
        printf("CS    = %o\nFLAGS = %o\n",data,io.ioflags);
        ohdear("sampling");
    }

    /* tell them to relax */
    printf("Finished sampling - patient can relax\n\n");
    fflush(stdout);
}


/*************************************************************************/
/* demux reads the data in extended memory and puts it back in           */
/* semi-demultiplexed form                                               */
/*************************************************************************/


demux()
{
    register int from,to,val;
    int tmp,next;
    int i;



    printf("Wait...\n");
    fflush(stdout);

    /* check data is multiplexed correctly and get
       battery voltages */
    memoffset = checkdata();  /* doesnt return if data bad */
```

```
    for(i=0; i<4; i++)        /* 4 bufferfuls gets all the data */
    {
        mapbuffer(i,memoffset);        /* read next buffer */
        val=bigbuf[0];
        bigbuf[0]=0;

        /* Demultiplexing loop - does it all in place,
           by calculating where each value should be stored (to)
           from its' original location (from).  Looping
           behaviour is avoided by noticing that all original
           values are negative (unused bits are set), whereas
           demultiplexed values are stored positive */
        from=0;
        next=1;
        while( next < BUFSIZE )
        {
            to = (from>>6) + ( (from&077) << 8 );
            if( bigbuf[to] >= 0 )
            {
                bigbuf[to] = val&01777;
                while( next < BUFSIZE && bigbuf[next] >= 0 )
                    next++;
                if( next < BUFSIZE )
                {
                    from = next;
                    val = bigbuf[from];
                    bigbuf[from] = 0;
                }
            }
            else
            {
                tmp = bigbuf[to];
                bigbuf[to] = val&01777;
                val = tmp;
                from = to;
            }
        }
        bigbuf[from]=val&01777; /* put back last value */
    }

}      /* data should now all be semi-demultiplexed in mem, starting at
          XMEMSTART(*32W)  + memoffset words  */


/*********************************************************************/
/* Checkdata returns the address of the start of valid               */
/* data , or exits if data is not valid                              */
/*********************************************************************/
checkdata()


{

    register unsigned  start;
```

```
    register int i;

    mapbuffer(0,0);

    /* now skip the first sample ( as it may have 65 channels-
     * a hardware fault) and point to the first point with
     * last channel bit (02000) not set
     * (the last two channels may have this bit set - hence
     *  the second if stmt)
     */

    for(i=0; ((bigbuf[i]&02000)==0)&&(i<BUFSIZE); i++);
    if(i>128) ohdear("skipping first sample");
    if(bigbuf[++i]&02000) i++;

    /*
     * We are at the start of potentially valid data
     * Check that the last channel bit is set every
     * 64 data words (only the first buffer of data
     * is checked)
     */

    start = i; /* save start */

    for( ; i<BUFSIZE; i+= 64)
        if( (bigbuf[i-1]&02000) == 0 || (bigbuf[i]&02000) != 0 )
                ohdear("checking data format");

    /*
     * Data is valid so get battery voltages, write header and
     * return with start of data
     *
     * Battery voltages are as follows:
     *
     *  -12 v battery  --  chan 60
     *  +12 v battery  --  chan 61
     *  +6  v battery  --  chan 62
     *
     * They are read from sample number 50 for these channels
     * ( a fairly arbitrary choice)
     */

    v1 = (bigbuf[start+64*50-5]) & 03777;  /* -12 batt */
    v2 = (bigbuf[start+64*50-4]) & 03777;  /* +12  "   */
    v3 = (bigbuf[start+64*50-3]) & 03777;  /* +6   "   */
    return(start);
}


getheader()
{
    register int i;

    for( i=0; i<SPC; i++) chan[i] = 0; /* clear area to be written */
    chan[0] = 0;
    chan[1] = (*rate)&03;
```

```
    chan[2] = v1;
    chan[3] = v2;
    chan[4] = v3;
    chan[5] = posn;
}


/* routine to map bigbuf into extended memory
 *
 * i is offset in 16kW from XMEMSTART
 * off is offset in words within that 16kW range
 *
 */

mapbuffer(i,off)
int i;
int off;
{
    register seg,physadr;

    physadr = XMEMSTART + ( i << 9 );
    for(seg=1; seg<6; seg++)
    {
        if( phys(seg,127,physadr) != 0 )
            ohdear("mapping memory");
        physadr += 0200;
    }
    bigbuf = (int *)(020000) + off;

}


/*************************************************************/
/* Routine for simple examining and saving                   */
/*************************************************************/

int getout; /* flag set by quit() to show DELETE has been pressed */

inspect()

{
    int quit();
    int chnum,n,i,point;
    int badchan[CHNUM+1];
    double sf,volts[3];
    unsigned flags;
    char ans[2], label[10];


    signal(SIGINT, quit);

    flags=0;
    volts[0] =(double) (512-v1)*40.3/1024.0;
    volts[1] =(double) (512-v2)*40.3/1024.0;
```

```
volts[2] =(double) (512-v3)*20.0/1024.0;

for(i=0; i<=CHNUM; i++) badchan[i] = 0;
for(;;)
{
    getout = 0;
    clearts();
    enterregis(0);
    eraseplane(2);
    showplane(2);

    /* set up regis macrographs used by plotone */
    printf("@:p]V[]P[+1,@;");
    printf("@:v][+1,@;");
    fflush(stdout);
    endregis();

    /* print a message if batteries are low */
    if( volts[0]>-11.6 || volts[1]<11.6 || volts[2]<5.8)
    {
        printf("\033[1m");
        printf("                BATTERIES ARE LOW\n");
        printf("CHANGE BATTERIES AFER FINISHING THIS PATIENT\n\n");
        printf("\033[0m");
    }

    printf("Select an operation from the following codes\n\n");
    printf("    p  -->  plot all channels\n");
    printf("    c  -->  plot single channel\n");
    printf("    s  -->  save data\n");
    printf("    r  -->  sample again\n");
    printf("    x  -->  exit program\n");
    printf(">");
    scanf("%1s",ans);
    if( getout == 0 )
        switch(*ans)
        {
    case 'p':
    /* plot all channels */
        {
            clearts();
            enterregis(0);
            showplane(3);
            writeplane(2);
            plotall();
            endregis();
            if( getout )
            {
                enterregis();
                printf(";;;");
                endregis();
                break;
            }
            pickbad(badchan);
            flags |= BADDONE;
            labelbad(badchan);
```

```
            printf("\033[24;1HPick bad channels");
            fflush(stdout);
            reviewbad(badchan);
            clearts();
            break;
        }

    case 'c':
    /* plot a single channel */
        {
            clearts();
            getchnum(&chnum);
            getsf(&sf);
            getn(&n);
            getpoint(&point);
            getchan(chnum,chan);
            scale(chan,1,sf);
            sprintf(label,"%d",chnum);
            clearts();
            enterregis(0);
            showplane(2);
            writeplane(2);
            border(1,1,label);
            plotone(chan,SPC-2,1,1,WIDTH,HEIGHT,point,n);
            endregis();
            labelonebad( badchan[chnum] );
            askifbad( badchan , chnum );
            break;
        }

    case 's':
    /* save the data */
        {
            poweroff();
            alarm(0);
            if( (flags&BADDONE) == 0)
                pickbad(badchan);
            save(flags,badchan);
            return(1);
        }

    case 'r':
    /* repeat the sampling process */
        {
            clearts();
            return(0);
        }

    case 'x':
    /* exit from the program */
        {

            clearts();
            cleargs();
            return(1);
        }
```

```
        default:
            {
            }
        }    /* end of switch */

    }      /* end of for(;;) */


}



/* Routine to get channel chnum from memory
 * after demux has run.
 */

getchan(chnum,chan)
int chnum,chan[];
{
    register  int *c, *m;
    register int j ;
    int i,physadr;

    if( chnum<1 || chnum>CHNUM)   /* return if chnum not sensible */
        return(1);

    c = chan;
    for(i=0; i<4; i++)            /* channels are in 4 parts */
    {
        physadr = XMEMSTART + ( i * 512 ) + ((chnum-1) * 8);

        if( phys(1,127,physadr) != 0 )
        {
            ohdear("mapping memory ( getchan )");
        }
        m = (int *)(020000) + memoffset;

        for(j=0; j<256; j++)
            *c++ = *m++;
    }
    return(0);
}

quit()
{
    signal(SIGINT, quit);
    getout++;
}

poweron()
{

    unsigned data;

    data = READY|POWERON;
    if( ioctl( dr, SETDRCS, &data ) != 0) /* turn power on (cs reg)*/
```

```
        ohdear("turning power on");
}

poweroff() /* power off routine */

{
    unsigned data;

    data = READY & ~POWERON;
    if( ioctl( dr, SETDRCS, &data) != 0)
        {
      fprintf(stderr,"Error while turning power off -- GET HELP\n\n");
          exit(1);
        }
}

ohdear(err) /* error routine */
char *err;
{
    fflush(stdout);
    cleargs();
    fprintf(stderr,"Error while %s\n", err);
    poweroff();
    exit(1);
}

timeout()
{
    poweroff();
    setttnormal();
    cleargs();
    printf("\n Timeout! \n");
    close(dr);
    exit(1);

}

nasty() /* routine for unexpected signals */
{
    cleargs();
    printf("Program terminated unnaturally -- GET HELP\n");
    exit(1);

}

/*********************************************************************/
/* Routines to save the data in a named file                         */
/*********************************************************************/

save(flags,badchan)
unsigned flags;
int badchan[];

{
    int key,i,dataf;
    char newfn[50];
```

```
    clearts();

    if( flags&SAVED )
    {
        printf("Data has already been saved\n\n");
        fflush(stdout);
        sleep(2);
        return(-1);
    }

    /* get the identifying number for this set of data */
    if( (key = getid())<0 )
    {
        printf("ERROR - CANT ASSIGN KEY\n");
        fflush(stdout);
        sleep(2);
        return(-1);
    }
    /* set up data file */
    sprintf(newfn,"/user/icu/ecgdata/rcg.%04d",key);
    if ( (dataf = creat(newfn,0444)) == -1)
    {
        printf("Can't create data file\n");
        fflush(stdout);
        sleep(2);
        return(-1);
    }
    printf("Saving to file rcg.%04d ....\n",key);
    fflush(stdout);
    getheader();
    chan[0] |= SAVED;

    /* write out the data */
    write(dataf,chan,2048);
    for(i=1; i<=CHNUM; i++)
    {
        getchan(i,chan);
        chan[0] = badchan[i];
        write(dataf, chan, 2048);
    }
    close(dataf);
    chown(newfn,105,105);
    return(0);
}


/* file where last identification number was stored */
#define  LASTKEY "/user/icu/ecgdata/lastkey"
/* identification file */
#define  IDFILE  "/user/icu/ecgdata/ecg.id"

getid()
{
```

```
    char fname[20], lname[20], ur[10], *date, buff[80];
    FILE *idf, *lastkf;
    int key,i;
    long tp, time(), lseek();

    /* read last id number */
    if ( (lastkf = fopen(LASTKEY,"r") ) == NULL)
        return(-1);
    if( fscanf(lastkf,"%d",&key) != 1 )
        return(-1);
    key++;

    printf("Input patient's name (1 first name then surname) \n>");
    /* read name */
    scanf("%19s%19s",fname,lname);
    while( getchar() != '\n' ) /* flush input */ ;
    /* make first letter upper case, rest lower case */
    fname[0] = toupper(fname[0]);
    lname[0] = toupper(lname[0]);
    for(i=1; (fname[i] != '\0'); i++) fname[i] = tolower(fname[i]);
    for(i=1; (lname[i] != '\0'); i++) lname[i] = tolower(lname[i]);

    printf("\nInput patient's UR number \n>");
    /* read hospital unit record number */
    scanf("%6s",ur);
    while( getchar() != '\n' ) /* flush input */ ;

    /* get date and time from system */
    tp = time(0);
    date = ctime(tp);

    if( (idf = fopen(IDFILE,"a")) == NULL)
        return(-1);

    sprintf(buff,"%1d %s %s %s %s",key,fname,lname,ur,date);
    /* print id data to file and terminal */
    printf("\n     %s\n",buff);
    fprintf(idf,"%s",buff);

    fclose(lastkf);
    if ( (lastkf = fopen(LASTKEY,"w")) == NULL)
        return(-1);
    fprintf(lastkf,"%d\n",key);
    fclose(lastkf);
    fclose(idf);
    return(key);

}


/* Routine to pick bad channels
 */

pickbad(badch)
int badch[];
```

```
{
    int i,chnum,max,num,count[1024];

    /* loop for all channels */
    for(chnum=1; chnum<=(CHNUM); chnum++)
    {
        getchan(chnum,chan);
        max = num = 0;
        for(i=0; i<1024; i++) count[i] = 0;

        /* build value histogram */
        for(i=0; i<SPC; i++) count[(chan[i]&01777)]++;
        /* find most common value */
        for(i=0; i<1024; i++) if( count[i]>count[max] ) max = i;

        /* if most common value is not near saturation levels,
           find baseline by averaging around this value */
        if(max>100 && max<900)
        {
            num += count[max];
            for(i=1; num<=SPC/4 && i<=20
                     && (max+i)<1020 && (max-i)>20; i++)
            {
                num += count[max+i];
                num += count[max-i];
            }
        }
        /* If less than one quarter of all values lie within
           0.4mV of most common value then channel is bad */
        if(num<=SPC/4) badch[chnum] = 1;

        /* now try to pick non-contacting channels
           by the big -ve spike at the first couple of samples */
        if( abs(chan[0]-chan[1])>50 &&
            abs(chan[1]-chan[2])>20    )  badch[chnum] = 2;

    }
}


/* This routine labels the bad channels once
they have been plotted */
labelbad(badch)
int badch[];

{
    int x,y,chnum;
    enterregis(0);
    chnum = 1;
    for(x=0; x<11; x++)
        for(y=0; y<5; y++)
        {
            printf("P[%3d,%3d]",(x*WIDTH/11+5),(y*HEIGHT/5+5) );
            if( chnum <= CHNUM )
            {
                if( badch[chnum] == 1)
```

```
                printf("T(S1)'B'");
            else if( badch[chnum] == 2)
                printf("T(S1)'D'");
            else if( badch[chnum] == 3)
                printf("T(S1)'U'");
        }
        else
        {
            printf("T(S1)'IGNORE'");
        }
        chnum++;
    }
    endregis();

}

/* This routine labels a single plotted bad channel */
labelonebad(code)
int code;
{
    enterregis(0);

    printf("P[10,10]");
    if( code == 1)
        printf("T(S2)'B'");
    else if( code == 2)
        printf("T(S2)'D'");
    else if( code == 3)
        printf("T(S2)'U'");

    endregis();

}

/* This routine allows the operator to manually accept or
   reject channels */
reviewbad(badchan)
int badchan[];
{
    int x,y;
    char code;

    x = y = 0;
    code = ' ';
    enterregis(0);
    printf("P[%3d,%3d]",(x*WIDTH/11+5),(y*HEIGHT/5+5) );
    fflush(stdout);
    setttsingle();

    while( code != 'x' && code != 'X')
    {
        code = tolower(getchar());
        switch(code)
        {
        case 'u':
            {
```

```
                if(y>0) y--;
                else if(x<10) y=4;
                break;
            }

        case 'd':
            {
                if(y<4 && x<10) y++;
                else y=0;
                break;
            }

        case 'l':
            {
                if(x>0) x--;
                else
                {
                    if(y==0) x=10;
                    else x=9;
                }
                break;
            }

        case 'r':
            {
                if(x<9 || ( x==9 && y==0) ) x++;
                else x=0;
                break;
            }

        case 'g':
            {
                printf("W(R)T(S1)' '");
                badchan[x*5 + y+1] = 0;
                break;
            }

        case 'b':
            {
                printf("W(R)T(S1)'B'");
                badchan[x*5+y+1] = 3;
                break;
            }
        }

        printf("P[%3d,%3d]",(x*WIDTH/11+5),(y*HEIGHT/5+5) );
        fflush(stdout);
    }

    endregis();
    setttnormal();


}

/* asks if a single channel being examined in detail is bad */
```

```
askifbad( badchan,chnum )
int badchan[], chnum;
{

    char ans[2];

    printf("\033[24;1HIs this channel bad?(y/n)>");
    fflush(stdout);
    scanf("%1s",ans);
    if (*ans == 'y' || *ans == 'Y')
    {
        enterregis(0);
        printf("P[10,10]T(S2)'B'");
        endregis();
        badchan[chnum] = 3;
    }
}


/*
 * Routine to plot one channel with upper LH corner at box number
 * x,y. Box has size xs,ys.
 * Values are assumed to be ok to fit on the screen.
 *
 * Every nth point is plotted
 *
 * Points only are plotted if point != 0
 */

plotone(d,len,x,y,xs,ys,point,n)
int d[],len,x,y,xs,ys,point,n;
{
    int xx,yy,ybase,lasty,highylim,lowylim;
    register int i,xinc;

    xx = (x-1)*xs + 2;
    ybase = (y-1)*ys + ys/2;
    highylim = y*ys;
    lowylim = (y-1)*ys;
    yy = ybase + d[1];
    if( yy > highylim ) yy = highylim;
    if( yy < lowylim ) yy = lowylim;
    printf("P[%1d,%1d]V[",xx,yy);
    lasty = yy;
    xinc = 0;

    /* Regis macrographs are used to try to speed up the plotting */
    /* macrographs are as follows :                               */
    /*                                                            */
    /* p is  ]V[]P[+1,    used in point mode                      */
    /* v is  ][+1,        used to advance 1 point in line mode    */

    /* they are set up in inspect */

    for( i=1; i<len; i += n)
    {
```

```
        yy = ybase + d[i];
        if( yy > highylim ) yy = highylim;
        if( yy < lowylim ) yy = lowylim;
        if (point) printf("@p%1d",yy);
        else
        {
            if( yy == lasty ) xinc++;
            else
            {
                if( xinc )
                {
                    if( xinc > 1 )
                        printf("][+%1d,%1d",xinc,lasty);
                    else
                        printf("@v%1d",lasty);
                }
                printf("@v%1d",yy);
                lasty = yy;
                xinc = 0;
            }
        }
    }
    if( point ) printf("]V[]");
    else
    {
        if( xinc) printf("][+%1d,%1d]",xinc,lasty);
        else printf("]");
    }
}


/* Routine to plot all channels
   Much optimised, so rather messy !
 */

#include    "ecg.h"

plotall()
{
    int x,y,chnum;
    double sum;
    register int i,j;

    chnum = 1;
    for(x=1; x<=11; x++)
        for(y=1; y<=5; y++)
        {
            getchan(chnum++,chan);
            j = 1;
            sum = 0;
            for( i=1; i<SPC; i += 16 )
            {
                chan[j] = chan[i];
                sum += (double)(chan[j]);
                j++;
            }
```

```
            sum /= (j-1);
            for( i=1; i<j; i++)
            {
                chan[i] -= (int)(sum);
                chan[i] = chan[i]>>2;
            }
            plotone(chan,j-1,x,y,70,96,0,1);
            if (chnum>CHNUM || getout )
            {
                return;
            }
        }
}


/* Routine to divide the graphics screen up into xnum by ynum boxes
 * Will number the boxes if label is null or put character string
 * label in top right corner
 */

border(xnum,ynum,label)
int xnum,ynum;
char *label;
{
    int x,y,i,j,num;

    for(i=0; i<=xnum; i++)
    {
        x = i*WIDTH/xnum;
        printf("P[%3d,0]V[%3d,%3d]",x,x,HEIGHT);
    }
    for(i=0; i<=ynum; i++)
    {
        y = i*HEIGHT/ynum;
        printf("P[0,%3d]V[%3d,%3d]",y,WIDTH,y);
    }

    if (*label != '\0')
    {
        for(j=0; label[j] != '\0'; j++);
        printf("P[%3d,10]T(S2)'%s'",(WIDTH-20*j),label);
    }
    else
    {
        num = 1;
        for(i=0;i<xnum;i++)
        {
            x = (i+1)*WIDTH/xnum-30;
            for(j=0; j<ynum; j++)
            {
                y = j*HEIGHT/ynum +10;
                if( num <= CHNUM )
                    printf("P[%3d,%3d]T(S1)'%3d'",x,y,num);
                num++;
            }
        }
```

```
    }
}


/* routines which
 * perform various screen functions
 *
 */

/* clear text screen */
clearts()
{
    printf("\033[H\033[0J");

}

/* enter graphics mode */
enterregis(debug)
int debug;
{
    if( debug ) printf("\033P3p");
    else printf("\033P1p");

}

/* end graphics mode */
endregis()
{
    printf("\033\\");
    fflush(stdout);
}

/* clear graphics screen */
cleargs()
{
    enterregis(0);
    printf("W(R)S(E)");
    showplane(3);
    endregis();

}


/* Routine to ask all the questions about
 * plotting parameters when examining a single channel
 */


getchnum(chnum)
int *chnum;
{
    clearts();
    do
        {
        printf("Which channel do you want (0 to %3d) >", CHNUM);
        if (scanf("%d",chnum) == 0)
```

```
            flin();
    }
    while( *chnum<0 || *chnum>CHNUM );
}

getsf(sf)
double *sf;
{
    do
        {
        printf("\n\nInput scaling factor (0.25 to 5) >");
        if( scanf("%f",sf) == 0)
            flin();
    }
    while( *sf<0.25 || *sf>5.0 );
}

getn(n)
int *n;
{
    do
        {
        printf("\n\nInput n - every nth point is plotted >");
        if( scanf("%d",n) == 0)
            flin();
    }
    while( *n<1 || *n>10 );
}

getpoint(point)
int *point;
{
    char ans[2];

    do
        {
        printf("\n\nDo you want points only(y/n))");
        if( scanf("%1s",ans) == 0)
            flin();
    }
    while( *ans!='y' && *ans!='Y' && *ans!='n' && *ans!='N');
    if( *ans =='y' || *ans=='Y' )
        *point = 1;
    else    *point = 0;
}


/* flush unwanted extra input */
flin()
{
    while( getchar() != '\n' );
}


/* Routine to scale and if required, offset data in  chan
 */
```

```
scale(chan,center,sf)
int chan[], center;
double sf;

{
    int i, base;

    if ( center )
    {
        /* get baseline */
        base = base1(chan);
        /* subtract it from each value */
        for(i=0; i<SPC; i++)
            chan[i] -= base;
    }

    /* scale values if scaling factor is not 1.00000 */
    if( sf != 1.0 )
        for( i=0; i<SPC; i++)
            chan[i] = (int)( ((double)chan[i])/sf);

}


/* This routine finds the baseline of a channel by finding
 * the most common value and averaging around it
 * Very similar code to pickbad routine
 */

base1(d)
int d[];
{
    int i,j,max,num,count[1024];
    double  sum;

    sum = 0;
    max = num = 0;
    for(i=0; i<1024; i++) count[i] = 0;
    for(i=0; i<SPC; i++) count[(d[i]&01777)]++;
    for(i=0; i<1024; i++) if( count[i]>count[max] ) max = i;
    if(max>100 && max<900)
    {

        sum += ((double)max)*count[max];
        num += count[max];
        for(i=1; num<=SPC/4 && i<=20 && (max+i)<1020 && (max-i)>20; i++)
        {
            j = max+i;
            sum += ((double)j)*count[j];
            num += count[j];
            j = max - i;
            sum += ((double)j)*count[j];
            num += count[j];
        }
    }
```

```
    if(num>SPC/4) return((int)(sum/num));
    else
    /* if channel seems to be bad just average all the values */
    {
        sum = 0;
        for(i=10; i<SPC-10; i++) sum += (double)d[i];
        return( (int)( sum/(SPC-20) ) );
    }
}


/* routines to set terminal input to single character, no echo;
 * and to reset to normal
 *
 */

#include <sgtty.h>

setttsingle()
{
    struct sgttyb tt;

    ioctl(0,TIOCGETP,&tt); /* get terminal parameters */
    tt.sg_flags &= ~ECHO; /* no echo */
    tt.sg_flags |= CBREAK; /* read single character at a time */
    ioctl(0,TIOCSETP,&tt);  /* set new parameters */

}

setttnormal()
{

    struct sgttyb tt;

    ioctl(0,TIOCGETP,&tt);
    tt.sg_flags &= ~CBREAK;
    tt.sg_flags |= ECHO;
    ioctl(0,TIOCSETP,&tt);
}

/* More strange regis graphics routines
 */

writeplane(n)
int n;
{
    printf("W(F %1d)",n);
}

eraseplane(n)  /* doesn't preserve position */
int n;
{
    writeplane(n);
    printf("W(E)P[0,0]W(S [,480])V[768]W(S0)W(R)");

}
```

```
showplane(n)
int n;
{
    int b1,b2,b3;

    switch(n)
    {
        case 1 : {
                    b1=74;
                    b2=0;
                    b3=74;
                    break;
                 }
        case 2 : {
                    b1=0;
                    b2=74;
                    b3=74;
                    break;
                 }

        default :{
                    b1=49;
                    b2=74;
                    b3=100;
                 }
    }
    printf("S(M1 (L%0d))",b1);
    printf("S(M2 (L%0d))",b2);
    printf("S(M3 (L%0d))",b3);
    fflush(stdout);
}
```

QRS onset location

The onset of the QRS complex in located automatically by the routine listed on the next page. The input data to this routine are a set of values which are the averages of the absolute values of all leads at each sampling time.

The data is differentiated, and a the location of the first value to exceed a specified threshold derivative value is found. From this point, the position where the derivative next becomes negative is found. If this position is less than 6 mS on from where the threshold value was exceeded, it is assumed that a noise spike has been found and the search for a point exceeding the threshold resumes. Otherwise it is assumed that the point just reached lies near the peak of the QRS complex. If no QRS peak is found by the end of the data, the search is restarted with a lower derivative threshold. This improves the chances of finding a QRS complex with a slow rate of rise (as occurs in some conduction defects or ventricular extrasystoles).

Once a point near the peak of the QRS complex has been found, a search is conducted backwards until the undifferentiated data reaches the baseline noise level. The point at which this occurs is designated the QRS onset.

```
/* qrs - picks the qrs onset from data stored in dt.
 *
 * This data should be the average of the absolute values
 * of all the good channels
 */
qrs(dt)
int dt[];

{
    int i, j, sum1, sum2, thresh, fpat, found, drv[SPC];

    /* differentiate data and store in drv */
    diff(dt,drv);
    fpat = found = 0;

    /* set derivative threshold = 3 */
    thresh = 3;

    /* loop while qrs not found, lowering threshold on each scan */
    while( (found == 0) && (thresh>=2) )
    {
        i = 10;
        /* search along derivative until position of
           first point above threshold (fpat) is found.
           Then look for next point where derivative becomes
           negative.  If this is more than 5mS from fpat, we have
           found the peak of the QRS complex */
        do
        {
            /* look for derivative value which exceeds threshold */
            while( drv[i]<=thresh && i<SPC-10 ) i++;
            if( i<SPC-10 )
            /* if one was found, search for next point where
               derivative becomes negative */
            {
                fpat = i;
                while( drv[i]>0 && i<SPC-10 ) i++;
            }
        }
        while( i-fpat<=5 && i<SPC-10 );

        /* lower threshold */
        thresh--;
        if(i<SPC-10) found = 1;
        /* and loop if not found */
    }

    /* if QRS peak found near either end of data , return 0
       (assumed not found) */
    if ( i<50 || i>=SPC-10 ) return(0);

    /* now find onset by comparing signal values
       with noise before QRS.  Start with i at position
       of peak, and decrement until signal fades into noise */
    do
    {
```

```
        sum1 = sum2 = 0;
        for(j=1; j<=20; j++)
        {
            /* average from 40mS to 20mS before i */
            sum1 += dt[i-40+j];
            /* average from 20mS before i to i */
            sum2 += dt[i-20+j];
        }
        i--;
    }
    while( sum2-sum1>=20 && i>=50 );

    /* keep decrementing i until data becomes flat or
       increases in value. */

    while( dt[i-1]<=dt[i] && (flat(dt,i)==0) ) i--;

    /* if QRS onset is in a usable postion, return it */
    if ( i<40 ) return(0);
    return(i);
}

flat(dt,i)
int i, dt[];
{
    int j,flt;

    flt = 1;
    for(j=0; j<4; j++)
        if( dt[i-j] != dt[i-j-1] ) flt = 0;
    return(flt);
}

diff(dt,drv)
int dt[], drv[];
{
    int i;

    drv[0] = dt[1] - dt[0];
    for(i=1; i<SPC-1; i++)
        drv[i] = dt[i+1] - dt[i-1];
    drv[SPC-1] = dt[SPC-1] - dt[SPC-2];
}
```

APPENDIX C

MODELLING SOFTWARE I

This appendix contains descriptions of the algorithms which approximate the input CT scan data by a rectangular grid. Program listings are also included.

Construction of a grid approximation of an area

When building the torso models, it is necessary to approximate a region where the boundary has been specified by a large number of square or rectangular elements. This can be done manually using graph paper and counting squares, but such a procedure is very time consuming, especially where the number of elements is large. The two programs listed in this appendix perform the task automatically, needing only a list of points on the boundary of the region, and a specification of the 'mesh' of the final grid approximation.

grid1

The first program, grid1, generates a detailed approximation of the region using a square grid. This grid is composed of a series of horizontal lines at y=SIZE/2 + n * SIZE, n=0,1,2,.... and vertical lines at x=SIZE/2 + n * SIZE, n=0,1,2,.... , where SIZE is some appropriate dimension (2.5mm in the torso models). The region is 'sampled' at the points of intersection of these lines. A 'bitmap' array representing the region is constructed, where each element in the array corresponds to a grid intersection point, and is non-zero if the point is inside the region. This array is initially all zero (Figure C.1a) and is filled as follows:

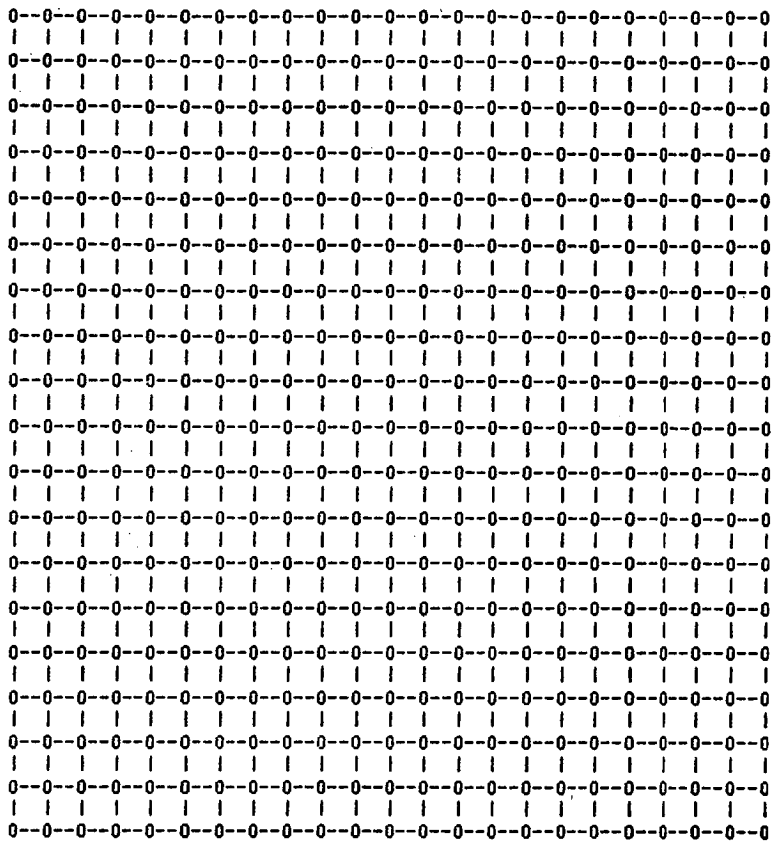
Start with any point on the boundary of the region. Join this point to the next point on the boundary. The line thus formed will cut a number (possibly none) of the horizontal grid lines. For each of these grid lines, complement all the elements in the corresponding row of the bitmap which are to the right of the line

between the two boundary points (Figure C.1b).

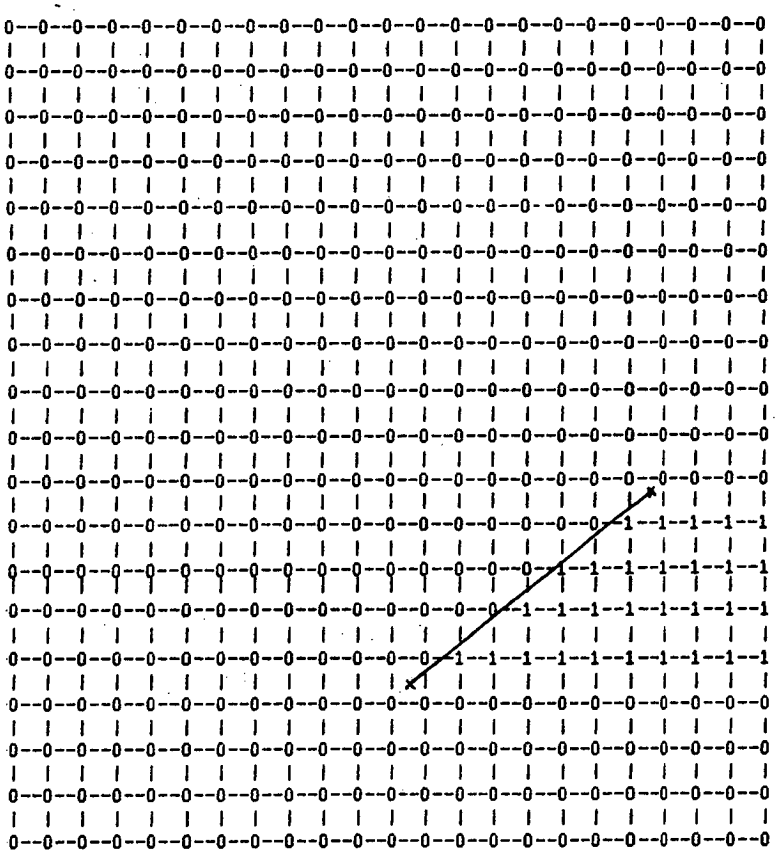
Continue around the boundary, joining a line to each successive boundary point, and complementing the appropriate bitmap elements (Figure C.1c). When the starting point is reached again, all bitmap elements inside the region will be non-zero, while all those elements outside the region will be zero(Figure C.1d). This method depends on the fact that any element inside the boundary region will be complemented an odd number of times, while elements outside the boundary will be complemented an even number of times, no matter how convoluted the boundary may be.

grid2

The second program, grid2, generates the non-uniform model grid from the square grid produced by grid1. A file containing the widths and heights of the grid elements is read. Each element is in turn located on the square grid produced by grid1. The appropriate tissue code for the element is then determined by examining the square grid points within the element.

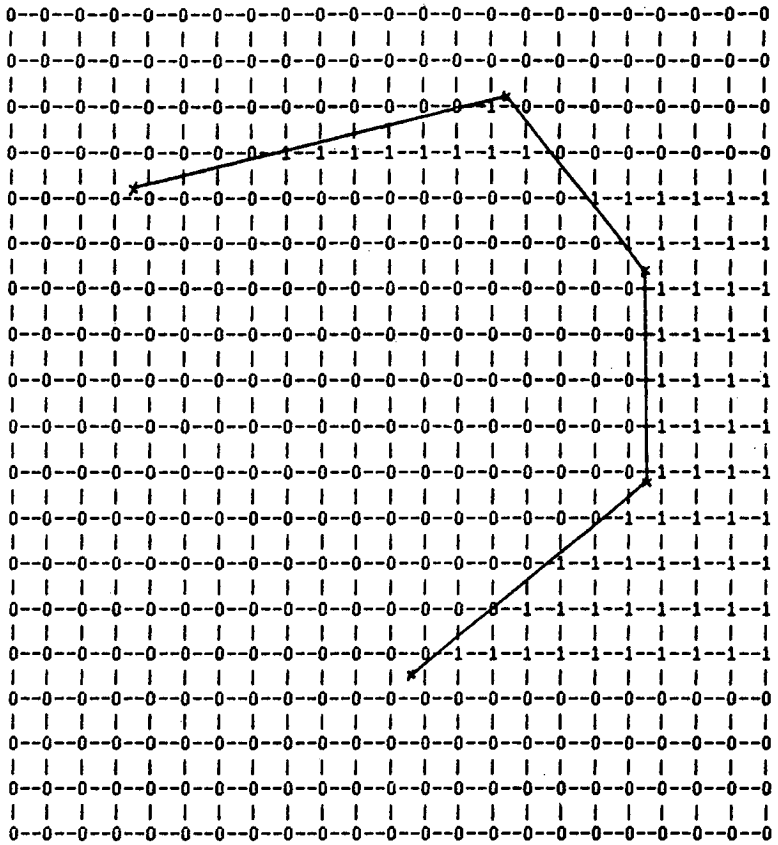


(a)

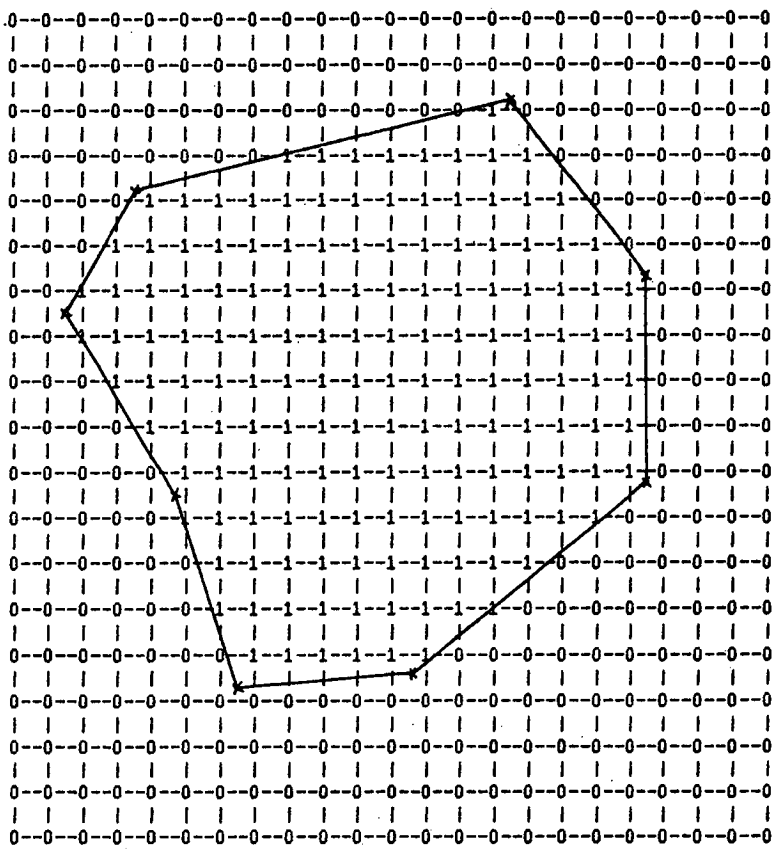


(b)

Figure C.1 (2 pages).
(a) Empty grid
(b) First section of boundary in place.



(c)



(d)

Figure C.1 continued.
(c) Continuation of boundary
(d) Area complete.

```
/*   grid1.c
 *
 *   A program to read co-ordinates of an area and cut it up
 *   into a square grid. The coded grid is printed to
 *   the standard output.
 *
 *   Output is in the following form, repeated as many times
 *   as there are input slices  :
 *
 *   Slice n
 *   2 A 3 B 7 D 3 A
 *   5 A 1 B
 *   1 ? 3 B 8 A
 *   .
 *   .
 *   .
 *
 *   Each number is the number of elements of each following code type.
 *   The special code ? signifies empty elements.
 *
 *   The grid size can be changed by changing the include file body.h
 *   SIZE = 2.5, MAXX = 180, MAXY = 140 are suitable for a human
 *   torso. DO NOT change these without afterwards re-compiling all
 *   other programs which include body.h
 *
 *   This program must be compiled with separate i/d space as follows
 *
 *      cc -O -i grid1.c -lm -o grid1
 *
 *   Stephen Walker
 *   Uni of Tas
 */

#include <math.h>
#include <stdio.h>
#include <ctype.h>
#include "body.h"


char bitmap[MAXX][MAXY];   /* grid for each sub-area */
char mainmap[MAXX][MAXY];  /* final grid */
float xp[500];             /* array of x co-ords on edge */
float yp[500];             /* array of y co-ords on edge */


main(argc,argv)
int argc;
char *argv[];
{

    char code[2];             /* property code for sub-area */
    char c;
    int pcount;               /* number of co-ordinates for this area */
    int slicenum;             /* slice number */
    int i,j;
```

```
double temp;
int slith;                  /* slice spacing in mm */

/* get slice spacing argument */
if( argc == 2 )
{
    slith = atoi(*++argv);
    if( slith < 2 || slith > 50 )
    {
        fprintf(stderr,"Ridiculous slice spacing: 10 mm assumed\n");
        slith = 10;
    }
}
else
{
    fprintf(stderr,"No slice spacing given: 10 mm assumed\n");
    slith = 10;
}

/* first output the numbers defining the grid (in this
   case the grid is square, with all boxes 2.5mm on each
   side.  This output is only done for compatibility
   with the output of grid2)
*/
printf("Row heights\n");
for(i=0; i<MAXY; i++)
    printf("1\n");
printf("Column widths\n");
for(i=0; i<MAXX; i++)
    printf("1\n");
printf("Thickness of slices (mm)\n%3d\n",slith);



/* main loop for each slice - keep going
   while another slice can be read from
   standard input
*/
while( scanf("%*s %d",&slicenum) == 1 )
{
    flin();                             /* flush rest of input line */
    printf("Slice %2d\n",slicenum); /* print this slice number  */

    for( i=0; i<MAXX; i++)
        for(j=0; j<MAXY; j++)
            mainmap[i][j] = 0;          /* clear main grid array */

    /* loop for each coded region */
    while( getchar() == 'C' )
    {
        scanf("%*s %1s",code);          /* read tissue type code  */
        flin();                         /* flush rest of input line  */
        pcount=readcoords();            /* read all co-ords on edge  */
        fill(pcount);                   /* fill up grid for sub-area */
        transfer(code[0]);              /* transfer to main grid */
    }
```

```
        output(mainmap);                /* print out grid in coded form */
    }

}


/* Routine to read the coordinates of points on the edge
   of a region  (as output by 'coords')
   Returns number of points read
*/
readcoords()
{
    float x,y;
    int pcount;

    pcount=0;

    /* loop while x and y can be read */
    while( scanf("%f %f",&x,&y) == 2 )
    {
        flin();                     /* flush rest of input line */
        xp[pcount] = x;             /* store x */
        yp[pcount] = y;             /* store y */
        pcount++;                   /* increment count */
    }
    return(pcount);
}


/* This routine fills in a grid for one area. The grid consists of
 * lines at x= SIZE/2 + n*SIZE , n= 0,1,2,.....  and
 *          y= SIZE/2 + n*SIZE , n= 0,1,2,.....
 *
 * Each point where a horizontal and vertical grid
 * line intersect corresponds to an element
 * in the array bitmap. If such a point falls inside the
 * area , then the corresponding element in bitmap will be set
 *
 * This grid works best if the area just touches the x and y axes,
 * this will be the case in general for torso slices from the program
 * 'coords', if they have been processed by 'offset'.
 *
 * The algorithm used here is as follows:
 *
 *      Get two successive points on the area's boundary.
 *      Sort out which has the higher y value
 *      Compute the first and last grid rows cut by line joining
 *                      the points
 *      For each of these rows, complement each element to the right
 *                      of the line, in bitmap
 *      Repeat
 *
 * When the starting point is reached again, all elements inside
 * the area should be set to 1.
 */
```

```
fill(pcount)
int pcount;

{
    int i,j,k,h,l,firstrow,lastrow,firstcol;
    double temp;

    for(i=0; i<MAXX; i++)
        for(j=0; j<MAXY; j++)
            bitmap[i][j] = 0;   /* clear temporary grid */

    for( k=1; k<pcount; k++)    /* loop for each co-ordinate on edge */
    {
        /* find higher point */
        if( yp[k] > yp[k-1] )
        {
            h=k;
            l=k-1;
        }
        else
        {
            h=k-1;
            l=k;
        }

        /* calculate first grid line cut */
        firstrow = (int)floor( (yp[l]-(SIZE/2))/SIZE ) + 1;
        /* calculate last grid line cut */
        lastrow = (int)floor( (yp[h]-(SIZE/2))/SIZE ) ;

        /* check we havent run out of array */
        if(firstrow<0 || lastrow >= MAXY )
            quit("grid: area too large - too many grid points\n");

        /* loop for each row cut */
        for(j=firstrow; j<=lastrow; j++)
        {

            /* calculate first column to right of cut */
            temp = ( xp[h]-xp[l] ) / ( yp[h]-yp[l] );
            temp *= ( ( (double)j + 0.5 )*SIZE - yp[l] );
            temp += xp[l];

            firstcol = (int)floor( (temp-SIZE/2)/SIZE ) + 1;
            if( firstcol<0 || firstcol>=MAXX )
                quit("grid: area too big - too many grid points\n");

            /* complement each element */
            for(i=firstcol; i<MAXX; i++)
                bitmap[i][j] = ( bitmap[i][j] == 0);
        }
    }
}

/* Routine to add a gridded region to the whole slice.
   Each region overwrites any old one lying in the same
```

```
   position.  It is important therefore to have the outermost
   boundary coordinates appearing first in the input to this
   program
*/
transfer(code)
char code;
{
    int i,j;

    /* copy any occupied element from bitmap to mainmap */
    for( i=0; i<MAXX; i++ )
        for( j=0; j<MAXY; j++ )
            if( bitmap[i][j] ) mainmap[i][j] = code;

}

/* Print out the grid for a slice, using run length coding
*/
output(mainmap)
char mainmap[][MAXY];
{
    int i,j,count;
    int emptyrows;              /* counter for repeated empty rows at end */
    char code,oldcode,label[80];

    emptyrows = 0;

    /* for each row */
    for(j=0; j<MAXY; j++)
    {
        count = 0;              /* zero count */
        code = mainmap[0][j];   /* get initial code */

        /* run along row */
        for(i=0; i<MAXX; i++)
        {
            /* if repeated code, inc count */
            if( mainmap[i][j] == code ) count++;
            /* otherwise ... */
            else
            {
                /* translate null codes to something printable */
                if( code == '\0' ) code = '?';

                /* print previous empty rows, if any */
                while( emptyrows )
                {
                    printf("%3d %c\n",MAXX,'?');
                    emptyrows--;
                }

                /* print last code and its' count */
                printf("%3d %c ",count,code);

                /* get new code */
                code = mainmap[i][j];
```

```
                count = 0;
                i--;
            }
        }

        if( code == '\0' ) code = '?';

        /* detect an entire empty row */
        if( code == '?' && count == MAXX )
            emptyrows++;
        /* else print last code in row */
        else
        {
            while( emptyrows )
            {
                printf("%3d %c\n",MAXX,'?');
                emptyrows--;
            }
            printf("%3d %c\n",count,code);
        }
    }
}


/* Useful routine to get rid of unwanted input */
flin()
{
    while( getchar() != '\n' )
        /* loop */ ;
}

/* Useful routine for when things go wrong */
quit(s)
char *s;
{
    fprintf(stderr,"%s",s);
    exit(1);
}
```

```
/*   grid2.c -
 *
 *   A program to read 2.5mm grid output by grid1 and convert
 *   into a non-uniform grid. The coded grid is printed to
 *   the standard output.
 *
 *   Output is in the following form, repeated as many times
 *   as there are input slices  :
 *
 *   Slice n
 *   2 A 3 B 7 D 3 A
 *   5 A 1 B
 *   1 ? 3 B 8 A
 *   .
 *   .
 *   .
 *
 *   Each number is the number of nodes of each following code type.
 *   The special code ? signifies empty elements
 *
 *   Stephen Walker
 *   Uni of Tas
 */

#include <math.h>
#include <stdio.h>
#include <ctype.h>
#include "body.h"


char ingrid[MAXX][MAXY];   /* initial grid */
char outgrid[NCOL][NROW];  /* final grid */

/* these arrays contain values in units of initial grid mesh size */
int rowht[NROW];            /* height of each row in final grid */
int colwd[NCOL];            /* width of each column in final grid */

/* this one is in millimetres */
int slith[NSLI];            /* thickness of each slice */

main(argc,argv)
int argc;
char *argv[];
{

    int slicenum;            /* slice number */
    int i;
    char c;

    /* read non-uniform grid spacings */
    getgridsizes(argc,argv);

    /* output grid sizes */
    printf("Row heights\n");
    for(i=0; rowht[i] != 0; i++)
```

```
        printf("%d\n",rowht[i]);

    printf("Column widths\n");
    for(i=0; colwd[i] != 0; i++)
        printf("%d\n",colwd[i]);

    printf("Thickness of slices (mm)\n");
    for(i=0; slith[i] != 0; i++)
        printf("%d\n",slith[i]);

    /* run up to start of first slice */
    while( (c=getchar()) != 'S' && c != EOF )
        /* loop */ ;

    if( c == EOF )
        quit("grid2: EOF before first slice\n");

    /* main loop - continue while each slice number can be read */
    while( scanf("%*s %d",&slicenum) == 1 )
    {
        flin();                 /* flush rest of input line */

        /* read input grid (produced by grid1) */
        getingrid();
        /* make output grid */
        makeoutgrid();
        /* print this slice number  */
        printf("Slice %2d\n",slicenum);
        /* print grid */
        output();

    }
}


/* Routine to read the dimensions of the non-uniform grid
   from a file specified in program arguments
*/
getgridsizes(ac,av)
int ac;
char *av[];
{
    FILE *gf;
    int t,i,sum;
    char c;

    /* check a file name has been specified */
    if( ac != 2 )
        quit("Usage: grid2 gridsizefile\n");

    /* open the file */
    if( (gf=fopen(*++av, "r")) == NULL )
        quit("Can't open gridsizefile\n");

    /* run up to first 'R' */
    while( (c=getc(gf)) != 'R' && c != EOF) /* loop */ ;
```

```
    if( c == EOF )
        quit("EOF before row heights\n");
    while( getc(gf) != '\n' ) /* throw away rest of line */ ;

    sum = i = 0;
    /* loop to read row heights */
    while( fscanf(gf,"%d",&t) == 1 && i < NROW )
    {
        rowht[i++] = t;   /* store height value */
        sum += t;
        if( sum >= MAXY )  /* check grid is no bigger than fine grid */
            break;
    }

    /* fill rest of rowht array with 0's */
    while( i < NROW )
        rowht[i++] = 0;

    /* read up to 'C' in input */
    while( (c=getc(gf)) != 'C' && c != EOF ) /* loop */ ;
    if( c == EOF )
        quit("EOF before column widths\n");
    while( getc(gf) != '\n' ) /* throw away rest of line */ ;

    sum=i=0;
    /* loop to read column widths */
    while( fscanf(gf,"%d",&t) == 1 && i < NCOL )
    {
        colwd[i++] = t;   /* store width value */
        sum += t;
        if( sum >= MAXX )  /* check grid size again */
            break;
    }
    while( i < NCOL )
        colwd[i++] = 0;   /* clear rest of array */

    /* read up to 'T' */
    while( (c=getc(gf)) != 'T' && c != EOF ) /* loop */ ;
    if( c == EOF )
        quit("EOF before slice thicknesses\n");
    while( getc(gf) != '\n' ) /* throw away rest of line */ ;

    i=0;
    /* loop to read slice thicknesses */
    while( fscanf(gf,"%d",&t) == 1 && i < NSLI )
    {
        slith[i++] = t;   /* store width values */
    }
    while( i < NSLI )
        slith[i++] = 0;   /* clear rest of array */
}


/* Routine to read 1 slice from standard input
 * It assumes the first line of the slice ( "Slice nn" )
 * has already been read (this is done in the main loop in main)
```

```
*/

getingrid()
{
    int x,y,n;
    char c[2];

    x=y=0;

    /* loop while we can read another code count */
    while( scanf("%d",&n) == 1 )
    {
        /* read tissue code */
        if( scanf("%1s",c) != 1 )
            quit("Illegal input grid (no code)\n");

        /* fill ingrid array with n codes */
        while( n-- )
        {
            ingrid[x++][y] = c[0];
            /* check for end of array row */
            if( (x%MAXX) == 0 )
            {
                x = 0;
                y++;
                /* move x,y to start at next array row and
                   check also at end of input line */
                if( n || getchar() != '\n' )
                    quit("Illegal input grid (line sync)\n");
            }
        }
    }

    /* check we finished at end of an array row */
    if( x )
        quit("Illegal input grid (incomplete last line)\n");

    /* fill rest with code for empty */
    for(; y<MAXY; y++)
        for(x=0; x < MAXX; x++)
            ingrid[x][y] = EMPTY;
}

/* Routine to fill up the output grid - most of the work is
 * done by the next routine `code'
 */
makeoutgrid()
{
    int x,y;
    char code();

    /* loops for each element in outgrid */
    for( y=0; y<NROW; y++)
    {
        for(x=0; x<NCOL; x++)
        {
```

```
            outgrid[x][y] = code(x,y);
        }
    }

}

/* Routine to find the most common code in all the fine grid points
 * contained in one large grid box at x , y
 */
char code(x,y)
int x,y;
{
    int inx,iny,max,i,j,k;
    char c,maxcode;
    struct {
              char code;
              int count;
           }
           count[20];    /* structure for counting different
                            codes within a large grid box -
                            allows for 20 different codes -
                            this should be plenty */

    /* first find where we are in the fine grid */
    inx = iny = 0;
    for( i=0; i<x; i++)
        inx += colwd[i];  /* add up column widths */
    for( i=0; i<y; i++)
        iny += rowht[i];  /* add up row heights */

    /* check we are not outside the fine grid area */
    if( (inx+colwd[x]) >= MAXX || (iny+rowht[y]) >= MAXY )
        return(EMPTY);

    /* now run through the fine grid points and count the various
     * codes
     */

    /* clear count array */
    for(k=0; k<20; k++)
    {
        count[k].code = (char)0;
        count[k].count = 0;
    }

    /* do the counting */
    for(j=iny; j < (iny+rowht[y]); j++)
    {
        for(i=inx; i < (inx+colwd[x]); i++)
        {
            /* get each fine grid code */
            c = ingrid[i][j];
            /* loop to find correct cond count to increment */
            for( k=0; k<20; k++)
            {
                /* if code has not been seen before, insert in table
```

```
                */
                if( count[k].code == 0 )
                {
                    count[k].code = c;
                    count[k].count = 1;
                    break;
                }

                if( c == count[k].code )
                {
                    count[k].count++;
                    break;
                }
            }
        }
    }

    max = 0;
    /* find most common code */
    for(k=0; count[k].count; k++)
        if(count[k].count>max || (count[k].count==max && count[k].code!=EMPTY)
        {
            max = count[k].count;
            maxcode = count[k].code;
        }

    return(maxcode);

}


/* routine to print output grid */
output()
{
  /* This routine is very similar to the routine
     'output()' in grid1.c and has been omitted here
     to save space

     Routines 'flin()' and 'quit()' have also been omitted
  */
}
```

APPENDIX D

MODELLING SOFTWARE II

This appendix contains listings of the programs used to determine the set of nodes within a given torso (numbers.c) and generate the system of equations for the potentials throughout the model (eqns.c).

```
/* numbers.c  - Organise nodes from a set of coded
 *           cross sections
 *
 * This program reads coded slices from the standard input. It
 * assumes the input is in the form output by the program `grid2'.
 * Two slices are kept in memory at a time. Nodes are generated
 * between the slices at points where the grid lines
 * intersect. Nodes also lie on the top of the first slice and the
 * bottom of the last slice.  Those nodes which are in the
 * volume or on its' surface are numbered consecutively,
 * starting from 1.
 *
 * For each node , whether in the volume or not, a record is output
 * to a file. This record contains the node number ( 0 if not in the
 * volume ) and the tissue codes of the 8 surrounding blocks of
 * the grid
 *
 *      Stephen Walker
 *      Uni of Tas
 */

#include <stdio.h>
#include "body.h"

main()
{

    char upperslice[NCOL][NROW];   /* slice above node plane */
    char lowerslice[NCOL][NROW];   /* slice below node plane */

    skipgridsizes();               /* skip grid sizes ! */
    clearslice(upperslice);        /* clear upperslice */

    /* loop while next slice can be read */
    while( getslice(lowerslice) )
    {
        /* generate nodes  */
        makenodes(upperslice,lowerslice);
        /* move lower to upper slice */
        transfer(lowerslice,upperslice);
    }

    /* still need to make nodes on bottom of last slice , so .. */

    clearslice(lowerslice);                /* clear lower slice */
    makenodes(upperslice,lowerslice);      /* generate nodes    */
}


/* Routine to skip input up to "Slice 1" line */
skipgridsizes()
{
    char c;

    while( (c=getchar()) != 'S' && c != EOF ) /* loop */
    if( c == EOF )
```

```
        quit("numbers: EOF before first slice\n");
}


/* Routine to fill a slice with the code for empty space */
clearslice(slice)
char slice[NCOL][NROW];
{
    int i,j;

    for(i=0; i<NCOL; i++)
        for(j=0; j<NROW; j++)
            slice[i][j] = EMPTY;
}


/* Routine to read a slice from the standard input */
getslice(slice)
char slice[NCOL][NROW];
{

    int x,y,n;
    char c[2];

    if( scanf("%*s %*d") == EOF )    /* skip "Slice nn" line */
    {
        return(0);                   /* return if EOF */
    }

    x=y=0;

    /* loop while we can read another code count */
    while( scanf("%d",&n) == 1 )
    {
        if( scanf("%1s",c) != 1 )
            quit("numbers: Illegal input grid (no code)\n");
        while( n-- )
        {
            slice[x++][y] = c[0];
            /* check for end of array row */
            if( (x%NCOL) == 0 )
            {
                x = 0;
                y++;
                /* if at end of array row
                /* check also at end of input line */
                if( n || getchar() != '\n' )
                    quit("numbers: Illegal input grid (line sync)\n");
            }
        }
    }

    /* check we finished at end of array row */
    if( x )
        quit("numbers: Illegal input grid (incomplete last line)\n");
```

```
    /* fill rest with code for empty */
    for(; y<NROW; y++)
        for(x=0; x < NCOL; x++)
            slice[x][y] = EMPTY;

    return(1);
}


/* Routine to make the nodes in a plane between two slices
 * The first time it is called it creates the node file
 */

FILE *nf;       /* node file pointer */
int first=1;    /* flag for first time */
unsigned number;    /* number of node in volume */

makenodes(upperslice,lowerslice)
char upperslice[NCOL][NROW];
char lowerslice[NCOL][NROW];
{
    int x,y,i;
    int invol;          /* flag, set if node in volume */
    char code[8];       /* codes for 8 neighbouring grid boxes */
    struct numtab nrec;  /* output record buffer */

    /* create node file if first call */
    if( first )
    {
        if( (nf=fopen(NODEFILE,"w")) == NULL )
            quit("numbers: Can't create node file\n");

        number = 0;     /* initialise number */
        first = 0;      /* reset first time flag */
    }

    /* now create the nodes. There are (NCOL+1)  * (NROW+1)
     * possible nodes in each node plane
     */

    for( y=0; y<NROW+1; y++)
    {
        for(x=0; x<NCOL+1; x++)
        {
            /* start off with all surrounding codes empty */
            for(i=0; i<8; i++)
                code[i] = EMPTY;

            /* now get all the surrounding codes,
               checking all the time that we don't
               stray outside the slice
            */
            if( x>0 && y>0 )
                code[0] = upperslice[x-1][y-1];
```

```
if( x<NCOL && y>0 )
    code[1] = upperslice[x][y-1];

if( x>0 && y<NROW )
    code[2] = upperslice[x-1][y];

if( x<NCOL && y<NROW )
    code[3] = upperslice[x][y];

if( x>0 && y>0 )
    code[4] = lowerslice[x-1][y-1];

if( x<NCOL && y>0 )
    code[5] = lowerslice[x][y-1];

if( x>0 && y<NROW )
    code[6] = lowerslice[x-1][y];

if( x<NCOL && y<NROW )
    code[7] = lowerslice[x][y];

/* check if node is connected to body
   by some conductive tissue */
invol = 0;
for(i=0; i<8; i++)
    if( code[i] != EMPTY && code[i] != BONE )
    {
        invol++;
        break;
    }

/* print out a node in the torso */
if( invol )
{
    number++;
    nrec.num = number;
    for(i=0; i<8; i++)
    {
        nrec.code[i] = code[i];
    }
    fwrite(&nrec, sizeof(nrec), 1, nf);
    /* debugging
    printf("%6u ",nrec.num);
    for(i=0; i<8; i++)
        printf("%c ",nrec.code[i]);
    printf("\n");
    */
}
/* print out an empty node */
else
{
    nrec.num = 0;
    for(i=0; i<8; i++)
        nrec.code[i] = EMPTY;
    fwrite(&nrec, sizeof(nrec), 1, nf);
}
```

```
        }
    }
}


/*Routine to transfer first argument slice to second argument slice */
transfer(slice1,slice2)
char slice1[NCOL][NROW], slice2[NCOL][NROW];
{
    register int i,j;

    for(i=0; i<NCOL; i++)
        for(j=0; j<NROW; j++)
            slice2[i][j] = slice1[i][j];
}

quit(s)
char *s;
{
    fprintf(stderr,"%s",s);
    exit(1);
}
```

```
/*   eqns.c - generate the equations for a torso model
 *
 *   This program uses the files 'nodes' and 'grid'
 *   which have been set up by the programs 'numbers'
 *   and 'grid2' to create the finite difference
 *   equations which approximate the field equations
 *   in the volume.
 *
 *   It is assumed that the node file already
 *   includes the source nodes.
 *   These can be set up by using the program 'sources' .
 *
 *
 *   Two files are written. They are arranged as follows:
 *
 *   'numfile'   contains the number of equations generated, written
 *               as an ascii decimal number on line 1.
 *               line 2 contains the maximum distance between an
 *               equation and any of the variables in it,
 *               that is:
 *                  if for equation number i we have
 *
 *                      d   = max( abs(i-j) ) where j ranges over the numbers
 *                       i     j              of the variables on the RHS of
 *                                            eqn i
 *
 *                  then dist = max( d  )
 *                                 i (  i )
 *               This is used by 'solve' to make sure it always has the
 *               values involved in any equation accessible at the right
 *               time. ( Because this computer's adress space is
 *               too small ! )
 *
 *               The rest of numfile contains the unique values of
 *               the equation coefficients.
 *
 *
 *   'cfile' identifies the coefficients and variables involved
 *           in each equation.  Consider an equation
 *
 *              Vi = aVj + bVk + cVl + dVm
 *
 *           cfile will contain a sequence of integers
 *
 *               Pa j Pb k Pc l Pd m 0
 *
 *           where Pa, Pb, Pc, and Pd are pointers to the locations
 *           of the values a, b, c and d in the table in numfile.
 *           The zero entry indicates the end of the equation.
 *           In the special case of an equation for a source
 *           point, the only entry is  -1 ( no following zero ).
 *           Use of the pointers and table of coefficient values saves
 *           space, as many of the equations are identical as far as
 *           coefficients are concerned. The table is printed to the
 *           standard output.
 *
```

```
 *
 *                      Stephen Walker
 *                      University of Tasmania
 */

#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include "body.h"

int rowht[NROW];        /* grid row heights */
int colwd[NCOL];        /* grid column widths */
int slith[NSLI];        /* grid slice thicknesses in mm */
int npcount;            /* number of node planes */

struct numtab node;     /* current node record */
int x,y,z;              /* grid coords of this node */
unsigned nhbnum[6];     /* real numbers of node neighbours */
double coeff[6];        /* coefficients in equation */

unsigned number;        /* number of equations generated */
unsigned sources;       /* number of source nodes encountered */
unsigned dist;          /* maximum difference in real node numbers for
                           nodes in the same equation */

long recordnum;         /* number of current record read from node file */

FILE *nf;               /* pointer to node file */

#define HASHSIZE   5003 /* prime fairly large */

struct hasht
{
    int cnum;
    double cval;
}
table[HASHSIZE];        /* hash table for coefft values */

main()
{
    long ftell();


    getgridsizes();             /* get grid sizes */

    /* open node file */
    if( (nf=fopen(NODEFILE,"r")) == NULL )
        quit("eqns: Can't open node file\n");

    /* calculate number of slices in body by dividing the number
       of bytes in the node file by (NCOL*NROW*sizeof(node))  */
    fseek(nf,0l,2);     /* seek to eof */
    npcount = (int)( ftell(nf)/((long)(NCOL+1)*(NROW+1)*sizeof(node)) );
    fseek(nf,0l,0);
```

```
    dist = 0;


    /* loop while another node can be read */
    while( fread( &node, sizeof(node), 1, nf) == 1 )
    {
        /* generate equation if a real node */
        if( node.num )
        {
            neighbours();       /* find the neighbours of this node */
            coeffs();           /* calculate coefficients */
            output();           /* output the equation */
            recordnum++;        /* increment record number */
            resetfile();        /* seek file to read next record
                                   (getting neighbours will have moved it) */
            number++;           /* count number of equations */
        }
        /* otherwise just increment record number in node file */
        else
        {
            recordnum++;
        }
    }

    /* summarise at end */
    summary();

}



getgridsizes()
{
    /* This routine is similar to 'getgridsizes' in grid2.c
       listed in Appendix C
    */
}



/* This routine finds any neighbours of a node
 *
 * Neighbours are looked for in each of 6 directions from a node
 * For a neighbour to be stored , it must be connected by a non-empty
 * grid box to the node. The routine `connect' checks this
 */

neighbours()
{
    long offset;                /* offset of a neighbour */
    long tmp;                   /* temp for calculating max distance */
    struct numtab nbnode;       /* neighbour node */
    int i;

    /* clear neighbour array */
    for(i=0; i<6; i++)
```

```
{
    nhbnum[i] = 0;
}

/* return if source node */
if( node.code[0] == SOURCE )
    return;

/* calculate x,y,z co-ordinates of node */
z = (int)(recordnum / ((NCOL+1)*(NROW+1)));
y = (int)(recordnum % ((NCOL+1)*(NROW+1)));
x = y % (NCOL+1);
y /= (NCOL+1);

/* now find the neighbours */

/* left */
if( x > 0 && connect(0,2,4,6) )
{
    offset = (x-1) + y*(NCOL+1) + (long)z*(NCOL+1)*(NROW+1);
    fseek(nf,offset*sizeof(node),0);
    fread(&nbnode,sizeof(node),1,nf);
    nhbnum[0] = nbnode.num;
}

/* right */
if( x < (NCOL+1)-1 && connect(1,3,5,7) )
{
    offset = (x+1) + y*(NCOL+1) + (long)z*(NCOL+1)*(NROW+1);
    fseek(nf,offset*sizeof(node),0);
    fread(&nbnode,sizeof(node),1,nf);
    nhbnum[1] = nbnode.num;
}

/* forward */
if( y < (NROW+1)-1 && connect(2,3,6,7) )
{
    offset = x + (y+1)*(NCOL+1) + (long)z*(NCOL+1)*(NROW+1);
    fseek(nf,offset*sizeof(node),0);
    fread(&nbnode,sizeof(node),1,nf);
    nhbnum[2] = nbnode.num;
}

/* backward */
if( y > 0 && connect(0,1,4,5) )
{
    offset = x + (y-1)*(NCOL+1) + (long)z*(NCOL+1)*(NROW+1);
    fseek(nf,offset*sizeof(node),0);
    fread(&nbnode,sizeof(node),1,nf);
    nhbnum[3] = nbnode.num;
}

/* above */
if( z > 0 && connect(0,1,2,3) )
{
    offset = x + y*(NCOL+1) + (long)(z-1)*(NCOL+1)*(NROW+1);
```

```
        fseek(nf,offset*sizeof(node),0);
        fread(&nbnode,sizeof(node),1,nf);
        nhbnum[4] = nbnode.num;
    }

    /* below */
    if( z < (npcount-1) && connect(4,5,6,7) )
    {
        offset = x + y*(NCOL+1) + (long)(z+1)*(NCOL+1)*(NROW+1);
        fseek(nf,offset*sizeof(node),0);
        fread(&nbnode,sizeof(node),1,nf);
        nhbnum[5] = nbnode.num;
    }

    /* calculate maximum distance to neighbours */
    for(i=0; i<6; i++)
    {
        if( nhbnum[i] )
        {
            tmp = (long)nhbnum[i] - (long)node.num;
            if( tmp < 0l ) tmp = -tmp;
            if( tmp > (long)dist )
                dist = (unsigned)tmp;
        }
    }
}


/* checks that a node is connected to a neighbour by some
   conductive tissue
*/
connect(a,b,c,d)
int a,b,c,d;
{
    int ind[4];
    int i;

    ind[0] = a;
    ind[1] = b;
    ind[2] = c;
    ind[3] = d;

    for(i=0; i<4; i++)
        if( node.code[ind[i]] != EMPTY && node.code[ind[i]] != BONE )
            return(1);

    return(0);
}


/* This routine calculates the coefficients in the equation
 */

coeffs()
{
    /* conductances of half of 8 surrounding boxes in each of
```

```
 * x , y , and z directions */
double cond[8][3];

double cd[6];  /* conductance to each neighbouring node */
double temp,k;
int i;

/* first check if source node */
if( node.code[0] == SOURCE )
{
    sources++;
    return;
}

conduct(cond);    /* calculate conductances of each surrounding partial
                   * box in each co-ordinate direction. Note that
                   * conductance will be zero for any empty box */

/* clear coefficient and conductance arrays */
for( i=0; i<6; i++)
{
    coeff[i] = 0.0;
    cd[i] = 0.0;
}

/* calculate conductances to each neighbour
 * Note that if there is a neighbour, at least one partial conductance
 * must be non-zero */

/* left */
if( nhbnum[0] )
{
    temp = cond[0][0];
    temp += cond[2][0];
    temp += cond[4][0];
    temp += cond[6][0];
    cd[0] =  temp;
}

/* right */
if( nhbnum[1] )
{
    temp = cond[1][0];
    temp += cond[3][0];
    temp += cond[5][0];
    temp += cond[7][0];
    cd[1] =  temp;
}

/* forward */
if( nhbnum[2] )
{
    temp = cond[2][1];
    temp += cond[3][1];
    temp += cond[6][1];
    temp += cond[7][1];
```

```
        cd[2] =  temp;
    }

    /* back */
    if( nhbnum[3] )
    {
        temp = cond[0][1];
        temp += cond[1][1];
        temp += cond[4][1];
        temp += cond[5][1];
        cd[3] =  temp;
    }

    /* above */
    if( nhbnum[4] )
    {
        temp = cond[0][2];
        temp += cond[1][2];
        temp += cond[2][2];
        temp += cond[3][2];
        cd[4] =  temp;
    }

    /* below */
    if( nhbnum[5] )
    {
        temp = cond[4][2];
        temp += cond[5][2];
        temp += cond[6][2];
        temp += cond[7][2];
        cd[5] =  temp;
    }

    /* now calculate coefficients */

    k=0.0;
    for(i=0; i<6; i++)
        k += cd[i];

    for( i=0; i<6; i++ )
    {
        coeff[i] = cd[i] / k;
    }
}


/* This routine calculates the conductance of each half-box
 * in each of the co-ordinate directions */

conduct( cond )
double cond[8][3];
{
    double xl;    /* dimension of box in x direction */
    double yl;    /* dimension of box in y direction */
    double zl;    /* dimension of box in z direction */
    double temp;
```

```
    char co;
    int i;

    /* clear conductances */
    for( i=0; i<8; i++)
    {
        cond[i][0] = 0.0;
        cond[i][1] = 0.0;
        cond[i][2] = 0.0;
    }

    /* loop for each of the 8 partial boxes around
       a node
     */
    for( i=0; i<8; i++ )
    {
        if( (co = node.code[i]) != EMPTY && co != BONE )
        {
            /* get dimensions of box */
            dimension(i,&xl,&yl,&zl);
            temp = (double)resist( co );
            /* calculate conductances in each co-ordinate direction
            */
            cond[i][0] = ( (yl/2)*(zl/2) )/(temp*xl);
            cond[i][1] = ( (xl/2)*(zl/2) )/(temp*yl);
            cond[i][2] = ( (xl/2)*(yl/2) )/(temp*zl);
        }
    }

    /* DEBUGGING
    fprintf(stderr,"Dim %f %f %f\n",xl,yl,zl);
    for( i=0; i<8; i++ )
    {
        fprintf(stderr,"%f %f %f\n",cond[i][0],cond[i][1],cond[i][2]);
    }
    */
}


/* This routine calculates the dimensions of the i'th box next to
 * the current node
 * No checking of subscripts for the rowht, colwd, or slith arrays
 * is done as the routine should only be called if the box exists
 * and is non empty.
 *
 * Dimensions returned are in centimetres ( to be compatible with
 * resistances expressed in ohm.cm )
 */
dimension(i,xl,yl,zl)
int i;
double *xl, *yl, *zl;
{
    /* switch depending on which box around node it is */
    switch( i )
    {
        case 0 :
```

```
{
    *xl = (double)colwd[x-1]*SIZE/10.0;
    *yl = (double)rowht[y-1]*SIZE/10.0;
    *zl = (double)slith[z-1]/10.0;
    return;
}

case 1 :
{
    *xl = (double)colwd[x]*SIZE/10.0;
    *yl = (double)rowht[y-1]*SIZE/10.0;
    *zl = (double)slith[z-1]/10.0;
    return;
}

case 2 :
{
    *xl = (double)colwd[x-1]*SIZE/10.0;
    *yl = (double)rowht[y]*SIZE/10.0;
    *zl = (double)slith[z-1]/10.0;
    return;
}

case 3 :
{
    *xl = (double)colwd[x]*SIZE/10.0;
    *yl = (double)rowht[y]*SIZE/10.0;
    *zl = (double)slith[z-1]/10.0;
    return;
}

case 4 :
{
    *xl = (double)colwd[x-1]*SIZE/10.0;
    *yl = (double)rowht[y-1]*SIZE/10.0;
    *zl = (double)slith[z]/10.0;
    return;
}

case 5 :
{
    *xl = (double)colwd[x]*SIZE/10.0;
    *yl = (double)rowht[y-1]*SIZE/10.0;
    *zl = (double)slith[z]/10.0;
    return;
}

case 6 :
{
    *xl = (double)colwd[x-1]*SIZE/10.0;
    *yl = (double)rowht[y]*SIZE/10.0;
    *zl = (double)slith[z]/10.0;
    return;
}

case 7 :
```

```
        {
            *xl = (double)colwd[x]*SIZE/10.0;
            *yl = (double)rowht[y]*SIZE/10.0;
            *zl = (double)slith[z]/10.0;
            return;
        }
    }

}


/* Convert a tissue code into a resistance value in ohm.cm
   Done by looking up a table defined in body.h
*/
resist(code)
char code;
{
    char c;
    int i;

    i=0;
    while( (c=codetab[i].co) )
    {
        if( code == c )
            return( codetab[i].val );
        i++;
    }

    quit("eqns: Unknown resistance code\n");

}

/* This routine writes the coded equation out to the CFILE
 * It calls a hashing routine to convert coefficients to an index
 * in a table of values
 *
 * CFILE is created the first time the routine is called
 */


output()
{
    static int first = 1;
    static FILE *cfd;
    unsigned cfnum[6];  /* coefficient indices */
    unsigned zero = 0;  /* store for zero written at end of each eqn */
    int i,j;

    /* if first call, create CFILE */
    if( first )
    {
        if( (cfd=fopen(CFILE,"w")) == NULL )
            quit("eqns: Can't create CFILE\n");
        first = 0;
    }
```

```
    /* insert values in hash table and get indices (cfnum) */
    hash(coeff,cfnum);

    /* write out equation */
    for( i=0; i<6; i++ )
    {
        if( nhbnum[i] )
        {
              fwrite(&cfnum[i],2,1,cfd);
              fwrite(&nhbnum[i],2,1,cfd);
        }
    }
    fwrite(&zero,2,1,cfd);
}


/* This routine hashes the coefficients of an equation into a
 * table. The hash function is fairly primitive - coefficients
 * are all between 0 and 1 , so
 *
 *              hash = (int)( coefft*10000 + 0.5 ) % HASHSIZE
 *
 */


hash(cf,cn)
double cf[];
unsigned cn[];
{
    static int num = 0;  /* count of unique values */
    unsigned hash;
    unsigned attempts;
    int i;
    double diff;


    /* loop for each coefficient */
    for( i=0; i<6; i++)
    {
        /* check there is a neighbour */
        if( nhbnum[i] )
        {
            /* calculate hash */
            hash = (int)( cf[i]*10000 + 0.5 ) % HASHSIZE;
            attempts = 0;
            /* loop to put it in the table */
            do
            {
                /* check if table entry occupied */
                if( table[hash].cnum )
                {
                    /* check if same value */
                    diff = fabs( table[hash].cval - cf[i] );
                    if( diff < 0.000000001 )
                    {
```

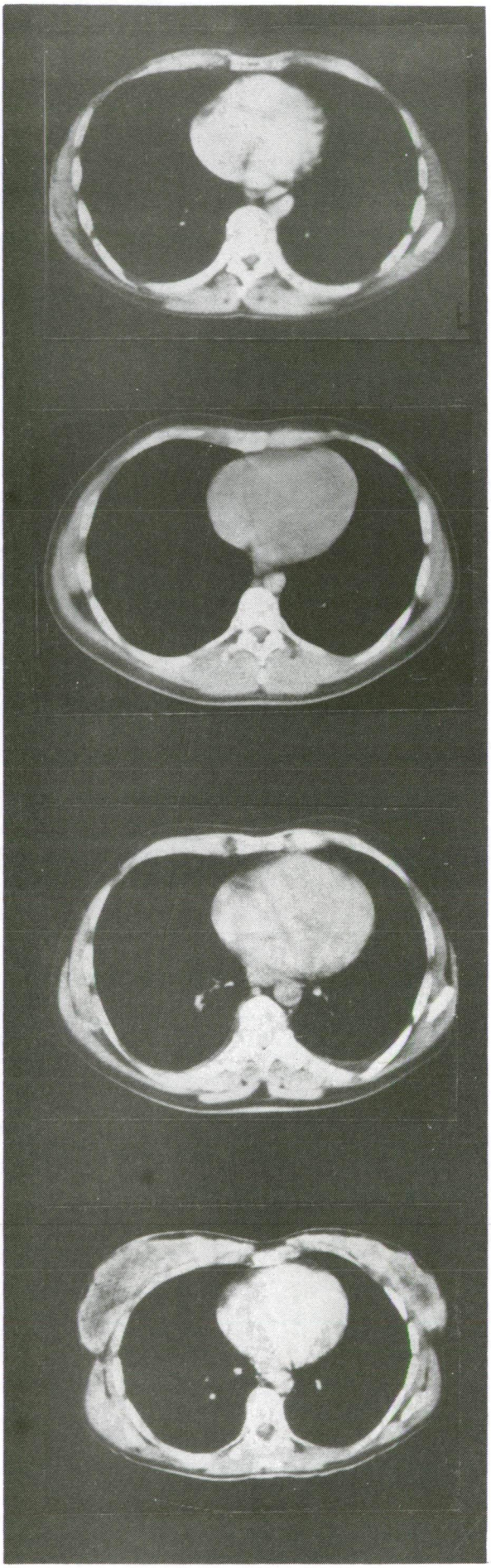
```
                        cn[i] = table[hash].cnum;
                        break;
                    }
                    /* if not, skip along table */
                    else
                        hash = (hash+5) % HASHSIZE;
                }
                /* fill unoccupied table entry */
                else
                {
                    table[hash].cnum = ++num;
                    table[hash].cval = cf[i];
                    cn[i] = num;
                    break;
                }
                attempts++;
            }
            while( attempts <= HASHSIZE );

            if( attempts > HASHSIZE )
                quit("eqns: hash table full\n");
        }
    }
}

/* seek node file to read next record */
resetfile()
{
    long offset;

    offset = recordnum * sizeof(node);
    fseek(nf,offset,0);
}


/* print numfile */
summary()
{
    int i;
    FILE *nfd;

    /* summarise number of eqns and max dist to NFILE  */
    if( (nfd=fopen(NFILE,"w")) == NULL )
        quit("eqns: Can't open NFILE\n");

    fprintf(nfd,"%d eqns generated\n",number);
    fprintf(nfd,"%d Max dist between variables\n",dist);

    /* print hash table to NFILE */
    for(i=0; i<HASHSIZE; i++)
    {
        if( table[i].cnum )
        {
            fprintf(nfd,"%d %12.10f\n",table[i].cnum,table[i].cval);
        }
    }
```

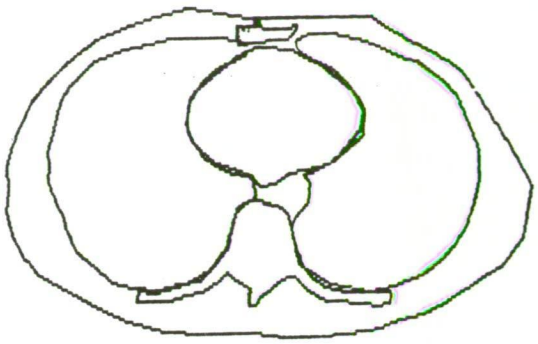
```
    fclose(nfd);

    /* warn if no sources encountered */
    if( sources == 0 )
    {
        fprintf(stderr,"eqns: No source nodes???\n");
    }
}


quit(s)
char *s;
{
    fprintf(stderr,"%s",s);
    exit(1);
}
```

APPENDIX E

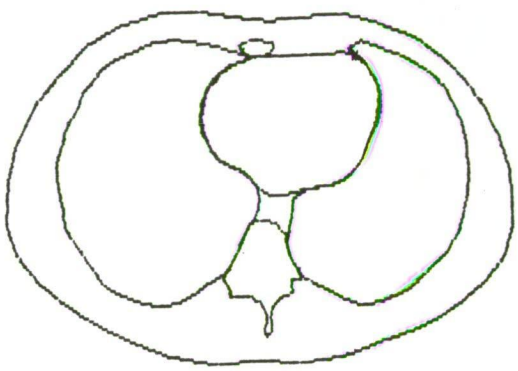
TORSO GEOMETRIES

This appendix contains a sample CT scan slice and corresponding computer plot from each of the torso models used in chapter five. The full set of CT scans for each torso were not included due to the huge amount of data involved. This data is available in digitised form on request.

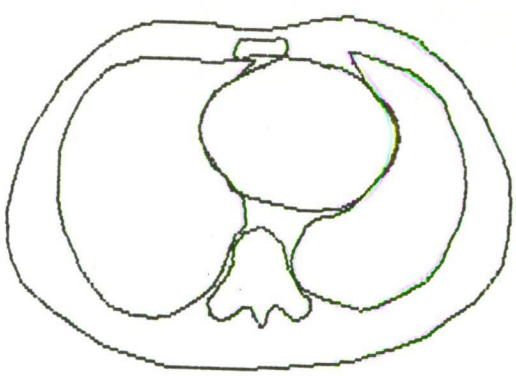




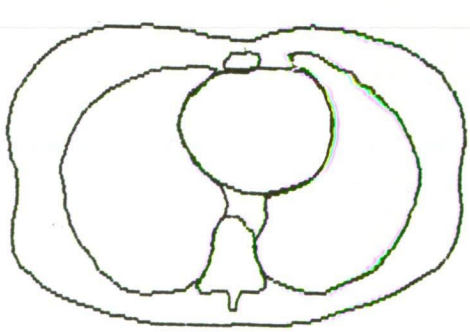
MODELS 1-4



MODEL 5



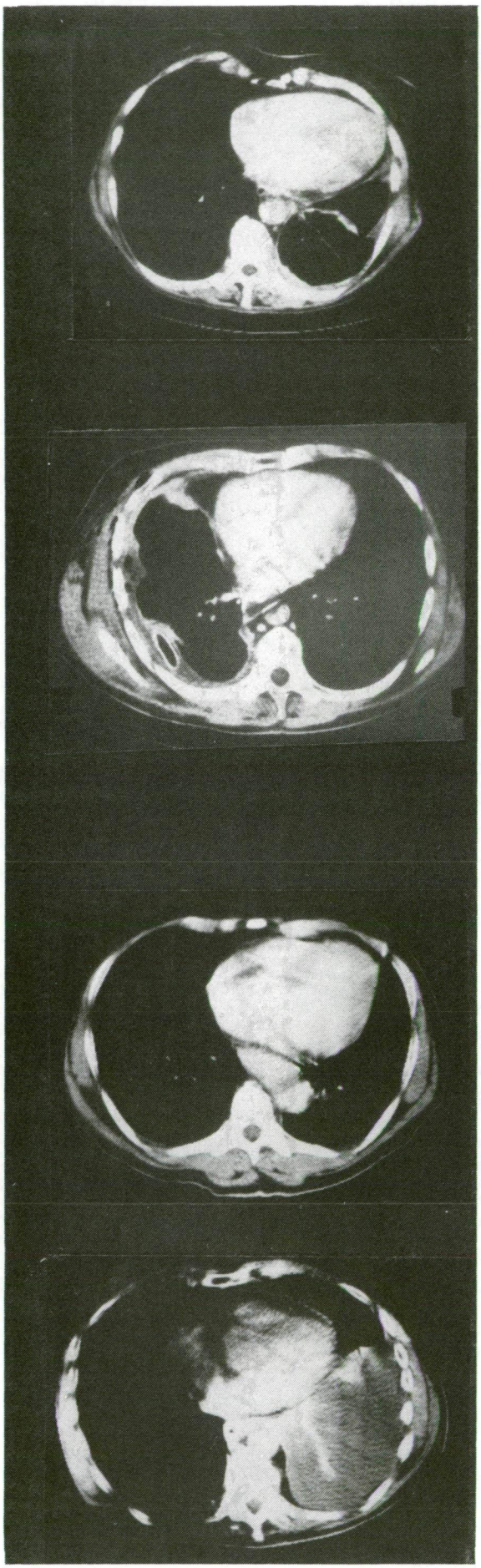
MODEL 6

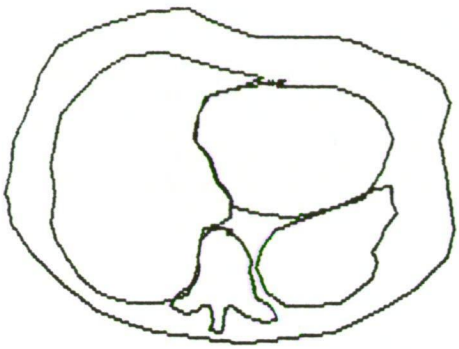


MODEL 7

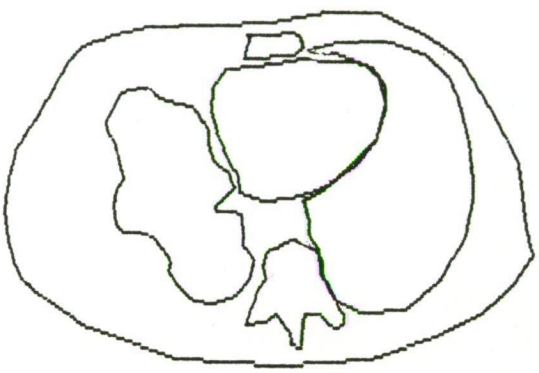
R L

Figure E.1 (2 pages). Sample CT scan image and corresponding digitised slice for each torso model

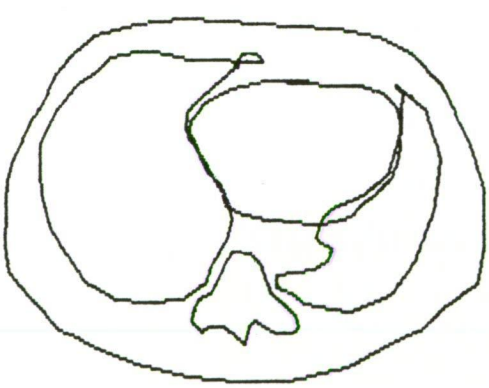




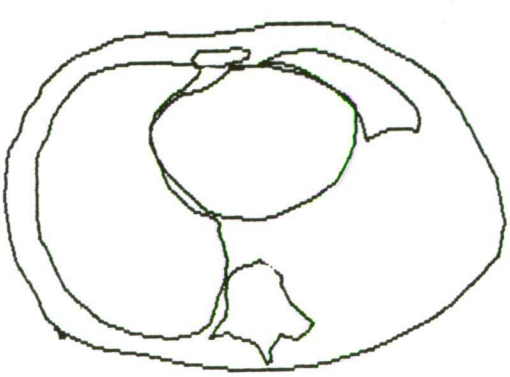
MODEL 8



MODEL 9



MODEL 10



MODEL 11

R L

Figure E.1 continued.

APPENDIX F

FORWARD TRANSFER MATRICES

This appendix contains the forward transfer matrices for each of the torso models used in chapter five. The values of all the matrix elements have all been scaled so that when epicardial potentials are specified in millivolts, torso surface potentials are obtained in microvolts.

Forward Transfer Matrix, Model 1

Columns 1 - 12

36.1	49.4	171.6	190.3	85.9	10.9	12.1	23.8	30.8	23.6	6.2	3.5
71.8	78.4	182.0	150.7	86.2	23.1	22.4	24.4	17.2	16.9	12.4	5.5
62.9	83.9	174.7	105.0	53.5	25.8	34.3	31.9	13.2	10.6	14.7	8.9
47.2	73.6	169.7	93.3	42.4	20.5	34.7	37.9	13.6	9.0	12.2	10.0
40.0	67.2	165.8	88.3	37.8	17.8	33.8	41.0	14.0	8.4	10.8	10.4
36.7	64.0	168.2	89.3	36.6	16.2	32.0	42.8	14.9	8.3	9.9	10.1
29.4	54.6	179.1	105.6	37.8	11.9	23.8	45.2	20.2	9.3	7.3	7.8
23.1	39.0	150.7	138.0	46.0	8.3	13.7	34.1	33.6	14.2	5.1	4.6
23.4	36.2	139.5	151.9	55.2	8.0	11.3	28.2	37.7	18.3	4.8	3.7
33.3	30.4	96.2	170.2	149.0	12.3	7.6	13.9	36.3	62.4	7.2	2.2
164.8	68.9	104.3	101.6	139.0	52.4	19.8	13.2	11.0	28.1	25.3	4.3
110.5	87.7	111.1	70.2	54.7	58.9	44.4	20.1	8.5	12.4	32.7	9.7
56.0	78.9	133.6	71.4	37.8	30.3	48.9	31.5	10.4	8.4	18.6	13.2
36.2	62.5	143.8	73.5	31.9	17.9	37.8	40.2	12.4	7.4	11.3	12.2
33.6	59.8	148.9	76.0	31.7	16.0	34.6	42.7	13.2	7.5	10.1	11.5
26.5	51.0	169.3	89.6	32.2	11.2	24.7	51.8	18.1	8.1	7.0	8.5
20.3	34.9	139.1	124.7	39.3	7.5	13.2	36.2	34.2	12.5	4.7	4.6
19.4	30.1	117.9	134.0	46.8	6.9	10.1	26.9	40.2	17.1	4.3	3.5
14.4	11.7	36.9	66.4	74.3	8.6	3.8	7.5	27.9	91.8	6.1	1.3
38.8	10.7	17.7	22.1	42.1	75.4	5.2	3.2	5.2	54.6	89.4	1.3
52.8	31.2	38.6	24.4	21.3	68.4	28.5	9.5	4.0	8.3	60.4	8.0
36.4	45.5	73.2	39.7	21.7	28.8	44.8	22.8	7.5	6.1	21.7	15.6
26.4	44.5	96.5	49.8	22.0	15.6	35.3	33.6	10.1	5.9	10.8	13.3
25.1	45.5	112.2	56.8	23.6	13.2	31.7	40.3	11.5	6.2	8.8	12.1
17.9	34.5	121.0	66.9	23.2	8.1	19.4	55.8	17.8	6.9	5.4	7.8
14.5	24.1	95.5	93.5	29.6	5.8	10.1	30.4	33.9	11.1	3.8	3.8
13.9	20.8	80.1	97.0	35.5	5.5	8.0	21.6	38.5	15.9	3.6	3.0
5.9	8.0	25.4	26.8	14.1	3.6	4.7	10.0	12.6	11.8	2.9	2.1
4.9	6.5	18.4	15.9	7.7	4.0	4.7	8.1	6.7	5.4	3.8	2.1
9.1	11.1	25.8	17.8	8.2	8.3	9.7	11.4	6.0	3.7	7.5	4.2
12.2	17.0	38.6	24.4	10.9	9.6	15.4	16.9	7.5	4.3	7.9	6.8
13.3	21.2	51.1	30.5	12.9	8.8	18.3	22.9	8.9	4.7	6.7	8.3
13.2	22.7	59.7	34.8	14.1	7.8	18.1	28.3	10.2	5.0	5.8	8.3
11.5	20.3	66.6	45.7	16.3	5.9	13.1	32.7	16.2	6.3	4.2	5.8
9.5	15.4	55.7	53.7	18.6	4.6	8.1	21.8	23.8	8.5	3.2	3.4
8.5	12.9	45.6	49.5	19.3	4.2	6.6	16.6	23.6	10.5	3.0	2.8
6.9	10.4	32.4	29.5	12.5	4.1	6.8	13.9	13.2	7.5	3.2	3.1
7.0	10.4	30.7	25.3	10.6	4.6	7.5	13.8	10.6	5.9	3.7	3.4
8.3	12.4	33.5	24.5	10.2	5.9	9.9	15.6	9.2	4.8	4.8	4.5
9.5	14.4	37.9	26.2	10.9	6.5	11.8	17.7	9.4	4.8	5.2	5.4
10.2	16.2	42.8	28.6	11.7	6.6	13.1	20.3	9.9	4.9	5.1	6.1
10.3	17.0	46.6	30.8	12.3	6.4	13.4	22.7	10.6	5.1	4.9	6.3
9.7	16.4	51.3	38.2	14.2	5.4	11.1	24.4	14.7	6.1	4.0	5.1
8.5	13.6	46.1	42.3	15.5	4.5	8.1	19.4	18.8	7.5	3.3	3.5
7.9	12.2	40.8	39.4	15.3	4.2	7.2	16.7	18.1	8.2	3.2	3.2

Forward Transfer Matrix, Model 1

Columns 13 - 25

4.3	14.4	9.3	12.7	8.5	7.4	17.5	13.7	40.6	43.6	25.9	95.5	62.7
4.2	8.1	6.2	23.7	12.9	7.4	10.2	9.7	47.0	48.1	23.1	63.3	44.8
6.0	6.7	4.1	29.6	20.6	10.7	9.3	7.4	65.0	72.7	32.1	69.4	47.1
7.5	7.2	3.7	26.0	23.4	13.6	10.4	6.9	72.3	88.9	40.1	82.3	53.7
8.3	7.5	3.5	23.9	24.4	15.2	11.1	6.8	75.7	97.0	44.4	89.6	57.4
8.6	8.0	3.5	22.2	23.7	15.7	11.8	6.8	74.9	97.2	45.6	93.9	59.1
8.7	10.7	4.0	16.8	18.6	15.4	15.2	7.5	66.3	85.6	45.2	110.3	63.8
7.0	17.1	6.2	11.9	11.7	12.3	23.1	10.7	57.0	67.2	41.9	145.6	78.0
5.9	18.9	7.9	11.1	9.7	10.5	24.9	12.9	52.7	59.2	38.2	148.1	81.8
2.8	17.9	23.1	13.6	5.6	5.2	19.7	30.6	35.2	32.3	20.0	87.3	85.7
2.4	5.4	8.9	42.3	10.5	4.4	6.6	13.8	47.1	33.5	14.4	39.2	38.7
4.1	4.4	4.6	62.0	23.5	7.5	6.2	8.6	81.2	64.6	23.8	48.7	40.0
6.7	5.6	3.4	39.5	30.8	12.5	8.4	6.7	87.5	98.0	38.3	72.1	51.1
8.8	6.9	3.2	25.9	29.1	16.6	10.6	6.5	87.4	115.0	49.7	92.1	61.0
9.3	7.4	3.2	23.4	27.5	17.4	11.2	6.5	84.8	113.8	51.1	96.2	62.4
10.3	9.9	3.6	16.7	20.6	18.3	14.7	7.0	72.2	96.9	51.4	114.0	66.3
7.8	18.0	5.6	11.5	11.9	13.6	25.0	10.1	61.1	72.7	46.7	161.9	82.8
6.1	21.2	7.7	10.4	9.4	11.1	29.0	13.2	57.4	63.6	42.9	177.4	93.2
2.0	21.3	58.3	12.9	3.8	4.2	26.9	97.3	42.5	30.7	20.2	103.4	225.8
0.8	3.3	36.6	159.9	3.6	1.6	4.2	129.0	93.9	16.3	7.1	24.2	153.8
2.5	2.4	3.9	187.9	26.6	5.4	3.8	9.7	219.5	78.0	21.0	38.7	45.1
6.1	4.5	2.7	57.1	46.4	13.1	7.6	6.3	149.4	155.6	46.9	78.6	62.1
8.9	6.1	2.7	27.8	35.8	18.9	10.2	6.3	118.0	162.5	62.7	103.1	73.1
10.3	6.9	2.9	22.4	31.3	21.0	11.3	6.4	103.7	149.0	65.4	109.2	73.4
13.4	10.7	3.3	14.2	20.9	25.9	17.2	7.1	84.6	119.4	70.9	145.8	82.0
7.9	20.3	5.3	10.3	10.9	14.7	31.1	10.5	70.9	81.0	57.2	220.2	103.7
5.8	23.2	7.8	9.6	8.7	11.5	35.4	14.5	66.2	70.0	49.9	234.8	119.1
3.6	9.9	8.2	10.7	7.2	9.2	19.1	27.0	109.1	80.2	51.8	221.4	314.8
3.0	5.1	3.7	20.0	7.6	8.3	10.0	18.0	239.4	98.8	50.1	140.9	307.1
4.1	4.4	2.2	36.5	15.8	11.2	8.3	7.4	330.4	176.1	56.8	108.1	115.8
5.9	5.3	2.3	30.0	24.7	15.5	9.9	6.7	218.5	215.5	69.9	121.6	102.6
7.9	6.1	2.5	22.0	28.1	20.0	11.3	6.7	158.9	213.5	81.5	133.2	100.4
9.6	6.9	2.6	17.9	27.0	24.2	12.7	6.8	134.0	195.2	90.8	143.6	100.7
10.5	11.1	3.3	13.1	18.6	24.9	20.0	8.0	106.6	143.4	89.8	195.4	110.5
7.1	16.9	4.5	10.2	10.8	15.7	31.0	10.5	89.9	98.4	71.6	275.1	132.0
5.5	17.4	5.9	9.6	9.1	12.7	32.9	13.7	87.7	88.5	62.9	292.3	158.9
5.1	10.1	4.6	12.1	10.7	13.3	19.8	13.9	130.8	116.2	71.7	239.1	209.0
5.2	8.0	3.6	15.0	12.3	14.0	15.6	11.5	170.8	139.9	75.7	200.8	194.2
5.8	6.8	2.8	19.4	16.4	15.7	13.1	8.6	200.7	178.5	78.8	165.5	143.9
6.5	6.8	2.7	19.7	19.5	17.4	13.0	7.9	184.0	193.3	82.1	159.6	127.8
7.4	7.1	2.7	18.2	21.5	19.6	13.5	7.7	161.3	195.2	87.4	161.6	121.3
8.2	7.5	2.8	16.5	21.7	21.8	14.2	7.7	145.4	188.3	92.6	167.5	119.4
8.5	10.6	3.3	13.3	17.0	21.5	19.8	8.6	120.3	149.5	90.1	210.0	126.8
6.8	13.9	4.2	11.0	11.8	16.2	26.5	10.5	105.3	113.4	77.6	266.2	145.6
5.9	13.6	4.7	10.8	10.7	14.4	26.4	12.1	106.4	108.0	72.7	272.9	164.9

Forward Transfer Matrix, Model 2

Columns 1 - 12

42.8	53.9	121.2	214.4	101.7	10.0	24.2	33.3	67.8	23.2	4.9	6.9
94.6	104.5	137.5	145.3	99.6	22.5	45.8	36.0	35.3	14.5	10.4	11.5
82.8	123.0	140.6	96.2	58.4	26.7	69.7	48.0	25.5	8.6	13.6	18.2
63.3	112.4	142.9	88.1	46.9	22.2	72.5	57.3	25.6	7.6	12.0	20.9
55.0	105.2	142.6	84.7	42.4	20.0	72.2	62.0	25.8	7.2	11.1	22.0
51.5	100.9	146.3	87.3	41.8	18.5	69.2	64.6	27.3	7.3	10.4	21.5
43.2	84.4	155.9	113.1	45.5	14.2	53.4	65.9	39.7	9.1	7.8	16.9
29.7	47.4	126.0	179.7	58.8	8.2	25.8	45.8	75.9	15.7	4.4	8.2
28.5	41.7	112.8	193.7	67.3	7.4	21.4	38.8	83.0	19.1	3.9	6.7
34.0	28.7	62.6	208.9	203.9	8.1	12.1	16.9	69.1	68.7	3.9	3.4
243.7	85.2	73.2	92.0	188.4	49.4	31.9	16.3	17.6	23.3	18.5	6.8
155.9	130.0	81.9	56.4	55.3	69.9	84.3	27.4	13.5	7.8	34.2	18.0
72.4	122.1	107.4	60.9	37.8	33.8	99.1	46.7	17.5	5.9	19.4	26.8
49.0	99.4	122.8	67.4	34.2	20.5	81.6	61.7	21.5	6.0	12.3	26.7
46.0	95.3	129.3	71.1	34.5	18.4	75.7	66.2	23.1	6.2	10.9	25.4
37.7	80.6	151.8	90.7	36.6	13.3	56.9	79.5	33.3	7.4	7.7	19.5
25.1	41.4	118.9	166.6	49.3	7.2	23.8	49.6	79.3	13.9	4.0	8.0
22.8	33.7	96.3	177.8	57.5	6.2	18.2	37.2	90.7	18.7	3.4	6.0
10.3	8.1	18.8	68.1	89.2	4.4	4.4	7.2	48.9	116.1	2.9	1.5
56.7	8.9	7.7	16.2	59.3	88.0	6.2	2.6	6.6	65.4	105.2	1.5
60.5	35.5	20.8	13.8	15.0	82.7	46.8	10.1	4.4	4.1	86.3	13.6
42.4	65.8	54.5	30.7	18.5	32.7	89.4	32.4	11.1	3.7	26.3	32.8
34.0	70.3	80.9	43.7	21.7	18.3	78.2	52.5	16.4	4.4	12.8	31.1
33.3	72.9	98.2	51.9	24.3	15.4	71.7	65.2	19.5	4.9	10.1	28.5
24.3	54.6	112.8	71.0	25.6	9.5	46.2	88.4	33.7	6.3	5.9	19.1
17.0	28.1	84.3	131.8	36.9	5.3	18.0	44.9	82.0	12.6	3.1	6.6
15.4	22.9	65.6	135.1	44.3	4.6	13.8	30.5	90.5	18.7	2.7	5.0
6.2	9.9	20.9	34.5	16.3	3.2	9.2	14.3	27.3	13.2	2.7	4.2
5.9	9.3	16.3	19.9	8.8	4.3	10.2	12.4	14.4	5.8	4.5	4.8
11.0	16.8	22.4	19.3	8.4	9.4	21.5	17.5	11.9	3.4	9.7	9.9
15.6	27.1	34.0	25.2	11.2	11.6	35.5	26.4	14.1	3.7	10.6	16.5
18.2	36.0	47.5	32.0	13.9	11.1	44.4	37.7	16.8	4.2	9.0	20.9
18.5	39.3	57.9	37.7	15.7	9.8	44.8	48.5	19.6	4.6	7.4	21.2
15.4	32.0	64.4	57.7	19.4	6.8	30.9	54.1	35.9	6.6	4.8	14.2
11.8	20.4	52.9	81.0	24.6	4.5	16.2	34.8	59.1	10.3	3.1	6.8
10.1	16.4	41.2	74.2	25.9	3.9	12.9	25.1	57.6	13.5	2.7	5.5
8.4	14.8	29.6	40.3	15.4	4.3	14.8	21.8	30.3	8.8	3.5	6.9
8.8	15.8	28.4	33.2	12.8	5.1	17.1	22.1	23.8	6.6	4.4	8.1
10.9	19.7	31.2	30.1	11.8	6.9	23.1	25.1	19.9	5.1	6.1	11.0
12.5	23.4	35.4	31.2	12.4	7.9	28.0	28.7	19.7	4.9	6.8	13.4
13.8	26.9	40.7	33.7	13.4	8.1	31.7	33.5	20.5	4.9	6.7	15.2
14.2	28.6	45.1	36.5	14.3	7.8	32.7	38.0	22.1	5.1	6.2	15.7
12.9	25.6	49.5	50.2	17.3	6.2	26.0	40.4	33.5	6.7	4.7	12.2
10.7	19.1	43.7	61.5	20.1	4.7	17.2	31.1	45.5	9.0	3.5	7.7
9.7	16.8	37.9	57.2	19.9	4.3	15.2	26.2	43.5	10.1	3.3	6.8

Forward Transfer Matrix, Model 2

Columns 13 - 25

8.2	31.4	9.0	8.7	16.6	8.4	38.0	10.8	19.9	27.8	14.7	61.0	41.1
8.3	16.5	5.1	16.6	26.1	9.3	20.4	6.2	23.1	34.7	13.4	37.0	25.9
11.2	12.6	3.2	22.8	41.2	13.4	16.6	4.4	32.4	52.4	17.8	35.8	24.9
13.7	13.0	2.9	21.6	48.0	16.8	17.5	4.3	36.9	63.3	21.9	40.3	27.9
15.0	13.4	2.9	20.9	51.1	18.6	18.2	4.3	39.2	69.0	24.3	43.1	29.7
15.6	14.2	3.0	19.6	50.0	19.3	19.3	4.5	38.7	68.5	25.0	45.1	30.7
15.5	20.3	3.8	15.0	39.5	18.2	26.9	5.6	33.4	57.4	24.4	55.9	34.8
11.8	38.0	6.9	8.7	20.4	11.9	48.9	9.5	25.9	36.9	21.1	85.5	49.0
10.3	41.2	8.4	7.8	17.0	10.2	52.4	11.2	24.1	32.5	19.5	88.2	52.8
4.4	32.8	24.1	6.4	8.5	4.5	36.3	24.4	13.5	16.0	9.2	48.4	51.2
3.8	8.3	5.5	23.8	15.7	4.3	9.9	6.6	16.9	19.0	6.4	17.4	16.0
6.7	6.8	2.4	51.4	42.4	8.3	9.0	3.6	38.7	46.0	11.6	21.2	17.1
11.9	9.1	2.2	34.7	62.2	15.0	12.7	3.5	46.4	74.4	20.4	32.9	24.7
16.1	11.5	2.5	24.3	63.6	20.8	16.3	4.0	48.3	86.9	27.9	43.4	31.6
17.1	12.4	2.6	21.8	60.6	22.1	17.5	4.2	46.6	85.1	29.2	46.0	32.7
19.2	17.8	3.2	15.5	46.5	23.7	24.6	5.0	38.2	68.8	29.8	56.8	36.0
13.5	41.7	6.4	8.3	20.4	13.2	54.9	9.2	28.3	39.7	24.4	99.5	53.4
10.6	47.8	8.8	7.2	15.8	10.4	63.1	12.3	26.7	34.0	22.1	109.2	63.6
2.3	39.9	89.6	5.8	4.5	2.8	51.9	129.7	17.1	13.1	8.3	58.7	196.2
0.8	4.3	41.7	161.9	4.3	1.0	5.6	132.6	69.4	8.1	2.8	10.6	132.5
3.0	2.6	1.6	269.0	48.1	4.4	4.0	3.9	165.1	62.2	8.9	14.2	19.3
10.1	6.6	1.6	65.3	101.6	14.8	10.3	3.3	105.0	146.9	25.7	36.3	32.5
16.3	9.8	2.1	30.5	86.3	23.8	15.2	4.0	77.4	139.3	37.7	51.4	41.7
19.6	11.4	2.3	23.1	75.4	28.1	17.3	4.3	64.0	120.7	40.7	55.2	42.0
25.8	20.3	3.1	13.9	51.2	35.8	30.4	5.5	49.2	89.6	46.9	81.3	49.7
14.8	49.5	6.4	7.7	18.8	14.5	71.7	10.4	36.4	46.1	32.9	147.0	73.2
10.1	55.1	10.0	6.8	14.5	10.6	80.9	15.7	33.8	38.8	27.3	154.2	93.0
5.9	21.7	11.2	10.4	14.8	9.2	41.2	35.3	84.4	60.0	36.0	167.5	340.6
5.2	11.1	4.7	24.0	17.8	9.2	20.7	21.3	224.2	86.2	38.9	110.3	309.7
7.0	8.6	2.2	52.7	39.1	12.6	15.5	7.0	314.3	164.0	40.9	74.8	99.8
10.1	9.8	2.2	42.6	62.6	17.9	17.4	5.9	194.2	202.0	47.5	77.4	78.9
14.1	11.5	2.4	29.1	72.3	24.5	19.8	5.8	126.5	189.2	55.5	83.2	74.5
18.1	13.2	2.6	21.8	69.0	31.3	22.5	6.0	98.9	164.5	63.4	90.1	73.9
20.1	24.7	3.7	14.1	45.2	31.8	41.3	7.8	73.6	112.4	64.0	133.8	85.0
13.7	41.5	6.0	9.5	22.0	16.7	69.9	11.8	58.7	68.0	48.2	198.4	110.0
9.8	41.7	8.6	8.8	18.0	12.8	73.5	17.4	58.2	60.7	41.3	211.1	148.8
9.0	23.0	6.4	13.7	24.4	14.7	42.9	17.4	106.9	94.8	53.7	184.6	209.4
9.2	17.8	4.7	18.5	29.5	16.0	33.0	13.7	148.9	120.6	58.3	155.0	188.9
10.2	14.6	3.4	25.9	41.0	18.4	26.5	9.3	178.1	159.3	59.6	122.1	130.5
11.5	14.2	3.1	26.5	49.2	20.7	25.5	8.2	159.0	172.6	60.7	113.8	110.6
13.3	14.7	3.1	23.8	54.4	23.9	26.1	7.7	133.5	171.4	64.1	113.1	101.9
15.0	15.7	3.2	20.7	54.7	27.1	27.7	7.7	115.9	161.8	68.2	116.8	99.1
15.9	23.9	4.1	15.2	41.2	26.2	41.8	9.2	90.6	121.6	66.2	151.8	107.0
12.7	33.1	5.6	11.6	25.9	17.9	58.5	12.3	77.0	86.5	55.8	199.1	130.1
10.7	32.2	6.6	11.3	23.5	15.6	58.2	15.0	79.3	82.7	52.1	205.5	156.3

Forward Transfer Matrix, Model 3

Columns 1 - 12

36.6	47.6	168.5	172.5	87.7	11.5	11.6	57.6	26.6	32.9	6.0	3.3
55.5	69.6	189.0	149.9	86.4	18.1	18.9	57.0	16.3	26.8	9.4	4.8
56.1	77.3	171.4	102.4	58.9	23.5	31.1	64.2	12.5	20.3	12.9	8.0
42.1	67.8	170.4	90.3	45.6	18.7	31.4	76.1	12.6	16.1	10.8	9.0
35.6	61.7	169.4	85.6	40.1	16.1	30.4	82.5	12.8	14.4	9.5	9.2
32.5	58.5	174.2	86.8	38.6	14.5	28.3	87.6	13.4	13.8	8.6	8.8
26.2	49.9	200.9	105.5	39.6	10.1	19.2	101.8	17.3	13.7	5.9	6.1
21.9	37.7	171.9	125.2	45.3	7.7	11.9	90.1	27.3	18.3	4.3	3.9
22.8	34.7	149.6	137.7	54.8	7.9	10.2	74.0	32.3	24.1	4.3	3.3
35.9	29.8	95.4	156.1	151.4	14.0	7.2	33.4	32.3	83.9	6.8	2.0
133.8	57.5	98.2	102.2	173.6	45.4	15.7	26.6	10.5	66.8	20.6	3.4
100.1	82.5	108.9	68.9	62.1	55.7	42.1	39.2	8.1	28.6	30.2	9.1
50.6	74.2	133.5	69.2	41.2	28.5	46.4	60.4	9.7	16.6	17.1	12.4
32.2	57.7	147.2	71.1	33.8	16.3	34.8	78.5	11.4	12.8	10.0	11.2
29.8	54.9	153.7	73.6	33.3	14.4	31.3	84.3	12.1	12.4	8.8	10.3
22.6	45.1	185.9	86.5	32.2	9.3	20.0	114.9	15.5	11.6	5.6	6.8
18.6	33.1	162.2	110.4	37.5	6.7	11.2	100.7	27.1	15.5	3.9	3.8
18.7	28.7	128.5	121.4	46.1	6.8	9.1	71.9	35.0	22.1	3.8	3.1
13.9	10.6	36.0	58.1	71.0	8.9	3.3	18.1	25.1	117.9	5.0	1.1
20.4	5.1	8.7	9.8	30.1	72.0	2.5	3.3	1.9	156.4	80.4	0.6
47.4	29.2	38.5	23.5	24.4	66.2	27.5	19.0	3.7	21.3	58.0	7.7
32.9	42.9	75.2	38.5	23.6	27.4	43.2	44.4	7.0	12.1	20.5	15.0
23.5	41.3	100.6	48.2	23.2	14.4	33.0	65.4	9.4	10.1	9.8	12.5
22.1	41.7	117.6	54.8	24.5	11.9	28.9	78.5	10.6	10.1	7.7	11.0
14.2	28.7	135.2	59.9	21.2	6.3	14.7	132.5	14.6	8.7	4.0	5.9
13.2	22.7	111.4	82.2	28.0	5.3	8.8	87.7	27.6	13.6	3.2	3.3
13.3	19.8	88.3	88.1	34.8	5.3	7.2	58.8	34.4	20.3	3.2	2.7
5.3	7.5	29.2	24.3	13.5	3.4	4.3	27.2	11.2	15.6	2.5	1.9
4.3	6.0	21.5	14.7	7.4	3.7	4.3	21.1	6.0	7.8	3.3	1.9
8.1	10.3	29.3	17.0	8.5	7.9	9.0	27.1	5.6	6.1	6.9	3.9
10.9	15.8	43.0	23.5	11.3	9.0	14.3	38.2	7.0	6.9	7.3	6.3
11.8	19.6	56.5	29.5	13.3	8.1	16.8	50.0	8.4	7.3	6.0	7.6
11.5	20.6	66.3	33.7	14.3	7.0	16.1	61.6	9.6	7.4	5.0	7.4
9.1	16.7	71.9	38.1	14.3	4.8	10.4	83.3	12.9	7.6	3.3	4.8
8.6	14.4	64.9	47.1	17.5	4.2	7.3	63.3	19.7	10.4	2.8	3.1
7.9	12.2	52.3	44.6	18.6	3.9	6.1	46.4	20.8	13.3	2.6	2.6
6.2	9.8	38.1	27.0	12.1	3.8	6.2	38.0	11.6	9.9	2.8	2.8
6.2	9.8	36.1	23.6	10.4	4.2	6.8	36.6	9.5	8.0	3.3	3.2
7.4	11.5	39.0	23.4	10.3	5.4	8.9	38.9	8.5	7.0	4.3	4.1
8.4	13.3	43.6	25.2	11.0	6.0	10.7	42.8	8.7	7.0	4.7	4.9
9.0	14.8	49.0	27.5	11.9	6.0	11.8	48.2	9.2	7.1	4.5	5.5
9.0	15.4	53.5	29.7	12.4	5.6	11.7	53.9	10.0	7.3	4.2	5.5
8.1	14.1	56.1	32.7	12.9	4.6	9.4	61.5	12.1	7.7	3.3	4.4
7.6	12.7	53.8	37.2	14.6	4.1	7.4	55.1	15.6	9.4	2.9	3.3
7.2	11.5	47.8	35.5	14.7	3.9	6.7	46.7	15.7	10.4	2.8	3.0

Forward Transfer Matrix, Model 3

Columns 13 - 25

10.8	12.3	11.5	11.4	8.0	13.4	15.2	16.2	36.2	40.4	28.5	82.4	51.5
10.3	7.7	8.6	17.6	11.2	12.7	9.9	12.1	40.0	43.7	26.0	60.2	38.3
12.4	6.3	6.7	25.6	18.4	15.7	8.9	10.2	56.7	63.7	32.9	63.6	40.3
15.1	6.6	5.6	22.6	20.8	19.4	9.8	9.1	62.8	77.4	40.4	74.1	45.5
16.6	6.9	5.1	20.6	21.5	21.4	10.4	8.6	65.3	83.8	44.4	79.9	48.3
17.4	7.2	5.0	18.8	20.6	22.3	10.8	8.5	63.7	82.9	45.6	82.7	49.1
18.7	8.9	5.1	13.0	14.6	22.8	12.8	8.6	51.4	66.8	43.7	88.7	48.9
17.4	13.8	7.1	9.7	9.9	21.4	18.9	11.7	46.6	56.7	43.4	116.9	61.0
14.9	16.1	9.3	9.4	8.6	18.9	21.3	14.8	45.2	52.7	40.8	124.9	67.4
6.8	15.7	28.6	11.3	5.2	8.8	17.2	36.7	29.8	29.0	20.7	74.0	68.3
5.0	5.1	17.9	32.1	8.3	6.4	6.2	21.6	36.4	27.3	14.2	34.6	30.5
8.0	4.2	9.1	56.7	21.9	10.6	6.0	13.3	73.8	58.4	24.0	45.0	33.7
12.8	5.3	5.7	36.1	28.8	17.1	8.1	9.4	79.3	88.4	38.3	66.6	44.2
16.9	6.4	4.7	22.8	26.4	22.6	10.1	8.4	77.2	101.8	49.4	83.9	52.4
18.0	6.8	4.6	20.3	24.5	23.8	10.6	8.3	73.9	99.5	50.9	87.0	53.1
22.2	8.5	4.4	12.9	16.4	27.3	12.8	7.9	57.0	77.4	51.1	94.3	51.8
20.1	14.2	6.2	9.1	10.0	24.6	20.1	10.8	49.8	61.4	49.2	129.2	64.5
15.5	18.3	9.0	8.8	8.5	20.2	25.2	15.1	50.0	57.5	45.9	152.3	78.5
4.5	19.6	70.6	9.2	3.3	6.6	24.7	121.8	35.4	27.1	19.7	90.7	197.8
0.8	1.2	151.1	110.2	1.7	1.1	1.6	215.1	47.0	7.1	3.2	9.2	59.6
4.7	2.2	8.5	182.9	25.7	7.1	3.6	16.5	213.2	73.7	20.5	36.0	39.0
11.4	4.3	4.6	54.3	44.8	16.9	7.5	9.0	141.9	146.3	45.9	74.5	55.9
16.5	5.8	4.0	25.3	33.4	24.3	10.1	8.2	108.2	148.9	61.5	97.4	65.3
19.1	6.5	4.0	19.6	28.4	27.4	11.1	8.0	92.6	133.1	64.3	102.0	64.5
31.1	9.1	3.7	10.5	16.1	40.1	15.3	7.5	66.2	94.7	73.3	122.4	64.0
21.1	16.5	5.9	8.5	9.7	27.8	25.9	11.5	61.2	73.3	62.4	183.6	85.5
14.7	20.7	9.2	8.1	8.1	20.5	32.0	17.0	59.8	65.7	53.3	209.7	105.1
8.7	9.0	10.6	9.4	6.7	14.2	17.8	34.6	103.9	76.3	52.6	208.2	302.2
6.9	4.7	5.4	18.7	7.0	11.9	9.4	24.9	234.1	94.3	49.7	133.3	297.6
8.7	4.1	3.2	35.0	14.9	14.9	8.1	10.2	324.2	168.6	55.7	103.2	109.5
12.0	5.1	3.3	28.3	23.2	20.2	9.9	9.0	210.3	204.6	68.3	116.7	95.6
15.4	6.0	3.4	20.1	26.0	25.6	11.4	8.7	148.9	199.0	79.4	128.6	92.7
18.8	6.8	3.5	15.7	24.2	31.0	12.9	8.6	121.6	176.3	88.3	139.7	92.1
26.1	9.3	3.7	10.5	15.7	40.6	17.8	8.9	93.3	127.8	99.3	175.0	95.1
19.0	14.2	5.2	8.8	10.3	28.5	26.8	12.0	82.9	94.9	78.2	240.0	115.9
13.9	15.5	7.0	8.4	8.7	21.6	30.1	16.4	82.0	85.0	66.3	268.0	145.8
12.3	9.1	5.8	10.9	10.0	20.5	18.3	17.5	124.1	110.4	73.1	222.4	197.1
12.0	7.3	4.7	13.7	11.4	20.4	14.7	14.8	163.2	132.1	75.8	188.5	183.7
12.7	6.4	3.8	17.9	15.1	21.7	12.7	11.2	191.8	167.8	78.0	157.3	135.0
13.9	6.5	3.6	18.0	17.9	23.6	12.8	10.2	174.1	180.5	80.9	152.7	119.1
15.5	6.8	3.6	16.3	19.5	26.3	13.4	9.8	150.1	180.0	85.8	155.7	112.4
17.4	7.3	3.7	14.4	19.2	29.2	14.4	9.7	132.4	169.7	91.0	163.0	110.3
20.0	9.0	3.9	11.4	15.1	32.8	17.7	10.2	111.1	138.7	96.2	192.2	114.8
17.4	11.7	4.9	9.8	11.4	27.7	23.1	12.3	99.0	110.6	83.6	234.9	130.0
14.8	12.0	5.7	9.6	10.3	23.7	23.9	14.6	100.1	104.0	76.2	248.3	151.2

Forward Transfer Matrix, Model 4

Columns 1 - 12

45.3	55.3	122.0	198.6	102.5	10.7	25.1	52.6	61.3	24.7	5.1	7.2
79.5	95.7	141.0	145.4	97.7	19.5	42.6	54.0	36.2	17.3	9.1	10.9
78.2	115.8	136.0	98.0	62.7	25.4	65.1	62.4	26.8	11.5	12.8	16.8
59.5	105.0	138.5	90.2	49.8	21.0	67.5	73.5	27.2	9.8	11.2	19.3
51.4	97.6	138.6	87.2	44.8	18.8	66.8	79.2	27.7	9.2	10.4	20.2
47.9	92.9	142.5	90.1	44.0	17.2	63.4	83.2	29.3	9.2	9.6	19.6
39.8	75.7	157.2	116.1	47.7	12.8	46.6	90.4	40.3	10.8	6.9	14.6
30.4	48.9	133.0	161.4	57.2	8.5	26.7	76.2	66.1	15.9	4.4	8.5
29.4	42.9	116.6	177.1	66.3	7.8	22.4	64.5	74.7	19.8	4.0	7.1
38.0	29.7	62.9	198.0	204.8	9.1	12.5	26.7	64.9	73.1	3.9	3.6
216.2	75.3	68.9	93.7	224.5	44.5	27.9	21.5	17.8	38.0	16.2	5.9
149.9	125.8	79.5	57.3	59.9	68.5	82.0	34.9	14.3	12.3	33.3	17.3
69.0	117.1	104.2	62.4	40.2	32.8	95.7	58.1	18.7	8.1	18.8	25.6
45.9	93.1	119.5	69.6	36.0	19.5	76.9	77.1	23.3	7.7	11.6	25.0
42.9	88.5	125.9	73.6	36.3	17.3	70.5	83.1	25.1	7.8	10.2	23.5
34.0	71.3	151.5	93.2	37.5	11.9	49.5	106.9	34.8	8.7	6.7	16.9
25.2	42.3	127.0	147.4	47.1	7.3	24.5	85.4	68.3	13.8	3.9	8.2
23.5	34.8	100.0	163.0	56.3	6.5	19.1	62.4	82.5	19.2	3.4	6.4
10.6	8.1	18.9	63.7	87.0	4.6	4.4	11.5	46.7	123.6	2.5	1.5
33.3	4.6	3.8	6.9	45.4	81.7	3.2	1.7	2.5	148.8	92.0	0.7
58.1	34.4	20.4	13.9	16.4	81.8	46.0	12.8	4.6	7.6	85.4	13.4
40.6	63.2	53.5	31.6	19.6	32.1	87.4	40.3	11.9	5.1	25.8	32.0
32.0	66.2	79.4	45.3	22.8	17.5	74.8	65.2	18.0	5.6	12.3	29.7
30.9	67.6	96.2	54.1	25.4	14.5	67.2	80.8	21.4	6.2	9.5	26.7
20.9	46.4	113.7	69.1	24.6	8.1	38.7	125.4	33.5	6.8	5.0	15.9
17.2	28.9	90.4	116.7	35.1	5.4	18.8	78.6	71.3	12.4	3.1	7.1
15.9	23.6	68.6	124.7	43.3	4.8	14.6	51.4	84.0	19.2	2.7	5.4
6.1	9.8	22.5	32.2	15.9	3.1	9.2	23.5	25.6	14.1	2.5	4.2
5.6	9.0	17.5	19.1	8.6	4.1	9.8	19.6	13.8	6.6	4.2	4.6
10.4	16.0	23.4	19.3	8.7	9.1	20.6	25.3	12.0	4.1	9.4	9.5
14.8	25.6	34.8	25.7	11.6	11.2	33.9	36.4	14.6	4.4	10.2	15.8
17.0	33.6	48.1	33.1	14.5	10.6	41.9	50.4	17.9	5.0	8.6	19.7
17.0	35.9	58.5	39.2	16.3	9.1	41.1	64.4	21.3	5.4	6.8	19.5
13.2	27.3	62.9	50.2	17.5	5.9	26.6	80.6	31.7	6.5	4.2	12.5
11.7	20.7	56.7	71.6	23.2	4.5	16.8	60.3	51.7	10.2	3.1	7.3
10.2	16.7	43.9	68.1	25.2	3.9	13.4	42.5	53.1	13.9	2.7	5.7
8.2	14.6	32.0	37.8	15.1	4.2	14.6	35.7	28.3	9.2	3.4	6.8
8.5	15.3	30.6	31.8	12.7	4.9	16.4	34.9	22.7	7.1	4.2	7.8
10.3	18.7	33.1	29.8	12.0	6.6	21.8	37.7	19.8	5.7	5.8	10.4
11.8	22.0	37.2	31.3	12.7	7.5	26.3	41.9	20.0	5.5	6.4	12.6
12.9	25.0	42.4	34.2	13.8	7.6	29.4	48.0	21.3	5.6	6.2	14.1
13.1	26.2	46.8	37.4	14.7	7.2	29.7	54.3	23.2	5.9	5.6	14.3
11.6	22.8	49.1	44.3	16.0	5.7	23.5	60.0	29.7	6.7	4.3	11.1
10.5	19.1	46.7	54.6	18.9	4.7	17.6	52.6	39.9	9.0	3.4	8.0
9.6	16.9	40.8	52.2	19.2	4.3	15.5	44.0	39.5	10.3	3.2	7.0

Forward Transfer Matrix, Model 4

Columns 13 - 25

11.9	28.2	9.1	8.7	17.2	11.5	34.4	11.0	19.4	28.4	17.0	55.8	37.2
11.8	17.1	5.7	14.6	24.9	11.8	21.4	7.0	22.2	34.0	15.9	39.3	25.6
14.2	13.4	3.9	21.3	38.0	14.6	17.8	5.3	30.3	48.5	19.0	37.8	24.3
17.1	14.1	3.5	20.1	44.1	17.9	19.2	5.1	34.4	58.4	23.2	43.0	27.2
18.7	14.5	3.4	19.3	46.8	19.7	20.1	5.1	36.5	63.5	25.5	46.1	29.0
19.5	15.4	3.5	18.0	45.4	20.4	21.2	5.2	35.8	62.6	26.3	48.1	29.8
20.0	20.6	4.3	13.1	34.2	20.1	27.5	6.1	29.8	50.6	25.9	56.0	32.7
17.5	33.1	6.6	8.6	21.1	16.8	42.9	9.2	25.0	37.5	24.7	76.3	43.6
15.3	36.9	8.3	7.7	17.9	14.8	47.2	11.2	23.6	33.7	22.9	80.0	48.0
6.4	30.6	24.9	6.0	8.9	6.3	33.8	25.5	12.8	16.3	10.4	44.6	46.5
4.8	8.3	8.5	20.2	13.8	4.8	10.0	8.9	14.6	17.0	6.8	17.3	14.5
8.3	7.3	3.6	49.9	40.7	8.8	9.8	4.9	37.2	43.8	12.1	22.4	16.3
14.4	10.0	2.9	33.4	59.3	15.6	14.1	4.3	44.3	70.4	21.0	35.2	24.1
19.6	12.8	3.0	22.9	59.4	21.5	18.3	4.7	45.5	81.1	28.7	46.8	30.8
20.9	13.8	3.1	20.3	56.0	22.9	19.7	4.9	43.6	78.7	30.1	49.6	31.9
24.7	18.8	3.5	13.5	40.3	25.5	26.2	5.4	34.1	60.7	31.7	58.8	33.9
20.4	35.9	6.0	8.1	21.1	19.5	47.7	8.9	27.2	40.4	28.9	88.2	47.2
15.8	43.3	8.7	7.1	16.9	15.5	57.4	12.4	26.1	35.5	25.8	99.8	58.5
3.4	38.5	93.6	4.4	4.5	3.8	50.0	139.7	15.5	12.9	8.8	55.5	186.3
0.5	1.6	148.7	113.1	2.1	0.6	2.1	210.0	35.6	3.8	1.3	4.2	52.0
3.8	2.7	3.0	267.4	47.4	4.7	4.3	6.3	163.6	60.8	8.9	14.6	17.8
12.3	7.3	2.1	64.3	99.3	15.2	11.5	4.1	102.7	142.8	25.6	38.3	31.5
19.7	11.0	2.4	29.3	82.4	24.3	17.3	4.7	74.1	132.6	37.7	55.2	40.6
23.6	12.9	2.7	21.7	70.5	28.5	20.0	4.9	60.3	112.6	40.9	60.0	41.0
35.3	20.9	3.1	11.7	43.0	39.7	32.0	5.7	43.2	77.7	50.9	83.4	45.4
22.6	43.0	6.1	7.6	20.2	22.7	62.9	10.2	35.4	48.3	39.1	130.8	65.9
15.0	51.2	10.0	6.7	15.6	15.7	75.6	16.2	33.1	40.4	31.0	143.7	87.7
8.8	20.6	12.1	9.9	14.8	11.9	39.5	39.2	82.4	59.0	37.1	162.0	334.1
7.6	10.7	5.7	23.2	17.2	11.0	20.2	26.4	221.2	83.9	39.0	107.6	303.7
9.6	8.8	2.6	51.9	37.6	14.0	16.0	8.6	311.1	159.8	40.6	74.8	96.9
13.4	10.4	2.6	41.6	60.1	19.2	18.6	6.9	190.2	195.8	47.0	78.9	76.4
18.1	12.5	2.7	27.9	68.5	25.7	21.8	6.7	121.6	180.5	55.0	86.6	72.1
23.1	14.8	2.9	20.2	63.5	32.5	25.6	6.8	92.7	152.5	62.9	96.3	71.8
29.9	22.8	3.6	12.6	40.7	39.9	39.5	7.9	69.3	106.6	75.1	133.6	79.4
21.3	36.6	5.8	9.3	23.5	25.1	62.4	11.9	57.9	71.3	55.2	180.3	101.6
14.9	38.8	8.7	8.5	18.9	18.1	69.1	18.2	57.0	62.0	44.9	199.4	142.2
13.6	21.7	6.7	13.2	24.2	18.8	40.9	19.1	103.8	92.9	55.5	177.2	202.5
13.5	17.3	5.1	17.7	28.5	19.3	32.3	15.4	144.9	116.5	59.1	150.6	182.9
14.5	14.7	3.7	24.9	38.8	21.0	27.0	10.6	173.1	152.8	59.7	121.3	126.0
15.9	14.7	3.4	25.4	46.4	23.1	26.7	9.3	153.4	164.5	60.7	114.7	106.6
18.1	15.5	3.4	22.4	50.7	26.2	27.9	8.8	127.1	161.5	64.0	115.8	98.3
20.5	16.9	3.5	19.1	49.9	29.5	30.3	8.7	108.2	149.2	68.3	121.8	95.9
23.1	21.9	4.1	14.2	38.4	32.1	39.3	9.7	88.1	118.2	73.9	149.6	102.7
19.7	29.3	5.5	11.4	27.2	25.4	52.6	12.5	76.0	89.5	62.3	182.7	120.9
16.4	29.5	6.7	11.0	24.2	21.3	54.1	15.8	77.5	83.5	56.0	193.1	148.4

Forward Transfer Matrix, Model 5

Columns 1 - 12

29.6	24.9	145.2	215.2	60.2	19.5	10.1	14.9	51.7	29.7	15.6	9.7
59.8	51.3	196.3	183.6	61.4	35.1	19.6	16.9	26.4	19.2	28.2	17.1
67.8	64.9	167.1	114.3	47.5	47.0	32.4	21.7	19.0	14.6	41.7	28.7
48.2	54.8	157.1	92.0	33.5	37.7	35.5	28.5	20.1	11.8	37.1	34.4
37.3	46.0	153.3	85.1	27.4	30.5	33.6	32.9	21.9	10.6	31.6	34.6
33.2	42.0	156.3	86.3	25.8	27.2	31.2	35.2	23.7	10.5	28.4	32.4
24.1	30.2	169.9	112.2	25.1	18.8	20.7	36.5	36.1	12.1	19.2	21.3
17.4	18.1	119.5	149.8	30.8	13.4	10.8	23.4	58.9	19.1	12.9	11.6
18.0	16.9	108.5	162.9	38.0	13.8	9.1	18.6	62.8	24.5	12.5	9.7
23.7	12.2	62.2	149.1	104.5	23.6	5.3	8.2	62.3	87.4	16.6	5.3
155.1	29.3	49.3	65.0	185.6	144.6	10.0	4.8	11.6	72.4	69.6	8.2
107.6	57.1	75.1	53.1	48.3	132.5	35.8	10.5	9.2	19.2	126.7	31.1
51.9	56.6	111.2	63.8	29.0	53.8	48.8	21.7	14.5	10.8	61.7	52.5
34.4	43.6	134.9	73.5	24.4	30.2	36.1	31.7	19.9	9.8	33.2	40.0
30.9	39.9	141.7	76.8	23.4	26.3	32.4	35.1	22.0	9.8	28.6	35.4
22.4	29.0	158.9	93.7	21.9	18.1	21.8	42.2	32.5	10.7	19.2	23.0
16.1	17.6	117.2	136.2	26.4	12.7	11.2	26.0	58.4	16.4	12.7	12.2
15.7	15.2	98.6	144.0	32.0	12.5	8.9	19.2	63.2	21.8	11.8	9.7
11.3	6.6	34.7	66.9	40.8	15.1	3.8	7.0	40.7	58.8	13.8	4.5
12.5	2.9	8.6	14.6	22.6	55.6	1.6	1.6	7.0	64.2	81.3	1.7
16.3	8.7	14.5	10.3	8.1	43.8	9.0	3.2	3.1	5.7	162.5	12.3
25.3	24.8	52.6	32.1	14.3	37.2	29.5	13.3	10.1	6.8	70.5	51.3
24.4	30.5	87.8	49.8	17.0	25.2	31.7	24.6	16.1	7.8	33.2	45.0
23.9	31.4	110.4	61.0	18.5	21.8	29.2	32.8	20.0	8.5	25.6	35.9
16.0	20.5	117.2	76.7	17.3	13.5	16.8	43.4	33.3	9.7	15.1	19.0
12.8	14.0	90.3	104.5	20.7	10.7	9.8	24.2	56.3	14.1	11.4	11.2
12.7	12.2	76.3	113.5	26.6	10.7	7.6	16.7	59.6	20.3	10.7	8.7
6.3	5.4	27.8	36.6	14.3	8.0	4.3	8.3	23.9	18.1	9.8	5.7
4.5	4.2	19.0	20.4	7.5	7.0	3.8	6.4	12.7	9.1	12.1	5.5
6.0	6.0	22.6	19.4	6.3	8.8	6.2	7.9	10.8	5.5	19.2	9.7
9.4	10.5	36.3	27.3	8.5	11.6	11.2	12.7	13.6	6.0	20.3	18.3
11.0	13.4	48.9	34.4	10.2	11.9	14.0	17.6	16.3	6.7	17.5	21.4
11.5	14.4	58.7	40.6	11.3	11.5	14.3	22.5	19.3	7.3	15.3	20.2
10.4	12.6	66.2	53.6	12.8	9.7	11.1	26.3	29.1	8.9	12.1	14.3
9.1	9.7	56.5	67.2	15.5	8.5	7.4	16.9	42.1	12.4	9.9	9.3
8.4	8.2	46.6	62.5	16.7	8.2	6.1	13.0	40.1	15.2	9.4	7.8
6.7	6.9	34.5	38.1	11.2	7.4	6.0	11.5	24.5	11.0	10.0	8.2
6.5	6.9	32.7	33.0	9.6	7.4	6.3	11.5	20.5	9.0	10.9	9.1
6.9	7.6	33.6	31.1	8.9	7.9	7.2	12.1	18.5	7.7	12.4	10.7
7.6	8.6	37.1	32.4	9.2	8.5	8.4	13.6	18.6	7.5	13.0	12.5
8.2	9.4	40.8	34.6	9.7	8.8	9.2	15.3	19.5	7.6	12.9	13.6
8.5	9.9	44.2	37.1	10.1	8.9	9.5	16.8	20.8	7.8	12.5	13.7
8.4	9.7	47.7	43.4	11.2	8.5	8.8	17.9	25.6	8.8	11.3	12.0
7.9	8.5	45.3	49.8	12.6	7.9	7.1	14.9	31.9	10.8	10.0	9.4
7.5	7.8	41.1	47.4	12.7	7.7	6.4	13.2	30.8	11.6	9.7	8.6

Forward Transfer Matrix, Model 5

Columns 13 - 25

5.9	12.1	11.4	16.7	10.2	15.2	16.0	20.1	49.9	41.9	19.8	85.1	69.3
5.7	6.4	6.3	23.7	14.1	13.1	8.7	11.3	44.6	40.3	16.0	52.4	42.5
7.3	5.2	4.6	34.7	22.5	16.5	7.5	8.8	58.1	55.1	19.6	52.1	41.3
9.8	6.0	4.1	35.1	29.3	22.1	8.7	8.5	72.3	73.6	25.9	64.5	49.4
11.3	6.7	4.0	32.9	31.9	25.9	9.8	8 7	79.8	84.3	30.2	73.9	55.6
12.0	7.3	4.1	30.6	31.1	27.4	10.6	9.0	80.7	85.9	31.8	78.8	58.5
12.2	10.5	5.2	22.5	22.6	28.3	14.3	11.1	74.7	76.1	32.4	95.8	68.0
9.8	16.3	8.6	17.3	14.3	25.0	21.6	17.3	72.0	64.4	31.6	126.7	89.4
8.1	16.8	10.8	16.6	12.1	21.8	22.8	20.8	67.9	57.8	28.8	126.7	93.4
3.7	13.0	33.1	19.3	6.4	10.8	18.2	52.8	47.7	32.3	16.0	84.2	102.0
1.7	2.5	13.9	42.7	5.8	4.3	3.7	20.6	27.7	15.4	5.8	20.9	29.6
3.8	2.6	4.6	80.3	18.8	8.8	3.9	8.1	58.7	37.5	11.3	28.6	26.6
7.9	4.5	3.5	57.2	38.1	18.5	6.8	7.4	83.0	76.6	23.1	53.8	43.4
11.4	6.3	3.8	36.5	37.7	26.7	9.5	8.4	88.8	95.4	32.2	74.6	57.0
12.5	6.9	3.9	32.3	35.1	29.0	10.4	8.8	87.9	95.9	34.2	80.4	60.4
14.1	10.0	4.6	23.4	25.3	32.7	13.9	10.4	81.4	86.0	36.4	98.6	69.7
11.0	16.6	7.6	17.6	15.4	27.8	21.9	15.9	76.5	69.8	34.3	132.1	90.4
8.7	17.9	10.1	16.8	12.7	23.9	24.8	20.2	74.3	63.1	31.9	141.9	101.1
3.7	12.8	38.0	23.2	6.9	12.9	23.7	87.3	77.0	44.1	22.2	131.2	213.2
0.8	2.0	43.9	171.7	2.1	3.0	3.9	191.8	88.6	13.3	5.7	27.4	171.6
1.4	1.1	1.9	368.8	11.1	4.4	2.0	5.4	221.7	33.4	7.8	18.7	24.8
5.7	3.6	2.6	116.3	55.8	16.7	6.2	6.7	166.7	118.3	26.2	56.5	50.9
10.2	5.5	3.3	45.6	51.7	27.5	9.2	8.2	124.0	137.1	38.5	80.9	65.2
12.9	6.7	3.7	33.0	41.2	33.0	10.8	8.9	106.9	123.2	41.8	90.0	68.9
16.5	11.1	4.6	21.6	25.1	42.8	16.2	11.1	95.9	103.4	47.7	121.4	83.9
11.8	18.1	6.9	17.8	15.6	32.1	25.2	15.7	89.3	80.2	41.2	161.2	105.1
8.3	18.8	10.1	17.0	12.5	25.1	28.9	21.5	84.6	69.2	36.0	172.1	120.4
4.8	9.7	12.6	22.9	10.2	18.8	21.0	40.4	130.5	74.5	36.6	185.0	264.7
3.7	5.2	6.3	41.9	10.3	15.3	11.2	28.4	272.6	85.9	34.1	121.4	251.4
4.3	4.4	3.2	62.4	18.1	17.4	8.9	11.7	365.4	133.6	36.6	92.9	112.9
6.6	5.4	3.2	48.1	33.9	24.8	10.4	9.9	233.3	191.1	47.1	104.1	96.6
8.9	6.4	3.5	34.4	38.1	32.1	11.9	10.1	173.2	190.7	55.5	115.3	96.5
11.1	7.4	3.8	27.9	34.6	39.1	13.5	10.7	147.6	170.3	61.7	126.5	99.1
13.1	11.1	4.7	21.8	23.9	44.8	18.8	12.5	124.2	129.4	62.4	156.6	109.6
9.5	16.2	6.8	18.7	15.1	32.0	28.5	17.1	110.7	92.6	48.8	204.6	134.8
7.4	15.8	8.7	18.4	12.9	26.5	30.6	21.8	109.6	84.1	44.0	218.4	159.6
6.6	10.2	6.9	23.2	15.1	26.4	21.3	21.0	154.2	107.6	50.3	194.3	187.1
6.5	8.6	5.5	27.3	17.1	26.5	17.6	17.7	189.7	124.9	52.6	170.9	171.9
6.8	7.6	4.6	31.3	20.4	27.4	15.4	14.6	212.2	145.7	54.0	151.0	144.3
7.5	7.6	4.3	30.8	24.0	29.7	15.1	13.3	197.4	159.9	56.7	146.3	130.2
8.3	7.9	4.3	28.6	25.7	32.5	15.4	12.9	179.0	162.5	59.8	148.1	125.3
9.0	8.4	4.4	26.5	25.6	35.3	16.2	13.0	164.9	157.5	62.4	152.8	124.2
9.7	10.4	5.0	23.3	21.8	37.0	19.5	14.1	146.1	135.7	61.9	172.2	130.2
8.5	13.1	6.2	20.8	16.5	31.8	25.1	17.0	131.9	107.9	53.7	203.5	148.0
7.6	12.7	6.9	20.9	15.3	29.0	25.5	19.0	133.3	103.2	51.1	208.7	162.5

Forward Transfer Matrix, Model 6

Columns 1 - 12

19.1	17.8	106.1	217.3	88.7	20.3	16.7	18.3	24.3	40.2	8.1	7.2
41.1	32.5	126.1	191.8	121.3	44.6	32.0	17.4	13.9	32.3	16.5	11.9
49.2	44.7	129.6	130.6	97.0	63.4	57.3	21.6	7.9	21.4	24.2	21.4
38.6	42.8	129.7	113.1	71.0	54.9	65.6	26.9	8.1	17.0	22.3	26.3
29.0	37.6	134.2	109.7	55.6	42.3	62.7	32.8	9.2	14.6	17.9	26.9
22.3	31.9	139.0	112.8	46.5	32.2	54.6	37.8	11.0	13.5	14.1	24.6
14.5	20.8	131.3	147.8	41.7	19.2	31.6	37.3	18.7	16.2	8.7	14.9
11.6	14.7	102.3	170.8	44.5	14.6	19.7	28.5	25.3	22.3	6.7	9.6
12.0	13.7	94.1	181.9	52.0	14.5	16.8	24.0	27.0	27.9	6.5	8.1
16.5	12.9	73.7	189.0	103.2	19.2	12.3	13.5	27.5	68.0	7.7	5.4
60.8	24.8	77.5	138.3	228.9	68.3	22.7	9.8	10.1	58.2	22.9	8.0
78.2	37.4	74.0	82.3	120.4	131.2	52.7	11.7	4.9	32.1	47.8	18.2
47.8	44.4	93.1	82.9	70.3	86.5	83.1	19.0	5.8	18.2	36.2	32.9
31.5	40.6	111.3	89.3	53.3	52.2	80.2	27.7	7.3	14.0	22 9	35.4
20.7	31.0	123.3	94.9	41.0	31.9	61.1	37.8	9.5	12.1	14.5	29.4
14.6	22.3	137.6	120.3	36.9	20.4	37.7	44.4	15.0	13.3	9.4	18.1
11.1	14.5	102.2	161.4	40.4	14.3	20.5	30.8	24.8	19.9	6.7	10.2
10.7	12.6	87.2	168.1	45.0	13.4	16.3	24.4	27.4	25.3	6.2	8.1
9.1	6.8	39.1	106.0	63.3	13.5	7.8	9.4	24.0	84.0	6.2	3.9
21.0	4.8	13.7	26.4	75.1	91.2	6.1	2.5	3.5	217.9	44.5	2.4
40.9	15.1	27.3	30.1	54.3	184.7	30.5	5.3	2.3	29.2	107.2	13.0
26.2	23.3	46.5	41.1	35.2	84.4	76.3	12.2	3.8	11.8	56.1	53.7
18.6	25.3	68.2	54.5	30.4	39.4	76.6	22.6	5.8	9.7	21.2	51.0
15.8	24.3	86.0	66.3	29.8	28.2	62.0	32.1	7.6	9.8	14.2	36.7
11.3	17.7	108.7	91.1	28.0	17.1	34.9	47.9	13.2	11.1	8.3	18.7
8.8	11.8	83.2	125.3	30.0	12.4	18.9	32.1	23.8	15.9	6.2	10.1
8.7	10.3	71.0	140.0	36.5	11.8	14.4	22.5	26.6	22.6	5.8	7.6
4.3	4.2	24.4	50.5	21.0	8.3	6.9	9.3	13.2	26.3	5.0	4.2
3.1	2.4	11.8	19.6	10.0	11.9	4.9	5.0	4.7	13.7	9.9	3.2
5.6	3.8	13.5	17.3	10.5	25.8	9.5	5.6	3.4	9.0	22.7	6.7
6.9	6.8	22.5	24.2	12.3	22.7	20.1	9.4	4.3	6.8	18.4	16.8
7.5	9.6	34.2	33.1	14.5	18.2	28.3	14.9	5.5	6.9	12.1	22.9
7.8	11.4	49.0	45.1	17.0	15.4	30.1	23.5	7.5	7.9	8.9	21.2
6.9	10.1	60.6	63.7	18.8	11.9	21.6	32.6	12.2	9.8	6.6	13.4
6.4	8.2	54.7	87.2	22.3	10.2	14.4	23.9	19.7	14.1	5.6	8.4
6.0	7.0	45.3	86.6	24.0	9.5	11.4	17.6	20.4	18.0	5.2	6.7
4.3	5.0	28.7	47.0	15.0	8.9	10.0	13.1	11.7	12.9	5.8	6.6
4.1	4.7	25.6	38.4	12.8	9.7	10.3	12.1	9.2	10.6	6.8	7.1
4.3	5.0	25.4	35.3	12.2	10.9	11.4	12.2	8.1	9.3	7.8	8.1
4.6	5.5	27.4	35.8	12.4	11.3	13.3	13.3	7.9	8.7	7.9	9.6
4.9	6.1	30.1	37.7	12.9	11.2	14.8	14.8	8.2	8.6	7.6	10.8
5.1	6.7	33.9	41.3	13.7	11.0	16.0	17.0	8.8	8.8	7.1	11.4
5.2	6.8	38.3	49.4	15.2	10.3	15.1	19.3	10.9	9.9	6.4	10.2
5.1	6.4	38.8	60.5	17.3	9.4	12.4	18.0	14.5	12.4	5.7	7.9
4.9	5.9	35.8	60.0	17.5	9.1	11.2	16.0	14.7	13.6	5.5	7.1

Forward Transfer Matrix, Model 6

Columns 13 - 25

6.2	9.0	14.1	13.7	10.3	12.5	28.2	33.4	34.7	60.2	24.5	84.4	84.7
5.4	5.1	11.4	23.9	14.4	9.9	16.0	22.6	32.0	54.9	17.8	51.1	54.0
6.7	3.2	8.2	35.0	24.6	11.9	10.8	14.6	39.2	74.6	19.4	41.1	42.4
8.7	3.5	6.8	33.4	30.6	15.5	12.2	13.3	44.1	94.9	24.7	48.6	47.3
10.9	4.1	5.9	28.3	32.6	19.4	14.5	13.2	46.5	109.6	30.1	58.5	53.9
12.8	5.0	5.5	23.4	31.3	22.8	17.3	13.8	47.5	116.4	34.6	68.6	60.8
12.9	8.2	6.6	15.9	21.0	24.0	27.0	18.7	46.5	103.8	38.6	94.7	79.7
10.4	10.8	8.9	13.2	14.8	20.9	35.4	25.4	46.6	91.6	38.2	115.3	97.9
8.8	11.3	10.8	12.7	12.7	18.2	36.8	29.7	44.6	82.8	35.3	115.8	102.1
4.9	9.7	23.1	13.1	8.2	10.4	30.3	53.2	33.9	52.9	22.2	86.4	102.8
3.1	3.5	20.5	30.2	9.7	5.8	10.9	30.1	28.1	35.8	11.1	34.2	46.6
3.8	1.9	12.8	64.3	20.7	6.9	6.7	17.1	45.8	55.1	12.4	26.6	35.1
6.3	2.5	7.5	52.5	36.1	11.6	9.0	12.9	50.8	91.6	19.9	38.0	41.2
9.4	3.3	5.8	35.7	40.8	17.1	12.1	12.3	50.6	118.2	27.9	51.3	49.7
13.4	4.5	5.1	24.6	37.8	24.3	16.3	13.0	51.6	135.7	37.4	68.2	61.1
15.6	6.9	5.5	17.2	25.4	28.0	23.5	15.9	48.9	117.9	41.8	88.8	74.5
11.3	10.8	8.1	13.4	15.8	22.7	35.5	23.8	48.4	97.3	40.6	117.7	97.8
9.2	11.8	10.1	12.7	13.0	19.4	39.6	29.2	47.8	88.6	38.3	126.7	108.7
3.9	10.0	38.8	12.4	6.6	9.4	36.3	107.1	40.6	53.7	23.5	108.3	176.2
0.9	1.4	144.4	56.1	3.3	2.1	5.1	115.9	39.7	17.1	5.4	19.1	80.2
1.9	1.0	14.9	172.2	17.4	4.1	4.0	22.5	99.6	50.8	9.4	19.7	42.9
4.7	1.8	5.4	92.5	69.8	9.8	7.4	11.0	86.3	138.4	20.9	36.9	44.6
9.0	2.9	4.3	37.9	68.2	18.4	11.8	11.1	69.0	193.7	35.2	57.8	57.5
12.9	3.9	4.3	26.1	52.3	25.6	15.2	12.1	63.4	190.0	43.8	71.5	66.1
19.7	6.6	4.9	16.6	28.7	37.7	23.9	15.3	56.8	147.1	53.9	99.0	81.9
13.0	11.5	6.9	13.5	16.8	27.6	39.3	22.4	56.8	115.0	50.3	139.1	109.4
9.1	12.4	9.5	12.7	12.9	20.4	45.2	29.9	54.2	97.2	43.3	150.2	125.1
4.7	7.4	15.1	13.7	8.4	12.7	39.4	70.5	68.9	85.3	37.5	171.7	287.3
2.7	2.7	10.5	37.0	6.8	7.8	14.3	69.1	157.3	79.2	27.5	90.8	394.0
3.0	2.0	6.3	89.4	14.3	8.6	10.2	25.9	320.4	142.7	30.0	66.0	147.6
4.9	2.5	3.7	56.1	39.0	13.1	11.9	13.6	190.5	290.1	40.1	71.4	92.1
7.5	3.1	3.5	29.7	49.3	19.5	14.5	12.6	108.9	323.4	51.4	82.4	86.4
11.9	4.2	3.9	20.7	41.9	29.7	18.8	13.9	85.3	265.5	65.3	100.5	93.4
17.3	6.8	4.7	15.7	25.4	42.4	28.2	17.1	74.0	185.2	75.6	131.5	107.8
11.3	10.9	6.6	13.9	15.5	27.2	44.4	24.3	69.2	129.5	59.2	176.2	136.7
8.3	11.2	8.5	13.3	12.7	21.0	50.9	31.3	68.0	114.2	52.1	191.6	159.3
7.1	7.0	7.0	17.0	13.9	19.8	36.3	30.9	98.9	145.3	59.0	185.3	203.6
6.7	5.5	5.9	21.4	15.3	19.1	28.5	27.2	127.6	167.6	60.4	162.0	201.2
6.8	4.9	5.2	25.0	17.9	19.4	24.6	23.2	146.6	197.0	61.6	143.3	174.3
7.4	4.7	4.8	24.1	21.5	21.1	23.6	20.4	136.7	226.2	64.7	136.0	150.9
8.3	4.9	4.6	22.0	24.0	23.2	23.9	19.2	121.4	237.3	68.0	135.3	140.2
9.5	5.2	4.6	19.7	24.8	26.4	25.2	18.9	107.3	231.0	72.3	139.0	135.2
10.7	6.4	5.1	17.4	21.4	29.2	30.3	20.6	95.2	197.2	74.0	155.4	140.1
9.5	8.5	6.3	15.5	16.0	25.2	40.1	25.1	85.6	151.7	64.9	185.0	158.2
8.3	8.7	6.9	15.3	14.5	22.5	42.5	28.1	85.3	141.7	60.9	192.2	171.7

Columns 1 - 12

37.1	29.5	144.9	206.9	121.8	7.0	5.7	5.3	20.8	21.4	20.7	10.9
98.1	65.2	167.2	162.2	122.9	20.4	13.9	4.4	8.4	10.1	53.1	23.0
98.2	73.2	156.8	107.7	81.9	27.3	21.7	5.6	6.0	6.9	75.6	36.8
75.4	66.7	159.8	90.7	60.3	24.2	24.3	7.3	6.3	6.1	72.8	44.3
62.8	60.7	166.5	88.1	53.2	20.8	23.8	8.5	6.9	6.1	65.2	45.3
55.8	56.6	179.4	93.3	51.6	18.2	22.1	9.5	7.7	6.3	57.5	42.3
35.0	37.8	203.7	141.8	59.3	9.5	11.9	11.0	15.0	9.9	30.6	23.2
25.7	23.2	136.9	164.2	81.5	6.0	6.2	8.2	23.4	17.7	20.0	12.9
26.5	22.7	129.0	170.8	92.1	5.9	5.7	7.1	24.0	20.0	19.3	11.7
28.3	16.7	79.8	154.6	194.5	5.7	3.6	3.7	22.8	44.5	17.3	7.3
224.8	31.6	58.4	76.6	328.7	33.4	6.4	1.5	3.8	17.2	64.8	10.2
153.4	56.3	83.7	57.6	74.2	60.7	22.1	3.3	3.5	6.7	157.1	36.4
86.1	60.5	108.4	62.7	51.3	35.5	28.2	5.2	4.6	5.4	114.8	53.8
62.6	58.5	138.8	72.7	47.4	23.1	26.7	7.6	5.9	5.6	76.3	53.7
53.8	54.5	155.8	78.5	46.2	18.8	24.1	9.2	6.8	5.8	61.8	49.1
34.1	38.4	205.7	105.3	46.7	10.3	14.5	14.4	11.7	7.5	34.1	29.2
24.0	22.2	135.6	152.6	71.7	5.9	6.4	9.1	23.4	15.9	20.1	13.5
23.6	19.8	112.8	156.3	89.9	5.5	5.2	6.8	25.2	21.6	18.5	11.0
14.9	9.4	46.9	79.5	101.0	4.5	2.7	3.0	16.7	41.9	17.0	6.3
16.6	1.9	5.4	6.9	30.6	16.8	0.7	0.3	1.1	17.6	156.3	1.6
49.2	21.9	37.6	23.6	24.7	31.7	12.1	2.1	2.2	3.6	226.7	28.1
58.2	40.4	76.6	44.1	35.3	28.4	22.7	4.4	4.1	4.6	122.9	54.8
45.4	41.8	102.7	54.9	35.9	18.4	22.5	6.9	5.7	5.2	68.8	55.3
37.6	38.7	120.5	61.1	35.7	13.9	19.9	9.2	6.8	5.6	50.1	47.9
23.0	24.3	142.3	88.3	39.9	7.2	10.0	14.7	14.3	8.1	25.8	22.8
19.7	17.6	103.9	122.5	62.5	5.2	5.5	8.0	22.5	15.3	18.9	12.4
19.0	14.6	78.9	119.7	88.6	4.8	4.2	5.1	23.6	26.0	17.2	9.4
11.4	9.3	42.4	45.8	35.0	4.1	3.9	3.7	9.7	11.8	18.6	10.4
11.0	8.8	34.5	30.4	21.2	4.6	4.2	3.1	5.9	6.3	25.0	11.7
15.8	12.6	42.4	32.6	21.7	7.1	6.5	3.7	5.8	5.4	37.5	18.0
20.7	17.7	56.6	39.5	25.4	8.7	9.4	4.8	6.5	5.7	39.7	26.3
20.7	19.0	65.0	43.5	26.8	8.2	9.9	5.8	7.1	6.0	34.9	27.7
19.1	18.1	69.4	46.4	27.4	7.3	9.2	6.6	7.8	6.3	30.3	25.3
15.2	13.9	70.7	63.4	34.9	5.1	5.9	7.0	12.9	9.1	21.0	15.0
14.1	12.2	61.0	66.8	41.1	4.6	4.8	5.4	14.2	11.7	19.0	12.2
11.5	9.4	42.8	46.4	35.3	4.1	3.9	3.7	9.8	11.8	18.5	10.4
13.0	11.2	49.1	47.0	31.0	4.8	5.0	4.5	9.6	9.1	21.4	13.5
13.5	11.6	47.2	41.1	26.6	5.3	5.5	4.3	8.1	7.4	24.7	15.2
14.9	12.9	49.3	40.2	25.7	6.0	6.3	4.5	7.6	6.8	28.0	17.6
16.4	14.4	53.7	41.7	26.2	6.6	7.2	4.9	7.6	6.6	29.7	20.2
16.8	15.1	56.9	43.2	26.8	6.6	7.5	5.2	7.8	6.7	29.0	21.0
16.4	14.9	58.6	44.7	27.3	6.3	7.4	5.5	8.2	6.9	27.5	20.4
14.4	12.8	59.2	54.0	32.2	5.1	5.7	5.6	11.0	8.7	21.9	15.1
13.8	12.0	55.9	55.0	34.2	4.8	5.2	5.1	11.4	9.7	20.8	13.7
13.0	11.2	49.2	47.2	31.1	4.8	5.0	4.5	9.7	9.1	21.4	13.5

Forward Transfer Matrix, Model 7

Columns 13 - 25

23.7	23.6	36.0	8.5	6.7	2.5	9.4	14.5	44.1	28.8	24.9	83.5	62.0
20.9	9.2	15.6	15.9	12.1	2.0	3.8	6.4	43.7	29.4	17.9	42.5	31.7
28.1	7.1	11.2	23.0	19.2	2.6	3.3	5.1	60.1	42.2	22.9	44.9	32.6
37.4	7.7	10.7	23.8	23.4	3.4	3.7	5.3	70.3	53.1	29.5	54.9	38.6
43.6	8.6	11.0	22.3	24.3	4.0	4.2	5.6	73.1	57.7	33.4	61.7	42.6
47.5	9.5	11.6	20.2	22.9	4.2	4.6	5.9	71.0	56.9	34.9	65.9	44.6
47.4	17.8	18.6	12.1	13.4	4.2	7.9	8.7	58.6	44.6	35.2	87.2	55.4
35.6	28.8	33.3	9.4	8.4	3.7	12.2	14.7	58.5	39.6	35.9	116.8	77.1
31.7	29.4	36.9	9.1	7.7	3.4	12.2	16.0	56.1	37.2	33.8	113.8	78.0
17.6	27.3	70.0	7.9	4.8	2.1	10.5	25.2	42.4	25.1	22.6	85.7	80.0
7.5	4.3	19.6	15.0	5.4	0.8	1.7	6.5	24.8	12.8	7.3	18.5	18.3
18.0	4.2	9.8	41.1	19.5	1.8	2.0	4.6	71.5	38.0	16.9	31.8	25.9
29.3	5.8	9.3	36.8	30.1	2.9	2.9	4.8	89.4	60.0	27.2	48.7	36.4
41.8	7.6	10.3	26.6	29.7	4.0	3.8	5.4	84.1	66.7	35.2	62.1	43.9
49.5	8.7	11.0	22.3	27.2	4.6	4.4	5.8	80.3	66.7	39.0	68.9	47.3
65.2	14.4	14.7	13.6	16.9	5.5	6.9	7.3	66.0	53.7	41.8	87.1	54.6
39.7	29.3	30.6	9.7	8.9	4.1	12.7	14.0	62.7	43.0	39.4	125.9	79.5
31.0	32.3	41.0	9.2	7.5	3.5	13.5	18.0	59.9	39.0	35.9	125.9	87.3
16.6	26.7	111.9	10.6	4.7	2.3	12.5	54.6	62.7	31.9	28.9	122.6	170.0
2.2	1.8	84.3	177.6	1.2	0.3	0.9	136.9	131.9	7.5	4.8	15.9	179.0
13.0	3.1	6.7	124.6	22.1	1.6	1.6	4.3	220.1	53.4	18.2	33.9	34.0
27.6	5.6	8.7	49.8	40.1	3.2	3.0	5.0	136.8	87.7	33.2	57.9	45.3
44.1	7.9	10.5	27.6	36.7	5.0	4.2	6.1	109.9	96.2	48.8	81.2	58.2
59.1	9.4	11.6	20.7	30.4	6.4	5.1	6.6	99.8	91.1	57.0	92.7	63.4
76.5	18.9	17.1	12.4	15.3	7.7	9.8	9.1	81.6	65.6	59.7	129.7	75.8
38.9	31.1	31.3	10.2	8.9	4.6	14.5	15.2	75.6	50.4	47.9	160.4	97.1
26.4	34.4	54.2	9.6	6.8	3.4	15.4	24.8	68.2	41.6	38.8	150.2	115.1
25.8	16.6	32.9	14.0	8.8	4.2	9.6	22.2	129.4	68.6	59.4	203.1	199.2
23.8	9.9	17.5	23.9	10.3	4.1	5.8	13.7	241.1	90.6	61.6	152.7	178.2
28.7	9.3	13.8	28.6	16.2	4.7	5.5	9.7	240.4	115.3	66.3	133.9	118.7
37.4	10.2	13.9	23.4	22.9	5.8	5.9	9.0	170.6	125.1	74.5	136.8	103.5
44.6	11.0	14.3	19.5	23.2	6.8	6.4	9.1	145.6	120.4	81.0	142.2	101.3
50.7	12.0	14.9	17.2	20.9	7.6	6.9	9.3	135.4	113.4	86.2	150.0	102.5
44.4	20.2	21.2	12.9	12.1	6.5	11.6	12.4	112.1	77.9	73.9	202.6	118.2
34.2	23.2	27.6	12.3	9.9	5.2	12.9	15.8	108.7	69.4	65.0	214.4	134.4
26.0	16.8	32.8	13.9	8.8	4.3	9.7	22.0	128.3	68.6	59.6	204.5	197.0
32.2	16.1	23.3	15.1	11.5	5.3	9.4	15.0	141.2	85.1	71.6	200.8	154.4
32.6	13.3	18.9	18.0	13.2	5.4	7.8	12.6	170.2	98.4	74.9	179.7	144.5
34.4	12.3	17.0	19.5	15.4	5.7	7.2	11.3	175.9	107.9	76.9	167.0	130.0
37.6	12.2	16.4	19.1	17.5	6.1	7.2	10.6	162.9	112.5	79.5	162.6	120.5
40.3	12.5	16.4	18.1	18.1	6.5	7.3	10.5	152.6	111.9	82.0	163.6	117.5
42.1	13.0	16.8	17.1	17.4	6.7	7.6	10.7	146.4	108.8	83.8	167.9	117.7
38.9	17.8	21.1	14.2	12.6	6.1	10.3	12.8	127.6	86.2	76.4	201.3	128.9
35.1	18.8	23.6	13.9	11.4	5.6	10.8	14.3	126.0	81.1	71.9	207.8	138.0
32.2	16.2	23.4	15.1	11.5	5.3	9.4	15.1	140.6	84.9	71.6	201.2	154.1

Forward Transfer Matrix, Model 8

Columns 1 - 12

54.0	21.4	87.7	211.1	118.8	15.0	1.7	4.0	32.8	29.7	25.4	26.9
114.4	49.0	112.1	182.9	146.5	29.4	3.9	3.7	15.2	17.7	45.4	55.7
128.9	87.6	120.8	108.6	84.6	40.5	8.2	5.1	8.2	8.4	68.8	117.1
85.8	96.1	134.6	85.9	52.3	31.3	10.2	7.2	8.1	5.6	58.1	156.6
68.5	91.9	144.6	81.6	43.4	25.9	10.1	8.5	8.7	5.0	49.7	162.3
54.9	80.6	168.3	89.3	39.8	20.4	8.8	10.6	10.8	5.0	39.9	144.0
33.5	38.2	179.1	161.2	47.8	10.6	3.8	12.2	29.2	10.0	21.7	63.1
29.6	19.0	99.2	193.9	63.5	9.0	1.8	6.6	43.9	19.5	18.8	30.6
36.6	18.1	87.3	202.2	83.7	10.8	1.6	5.0	41.0	25.5	20.7	26.5
50.6	14.9	56.5	152.0	122.5	17.1	1.2	2.7	28.7	49.2	29.2	20.3
162.4	21.2	45.8	98.6	184.9	57.0	1.8	1.7	11.2	40.6	69.2	25.4
251.2	51.0	41.5	44.2	66.0	111.4	7.0	1.7	3.4	9.4	184.0	86.3
101.2	99.3	68.5	45.4	34.7	57.1	16.5	3.6	4.2	4.0	118.1	231.4
58.2	94.8	120.0	62.5	33.3	24.9	12.1	7.5	7.1	4.0	50.3	207.7
49.0	82.8	143.8	69.4	32.4	19.6	10.1	9.8	8.7	4.2	39.7	173.4
28.1	39.8	209.6	110.9	32.5	9.6	4.4	18.9	21.6	6.3	20.6	73.8
25.4	19.3	104.2	179.2	51.0	8.0	1.9	8.0	45.7	15.7	17.8	33.5
28.6	16.1	80.6	181.5	64.7	8 9	1.5	5.3	44.8	22.7	19.1	26.3
22.7	8.3	33.7	82.1	51.8	10.0	0.8	2.3	23.6	35.3	24.5	16.6
20.7	4.3	14.7	33.7	31.3	16.1	0.5	1.0	8.6	29.9	45.6	9.1
26.0	4.1	4.9	6.2	7.0	35.3	0.9	0.4	1.4	5.1	335.1	21.2
24.7	18.3	12.0	8.5	6.4	26.3	5.9	0.8	1.3	1.3	195.6	213.0
15.5	29.4	34.3	17.9	8.6	9.4	6.2	2.8	3.2	1.7	30.7	250.5
23.6	47.5	94.4	41.8	17.2	10.3	7.1	8.8	7.5	3.2	25.0	158.6
17.9	24.8	137.4	81.5	22.9	6.6	3.0	17.5	22.2	6.2	17.0	56.2
18.6	14.2	74.1	128.5	37.0	6.5	1.5	6.7	41.3	13.7	17.3	29.5
20.8	11.8	56.9	128.4	46.6	7.4	1.2	4.3	39.7	21.0	18.7	23.5
11.5	7.0	28.5	51.0	22.2	5.5	0.9	2.7	17.7	13.4	20.7	20.8
8.7	5.6	20.5	31.8	14.1	5.3	0.8	2.1	11.0	8.9	27.4	19.9
7.4	5.6	17.7	22.1	9.2	5.3	0.9	1.9	7.5	5.2	39.0	26.7
7.2	7.6	20.8	21.3	8.4	4.6	1.3	2.3	7.0	4.1	29.0	41.5
8.5	11.6	30.9	25.8	9.8	4.5	1.9	3.4	7.9	4.1	20.3	56.5
9.4	13.1	39.5	30.8	11.2	4.6	2.0	4.6	9.5	4.6	18.3	53.3
11.5	11.4	50.9	59.4	18.9	4.9	1.5	5.8	20.7	8.1	17.6	33.5
12.7	9.4	42.0	71.5	24.4	5.3	1.1	4.1	25.2	11.6	18.4	25.7
12.4	8.0	33.7	61.7	24.6	5.5	1.0	3.1	21.5	13.7	19.5	22.5
10.0	7.7	30.2	45.6	17.5	4.9	1.0	3.1	16.1	9.5	20.5	25.8
9.1	7.4	27.7	38.6	14.9	4.7	1.0	2.9	13.5	8.1	22.1	26.9
8.6	7.6	26.7	34.1	13.1	4.6	1.1	2.9	11.8	6.9	23.4	29.8
8.5	8.1	27.7	33.0	12.5	4.5	1.2	3.0	11.3	6.3	22.5	33.4
8.7	9.0	30.2	33.6	12.5	4.5	1.3	3.4	11.4	6.1	20.8	36.5
8.9	9.4	32.3	35.2	12.9	4.5	1.4	3.6	11.9	6.2	20.0	36.6
10.1	9.2	36.7	47.6	16.7	4.7	1.2	4.0	16.7	8.1	19.1	30.7
10.7	8.5	35.0	52.9	19.0	4.9	1.1	3.6	18.8	9.6	19.3	27.0
10.4	8.0	32.0	49.4	18.7	4.9	1.0	3.3	17.5	9.9	19.9	25.9

Forward Transfer Matrix, Model 8

Columns 13 - 25

10.7	22.2	51.9	2.5	5.8	7.1	15.5	14.6	39.4	47.6	36.7	46.7	70.8
9.9	9.8	30.1	2.6	8.7	5.6	6.9	7.9	25.1	38.4	22.7	21.8	34.5
14.5	5.3	14.9	3.3	16.9	7.9	4.0	4.1	25.2	56.2	25.8	14.8	20.3
21.4	5.5	10.8	3.1	23.8	11.8	4.4	3.3	28.2	80.6	36.7	17.5	21.0
25.6	6.0	10.0	2.8	26.0	14.1	4.9	3.3	29.7	91.7	43.2	19.7	22.7
31.2	7.5	10.4	2.5	24.7	16.7	6.0	3.5	30.8	95.1	49.8	23.4	26.1
32.0	19.9	20.5	2.1	12.9	16.8	14.0	7.1	38.0	75.7	56.4	44.2	49.9
18.1	31.1	38.8	2.4	7.6	11.6	21.7	12.8	49.7	68.4	56.0	65.3	81.0
13.8	28.8	47.7	2.5	6.5	9.2	20.1	14.7	47.3	59.8	48.1	60.5	82.2
7.9	21.3	92.7	3.2	4.8	5.8	16.2	24.5	47.1	46.3	35.6	51.4	97.9
4.6	8.1	84.3	4.4	4.3	3.1	6.3	18.5	31.3	25.3	16.5	21.4	52.1
4.9	2.3	17.1	7.3	11.3	2.8	1.8	4.3	30.3	29.9	10.5	7.0	13.3
11.2	2.9	7.9	5.5	28.6	6.5	2.4	2.4	32.7	69.9	22.0	10.4	13.5
24.2	5.0	8.4	3.0	32.7	14.0	4.3	2.9	31.1	107.7	44.6	18.7	21.2
31.4	6.2	9.0	2.6	30.8	17.9	5.3	3.2	32.7	114.9	54.9	22.8	25.1
50.8	15.2	14.1	2.0	15.8	25.4	11.7	5.3	38.0	90.9	71.2	40.9	42.5
22.1	32.9	33.1	2.4	8.5	14.0	23.1	11.6	52.3	76.3	63.9	69.7	80.5
15.1	33.2	45.7	2.6	7.0	10.6	23.8	15.0	54.7	69.6	57.1	72.5	93.0
7.6	23.6	111.4	4.1	5.4	7.1	23.2	37.6	77.7	69.3	53.9	87.7	179.5
3.4	8.8	230.0	7.8	3.0	3.3	9.6	85.2	88.8	39.2	27.7	45.1	232.5
1.4	1.5	23.7	75.7	8.2	1.4	1.8	14.7	300.1	46.7	13.7	13.5	49.9
2.8	1.1	4.1	26.1	79.3	2.3	1.3	2.0	166.1	163.1	14.6	8.6	14.4
11.4	2.8	5.0	3.4	92.4	9.7	3.1	2.2	52.0	314.3	50.5	19.8	23.1
34.4	6.1	8.5	2.5	43.0	26.0	6.3	3.6	48.7	206.4	99.0	34.3	36.2
56.4	18.1	16.1	2.4	15.2	35.6	16.2	6.7	56.1	122.1	115.7	63.9	62.3
20.3	34.7	33.3	3.0	8.9	15.5	28.2	13.0	70.8	98.2	84.3	96.0	105.0
13.3	34.5	49.7	3.2	7.3	11.0	28.7	18.1	72.6	87.1	71.4	97.9	124.8
9.9	19.5	49.5	4.7	8.2	10.7	23.9	22.9	114.8	118.1	90.3	127.9	197.8
8.0	12.2	41.1	7.3	8.6	9.2	15.7	21.6	173.7	131.0	86.7	108.9	220.2
7.6	8.1	22.9	11.2	12.8	8.9	10.5	12.6	255.1	193.4	88.8	80.2	139.5
9.4	7.4	16.0	7.4	21.0	11.0	9.5	8.3	191.8	292.6	103.7	70.7	96.3
14.6	8.2	14.9	4.4	25.7	16.6	10.4	7.4	119.9	297.1	137.4	73.0	85.0
19.5	9.7	15.8	3.8	22.2	21.8	12.0	7.7	103.9	250.1	164.1	80.8	87.6
21.6	20.6	25.5	3.7	12.3	21.1	22.5	11.7	97.4	153.6	134.8	112.6	118.3
14.3	25.3	35.2	3.9	9.6	14.2	26.9	15.6	100.3	128.0	104.4	126.3	144.6
11.3	22.9	44.8	4.2	8.6	11.8	26.4	20.0	105.7	120.5	94.8	129.5	172.3
11.9	17.4	34.0	4.8	10.6	13.2	21.1	16.3	126.6	152.2	111.6	124.5	163.9
11.5	14.6	30.4	5.5	11.5	13.0	18.2	15.1	143.6	167.8	114.8	117.0	160.2
11.6	12.7	25.9	5.9	13.1	13.3	15.9	13.0	156.3	191.8	119.3	106.9	143.7
12.3	12.1	23.4	5.5	14.9	14.2	15.1	11.7	148.9	211.4	126.3	102.0	130.2
13.8	12.1	22.3	4.9	16.0	15.8	15.1	11.0	134.5	217.8	135.9	100.7	122.1
14.9	12.6	22.3	4.7	15.7	17.0	15.6	10.9	127.0	210.6	142.1	102.6	121.0
15.5	17.5	27.5	4.3	12.4	16.7	20.6	13.1	116.2	168.1	129.7	117.8	135.6
13.5	19.8	32.1	4.4	10.8	14.5	23.0	15.1	115.7	149.9	115.3	125.8	149.9
12.3	18.7	34.3	4.6	10.5	13.5	22.4	16.2	120.5	148.6	111.5	126.2	159.8

Columns 1 - 12

4.9	8.1	85.8	202.8	91.0	11.0	6.7	14.1	45.5	92.8	18.7	4.2
16.6	21.2	108.7	167.1	102.2	38.1	21.5	14.5	22.1	77.7	61.5	12.2
18.0	28.4	112.4	94.2	58.8	47.3	37.0	20.0	14.8	48.5	88.9	23.6
12.5	25.5	116.2	73.6	38.4	34.7	38.0	26.7	16.2	33.5	72.7	28.1
9.4	22.2	119.9	67.7	30.2	26.0	34.3	31.8	18.6	27.3	56.7	27.4
7.0	18.7	125.7	67.3	25.3	19.1	28.6	37.3	22.3	23.7	42.5	23.8
4.6	13.9	143.7	84.8	23.6	11.7	18.6	45.2	34.8	23.5	25.9	15.5
2.4	6.3	93.8	110.9	29.3	5.9	7.1	32.9	64.6	37.4	12.9	5.9
2.4	5.3	74.1	123.5	39.2	5.8	5.5	23.0	65.9	53.2	12.1	4.4
3.0	4.4	44.5	122.8	80.6	8.3	4.0	8.9	37.6	168.5	15.0	2.7
18.5	12.8	50.0	80.1	95.7	76.1	16.2	7.4	11.8	157.6	112.0	8.6
16.0	18.9	63.9	54.9	43.6	68.5	33.4	12.2	9.3	50.4	172.4	22.4
12.6	22.5	88.0	58.5	34.7	41.8	41.3	20.2	12.6	33.3	103.0	33.4
8.9	20.7	105.4	59.3	27.4	25.8	36.0	29.1	16.8	25.6	59.7	31.1
6.9	17.9	112.1	60.1	23.8	19.2	29.5	34.5	20.2	22.7	43.8	25.8
4.2	12.6	121.7	68.6	20.5	11.3	18.7	45.0	31.1	21.4	25.7	16.4
2.3	6.4	93.6	93.2	23.6	5.9	7.8	38.9	61.1	29.9	13.2	6.6
2.1	4.9	69.3	103.9	31.1	5.2	5.4	24.4	66.2	44.0	11.4	4.5
1.0	1.8	18.9	37.0	19.0	3.5	2.1	6.2	20.9	61.8	9.3	1.8
1.3	1.3	7.4	10.2	7.9	10.3	2.1	2.2	4.3	30.0	51.5	1.6
3.2	4.3	18.4	15.0	9.6	17.5	8.9	5.2	5.2	14.3	104.4	8.2
4.4	8.4	39.6	26.4	13.5	17.2	18.3	12.2	9.6	15.7	67.7	20.9
4.6	11.1	64.5	38.0	15.9	14.3	22.1	22.1	15.4	17.3	39.6	23.9
4.5	12.1	81.2	45.7	17.2	13.2	21.7	29.2	19.1	18.4	32.8	21.5
3.2	9.1	86.9	53.3	16.3	8.7	14.4	37.9	28.8	19.0	21.1	13.6
1.9	4.9	67.2	67.6	18.4	4.9	6.4	33.5	52.7	25.5	11.8	5.8
1.7	4.1	54.7	76.9	23.3	4.5	4.9	21.9	57.2	35.8	10.6	4.3
1.0	2.3	23.4	28.9	10.8	3.3	3.4	10.1	20.9	22.7	10.3	3.3
1.0	2.2	19.6	19.8	7.4	3.9	3.8	8.5	13.7	14.2	15.3	3.8
1.4	3.0	23.0	20.2	7.7	5.2	5.4	9.7	13.0	12.5	21.5	5.6
1.7	4.0	30.0	23.9	8.9	6.0	7.4	12.5	14.6	13.2	21.3	8.0
2.1	5.0	38.1	28.3	10.3	6.5	9.3	15.9	16.8	14.3	20.3	9.9
2.2	5.8	46.0	32.6	11.4	6.8	10.3	19.5	19.2	15.4	19.3	10.7
1.9	5.1	48.5	37.7	12.1	5.6	8.4	22.9	25.2	17.0	15.5	8.4
1.4	3.5	41.6	45.7	14.1	4.0	5.0	20.2	36.3	22.3	10.9	4.8
1.4	3.2	37.4	47.3	15.4	3.9	4.4	16.4	36.7	26.2	10.4	4.2
1.2	2.8	27.8	30.4	10.6	3.7	4.4	12.4	22.4	19.4	11.7	4.3
1.2	2.9	26.5	26.1	9.2	4.1	4.8	11.8	18.5	15.9	13.9	4.9
1.4	3.2	27.8	25.4	9.0	4.6	5.5	12.3	17.3	14.7	16.0	5.7
1.5	3.7	30.8	26.6	9.4	5.0	6.4	13.5	17.7	14.7	16.6	6.7
1.7	4.1	34.4	28.4	10.0	5.3	7.2	15.1	18.6	15.0	16.7	7.5
1.8	4.5	38.0	30.6	10.5	5.5	7.7	16.8	19.8	15.6	16.4	8.0
1.7	4.2	39.6	33.8	11.2	5.0	7.0	18.4	23.4	16.8	14.6	7.2
1.4	3.3	36.4	38.6	12.4	4.0	5.1	17.3	29.8	20.3	11.5	5.0
1.3	3.1	34.2	39.0	12.8	3.9	4.7	15.6	29.9	21.9	11.2	4.5

Columns 13 - 25

11.4	25.2	61.5	17.6	10.0	11.6	25.9	39.3	33.2	17.8	31.4	65.0	64.5
10.4	11.8	40.7	44.5	22.9	9.9	12.5	24.3	38.9	20.3	23.1	36.6	40.6
15.0	7.7	25.9	68.1	45.4	14.2	9.8	18.0	59.2	34.3	31.1	37.9	41.4
21.3	8.4	20.4	63.3	59.8	20.5	11.7	16.8	70.8	47.8	43.7	49.5	50.0
26.2	9.5	18.4	54.2	63.0	25.5	13.7	17.0	75.6	55.5	53.6	59.3	57.0
31.4	11.2	17.8	44.3	58.5	30.5	16.2	17.7	76.8	58.8	62.8	69.3	63.5
37.0	16.0	19.9	29.9	39.4	34.4	21.7	20.2	66.8	49.2	68.3	82.8	68.5
28.8	28.9	34.6	18.2	16.8	29.1	37.1	32.9	56.8	34.7	68.6	115.3	88.8
20.9	32.6	47.4	16.7	12.7	22.3	40.5	41.2	52.2	29.8	59.3	115.3	94.6
7.8	30.9	123.6	15.7	6.8	8.6	28.7	67.0	29.9	14.6	25.9	61.4	79.2
5.6	7.7	75.3	71.0	15.5	5.5	7.9	37.5	37.4	13.5	13.9	22.8	39.5
9.5	5.2	27.9	124.2	41.0	9.3	6.8	19.2	74.6	28.7	22.3	27.8	37.4
16.3	6.8	19.9	91.2	70.1	16.1	9.6	16.1	80.1	47.1	36.5	42.0	46.1
24.6	8.9	17.5	59.7	75.0	24.6	13.1	16.6	82.2	61.6	53.7	58.8	58.0
30.0	10.5	17.2	47.1	66.0	30.2	15.7	17.6	83.4	65.5	64.7	69.9	65.7
40.1	15.3	18.7	31.3	43.8	39.1	21.6	20.1	75.5	57.3	77.9	88.2	74.0
35.4	27.5	28.6	19.2	19.2	35.7	36.0	29.3	61.5	38.8	78.7	119.4	88.4
23.1	32.9	42.0	17.3	13.5	25.1	43.4	39.9	58.1	33.4	67.6	129.6	101.7
6.5	25.1	123.7	18.2	6.0	8.3	42.5	161.8	50.5	20.2	32.8	94.3	226.9
2.3	4.0	69.6	177.5	4.5	3.0	7.1	173.1	123.1	13.6	13.6	28.4	250.1
5.0	3.4	13.6	276.8	26.0	6.2	6.2	17.5	271.9	39.8	23.9	33.1	62.6
11.6	5.9	13.1	145.3	93.8	13.8	10.3	15.4	177.4	93.2	45.1	54.1	67.0
21.4	9.1	15.1	58.6	89.5	25.1	15.3	18.1	124.8	103.4	71.2	78.3	81.4
28.2	10.8	16.3	43.8	68.3	32.2	17.6	19.2	107.0	88.5	81.4	86.5	83.5
38.6	15.6	18.4	30.2	41.2	43.5	23.8	21.8	91.3	69.5	98.1	106.2	89.4
35.5	27.3	26.7	19.6	18.1	40.5	39.6	30.1	71.8	44.2	98.2	145.6	102.2
22.5	31.8	37.9	18.4	13.9	26.4	47.5	40.8	69.1	39.0	79.2	155.2	118.4
11.5	17.7	36.7	27.0	12.8	16.1	37.7	60.9	114.8	49.6	71.9	159.4	243.5
9.8	10.5	21.9	51.9	15.6	14.0	21.7	41.1	213.9	66.1	67.9	118.9	233.5
11.0	9.3	16.6	67.3	24.1	15.6	18.4	27.5	260.9	91.0	72.2	104.5	153.7
13.9	10.0	16.1	54.8	36.1	19.5	19.3	24.4	214.7	114.7	84.5	109.2	131.1
17.5	11.1	16.7	43.7	42.4	24.2	20.9	24.0	170.1	116.5	96.4	115.6	124.0
21.5	12.3	17.5	37.1	41.5	29.3	22.5	24.3	142.9	104.5	105.6	121.2	120.4
26.2	15.8	19.5	29.7	30.8	35.6	27.8	26.4	119.6	83.3	116.5	139.1	121.2
23.4	23.1	26.6	22.4	17.1	31.4	41.1	34.6	94.7	54.5	107.6	178.8	134.7
18.5	25.0	32.5	21.9	15.0	24.5	46.1	42.4	93.0	50.3	91.3	181.5	151.2
14.4	17.0	27.6	29.5	17.1	20.3	34.7	42.9	132.3	64.2	89.4	165.6	194.1
13.7	13.5	22.1	38.1	20.0	19.5	27.3	35.6	172.5	77.5	89.0	145.2	186.3
14.1	12.3	19.5	43.4	24.1	20.1	24.4	30.6	192.0	90.0	90.4	133.4	162.8
15.4	12.3	18.7	41.6	28.4	21.9	24.0	28.4	181.8	98.8	95.6	131.6	148.7
17.1	12.7	18.6	38.5	31.3	24.2	24.4	27.6	165.2	101.1	101.4	132.8	141.0
19.1	13.3	18.9	35.5	32.2	26.8	25.2	27.4	150.3	97.9	106.6	135.2	136.4
21.2	15.5	20.5	31.0	27.6	29.6	28.8	29.0	132.5	84.9	111.8	147.9	136.7
20.1	20.1	25.4	25.4	18.6	27.9	37.7	35.1	111.2	62.8	106.4	175.8	148.3
17.9	20.9	28.2	25.1	17.1	24.7	40.2	39.2	110.2	59.6	97.9	178.0	159.1

Columns 1 - 12

45.2	30.7	113.4	285.9	181.3	6.3	8.4	17.4	5.5	15.7	7.4	6.0
114.7	70.2	129.9	201.9	160.5	16.7	18.7	20.5	2.8	9.8	19.4	12.1
120.7	96.5	132.2	141.3	108.8	19.9	30.3	26.6	2.4	7.1	25.6	19.6
84.8	97.4	145.4	120.1	72.2	15.8	38.3	35.7	2.6	5.6	22.7	25.3
69.3	90.5	152.1	116.6	62.1	13.4	39.1	40.1	2.8	5.3	20.2	26.4
61.0	82.5	162.1	121.7	59.9	11.8	36.2	43.1	3.1	5.3	17.8	24.6
37.9	50.5	187.8	166.7	63.4	6.9	20.6	45.7	4.9	6.5	10.2	14.4
25.6	27.9	124.7	211.9	79.3	4.6	11.0	30.2	8.3	10.5	6.6	8.4
26.7	24.1	104.8	241.3	109.4	4.5	8.7	22.9	8.6	14.1	6.0	6.7
33.0	17.6	64.1	250.7	275.8	4.8	4.9	10.5	5.3	34.0	5.2	3.7
150.5	36.4	62.8	162.0	369.1	19.1	7.9	8.5	1.8	19.5	16.2	5.1
263.4	88.6	73.3	90.7	131.7	44.3	22.8	13.5	1.3	8.7	49.4	14.4
121.8	127.3	106.3	90.5	67.6	25.3	48.7	25.8	1.9	5.1	37.6	31.5
71.7	98.8	136.4	102.9	57.3	14.7	45.9	37.6	2.6	4.9	22.7	31.2
61.1	87.3	147.7	107.7	55.0	12.3	41.7	42.1	2.8	5.0	19.2	28.7
35.6	51.4	193.1	135.5	52.0	6.8	22.9	54.9	4.3	5.6	10.5	16.3
23.2	26.7	122.0	193.6	67.1	4.4	11.0	32.1	8.5	9.5	6.5	8.6
22.6	21.9	96.9	217.9	89.4	4.1	8.5	23.4	9.6	13.3	5.8	6.8
14.4	8.8	33.1	131.5	162.4	3.6	3.4	8.1	6.0	65.0	4.1	3.0
26.4	2.1	5.3	15.2	105.2	26.4	0.9	1.5	0.7	132.4	20.0	0.8
99.5	9.1	6.4	8.0	19.0	91.9	4.2	1.7	0.3	7.5	203.3	3.6
57.9	50.5	30.8	25.7	18.4	32.7	46.8	9.5	0.8	2.2	96.8	65.6
43.3	70.4	78.1	58.1	31.1	11.9	54.3	27.5	2.1	3.8	23.2	51.2
36.9	61.0	108.0	75.4	35.6	8.6	40.3	40.9	2.8	4.5	15.0	33.5
23.2	33.8	148.8	109.1	39.6	4.9	17.1	54.3	5.0	5.8	8.0	13.8
18.9	22.1	101.3	161.1	54.4	3.9	9.9	30.3	8.9	9.1	6.0	8.2
16.7	16.9	73.0	166.1	66.4	3.5	7.4	20.6	9.9	13.0	5.3	6.4
8.9	8.2	29.5	61.9	41.7	3.3	4.6	10.8	4.7	19.4	5.3	4.8
8.5	6.8	21.7	36.7	25.3	4.9	4.2	8.6	2.8	14.1	8.1	4.6
11.2	8.3	21.0	27.2	15.2	7.5	5.9	8.7	2.0	6.4	16.4	7.0
12.2	13.5	28.0	31.2	15.5	5.8	10.5	11.7	2.1	4.8	13.8	13.0
15.1	21.0	43.6	42.5	19.8	5.2	16.3	18.6	2.7	4.9	11.0	19.1
16.5	24.3	59.2	53.6	23.7	4.9	17.5	25.7	3.3	5.4	9.3	18.6
14.7	19.9	76.2	78.2	30.2	3.9	11.8	31.3	5.1	6.7	6.9	11.4
12.9	15.2	62.6	98.9	37.3	3.4	8.1	22.9	7.4	9.1	5.7	7.7
11.6	12.3	48.8	98.3	43.1	3.2	6.5	17.1	7.6	12.3	5.3	6.3
9.7	10.4	36.1	61.8	32.0	3.5	6.2	14.0	4.9	12.0	6.1	6.6
9.7	10.2	33.1	50.7	26.0	4.0	6.5	13.4	4.0	10.0	7.3	7.1
10.5	11.3	32.8	43.9	21.5	4.6	7.7	13.6	3.3	7.5	9.3	8.7
11.2	13.1	36.0	44.5	21.0	4.5	9.2	15.1	3.3	6.7	9.3	10.6
12.2	15.6	42.2	48.5	22.1	4.3	11.0	18.0	3.5	6.4	8.7	12.4
12.9	17.2	48.9	53.6	23.7	4.2	11.9	21.0	3.8	6.5	8.1	13.0
12.7	16.3	57.3	68.3	28.2	3.8	10.1	23.7	4.9	7.4	6.9	10.4
11.7	13.8	52.4	80.6	33.0	3.4	7.9	20.2	6.3	9.1	6.0	7.9
10.9	12.1	45.3	79.6	35.5	3.3	6.9	17.1	6.3	10.8	5.8	7.0

Forward Transfer Matrix, Model 10

Columns 13 - 25

16.6	9.6	7.0	6.6	5.2	3.8	9.0	10.8	30.7	33.4	26.7	61.3	56.0
16.2	5.3	4.0	14.1	9.3	3.4	4.9	5.9	32.2	36.9	22.4	36.6	31.4
20.4	5.1	3.0	18.9	14.7	4.3	4.8	4.7	42.0	52.6	28.6	39.0	30.7
27.4	6.1	2.6	18.3	19.0	5.8	5.7	4.4	50.5	70.8	38.5	48.8	36.0
30.9	6.7	2.6	17.0	20.0	6.6	6.3	4.4	53.3	77.7	43.3	54.1	39.1
33.4	7.3	2.6	15.5	18.9	7.1	6.8	4.6	52.8	77.5	45.4	57.8	41.2
37.7	10.9	3.3	9.7	12.0	7.9	9.9	5.9	47.3	64.6	47.9	75.5	51.9
31.9	16.9	5.5	7.3	7.8	7.5	15.6	9.6	50.5	59.4	51.3	109.9	77.8
25.2	16.0	7.0	6.7	6.4	6.1	15.3	11.9	46.0	50.9	44.2	104.7	81.9
11.3	8.8	14.3	5.3	3.4	2.8	8.7	21.3	28.2	26.6	21.9	60.3	77.6
7.2	3.1	6.7	11.6	4.0	1.6	3.0	8.5	20.2	17.1	11.0	21.5	25.6
10.5	2.8	3.3	30.9	11.0	2.3	2.6	4.6	39.6	32.6	15.7	21.8	20.1
20.1	4.5	2.3	28.2	23.0	4.4	4.2	3.8	54.3	68.0	30.9	37.7	29.0
29.3	6.2	2.4	19.4	23.3	6.4	5.9	4.2	57.4	84.8	43.6	52.2	38.1
32.9	6.9	2.5	16.9	21.9	7.1	6.5	4.4	57.3	86.7	47.7	57.4	41.2
44.5	10.2	2.9	10.2	13.7	9.3	9.3	5.3	51.2	73.8	54.5	75.4	50.9
35.3	18.4	5.1	7.5	8.2	8.4	16.9	9.2	54.7	64.9	57.2	120.0	81.0
27.5	18.4	6.9	6.9	6.7	6.8	17.8	12.1	52.4	57.4	50.8	123.6	92.5
11.3	11.0	33.2	6.3	3.3	3.2	12.4	55.4	47.3	36.8	30.8	104.6	201.1
2.2	1.5	118.5	39.3	0.9	0.7	1.8	160.5	74.5	12.2	8.0	23.3	219.8
2.0	0.7	4.4	197.8	4.0	0.6	0.8	8.1	259.0	22.2	6.8	10.6	28.4
9.2	2.3	1.2	114.2	76.7	2.4	2.4	2.3	154.9	124.0	23.5	26.1	23.1
26.1	5.6	2.1	25.2	43.9	6.7	5.7	4.1	96.5	164.6	58.3	60.1	46.2
38.2	7.6	2.5	15.8	28.4	9.4	7.6	4.9	80.2	138.2	73.5	76.2	55.0
54.2	13.2	3.3	9.2	13.3	12.6	12.4	6.3	65.1	93.7	79.4	105.7	68.4
37.9	21.1	5.1	7.6	8.4	9.5	19.9	9.6	64.0	74.2	67.9	146.4	94.2
28.2	20.9	7.3	7.3	6.8	7.6	22.0	13.5	66.0	68.7	62.1	164.3	120.1
17.7	12.0	12.8	9.6	5.8	5.6	14.9	26.8	101.8	77.0	61.2	182.0	269.9
14.2	7.8	10.6	16.6	5.9	4.6	9.5	24.3	182.1	89.7	59.0	140.3	289.2
13.7	5.9	4.6	33.9	9.4	4.5	6.8	10.3	340.4	147.9	60.5	95.3	129.7
17.4	6.4	3.2	26.8	18.1	5.7	7.2	6.9	260.3	228.5	72.0	92.9	92.5
26.1	8.0	3.2	17.4	24.0	8.4	8.8	6.6	155.0	233.0	95.5	106.3	88.0
35.0	9.8	3.4	13.4	21.4	11.0	10.5	7.0	119.1	187.2	110.1	119.8	90.3
43.2	15.1	4.2	9.9	13.0	12.8	15.6	8.4	94.7	125.5	107.5	151.6	102.1
35.2	20.7	5.5	8.6	8.7	10.2	22.1	11.0	87.3	95.2	86.6	192.0	125.8
27.3	19.1	7.5	8.4	7.4	8.1	22.5	14.9	87.8	86.5	76.0	204.5	157.5
23.0	13.3	7.9	10.8	8.1	7.3	16.1	16.5	121.5	105.0	80.4	189.4	197.4
21.9	11.3	6.8	13.5	8.9	7.1	13.3	14.6	155.9	122.2	83.4	169.9	189.3
21.8	9.7	5.1	17.5	11.2	7.1	11.2	11.0	198.3	155.3	86.6	143.0	147.4
23.6	9.7	4.5	16.8	13.7	7.7	11.0	9.5	184.4	177.9	92.4	136.9	127.2
27.4	10.3	4.2	14.6	15.7	8.9	11.6	8.8	153.4	182.4	101.8	138.7	117.0
31.5	11.2	4.2	13.0	15.8	10.2	12.4	8.7	132.3	170.3	108.7	143.6	113.6
35.6	14.5	4.7	10.7	12.3	11.1	15.8	9.6	110.6	133.9	106.1	164.9	120.0
32.1	17.8	5.7	9.6	9.3	9.7	19.8	11.6	101.5	108.0	91.7	192.7	138.3
27.7	17.0	6.8	9.5	8.3	8.5	19.9	13.9	102.4	101.1	84.2	200.3	159.7

Forward Transfer Matrix, Model 11

Columns 1 - 12

44.9	69.6	161.6	249.8	121.9	9.5	5.3	8.7	24.4	29.7	10.0	12.4
79.7	110.5	175.0	183.1	123.4	19.7	9.4	8.5	12.2	23.0	19.0	22.0
80.6	151.4	172.6	105.3	79.9	25.0	17.1	11.5	7.4	15.7	25.7	40.2
57.9	166.8	186.8	83.8	53.9	19.3	20.9	14.6	6.6	10.8	21.6	49.1
46.0	170.0	200.6	78.6	44.3	15.4	21.6	16.6	6.6	9.0	18.2	50.3
34.4	165.3	232.1	82.0	37.3	11.0	19.9	19.2	7.3	7.6	13.7	45.6
26.7	135.5	269.5	114.9	38.6	7.3	14.5	20.5	10.9	8.0	9.6	32.7
20.2	61.2	195.9	213.2	54.0	4.6	6.2	16.3	27.9	15.0	6.5	14.7
24.7	53.6	160.6	240.6	77.4	5.4	5.0	12.3	33.3	23.1	6.9	11.8
38.0	47.9	115.2	235.8	164.3	8.8	3.8	6.8	28.7	58.9	9.6	8.9
154.0	79.6	97.2	104.1	215.4	43.7	7.2	5.0	7.2	48.3	36.4	17.4
127.3	133.0	119.0	77.1	86.7	50.6	17.0	8.1	5.4	19.7	49.5	41.5
60.0	170.0	154.3	65.8	45.7	23.5	25.7	13.2	5.6	9.8	27.5	62.1
36.6	174.3	184.2	63.2	33.7	13.4	25.3	17.4	6.0	7.3	17.3	59.8
30.0	170.1	205.5	64.5	30.3	10.5	23.3	19.9	6.4	6.6	14.0	54.3
21.1	134.4	272.7	87.0	28.9	6.3	15.9	25.8	9.3	6.4	8.9	35.7
16.8	61.3	203.0	178.1	41.1	4.2	6.9	20.2	25.2	11.6	6.4	16.2
20.2	50.1	158.0	223.2	62.3	4.6	5.0	13.8	34.8	19.5	6.5	12.1
13.9	17.5	41.2	77.0	95.5	5.6	2.1	5.0	23.9	121.6	10.4	5.4
56.5	11.7	13.0	12.6	65.8	88.0	1.7	1.3	1.8	107.6	199.5	4.9
70.2	54.3	37.8	20.1	28.1	81.6	13.6	3.8	2.0	11.4	177.2	45.3
32.5	121.2	87.5	30.9	19.7	20.7	32.7	11.0	4.0	5.5	34.5	97.0
21.3	142.1	133.3	39.8	19.0	9.8	28.1	17.7	5.6	5.2	16.2	73.0
19.0	141.3	158.2	44.2	19.2	7.8	24.6	21.4	6.2	5.3	12.9	61.0
14.3	108.9	223.6	60.4	19.6	4.9	16.0	32.5	9.3	5.6	8.4	36.9
10.5	45.2	147.5	109.1	24.9	3.2	6.4	25.9	25.4	9.3	6.4	15.9
11.7	36.2	106.7	132.1	36.9	3.4	4.9	15.9	36.9	16.2	6.4	12.3
6.3	22.9	48.5	39.5	20.7	3.4	4.2	10.5	15.1	21.3	10.3	11.3
6.6	21.3	36.4	20.6	11.6	6.4	4.4	8.1	6.7	12.4	30.3	12.6
8.2	33.7	44.7	19.3	8.9	7.2	8.0	9.3	5.3	5.9	29.4	24.8
11.6	64.2	72.2	26.3	11.4	7.5	15.7	13.6	6.1	5.1	19.6	47.7
12.7	87.3	101.9	33.4	13.6	6.6	18.8	18.2	7.2	5.4	14.3	52.0
12.2	91.1	119.4	37.1	14.3	5.7	18.0	21.6	7.8	5.5	12.0	47.3
10.1	69.8	137.0	51.4	15.8	4.2	12.3	27.2	12.4	6 5	9.0	30.8
8.0	38.5	104.2	71.5	18.5	3.1	6.5	23.4	22.1	9.0	7.3	16.9
7.9	32.4	81.7	72.7	22.0	3.0	5.5	16.9	26.1	12.6	7.4	14.3
7.0	35.2	69.2	42.0	15.1	3.6	6.8	15.7	14.4	10.6	10.8	18.6
7.2	38.1	67.4	35.4	13.1	4.2	7.7	15.2	11.4	8.8	13.5	21.4
7.9	44.6	71.8	33.5	12.3	4.6	9.4	15.8	10.0	7.4	14.3	26.2
8.7	52.7	79.8	34.1	12.5	4.9	11.2	17.0	9.7	6.8	13.6	31.3
9.3	59.9	89.1	35.8	13.0	4.9	12.5	18.6	9.7	6.6	12.6	34.2
9.3	62.0	94.9	37.3	13.3	4.7	12.6	19.8	10.0	6.6	11.8	34.0
8.7	55.3	101.0	44.6	14.4	4.1	10.6	21.9	12.7	7.3	10.1	28.0
7.6	40.0	88.2	53.5	16.0	3.4	7.4	20.4	17.4	8.9	8.9	19.7
7.4	36.5	79.5	53.2	16.9	3.4	6.8	17.9	18.4	10.2	9.0	18.1

Forward Transfer Matrix, Model 11

Columns 13 - 25

10.4	66.5	22.6	2.4	13.4	3.9	3.8	3.8	26.4	23.0	18.7	21.7	35.7
9.3	37.1	15.0	4.5	21.3	3.9	2.4	2.9	34.7	28.2	16.4	14.3	24.3
12.3	30.8	10.6	6.6	38.1	5.6	2.4	2.5	52.0	45.8	22.0	15.0	23.9
15.6	33.1	8.6	6.1	47.6	7.3	2.8	2.2	56.4	57.0	27.7	17.6	25.9
17.5	35.4	7.9	5.4	50.0	8.2	3.0	2.0	55.7	60.6	30.8	19.1	27.1
19.9	39.5	7.5	4.2	47.0	9.1	3.4	1.9	50.5	59.2	33.5	20.7	28.1
21.1	52.3	8.5	3.0	35.2	9.0	4.1	2.0	40.9	48.7	33.4	23.3	29.8
19.9	107.9	16.4	1.9	17.5	7.1	6.7	3.3	34.1	34.9	33.5	36.1	45.0
15.6	106.9	22.3	1.9	14.0	5.6	6.1	4.0	30.8	29.2	27.9	33.9	47.1
8.6	69.1	41.2	2.1	10.0	3.2	3.8	6.0	24.1	19.2	16.4	22.3	47.4
5.6	22.5	22.6	7.0	16.2	2.4	1.5	4.5	38.7	20.9	10.4	9.3	22.8
8.9	23.1	12.1	11.8	36.9	4.1	1.8	3.1	69.7	42.9	16.9	11.8	22.1
14.5	30.3	8.0	8.2	57.7	6.9	2.6	2.2	69.6	66.3	27.4	17.1	25.9
18.8	36.1	7.3	5.5	60.1	9.0	3.2	2.0	62.6	72.2	34.6	20.8	29.4
21.3	39.2	7.2	4.6	56.8	10.1	3.5	2.0	58.2	71.0	37.8	22.4	30.6
26.2	52.7	7.8	2.9	40.0	11.4	4.5	2.0	44.8	56.8	40.6	25.7	32.2
24.8	118.2	14.5	2.0	19.9	8.8	7.9	3.2	38.6	40.7	40.8	41.5	48.2
18.1	125.4	21.1	1.9	14.9	6.4	7.2	4.1	34.0	32.8	32.5	39.7	51.9
7.5	96.3	158.6	2.4	7.2	2.9	5.8	23.7	34.5	21.0	19.0	39.3	162.7
1.9	10.2	116.2	29.1	5.3	0.8	0.8	40.9	111.6	12.6	5.6	7.4	93.0
4.6	12.4	9.7	49.9	44.3	2.3	1.1	4.1	230.6	53.1	11.4	8.4	22.8
13.3	29.2	6.4	13.4	98.1	6.8	2.7	2.1	128.3	119.8	31.3	19.9	31.5
21.1	42.4	7.4	6.0	82.3	10.7	4.0	2.3	88.6	109.4	46.3	28.0	40.3
25.0	47.6	7.7	4.7	71.4	12.6	4.5	2.3	77.7	100.4	51.9	30.7	42.4
35.9	67.3	8.7	2.9	45.9	16.0	6.2	2.4	57.6	75.8	59.9	36.5	44.6
35.8	170.2	15.8	2.2	21.5	12.1	12.2	4.0	51.8	53.8	60.7	62.9	67.4
23.7	199.2	25.0	2.1	16.7	8.4	11.5	5.5	48.7	45.5	47.8	65.6	80.8
16.9	142.7	56.1	3.4	16.6	6.9	11.0	14.7	84.8	57.4	49.8	92.1	233.6
12.8	65.7	30.5	10.2	18.7	5.8	5.9	15.9	221.7	78.4	45.9	60.9	249.9
13.8	50.1	12.6	11.6	36.6	6.7	4.7	5.8	325.8	140.6	47.5	42.2	97.5
18.9	55.6	10.1	8.2	67.3	9.6	5.3	3.6	185.0	167.9	56.6	41.8	69.2
24.4	63.2	10.3	5.6	70.3	12.4	6.0	3.4	119.4	138.2	65.0	44.6	65.7
28.4	68.7	10.6	4.6	63.8	14.3	6.6	3.3	101.0	123.3	70.6	46.9	65.7
36.3	105.3	12.9	3.3	42.5	16.0	9.5	3.8	80.3	94.0	77.5	59.1	72.9
35.5	180.3	18.5	2.6	24.1	12.9	13.7	5.0	68.8	68.2	72.4	79.3	89.8
26.5	205.6	26.4	2.6	20.6	9.9	13.6	6.6	68.0	61.6	61.4	85.6	109.0
24.6	136.9	25.9	3.9	27.3	10.4	11.3	7.6	108.1	88.1	70.6	88.9	147.2
23.5	108.8	21.1	5.0	31.6	10.4	9.6	7.1	140.5	105.4	71.9	79.2	142.5
23.8	95.1	17.1	5.4	38.7	11.0	8.6	5.8	154.0	123.2	72.9	69.4	117.2
25.0	90.3	15.2	5.3	45.6	11.8	8.3	5.0	142.1	130.3	74.0	64.2	100.6
26.8	90.1	14.4	4.8	49.2	12.8	8.3	4.6	125.1	127.1	75.8	62.4	92.5
28.3	92.3	14.3	4.5	48.7	13.5	8.5	4.5	115.8	122.2	77.8	62.8	90.3
31.8	114.6	15.8	3.8	40.1	14.1	10.2	4.7	99.0	103.9	80.2	70.0	93.2
31.4	154.8	19.9	3.2	28.7	12.6	12.6	5.6	88.5	84.4	76.7	83.4	106.6
28.0	163.8	23.4	3.2	26.4	11.3	12.7	6.5	88.9	80.4	71.6	87.4	118.9

REFERENCES

Ahmed, N., Milne, P. J. and Harris, S. G. (1975). "Electrocardiographic Data Compression via Orthogonal Transforms," IEEE Trans. Bio-Med. Eng. Vol. BME-22(6), pp.484-487

Barbato, E., Pileggi, F., Debes, A. C., Fujioka, T., Magalhaes, M. S., Tranchesi, J., SanJuan, E. and Decourt, L. V. (1958). "Study of the sequence of ventricular activation and the QRS complex of the normal human heart using direct epicardial leads.," Am. Heart J. Vol. 55(6), pp.867-880

Barnard, A. C. L., Duck, I. M. and Lynn, M. S. (1967). "The Application of Electromagnetic Theory to Electrocardiology: I. Derivation of the Integral Equations," Biophysical J. Vol. 7, pp.443-462

Barnard, A. C. L., Duck, I. M. and Lynn, M. S. (1967a). "The Application of Electromagnetic Theory to Electrocardiology: II. Numerical Solution of the Integral Equations," Biophysical J. Vol. 7, pp.463-491

Barnard, A. C. L., Holt, J. R. and Kramer, J. O., (1976). "Models and Methods in Inverse Electrocardiology: The UAB Choice," in The Theoretical Basis of Electrocardiology, ed. C. V. Nelson and D. B. Geselowitz, Clarendon Press, Oxford

Barr, R. C., Pilkington, T. C., Boineau, J. P. and Spach, M. S. (1966). "Determining Surface Potentials from Current Dipoles with Application to Electrocardiography," IEEE Trans. Bio-Med. Eng. Vol. BME-13(2), pp.88-92

Barr, R. C., Spach, M. S. and Herman-Giddens, G. Scott (1971). "Selection of the Number and Positions of Measuring Locations for Electrocardiography," IEEE Trans. Bio-Med. Eng. Vol. BME-18(2), pp.125-138

Barr, R. C., Ramsey, M. III. and Spach, M. S. (1977). "Relating Epicardial to Body Surface Potential Distributions by means of Transfer Coefficients Based on Geometry Measurements," IEEE Trans. Bio-Med. Eng. Vol. BME-24(1), pp.1-11

Barr, R. C. and Spach, M. S. (1978). "Inverse Calculation of QRS-T Epicardial Potentials from Body Surface Potential Distributions for Normal and Ectopic Beats in the Intact Dog," Circ Res Vol. 42(5), pp.661-675

Barr, R. C. and Spach, M. S. (1983). "Construction and Interpretation of Body Surface Maps," Progress in Cardiovascular Diseases Vol. XXVI(1), pp.33-42

Burger, H. C. and VanMilaan, J. B. (1943). "Measurements of the specific resistance of the human body to direct current," Act. Med. Scand. Vol. 114, p.584, (Cited in Rush et. al. 1963)

Burger, H. C. and VanDongen, R. (1961). "Specific electrical resistance of body tissues," Physics in Medicine and Biology

Vol. 5, p.431, (Cited in Rush et. al. 1963)

Burgess, M. J., Lux, R. L., Wyatt, R. F. and Abildskov, J. A. (1978). "The Relation of Localised Myocardial Warming to Changes in Cardiac Surface Electrograms in Dogs," Circ Res Vol. 43(6), pp.899-907

Cuppen, J. J. M. (1983) A Numerical Solution of the Inverse Problem of Electrocardiography, PhD Thesis Nijmegen University, The Netherlands.

Cuppen, J. J. M. and VanOosterom, A. (1984). "Model studies with the inversely calculated isochrones of ventricular depolarisation," IEEE Trans. Bio-Med. Eng. Vol. BME-31(10), pp.652-659

Dahlquist, G. and Bjorck, A., (1974). Numerical Methods, Prentice-Hall, Englewood Cliffs, New Jersey

Durrer, D., VanDam, R. T., Freud, G. E., Janse, M. J., Meijler, F. L. and Arzbaecher, R. C. (1970). "Total Excitation of the Isolated Human Heart," Circulation Vol. 41, pp.899-912

Frank, E. (1952). "Electric Potential Produced by Two Point Current Sources in a Homogeneous Conducting Sphere," J. App. Physics Vol. 23(11), pp.1225-1228

Franzone, P. C., (1980). "Regularization Methods Applied to an Inverse Problem in Electrocardiology," in Computing Methods in Applied Sciences and Engineering, ed. R. Glowinski and J. L. Lions, North-Holland Publ.

Gelernter, H. L. and Swihart, J. C. (1964). "A mathematical-physical model of the genesis of the electrocardiogram," Biophys. J. Vol. 4, pp.285-301

Geselowitz, D. B. (1967). "On bioelectric potentials in an inhomogeneous volume conductor," Biophys. J. Vol. 7, pp.1-11

Geselowitz, D. B., (1976). "Determination of multipole components," in The Theoretical Basis of Electrocardiology, ed. C. V. Nelson and D. B. Geselowitz, Clarendon Press, Oxford

Geselowitz, D. B. and Thorsson, H. T. (1983). "Multiple Dipole Inverse Solutions from Human Body Surface Electrocardiograms," IEEE Computers in Cardiology, pp.461-464

Guardo, R., Sayers, B. McA. and Monro, D. M., (1976). "Evaluation and analysis of the cardiac electrical multipole series based on a two dimensional Fourier technique," in The Theoretical Basis of Electrocardiology, ed. C. V. Nelson and D. B. Geselowitz, Clarendon Press, Oxford

Heringa, A., Uijen, G. J. H. and VanDam, R. T., (1981). "A 64-Channel System for Body Surface Potential Mapping," in Electrocardiology, ed. Z. Antaloczy and I. Preda, Budapest

Holt, J. H., Barnard, A. C. L., Lynn, M. S. and Svendsen, P. (1969). "A Study of the Human Heart as a Multiple Dipole Electric Source: I. Normal Adult Male Subjects," Circulation Vol. 40(5), pp.687-696

Holt, J. H., Barnard, A. C. L., Lynn, M. S. and Svendsen, P. (1969a). "A Study of the Human Heart as a Multiple Dipole Electric Source: II. Diagnosis and Quantitation of Left Ventricular Hypertrophy," Circulation Vol. 40(5), pp.697-710

Holt, J. H., Barnard, A. C. L., Lynn, M. S. and Svendsen, P. (1969b). "A Study of the Human Heart as a Multiple Dipole Electric Source: III. Diagnosis and Quantitation of Right Ventricular Hypertrophy," Circulation Vol. 40(5), pp.711-718

Horacek, B. M. (1974). "Numerical model of an inhomogeneous human torso," Adv. Cardiol. Vol. 10, pp.51-57

Kaufman, W. and Johnston, F. D. (1943). "The electrical conductivity of the tissues near the heart and its bearing on the distribution of cardiac action currents," Am. Heart J. Vol. 26, p.42, (Cited in Rush et. al. 1963)

Kernighan, B. W. and Ritchie, D. M., (1978). The C Programming Language, Prentice-Hall, Englewood Cliffs, New Jersey

Kilpatrick, D., Duffin, P., Vickery, J. C. and Bourdillon, P. J. (1979). "Apparatus for the Rapid Measurement of Electrocardiographic Body Surface Potentials," Med. Biol. Eng. Comput. Vol. 17, pp.257-260

Kolman, B., (1980). Introductory Linear Algebra with Applications, Macmillan

Liebman, J., Thomas, C. W., Rudy, Y. and Plonsey, R. (1981). "Electrocardiographic body surface potential maps of the QRS of

normal children," J. Electrocardiology Vol. 14(3), pp.249-260

Lo, G. C. (1977) Estimation of Epicardial Electrical Potentials from Body Surface Measurements Based on a Digital Simulation of the Thorax, PhD Thesis, University of London.

Lorrain, P. and Corson, D., (1970). Electromagnetic Fields and Waves, W. H. Freeman and Co., San Francisco

Lux, R. L., Smith, C. R., Wyatt, R. F. and Abildskov, J. A. (1978). "Limited Lead Selection for Estimation of Body Surface Potential Maps in Electrocardiography," IEEE Trans. Bio-Med. Eng. Vol. BME-25(3), pp.270-276

Lux, R. L., Burgess, M. J., Wyatt, R. F., Evans, A. K., Vincent, G. M. and Abildskov, J. A. (1979). "Clinically Practical Lead Systems for Improved Electrocardiography: Comparison with Precordial Grids and Conventional Lead Systems," Circulation Vol. 59(2), pp.356-363

Lux, R. L., Evans, A. K., Burgess, M. J., Wyatt, R. F. and Abildskov, J. A. (1981). "Redundancy Reduction for Improved Display and Analysis of Body Surface Potential Maps: I. Spatial Compression," Circ Res Vol. 49(1), pp.186-196

Lux, R. L., Evans, A. K., Burgess, M. J., Wyatt, R. F. and Abildskov, J. A. (1981a). "Redundancy Reduction for Improved Display and Analysis of Body Surface Potential Maps: II. Temporal Compression," Circ Res Vol. 49(1), pp.197-203

Martin, R. O. and Pilkington, T. C. (1972). "Unconstrained inverse electrocardiography: Epicardial potentials," IEEE Trans. Bio-Med. Eng. Vol. BME-19, pp.276-285

Martin, R. O., Pilkington, T. C. and Morrow, M. N. (1975). "Statistically constrained inverse electrocardiography," IEEE Trans. Bio-Med. Eng. Vol. BME-22, pp.487-492

Miller, W. T., III and Geselowitz, D. B. (1978). "Simulation Studies of the Electrocardiogram: I. The Normal Heart," Circ Res Vol. 43(2), pp.301-315

Miller, W. T., III and Geselowitz, D. B. (1978a). "Simulation Studies of the Electrocardiogram: II. Ischemia and Infarction," Circ Res Vol. 43(2), pp.315-323

Monro, D. M. and Bourdillon, P. J. (1978). "Techniques for surface and epicardial isopotential mapping," Adv. Cardiol Vol. 21, pp.11-14

Noble, D., (1979). The initiation of the heartbeat, Clarendon Press, Oxford

Okamoto, Y., Teramachi, Y. and Musha, T. (1983). "Limitation of the inverse problem in body surface potential mapping," IEEE Trans. Bio-Med. Eng. Vol. BME-30(11), pp.749-754

Plonsey, R., (1976). "Laws Governing Current Flow in the Volume Conductor," in The Theoretical Basis of Electrocardiology, ed. C. V. Nelson and D. B. Geselowitz, Clarendon Press, Oxford

Reek, E. J., Grimbergen, C. A. and VanOosterom, A. (1984) A low cost 64 channel microcomputer based data acquisition system for bedside registration of body surface maps, Draft copy obtained from A. Van Oosterom, Dept. of Medical Physics and Biophysics, University of Nijmegen, The Netherlands..

Roberts, D. E., Hersh, L. T. and Scher, A. M. (1979). "Influence of Cardiac Fiber Orientation on Wavefront Voltage, Conduction Velocity and Tissue Resistivity in the Dog," Circ Res Vol. 44(5), pp.701-712

Rudy, Y., Plonsey, R. and Liebman, J. (1979). "The effects of variations in conductivity and geometrical parameters on the electrocardiogram using an eccentric spheres model," Circ. Res. Vol. 44, pp.104-111

Rush, S., Abildskov, J. A. and McFee, R. (1963). "Resistivity of Body Tissues at Low Frequencies," Circ Res Vol. 12, pp.40-50

Rush, S. (1971). "An inhomogeneous anisotropic model of the human torso for electrocardiographic studies," Med. Biol. Eng. Vol. 9, pp.201-211

Schwan, H. P. and Kay, C. F. (1956). "Specific resistance of body tissues," Circ. Res. Vol. 4, p.664, (Cited in Rush et. al. 1963)

Selvester, R. H., Solomon, J. C. and Gillespie, T. L. (1968). "Digital computer model of a total body electrocardiographic surface map: An adult male-torso simulation with lungs,"

Circulation Vol. 38, pp.684-690

Smith, G. D., (1978). Numerical Solution of Partial Differential Equations: Finite Difference Methods, Oxford University Press, Oxford Applied Mathematics and Computing Science Series, Oxford

Spach, M. S. and Barr, R. C. (1975). "Ventricular intramural and epicardial potential distributions during ventricular activation and repolarisation in the intact dog," Circ. Res. Vol. 37, pp.243-257

Spach, M. S., Barr, R. C., Warren, R. B., Benson, D. W., Walston, A. and Edwards, S. B. (1979). "Isopotential Body Surface Mapping in Subjects of All Ages: Emphasis on Low Level Potentials with Analysis of the Method," Circulation Vol. 59(4), pp.805-821

Taccardi, B. (1963). "Distribution of heart potentials on the thoracic surface of normal human subjects," Circ. Res. Vol. 12, pp.341-352

Taccardi, B., DeAmbroggi, L. and Viganotti, C., (1976). "Body surface mapping of heart potentials," in The Theoretical Basis of Electrocardiology, ed. C. V. Nelson and D. B. Geselowitz, Clarendon Press, Oxford

Tikhonov, A. N. and Arsenin, V. Y., (1977). Solutions of Ill-Posed Problems, V. H. Winston and Sons, Washington

Uijen, G. J. H., Heringa, A. and VanOosterom, A. (1984). "Data reduction of body surface potential maps by means of orthogonal

expansions.," IEEE Trans. Bio-Med. Eng. Vol. BME-31(11), pp.706-714

VanOosterom, A. (1978). "Triangulating the Human Torso," The Computer J. Vol. 21, pp.253-258

VanOosterom, A. and Strackee, J. (1983). "The Solid Angle of a Plane Triangle," IEEE Trans. Bio-Med. Eng. Vol. BME-30(2), pp.125-126

Walker, S. J., Lavercombe, P. S., Loughhead, M. G. and Kilpatrick, D. (1983). "A Body Surface Mapping System with Immediate Interactive Data Processing," IEEE Computers in Cardiology, pp.305-308

Yajima, K., Kinoshita, S., Tanaka, H., Ihara, T. and Furukawa, T. (1983). "Body Surface Potential Mapping System equipped with a Microprocessor for the Dynamic Observation of Potential Patterns," Med. α Biol. Eng. α Comput. Vol. 21, pp.83-90

Yamada, K., Toyama, J., Sugenoya, J., Wada, M. and Sugiyama, S. (1978). "Body Surface Isopotential Maps: Clinical Application to the Diagnosis of Myocardial Infarction," Jap. Heart J. Vol. 19(1), pp.28-45

Yamashita, Y. (1981). "Inverse solution in Electrocardiography: Determining epicardial from body surface maps by using the finite element method," Japanese Circulation J. Vol. 45(11), pp.1312-1322

Yamashita, Y. (1982). "Theoretical Studies on the Inverse Problem in Electrocardiography and the Uniqueness of the Solution," IEEE Trans. Bio-Med. Eng. Vol. BME-29(11), pp.719-725

Zienkiewicz, O. C. and Morgan, K., (1983). Finite Elements and Approximation, John Wiley and Sons