City Research Online

City, University of London Institutional Repository

Citation: Fardipour, P. (1989). Computer modelling of the human cardiovascular system based on relational analysis. (Unpublished Doctoral thesis, City University)

This is the accepted version of the paper.

This version of the publication may differ from the final published version.

Permanent repository link: https://openaccess.city.ac.uk/id/eprint/34930/

Link to published version:

Copyright: City Research Online aims to make research outputs of City, University of London available to a wider audience. Copyright and Moral Rights remain with the author(s) and/or copyright holders. URLs from City Research Online may be freely distributed and linked to.

Reuse: Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

City Research Online: http://openaccess.city.ac.uk/ publications@city.ac.uk

THE CITY UNIVERSITY LIBRARY
Northampton Square London EC1V 0HB

COMPUTER MODELLING OF THE HUMAN CARDIOVASCULAR SYSTEM BASED ON RELATIONAL ANALYSIS

by

PARVIN FARDIPOUR

A thesis submitted for the
degree of Doctor of Philosophy
at City University

Department of Systems Science
City University
London

September 1989

TO

My parents,

for their encouragement and
financial support in the course
of this study.

CONTENTS

page

ACKNOWLEDGEMENTS

I am indebted to both my supervisors, Dr M S Leaning (University College, London) for his invaluable guidance and Professor E Carson (City University) for his excellent administrative work throughout the production of this thesis.

I offer my sincere thanks to [redacted] (University of Oslo) for beta testing the package 'PULSE'.

I am very grateful to [redacted] (Wyeth Research) for his invaluable help and support during the preparation of this thesis and proof-reading the thesis.

I also wish to thank the following individuals:

[redacted] (Tudu clinic) for his interesting discussions on liver and pericardium diseases.

[redacted] (Wyeth Research) for his discussion on data acquisition in real life.

Finally, I wish to express my gratitude to Wyeth Research for providing facilities during the preparation of this work.

ABSTRACT

This thesis describes the origins and developments of a novel design for the computer modelling of the human cardiovascular system. With this design, complex models of the cardiovascular system can be completely represented in a relational database. A simulation program then scans the database and produces a numerical and graphical simulation.

The validation of the approach is divided into two sections. The first is verification using a large complex 19-segment model developed by Beneken and De Wit (1967), which was then used by Pullen (1976) and Leaning et al. (1983). The model consists of circulatory dynamics together with its neural control. Various tests on the parameter values and overall results were checked against the results reported by Pullen (1976) and Leaning et al. (1983). Secondly, the approach has been validated, and the approach shown to be usable by an independent clinician in Norway who has built a small model using this novel design and checked the results. Finally the hepatic and splenic circulations were added to the above 19-segment model to form a 23-segment model, to illustrate the ease with which a model can be extended, thus further highlighting the validity of this novel design.

This new approach places emphasis on model building rather than programming skills or implementation. The advantages are that the generic equations are applicable to most cardiovascular models and therefore the parameter values, initial conditions and other quantitative data are explicit and thus could be changed in the relational database. Also the structure of model can be easily changed. The user does not need to change the simulation program which is universal.

The approach focuses on the particular area of cardiovascular physiology, but it has applicability to complex dynamics in general.

CHAPTER 1

INTRODUCTION

The cardiovascular system is the main transport system of the body. Waste products are transported from the tissues, while nutrients are transported to the tissues by the bloodstream. Neural and hormonal control mechanisms enable the system to adapt to the changing environment.

Modelling may be defined as the simplified representation or description of a system or complex entity, for example designed to facilitate calculations and predictions. Mathematical modelling and the computer simulation of the cardiovascular system are now widely used to provide a clearer understanding of the system, to reveal areas of uncertainty, to estimate and predict in a clinical environment, to test hypotheses and finally to explain phenomena and processes in physiology and medicine.

Until now, quantitative computer implementations of mathematical models have been rigidly bounded; meaning that model structure, parameter values, initial conditions and other model information are implicit (not very clear at the first attempt) and so could only have been altered by users with an in-depth knowledge of programming. In addition, for each model that was required by the user, a separate program was necessary. Furthermore, validation of these models was very difficult due to the complexity of the program structure. The lack of adequate documentation in computer models of physiological systems has contributed in no small way to the tedium and extra work necessary in working with them.

There have been various problems associated with cardiovascular modelling - such as models being more complex than is required, the model structure being usually implicit in the computer code and perhaps most importantly, the general difficulty in changing the underlying model, owing to the programming effort required. For a non-programmer to build a model, not only was an understanding of the methodology of model building required, but also a familiarity with computer programming. This made it very difficult for the clinician working in biomedical researchers (biologist to clinicians), to exploit the potential of mathematical or computer cardiovascular modelling in their work.

The prime aim of the work presented here was to overcome problems mentioned earlier and to build a system where non-programmers can build, simulate, edit and test models easily.

For such a system to be built, a methodology of cardiovascular modelling had to be generalized so that any clinician or researcher would be able to build the desired model. To produce a generalized cardiovascular modelling system, the generic equations for such a system had to be derived. This needs clinical knowledge as well as mathematical and modelling experience. The generic equations derived for the cardiovascular system are based on a lumped parameter approach which considers blood flow to be linear. In the lumped parameter approach, the system is modelled as a collection of linked segments, each segment functioning effectively as a reservoir. This implies that distributed effects are lumped together and treated as a homogeneous entity. These models are characterized by ordinary differential equations. In the distributed parameter approach, no assumptions of homogeneity are made and thus the non-uniform blood flow distribution is taken into account. This leads to models characterized by partial differential equations (Carson et al., 1983).

Once the generic equations were derived, the information on the paper had to be transferred to the computer. In the new approach described in this thesis, the simulation engine (where the simulation is carried out) is based on these generic equations and is universal, that is it does not have to be repeated for each cardiovascular model. The model is represented in a relational schema where the simulation engine scans the database and produces a numerical and graphical simulation. This approach overcomes the problems of earlier approaches as well as making the simulation faster and easier to handle. Some advantages are summarized below.

(i) Explicit representation of the model structure.
(ii) Parameter values, initial conditions and other quantitative data are explicit.
(iii) Model structure and parameters are easily changeable.
(iv) It is possible to query the representation and generate reports, including the listings of model equations.

(v) The quantitative time response may be simulated, when the model is complete.

(vi) The simulation details can also be explicit.

To transfer the data into the computer, a novel approach was designed containing two modules. A database - oriented architecture has been designed on the basis that all relevant information would be stored in a relational database system. This approach was adopted since relational databases technology is used widely as a means of accessing required data. Model simulation is performed by the main simulation engine written in Pascal. The link between these two modules is an application generator, which is used to transfer the required data across to the simulation engine.

In the subsequent chapters it will be shown that each of the research aims, proposed above, has been achieved and realized in a working computer package, capable of representing and simulating cardiovascular models, including those of great complexity.

Part of the work presented in the thesis was conducted in collaboration with a clinical research group, the Institute for Experimental Medical Research (Oslo, Norway). This is one of the few experimental groups which has adopted mathematical modelling as a serious investigative tool.

1.1 APPLICATIONS OF CARDIOVASCULAR MODELS

Applications of cardiovascular models lie predominantly in areas of research and clinical medicine. In research, the purpose is typically to investigate the response of the intact circulation to disturbances, inputs and changes (e.g. drug-induced changes in heart function), which is intuitively difficult. The effects of CNS control and cardiovascular drugs may also be included. In clinical medicine, circulatory problems in heart and liver disease can be investigated with models.

The new approach is shown to be useful, because of the flexibility and ease of designing models of appropriate complexity for particular problems.

There is a tendency for many cardiovascular models to be "comprehensive" (Beneken and De Wit, 1967; Pullen, 1976; Leaning et al., 1983), so that a wide range of situations could be modelled. A disadvantage is that validating the model becomes very difficult. One reason for having comprehensive models is the programming time required to produce an implementation for a new model. This problem is overcome in the new system since the model structure is explicit and can be very easily changed. Simulation languages have a similar advantage, but because they are based on mathematical equations, their conceptual basis is not so apparent as in the database approach. The database provides an interface between the user and the simulation. In this way the user does not need to manipulate the equations directly, but rather provides the information which those equations require in a physiologically explicit manner. Using this database approach instead of having one large cardiovascular model, it is likely that a user will have a set of smaller ones adapted to particular problems, e.g. for examining the effects of inotropic changes (such as reducing the stroke volume and increasing the end-diastolic pressure as in Piene model, Piene et al., 1983), the role of the splanchnic circulation in circulatory regulation, etc.

Finally, in 1979 there was a debate between two scientists Guyton (1979) and Yates (1979) who each favoured a different type of modelling. Guyton wrote a commentary about the usefulness of the large scale model while Yates favoured simpler models. The approach adopted here is intended to overcome such disagreements, since both a simple model and a complex one can be built by 'PULSE'.

The thesis shows how the design and implementation are achieved. Finally the package 'PULSE' is exposed to both alpha and beta site testing. The novel approach has been developed for circulatory models with neural control, and a full functioning system which has been tested thoroughly should prove to be of wider use.

1.2 ORGANIZATION OF THE THESIS

Chapter 2 reviews some key models developed for use in cardiovascular research. The chapter begins by a brief historical overview, and then focuses in more depth on two well-known pulsatile models. From the review, the limitations of this kind of modelling is demonstrated and the need for a flexible modelling approach will also be shown. The chapter also reviews some simulation programming languages and describes a computer implementation of cardiovascular models.

Chapter 3 is divided into two parts. The first deals with the modelling of the circulatory fluid mechanics, the second with the modelling of the neural control. In the first part the general conceptual and structural foundations of pulsatile cardiovascular models are examined. These are divided into three sections: the elastic compartment representations of the cardiovascular system, the circulatory network, and the circulatory regions. In the second part, the derivation of generic equations for the neural control, based on empirical studies of Katona (1967) and the black-box approach of Hyndman (1970), will be given.

Chapter 4 describes a novel approach to computer modelling, based on the relational database (Date, 1986). The chapter begins by outlining the design of a database - oriented architecture, followed by the development of the relational schema for cardiovascular models. The chapter concludes with a description of the general algorithm for simulating cardiovascular models.

Chapter 5 begins by presenting the approach to software development. This is followed by a discussion of the choice of software and hardware for implementation. The development of 'PULSE' software is then described. Finally the design of the database manipulation programs and the simulation engine is presented.

Chapter 6 discusses the validity of this novel modelling approach. This is divided into two parts, concerning first the internal validity (verification) and then the external validity. The most important aspects of internal validity are the consistency and completeness of the approach. The chapter achieves this by verifying the internal validity of a previous well-validated 19-segment model (Pullen, 1976; Leaning, 1980; Al-Dahan, 1984). External validity, which is based on the modelling objective of the user as well as the potential of the approach, was achieved in three parts, verification of the package and the approach by beta site testing, testing the usability of the package and the approach, as well as the technical validity of the approach. Two models were built to illustrate these criteria. One was an extension of the above mentioned 19-segment model (a 23-segment model), and the other was the 4-segment model built at the beta test site.

Chapter 7 contains the results of evaluation studies carried out by a Norwegian Clinical Research Group to whom the software package 'PULSE' was sent for testing and critical appraisal.

Chapter 8 makes a number of various suggestions for future work.

Chapter 9 summarizes the main conclusions of the work described in the thesis.

Appendices contain technical details, programs, listings, etc which have been omitted from the main text for simplicity. **Appendix 1** gives a listing of dictionary entries (`entity, attribute, ent_att`) used in the novel modelling approach. These entries are described in Chapter 3. **Appendix 2** contains a list of the circulatory and neural control generic equations (`result1.equ, result2.equ`) driven by a Pascal program. These equations are explained in Chapter 5. **Appendix 3** gives a complete listing of the simulation engine Pascal program. (`harvey.pas`). This program is described in detail in Chapters 4 and 5. **Appendix 4** lists the database files for the 19-segment model explained above. **Appendix 5** lists the database files for drug input for the 19-segment model explained above.

1.3 ADVICE TO THE READER

This section is prepared to guide different classes of users in reading this thesis. For clinicians and trained scientists, Chapters 2, 3, 6, 7 would be useful; for the mathematicians and engineers, Chapters 3, 4, 5 are interesting and for the computer modellers Chapters 2, 4, 5 are recommended.

CHAPTER 2

REVIEW OF SOME CARDIOVASCULAR MODELS AND APPROACHES TO COMPUTER SIMULATION

2.1 INTRODUCTION

Having examined the reasons for mathematical modelling of the cardiovascular system in Chapter 1, this chapter reviews some key cardiovascular models and approaches to computer simulation. The chapter has a brief historical overview in Section 2.2, and then focuses in Sections 2.3 and 2.4 on two well known pulsatile models: the model developed originally by Pullen and reported by Leaning et al. (1983a, b), which has 19 compartments and the 7 compartment model of Piene et al. (1983). A common feature of these two models is that they include CNS control. An examination will then be made as to which model is more useful, and in which aspect and area. Finally in Section 2.5, the chapter reviews some simulation programming languages followed by Section 2.6 which describes a computer implementation of cardiovascular models.

There have been various problems associated with cardiovascular modelling - such as being more complex than is required. The model structure is usually implicit in computer code leading to difficulty in identifying and changing the parameters. Perhaps most importantly, there are the general difficulty in changing the underlying model, owing to the programming effort required.

The comprehensive need for cardiovascular models lies in being able to model a wide range of situations. The disadvantage is that the validation of these models becomes very difficult. The aim of this chapter is to review some of these models and to emphasize the need for better modelling approaches.

2.2 BRIEF HISTORICAL BACKGROUND

Numerous attempts have been made to model the cardiovascular system in part and as a whole. In the seventeenth century William Harvey

(1628) proved that the heart, and not the liver, was the centre of the vascular system and that it propels the blood around a closed circuit by its rhythmical contractions, as would the repeated strokes of a man-made pump. The model was simple, though implicit, compartmental and obeyed the law of mass balance. Some properties of heart such as natural rhythmicity of the muscle (given a supply of oxygenated blood, the heart muscle will spontaneously and regularly beat, pumping blood) and the famous Starling law are very important in understanding Harvey's model. The Starling (1866-1927) law states that, within physiological limits, the external stroke work done by the heart is proportional to end-diastolic ventricular volume, i.e. the heart will automatically balance cardiac output with venous return, thus demonstrating that the heart and circulatory system is a self-innervating auto-regulative system.

In this section a brief historical review is given. The review is not exhaustive, but rather is intended to illustrate some of the more important developments which have been made in relation to physical aspects of the heart and circulation, in particular those dealing with pulsatile behaviour.

2.2.1 Models of the Auto-controlled Cardiovascular System

These models are based on the observations of the earlier models that the heart and circulation form a stable system independent of any other form of control (e.g. neural or hormonal). They therefore describe an instantaneous steady state behaviour of the cardiovascular system. Although the auto-controlled cardiovascular system can attain a steady state equilibrium in a constant environment, the lack of neural or hormonal control in the classical reflex causes it to be usually known as the 'uncontrolled' cardiovascular system.

One of the earliest models was that formulated by Van Harreveld et al. (1949) in which their model used a resistive-capacitive electrical analogue. The mean flow and mean pressures at the outlet and inlet of the heart, together with their changes when the system parameters varied, were represented. The resistance to blood flow is described by the resistance in the electrical circuit, the capacity for blood storage is described by the charge capacity, and the blood pressure is described by voltage. The model, like all the other work carried out in this period, represented the steady state behaviour of the cardiovascular system without taking into account the effect of neural or hormonal control.

Guyton (1955), used a graphical method for the determination of mean cardiac output, venous return and arterial pressure for both normal and abnormal conditions. In this method, curves of venous return and cardiac output are drawn against a single independent variable, right atrial pressure (cardiac response curves and venous return curves). The interaction of the venous return and cardiac output curves determines the operating point (when venous return and cardiac output are equal), thereby satisfying Starling's law. Guyton's method describes the steady state conditions that the system might achieve given limited environmental disturbances.

The next level of complexity in the development of cardiovascular models in terms of compartmental structure is Grodins's (1959) resistive-capacitive model. Grodins's model is based on Starling's law of the heart in which cardiac activity is represented by a linear relationship between end-diastolic ventricular volume and stroke work (where the proportionality constant is called the 'strength' of the ventricle). This model consists of 23 simultaneous equations which must be solved in order to determine the equilibrium values.

Noordergraaf et al. (1963) used an electrical analogue for the study of human haemodynamics in which he developed a mathematical model of the systemic arterial tree (115 segments) to find a quantitative interpretation of the amount of blood passing through the heart in a specific time by recording the recoil movements of the body that result from contraction of heart muscle in ejecting blood from the ventricles. The assumption he made was that the blood flow through the left ventricle was linear.

Beneken (1965) simulated a compartmental model to describe the haemodynamic behaviour of the central part of the human blood circulation. The description of a ventricle consists of two parts: one concerning the specific shape and one concerning the properties of the wall material i.e. heart muscle. A cylindrical shape has been assumed for the left ventricle cross-section and a constant length. The right ventricular cavity is assumed to be bounded by part of the outer surface of the left ventricle together with a spherical free wall bent around a part of the left ventricle. A time-varying relation between force and muscle length (compliance) of both ventricular segments is also included in this model. The model was simulated on an electronic analogue computer and was uncontrolled. In the analogue computer, which solves 57 equations, all the variables such as pressures, flows and volumes appear as voltages. Beneken investigated the effect of perturbing the parameters on the behaviour of the system as a whole.

2.2.2 Models of the Controlled Cardiovascular System

Whilst the cardiovascular system is stable in an unchanging environment, its stability from moment to moment in a changing environment depends upon a rapid neural control of the heart and blood vessels by the central nervous system.

There are two kinds of controlled haemodynamic model: (1) pulsatile models with CNS control, suitable for investigating short-term, typically haemodynamic effects (less than 5 minutes); and (2) non-pulsatile models for studying long term effects which include hormonal control.

2.2.2.1 Pulsatile haemodynamic models

Beneken and De Wit (1967) constructed a 19-compartment model of the cardiovascular system (Fig. 2.1). It consists of 4 heart chambers, a 7 compartment arterial tree, a 6 compartment venous tree and a 2 compartment pulmonary tree. All 4 heart compartments have time-varying elastances. The neural control of heart rate, myocardial contractility, peripheral resistance and venous tone is also included. The models of the two baroreceptors (carotid sinus and aortic arch) which send pressure information to the central nervous system are based on Katona's empirical studies on the dog (1967). In these, the relationship between arterial pressure and heart rate was examined, the net effect at the brain of all the baroreceptor impulses being considered as a single input signal to the model. The responses of the model were shown to agree with a range of circulatory responses over short periods of time. The model responds reasonably well to Valsalva manoeuvres and blood volume reduction tests. Details of the subsystems and their respective orders are given in Table 2.1. The complete equation set (of order

36) is solved by analogue computer to give pressures and flows with acceptable time-courses.

Beneken and Rideout (1968) demonstrated that the computer models of the circulation, based on lumped circuits approximations may be used for simulation studies of its pulsatile pressure, flow and volume relationship. A second model, coupled to such a basic circulation model, may be devised to simulate the flow of substances carried by the blood. Such a slave, or dependent, model is based on the notion that transport flow is proportional to concentration in the slave circuit multiplied by flow in the main circuit. The combined, or multiple, model may be used to control studies related to the transport of CO_2 or O_2.

Pullen (1976) based his model structure on the circulatory fluid mechanics model of Beneken and De Wit (1967), with the baroreceptor and neural control models of Katona (1967) and Hyndman (1970) and the "multiple modelling" technique of Beneken and Rideout (1968) to represent the transport of chemical substances in the blood stream. Pullen introduced an algebraic method for modelling the local effects of cardiovascular active drugs. This model is described in more detail in Section 2.3.

Piene (1983) developed a 7 segment model based on Beneken (1964) model. The model consists of 4 active heart chambers, 1 pulmonary bed, 1 systemic arterial bed and 1 venous bed. The heart chambers are contained in an elastic pericardium whose P-V (pressure - volume) relationship was linear. The values which are chosen for the model parameters resemble the dog circulatory system and the simulation was written in FORTRAN. The model mathematical representations are the same as for the 19-segment model (see Chapter 3 for derivation of generic equations). This model is described in more detail in Section 2.4.

2.2.2.2 Non-pulsatile models

The models of Section 2.2.2.1 represent, with varying degrees of validity, the major aspects of short-term cardiovascular dynamics and CNS control in response to a limited number of environmental changes.

In real life, the story is not so simple. The changes caused by the environment include fast-acting chemical effects, medium-term hormonal changes and longer-term fluid shifts and disease processes.

The non-pulsatile control model of the cardiovascular system developed by Boyers et al (1972) was used to study the normal responses to change of posture, blood loss, transfusion and autonomic blockade. The model can simulate the steady state responses of the cardiovascular system to stresses ranging in duration from a few seconds to many hours, and can also be used to study the regulation of interstitial fluid and total blood volumes. The model consists of the heart, large arteries, peripheral circulation and the effects of the ganglionic blocking agent Arfonad on the circulatory response to a large transfusion of blood. The results from all the tests of the model agreed closely with the observed measurements of the human circulation.

Guyton et al. (1972) developed a non-pulsatile lumped parameter model of the uncontrolled circulation to which was added a large number of short-term and long-term control mechanisms. The analysis consisted of 354 blocks, each of which represents one or more mathematical equations describing some physiological facet of circulatory function. The analysis is a framework to show how the different regulatory mechanisms operate together in the overall system.

Guyton divided the analysis into 18 different major systems that enter into circulatory control. This led to an equation set of order 37 (Table 2.2) which may be solved by digital computer from which 5 systems will be mentioned here.

(1) Circulatory dynamics: This is divided into 5 volume segments (the aorta, the veins, the right atrium, the pulmonary arteries, and the combination of pulmonary veins and left atrium), 5 resistances to flow segments (muscle, non-muscle, non-renal vasculatures, venous resistance, resistance between the large veins and the right atrium) and other circulatory dynamics such as cardiac hypertrophy.

(2) Electrolytes and cell water: This calculates extracellular fluid volume and total body water; the accumulation of sodium in the extracellular fluid and its concentration; the extracellular fluid potassium quantity and concentration; the rate of potassium excretion by the kidney; the transfer of fluid through the cell membrane and the intracellular fluid.

(3) Angiotensin control: This calculates control of angiotensin formation as a function of renal blood flow. The effect of sodium concentration on angiotensin formation is also determined.

(4) Aldosterone control: This calculates the effects of arterial pressure, potassium to sodium ratio, and angiotensin on aldosterone secretion rate. The accumulation of aldosterone in the tissues and its concentration is also calculated.

(5) Antidiuretic hormone control (ADH): This calculates the total effect on antidiuretic hormone secretion of extracellular ion concentration, of right atrial pressure and of autonomic simulation. The rate of secretion of antidiuretic hormone and antidiuretic hormone multiplier expressed as the functional effect of antidiuretic hormone as a ratio to its normal effect are also calculated.

In this model some of the control and haemodynamic systems operate with very short time constants of the order of 0.005 min (e.g. in the haemodynamic circuit), while others operate with longer time constants; as high as 40 days (e.g. describing ventricular hypertropy). Results from simulated experiments agree closely with results from equivalent animal or human experiments. A problem with this type of model (containing many assumptions and empirical models) is that it frequently produces valid overall behaviour whilst containing anomalous invalid behaviours of its sub-models. For more details of validation of this type of model, refer to Leaning (1980), where tests of a similar model developed by Uttamsingh et al. (1985) are reported.

2.3 AN OVERVIEW OF THE LEANING ET AL. (1983A, B) MODEL

The model circulatory structure is based on the Beneken and De Wit (1967) model, consisting of 19 elastic compartments (Fig. 2.1). This is based on an extensive programme of work at City University into the development and validation of dynamic mathematical models of the human cardiovascular system (Pullen, 1976; Leaning, 1980; Al-Dahan, 1984). The overall aim is to demonstrate the potentialities and limitations of mathematical modelling of the human cardiovascular system. The specific objective of this work was to study haemodynamic and drug effects in the CNS controlled cardiovascular system over a time scale of about two minutes (Fig. 2.2). The mathematical representation of this model is the same as the generic mathematical representation derived in Chapter 3.

Leaning (1980) and Al-Dahan (1984) have carried out extensive tests of Pullen's model based on a comprehensive and systematic programme of validation for the circulatory, neural control and local pharmacodynamic sub-systems of the model. Particular emphasis was given to the validation aspects of modelling methodology, including: the concept of model validity; programmes for model validation; and validation problems for models of complex systems. Subsequently, the results of an extensive programme of validation of the cardiovascular model were presented and analysed (Leaning, 1980). The limitations and, in addition, the possibilities of model reduction and development were discussed (Leaning et al., 1983a, b). Many of the validation problems arose from the large size of the model; 61 state equations and 179 parameters. The parameters and initial state values were obtained by wide literature research, and apply to a "normal, adult, conscious male". The model reproduces the haemodynamic changes expected in a variety of tests, such as, equilibrium conditions, passive tilting, blood volume changes and the Valsalva manoeuvre.

The problem of validation and identification arises in large models because of the large number of parameters which are not easy to identify. Nevertheless they proved to be valuable in designing and interpreting experiments, and generally in understanding cardiovascular dynamics and control (Huikeshoven et al., 1985). Chapter 6 implements this model and presents some key features together with verification and validation tests.

2.4 AN OVERVIEW OF THE PIENE ET AL. (1983) MODEL

The model structure is based on the Beneken (1964) model (Fig. 2.3). It consists of four active heart chambers, right and left ventricles (rv and lv) and right and left atria (ra and la). The rv was connected to the la by a lumped elastic compartmental model of the pulmonary circulation. The lv was connected to the venous system through a similar representation of the systemic arteries, and the venous bed was represented by a large compliance and a small inflow resistance to the right atrium. The four heart chambers were contained within an elastic pericardium.

The aim of this work was to examine the potency of a simple model of the circulatory system for the interpretation of experimental data. Piene showed that alterations of myocardial function of a compartment of the intact heart will affect the other heart compartments. The pressure-volume relationship of a specific heart compartment is described by the myocardial stiffness (or elastance), plus the effects of pericardial constraint. Piene also showed that, owing to the fact that the heart chambers are arranged in series, the end-diastolic dimensions have to adjust until the outputs on both sides of the heart are equal.

It is easy to misinterpret the observations of cardiac function in patients or animals, owing to the effects of ventricular interaction. The Piene model demonstrates how the analysis of data from experiments or clinical measurements may be aided by predictions from a simple cardiovascular model. Such models may also indicate which variables should be measured in order that an experiment can give the required information.

Not only does the model not show the abdominal, leg and head circulation (since all the systemic arteries are lumped together), but also the effects of drugs are not included. Although the model does not include the above features, it has the advantage of being easier to identify than the 19-segment model. The mathematical representation for this model is the same as the generic mathematical representation derived in Chapter 3.

2.5 USE OF GENERAL PURPOSE LANGUAGES FOR COMPUTER SIMULATION

In this section some programming languages will be reviewed with their advantages and disadvantages in relation to cardiovascular modelling. The next section will discuss the reasons for selecting Pascal as a simulation language.

Simulation can be considered a representation of a changing, or dynamic, system developed to simplify manipulation and study while using a computer for computations, comparisons, and analyses. A model is a representation of a system in which the process or interactions bear a close resemblance or relationship to those of the specific system being simulated or studied. Although almost any type of programming languages can be used to implement a simulation model, various specific languages have been developed for simulation application. Some of these languages will be discussed in part (4) of this section.

High-level programming languages can be divided into five categories; (1) fundamental algorithmic and procedural languages, (2) variants for interactive use, (3) non-numerical languages, (4) simulation languages and (5) process and numerical-control languages. (The following reviews are extracted mainly from Wexelblat, 1981 and Horowitz, 1984.)

(1) Fundamental algorithmic and procedural languages such as FORTRAN and Pascal:

FORTRAN, which stands for **FOR**mula **TRAN**slation was developed by IBM and came into general use in 1956. Its designers' main goals were to provide an easier way of writing scientific and engineering programs, to simplify the processing of large quantities of numerical data, and to produce efficient object codes. Its <u>strengths</u> are that almost any type of mathematical or business problem can be solved with FORTRAN. While only a few data structures are provided, they include the data structures most commonly used for scientific and engineering problem solving. Because FORTRAN is made up of a few fairly Englishlike commands, it is relatively easy to learn. Since FORTRAN has been standardized, it is possible to develop quite portable programs by limiting oneself to these official standards. Its <u>weaknesses</u> are that while FORTRAN can be used for almost any type of information processing task, it was developed before standard programming, and has been brought up to date only by lots of nonstandard extensions. Early versions of the language did not provide very sophisticated control structures. However, FORTRAN 77, does provide structural programming logical flow patterns. Finally, even though FORTRAN is easy to learn, its syntax rules are very strict. It is easy to make errors when keying in programs or choosing names for a program's variables that cause the program not to execute or worse, to result in execution errors. The variants of FORTRAN available on the market at the moment have overcome most of these weaknesses and the lastest version of FORTRAN (FORTRAN 8XX, not on the market yet) is supposed to solve all the earlier weakenesses of FORTRAN.

Pascal, named after the French inventor of the mechanical calculator, was developed by Niklaus Wirth, a Swiss computer science professor. It was made available for general use in 1971. Professor Wirth's main design goal was to develop a language for teaching structured programming concepts. In addition, he wanted a programming language that would provide a powerful set of information processing capabilities but still be easy to learn. Pascal's greatest <u>strength</u> is the sophistication of its control structures and data structures. Programmers are encouraged to follow the structured programming concepts through the language's command and syntax. As a result, Pascal code usually possesses a very clear, logical structure. In addition to providing many standard data structures, Pascal allows programmers to define new data structures to be used in their programs. Also, like BASIC, Pascal uses relatively few commands, which makes it possible to write simple programs with very little training. Finally, Pascal's popularity as a teaching language has made it widely available. While Pascal was designed to serve as a general-purpose programming language, its <u>weaknesses</u> lie in relatively weak input, output, and file-handling capabilities (actually Pascal does not have any file i/o defined, and is almost entirely implementation dependent) which have limited its use for business computing (Turbo Pascal which was used here, explained in Chapter 5, did cater for file handling).

FORTRAN, used for conventional simulation, is not an ideal choice as a single software environment because of its limitation in terms of lack of flexible data structures and the slow compilation time of PC compilers. Pascal on the other hand places emphasis on model-building, rather than knowledge representation such as symbolic languages which will be explained later in this chapter). Pascal has a hierarchical record type structure which would be advantageous to our purpose (this is explained in more detail in Chapter 5).

(2) Interactive languages such as BASIC:

BASIC, which stands for **B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode, was developed by John Kemeny and Thomas Kurtz of Dartmouth College and made available for general use in 1964. The design goals emphasized by Kemeny and Kurtz reflected their desire to develop an interactive programming language that would be easy for students to learn and to use. Its <u>strengths</u> are to be able to handle just about any processing task but excelling at few. It can be learnt and used easily. It has only a few commands, the purposes of which are self-evident, and it handles many programming details for the programmer. Its <u>weaknesses</u> are its limited control and data structures. It has a limited set of syntax rules for naming variables which can result in a program code that is extremely hard to understand. Also the official ANSI standard for BASIC is so limited that almost every version of the language extends the standard. As a result, BASIC programs are not that portable, despite the many computer systems that have BASIC compilers and interpreters. This weaknesses are being overcome in many newer versions of the language.

The limitations of BASIC will not allow us to choose it as a simulation language.

(3) Non-numerical language such as PROLOG:

PROLOG, whose name derives from **PRO**gramming in **LOG**ic, is gaining popularity in many areas of applications. With PROLOG the emphasis is on description rather than on action, more on what is to be done than on how it is to be done. This style of computing is called pattern-directed rule based programming and is useful in several applications of artificial intelligence (<u>strength</u>). Its <u>weakness</u> lies in not being able to easily handle calculations.

PROLOG is not recommended for our purpose here since these symbolic languages are not recommended languages for arithmetic problems because they are too slow at performing numerical calculation.

(4) Simulation languages:

Continuous and discrete systems are modelled using disparate methods (Spriet and Vansteenkiste 1982). Continuous systems are usually described by systems of differential equations. The language equipped to simulate them must come armed with some means of describing continuous state transitions and tools for solving differential equations (DYNAMO (Pugh, 1963) and CSMP (IBM, 1967)).

In discrete systems, the state changes occur in single, discontinuous steps at discrete times. Languages designed to simulate them must offer features that change the model's state at the appropriate moments. (e.g. GPSS, GASP, SIMSCRIPT, SIMULA). There are those like GPSS that model transaction-oriented systems and those like GASP that model event-oriented systems.

There are many simulation languages on the market at the moment and hence it is difficult to provide a comprehensive review. Of the discrete systems simulation languages only SIMSCRIPT is reviewed here since the work represented in this thesis deals with continuous system.

SIMSCRIPT is a typical, proprietary general-purpose simulation language system for a discrete system. It is based on the notion that the state of a system is definable and can be described in terms of entities (i.e. the specific objects or things of which a system is composed) and attributes (i.e. those properties that are associated with the entities as well), and as sets (i.e. groups of entities). All entities must be specific and explicit, with a complete list of their attributes and possible set memberships; this is the major prerequisite in the development of a simulation model.

From the continuous system the following simulation languages are mentioned here.

DYNAMO is a compiler for translating and executing 'system dynamic' simulation for a continuous system.

CSSL is a 'Continuous System Simulation Language' for solving ordinary differential equations. It is a non-procedural language with a NOSORT option which translates simulation definitions into FORTRAN. It is powerful and easy to use. It has an extensive MACRO definition capability. This provides the sophisticated user with many options for creating and extending the existing language features. Lack of a file management system limits its ability to handle large data sets.

ISIM is an 'Interactive Software SIMulation system' for solving ordinary differential equations. It has its own built-in numerical integration system and the user has to input the differential equations. A small program such as:

```
1 SIM:RESET:INTERACT:GOTO 1
:
DYNAMIC
X'' = -K*X' - X + 1
PLOT T,X,0,TFIN,0,1.6
PREPARE T,X,X'
VAL K = 0.2000
VAL CINT = 0.5000
```

will give the curve depicted in Fig. 2.4. Since ISIM is a FORTRAN-based simulation language the window facility is rather poor. The graphical simulation of 2 or more variables simultaneously in different windows is not possible.

Most cardiovascular models have been implemented as numerical simulations, usually in FORTRAN (including those in Section 2.3 and Section 2.4). Some researchers have used simulation languages (e.g. CSMP, CSSL, DYNAMO, and GASP) where the mathematical equations are entered directly (discussed above). Although preferable in many ways to FORTRAN and other languages, such as allowing programs to be written quickly and efficiently without the user being expert in computing topics, simulation languages do not allow models to be built in conceptual terms of "arteries", "veins", "atria", etc. which is the goal of the work presented here. Limitations of languages like FORTRAN were discussed above.

(5) Process and numerical-control languages:

These languages have been developed to facilitate the use of computers controlling industrial processes and machine tools. These vary widely, because each application can be very different. A detailed review is beyond the scope of this thesis.

2.6 COMPUTER IMPLEMENTATIONS OF CARDIOVASCULAR MODELS

Pascal was chosen as a simulation language in this thesis. The reason lies behind the record-type structure of Pascal, which will fit into dBASEIII+ entities perfectly, and also its fast compiler (Turbo Pascal compiler explained in Chapter 5) on a PC. The aim was to choose a simulation language which will place emphasis on model building and the disadvantages of the languages discussed above made the choice Pascal. Chapter 5 will give some more information about the simulation language used here.

2.7 CONCLUSION

A brief review of some of the more important cardiovascular system models has been presented. The chapter focused on two models, describing their structure and achievements.

From the review of these two models the key factors to successful cardiovascular modelling are as follows.

(1) They should be no more complex than the problem requires.

(2) Assumptions on which the model is based and the range of its validity should be made explicit.

(3) Parameter values should be easily changeable.

Taking these features into account, the chapter reviewed some cardiovascular models together with their limitations and the need for better modelling approach has been emphasized. Consequently the architecture described in Chapter 4 is aimed at producing a better approach to cardiovascular modelling.

The chapter also reviews some programming languages and concludes in choosing the Pascal language for implementation of this cardiovascular model.

In Chapter 3, the basic elements of circulatory and neural control models are described in detail. These elements are the conceptual and structural foundations of pulsatile cardiovascular modelling, also the generic mathematical equations will be derived in this chapter.

SUBSYSTEM	ORDER
19-SEGMENT CIRCULATORY FLUID MECHANICS MODEL	29
HEART RATE CONTROL	3
INTERVAL-STRENGTH RELATION OF HEART	1
PERIPHERAL RESISTANCE CONTROL	1
CAPILLARY PRESSURE AND BLOOD VOLUME CONTROL	1
INFLUENCE OF CORONARY FLOW ON HEART PERFORMANCE	1
TOTAL	36

Table 2.1 **Subsystems in the Beneken & De Wit Model (1967)**

SUBSYSTEM	ORDER
CIRCULATORY DYNAMICS	5
VASCULAR STRESS RELAXATION	1
CAPILLARY MEMEBRANE DYNAMICS	2
TISSUE FLUIDS, PRESSURES AND GEL	4
ELECTROLYTES AND CELL WATER	4
PULMONARY DYNAMICS AND FLUIDS	2
ANGIOTENSIN CONTROL	1
ALDOSTERONE CONTROL	1
A. D. H. CONTROL	2
THIRST AND DRINKING	0
KIDNEY DYNAMICS AND EXCRETION	0
MUSCLE BLOOD FLOW CONTROL AND PO_2	3
NON-MUSCLE OXYGEN DELIVERY	2
NON-MUSCLE, NON-RENAL LOCAL BLOOD FLOW CONTROL	3
AUTONOMIC CONTROL	2
HEART RATE AND STROKE VOLUME	0
RED CELLS AND VISCOSITY	2
HEART HYPERTROPHY OR DETERIORATION	3
TOTAL	37

Table 2.2 **Subsystems in the Guyton et al Model (1972)**

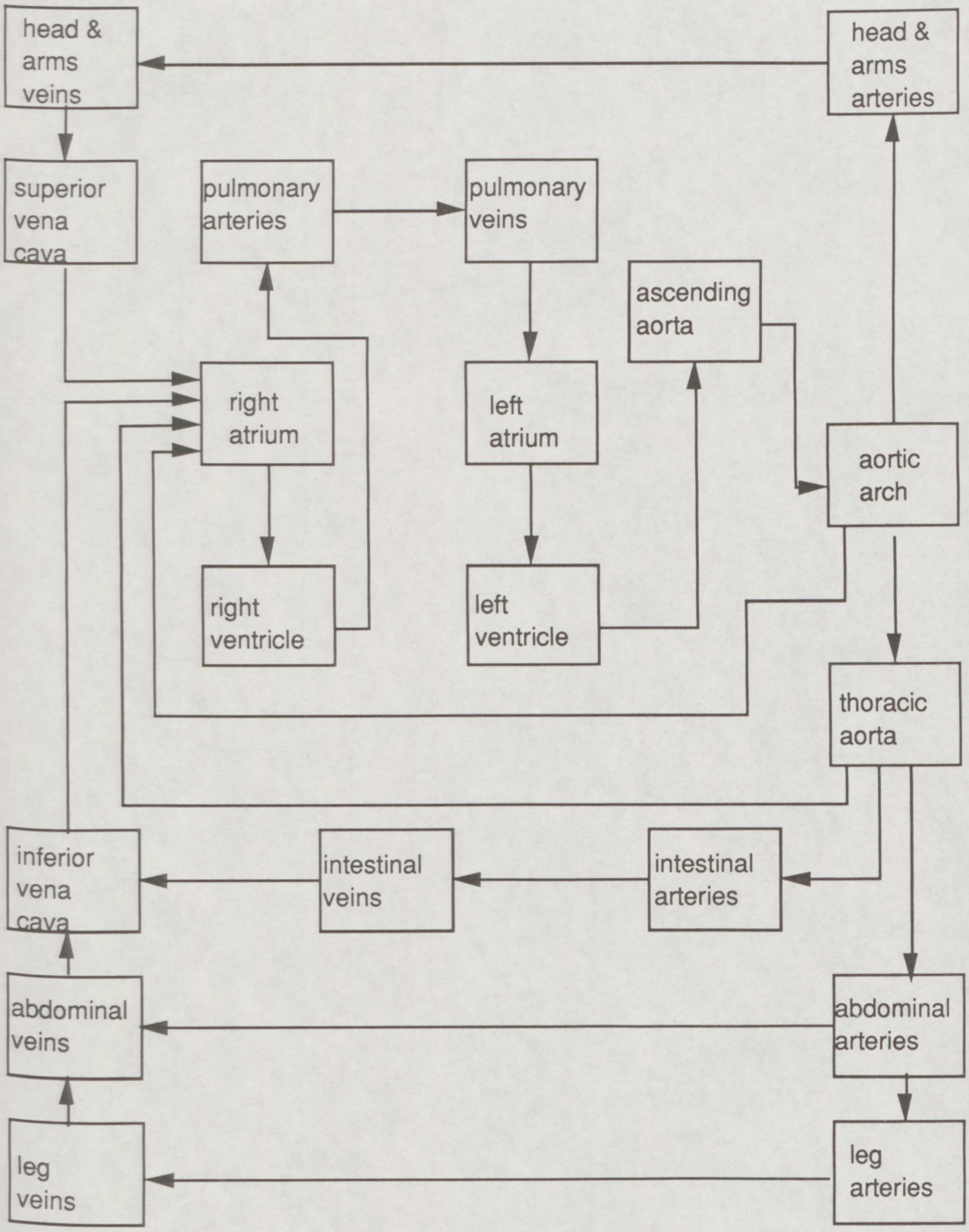


Figure 2.1 **Circulatory network of the 19 compartment cardiovascular model (Beneken et al., 1967; Pullen, 1976; Leaning et al, 1983)**

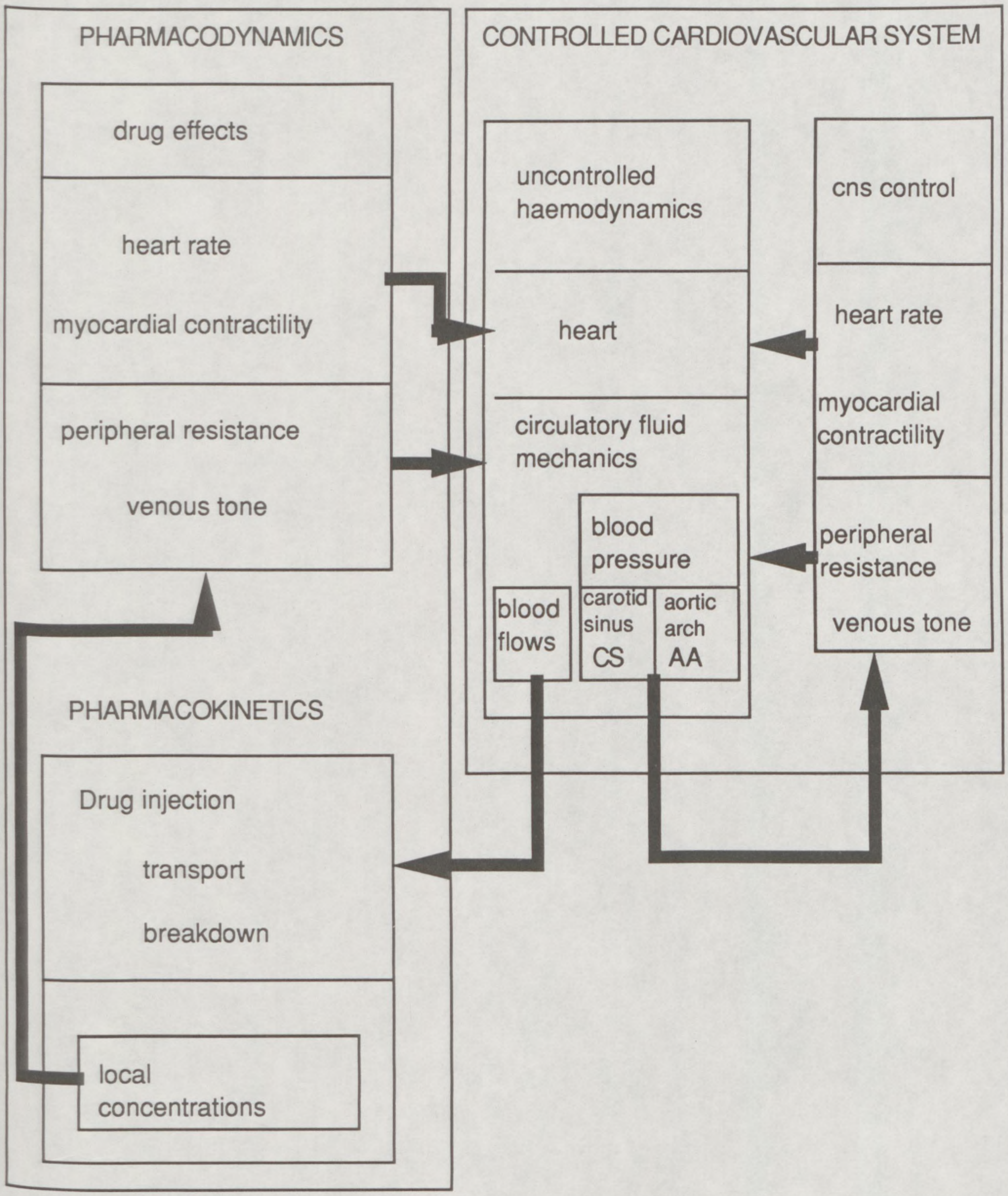


Figure 2.2 **Sub-system and main intractions (Leaning et al., 1983a)**

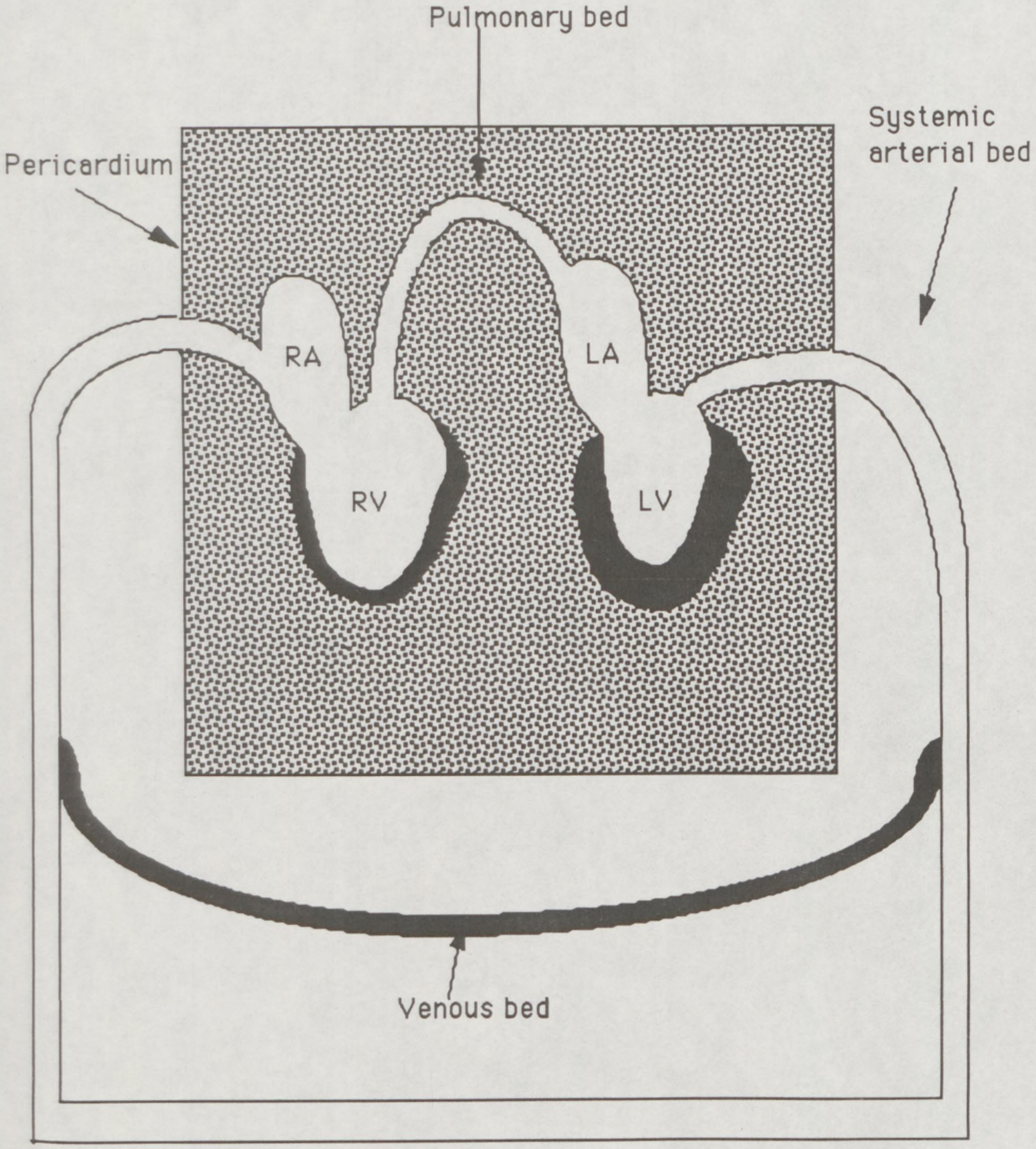
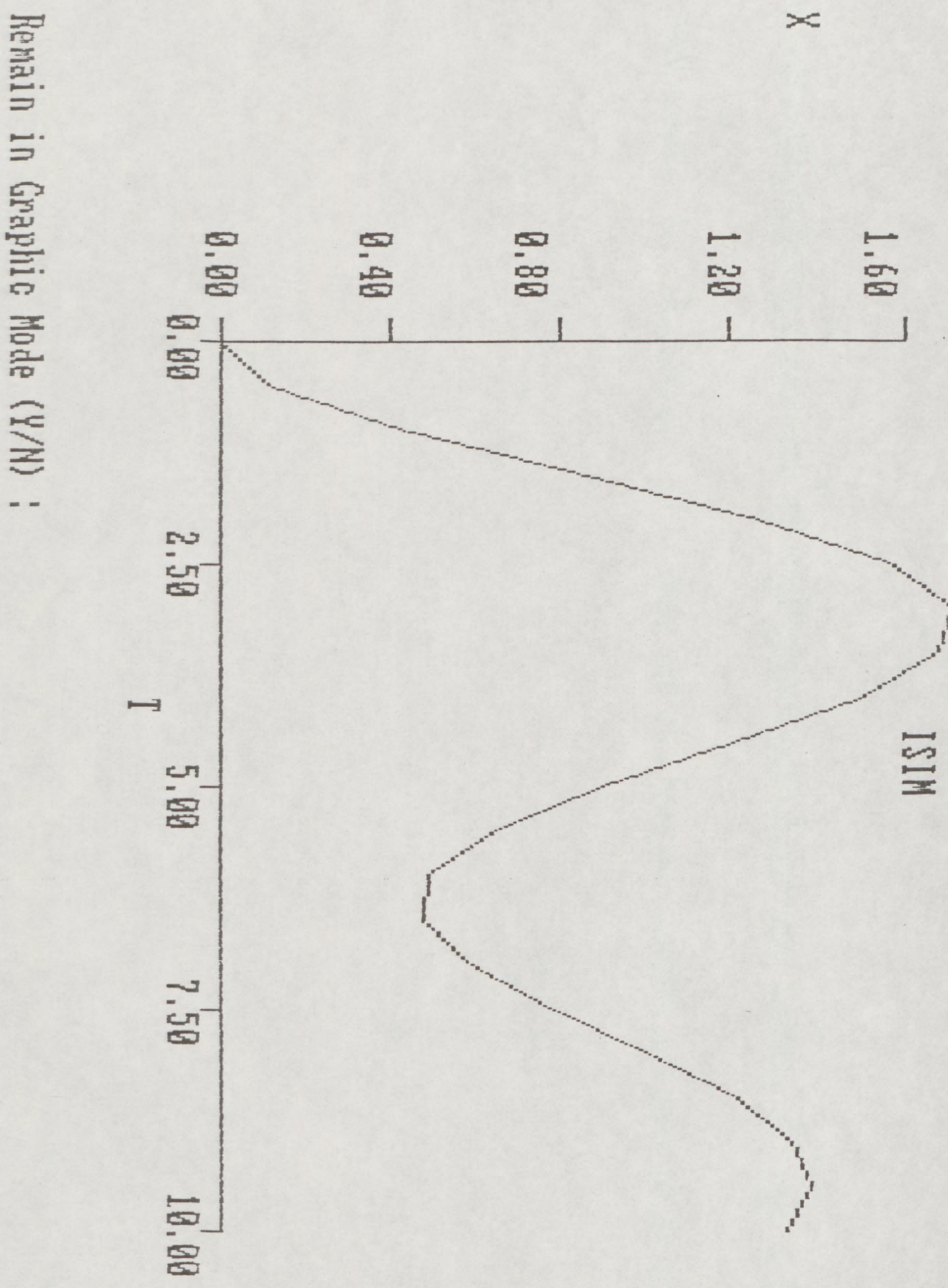


Figure 2.3 **Circulatory network of the Piene model (1983)**



```
Remain in Graphic Mode (Y/N) :
```

Figure 2.4 **ISIM program**

CHAPTER 3

CONCEPTUAL AND MATHEMATICAL FOUNDATIONS OF PULSATILE CARDIOVASCULAR MODELLING

3.1 INTRODUCTION

The last chapter reviewed a number of models of the cardiovascular system, and concluded by presenting the case for a flexible modelling system. Also it concluded with a brief review of computer languages, together with some simulation languages. This chapter will be divided into two parts, the first dealing with the modelling of the circulatory fluid mechanics and the second with the modelling of the neural control.

In the first part of the chapter some physiological background will be given followed by the conceptual and structural foundations of pulsatile cardiovascular modelling, together with their mathematical representations. These elements will form the basis of the modelling system (and database-oriented architecture) developed later.

In conceptualisation, the circulation is decomposed into a set of elastic compartments comprising a circulatory network and circulatory regions. There are distinct elastic compartments for the arteries, veins, atria and ventricles. The circulatory network is defined by the different connection types between arteries, veins, atria and ventricles. The circulatory region refers to anatomically distinct regions of the circulation, such as the pulmonary circulation.

Finally a complete and general mathematical representation for the volumes and pressures in these compartments (arteries, veins, atria, ventricles) and the flows between them will be illustrated.

In the second part of this chapter some physiological background to the central nervous control system will be given followed by derivation of generic equations for baroreceptors and for a 'bang-bang' (the terminology used by Hyndman 1970 to define the 'on-off' situation of the system) representation of 4 controllers of the neural control which will be the basis of the modelling system (and database-oriented architecture) developed later.

Finally the effect of neural control on the circulation will be discussed.

3.2 BASIC ELEMENTS OF CIRCULATORY MODELLING

3.2.1 Background Physiology

In general, the physiological literature defines the human cardiovascular system as a closed tubular system in which blood, propelled by a muscular heart, flows through vessels to and from all parts of the body.

The primary function of the heart is to serve as a muscular pump propelling blood into and through vessels to and from all parts of the body. The arteries, which receive the blood at high pressure and velocity and conduct it throughout the body, are thick-walled vessels with elastic fibrous tissue and muscle cells. The arterial tree is a branching system of arteries, from which blood enters simple endothelial tubes (i.e. tubes formed of endothelial, or lining, cells) known as capillaries. These microscopically thin capillaries are permeable to vital cellular nutrients (especially oxygen) and waste products, and thereby distribute and receive nutrients and wastes. From the capillaries, the blood, depleted of oxygen and burdened with waste products, moving more slowly and under low pressure, enters small vessels called venules, which converge to form veins, ultimately guiding the blood on its way back to the heart.

3.2.2 The Pattern of the Circulation

Blood flows through a continuous network of blood vessels that forms a double circuit connecting (1) heart and lungs, and (2) heart and all other tissues. The left ventricle pumps blood into the systemic circulation, which brings oxygenated blood to all the different organs and tissues. Blood returns to the right atrium of the heart somewhat de-oxygenated but loaded with carbon dioxide wastes. It is pumped by the right ventricle into the pulmonary circulation. The pulmonary arteries carry blood to the lungs, where gases are exchanged. Then pulmonary veins return the blood, rich in oxygen once more, to the left atrium. Blood then passes into the left ventricle and is pumped into the systemic circulation again, repeating the double cycle (Fig. 3.1).

(i) **Pulmonary circulation**

De-oxygenated blood returning to the heart from the systemic circulation passes into the right atrium, and then into the right ventricle. From the right ventricle it is pumped into the *pulmonary trunk*, a very large artery that divides almost immediately to form the *right* and *left pulmonary arteries*. These vessels deliver blood to the right and left lungs respectively. Upon reaching the lung each pulmonary artery gives rise to branches that service all regions of the organ.

Eventually blood flows into the extensive capillary networks in the walls of the air sacs, where carbon dioxide diffuses out of the blood and oxygen diffuses into it. *Pulmonary capillaries* deliver the new oxygenated blood to pulmonary venules, which join to form larger and larger veins. Two *pulmonary veins* exit from each lung and conduct oxygenated blood to the left atrium of the heart. Note that the pulmonary veins are the only veins that carry oxygenated blood, and the pulmonary arteries are the only arteries that transport de-oxygenated blood (Soloman & Davis, 1983).

(ii) **Systemic circulation**

Blood returning from the pulmonary circulation enters the left atrium, then passes into the left ventricle. From there it is pumped into the largest artery in the body, the *aorta*.

The first portion of the aorta, which travels upward, is known as the *ascending aorta*. Coronary arteries branch off from the ascending aorta and enter the heart muscle. After a short distance the aorta makes a U-turn, the *aortic arch*. Three large arteries that branch off from the aortic arch are (1) the *brachiocephalic (innominate) artery*, which supplies the right upper portion of the body; (2) the *left common carotid artery*, which supplies the left side of the head and neck; and (3) the *left subclavian artery*, which supplies the neck and left arm. As the aorta passes down through the thoracic and abdominal cavities it is called the *descending aorta*. The descending aorta is referred to as the *thoracic aorta* as it passes through the thorax. Below the diaphragm the descending aorta is called the *abdominal aorta*. Branches are given off to all the major organs and tissues. For example, the *renal arteries* branch off to the kidneys. In the lower abdominal cavity the aorta itself divides to form the left and right *common iliac arteries*, which deliver blood to the lower extremities and pelvic structures.

In the systemic circulation blood returns to the heart through two large veins, the *superior* and *inferior vena cava*. The inferior vena cava receives blood returning from the portion of the body below the level of the diaphragm. Two *brachiocephalic (innominate) veins* receive blood from the upper portion of the body and empty it into the superior vena cava (Soloman & Davis, 1983).

(iii) Coronary circulation

The heart is a large organ that requires a rich and continuous supply of nutrients and oxygen. Blood flowing through its chambers cannot serve these needs because the layers of the heart are far too thick to depend upon diffusion. However, the heart is equipped with its own complex of blood vessels, the coronary circulation. Two coronary arteries branch off from the ascending aorta as it leaves the heart, at a point slightly distal to the cusps of the aortic valve (Soloman & Davis, 1983).

3.2.3 Volume-Pressure Relationship of the Circulatory System

Systemic and pulmonary blood vessels are elastic structures. This is demonstrated by their inherent ability to recoil after a deformation-producing stress has been removed. In the intact body the deforming stress is an increase in the intravascular fluid volume. Such increases in the vascular volume will stretch the wall of the vessels and the recoil of the elastic vessel walls will increase the intravascular pressure. The ratio of vascular volume to vascular transmural pressure is called vascular compliance (Green, 1977). Although the term elastance is extremely descriptive, i.e. an increased elastance means greater elasticity, most physiologists prefer to define elastic behaviour in terms of compliance. Compliance is simply the reciprocal of elastance and can thus be defined as:

C = (change in volume)/(change in transmural pressure).

and transmural pressure = pressure inside - pressure outside.

thus, $C = (V-Vu)/(P-0)$

where Vu is the resting or unstressed volume, i.e. the volume contained within the compliant structure when the pressure, P, within the compliant structure is zero (0), and V is the volume above the unstressed volume.

3.2.4 Pressure-Flow Relationship of the Circulatory System

The physical principles which govern the flow of fluids through conducting passages, ie vessels, whether rigid or collapsible, are derived from the general laws of hydrodynamics. The fluid may be either liquid such as blood flowing through the cardiovascular system, or air such as we breathe. The difference in these fluids lies in their different densities and viscosities (Green, 1977). The basic expression for the flow through rigid tubes (for liquid) is that of Poiseuille's law, which states that the volume of fluid flowing past a point in the tube per unit time is proportional to the difference in pressure between the inflow and outflow ends of the tube ($P_I - P_O$) and the fourth power of the radius (r) of the tube, and inversely proportional to the length of the tube (l) and viscosity of the fluid (q).

In mathematical terms Poiseuille's law may be expressed as follows for conditions of horizontal flow:

$$(P_I - P_O)/F = 8ql/\pi r^4$$

The quantity ($8ql/r^4$) represents those factors which tend to retard flow (F) and is referred to as the resistance to flow (π is constant). The most commonly used relationship is thus $(P_I - P_O)/F = R$ where R is the resistance.

3.2.5 Conceptual and Structural Foundations

Leaning et al. (1983a) said:

'Conceptualisation, the first stage of model formulation, involves identification of the main sub-systems and choice of structural detail'.

To establish a conceptual foundation for cardiovascular modelling, it is necessary to examine each sub-system and its properties in depth.

3.2.5.1 Elastic compartments

Conceptually, cardiovascular system (circulation and heart) models can be considered to be a network of elastic compartments linked by resistive tubes (Fig. 3.2). The main sub-systems of cardiovascular models consist of arteries, veins, atria and ventricles. In compartmental modelling the real arteries, veins, atria and ventricles are lumped together and the level of detail depends on the purpose for which we are modelling. A more detailed description of these sub-systems follows.

(i) **Arteries**

In modelling the arteries, the inertial effect of blood has to be included due to its rapid acceleration. Also the wall visco-elasticity, geometric and elastic taper have to be taken into account. Arteries are thick-walled, highly elastic, high pressure vessels.

In reality an artery is an elastic tube whose diameter will vary with a pulsating pressure (McDonald, 1974). In addition, it will propagate pressure and flow waves, created by the injection of blood by the heart, at a certain velocity which is largely determined by the elastic properties of the wall.

The arterial wall belongs to the large class called the <u>visco-elastic</u> class. This class of substances exhibits properties appropriate to both an elastic solid and a viscous liquid.

Dick et al. (1968) have shown by a study on dogs that the non-linearity of the arteries is fairly small, and thus can be neglected. Figure 3.3 shows a typical arterial compartment - the 'ascending aorta', where K_i is wall visco-elasticity and L_{ji} is inertial effect.

(ii) **Veins**

Unlike arteries, veins are highly compliant, collapsible, large-capacity vessels with relatively low transmural pressures and non-linear modelling has to be applied to obtain an adequate representation. This is especially necessary in situations where large volume changes occur in certain venous segments, for example during simulated haemorrhage or tilt-table experiments.

When a venous segment collapses, the effective compliance and resistance change. The compliance increases to approximately 20 times its normal value when the transmural pressure becomes negative (Snyder and Rideout, 1969). Inertial effects are neglected due to the low blood acceleration and also wall viscosity effects are not considered significant in the short time-scale of the model. Valves at various locations in the venous system ensure unidirectional blood flow. Valves effect the flow of blood in the circulation and thus in modelling the venous compartment they have to be considered. Figure 3.4 shows a typical venous compartment - the 'abdominal veins', where α_i is a factor by which a normal compliance changes and β_i is a constant effecting flow, when flow is less than zero and no valves exists.

(iii) The heart chambers

Heart pumping is represented by the pressure-volume curve which rises during heart contraction and this can be shown as the time-varying elastances (reciprocal compliance) of the cardiac compartments (Fig. 3.5).

Figures 3.5(i) and (iii) show the time-varying elastances of the right and left atria. At the start of the cardiac cycle, the elastance of the atrium increases and reaches its maximum just before the ventricle starts to contract. When the ventricle starts to contract, the atrium elastance starts to decrease and reaches its minimum value very quickly.

Figures 3.5(ii) and (iv) show the time-varying elastance of the ventricles. While the atrium is contracting, the ventricular elastance stays at its minimum value until the ventricle starts to contract. Then the elastance increases and reaches its maximum value, but shortly afterwards starts to decrease rapidly to its minimum value just before the ventricular contraction is over.

(a) Ventricles

The ventricles are the lower chambers of the heart. The right ventricle pumps blood into the pulmonary circulation (pulmonary arteries), while the left ventricle pumps blood into the systemic circulation (aorta). These valves are called semilunar valves (they have three cusps shaped like half moons). The semilunar valve between the left ventricle and the aorta is known as the aortic valve, and the one between the right ventricle and the pulmonary trunk as the pulmonary valve.

(b) Atria

The atria are the upper chambers of the heart. The right and left atria, which receive blood returning to the heart from the veins and lungs, act as reservoirs between contractions of the heart. There are valves between the atria and the ventricles to prevent backflows (the atrio-ventricular valves). The resistances of these valves when open are considered in the mathematical representation of flow from atria to ventricles.

The right atrium receives blood from the coronary and bronchial vascular beds as well as from the inferior and superior vena cava, while the left atrium receives blood from the pulmonary circulation. The sequence of events that occurs during one complete heart beat is referred to as the <u>cardiac output</u>. Each complete cycle normally lasts for about 0.8s and thus occurs about 75 times per minute. It consists of both contraction in which blood is forced out of the heart (known as systole), and a subsequent relation in which the heart fills with blood (known as diastole).

3.2.5.2 Circulatory networks

A circulatory network consists of a set of elastic compartments connected by various connection types. There are 6 different connection types in the circulation. The first group of connection types is associated with the heart. There are 4 <u>heart connections</u>, depicted in Fig. 3.6, between the veins and atria (VeinAtr), between the ventricles and arteries (VentArt), between the atria and ventricles (AtrVent) and between arteries and atria (ArtAtr). There are also <u>arterial connections</u> between two arterial compartments (ArtArt) (Fig. 3.7), <u>arterio-venous connections</u> between arteries and veins compartments (ArtVein) (Fig. 3.8), and, lastly, <u>venous connections</u> between two venous compartments (VeinVein). <u>Valves</u> at various locations in the venous system ensure unidirectional blood flow. In the circulatory fluid mechanics model of Beneken and De Wit (1967), a venous valve is included between the segments representing leg veins and abdominal veins to prevent the backflow of blood due to gravity (Fig. 3.9).

3.2.5.3 Circulatory regions

It is also important to identify a <u>circulatory region</u>, which is a set of connected elastic compartments. There exist three circulatory regions of main interest:

1) The heart region where the heart has 4 chambers, two atria and two ventricles. The atria are connected to the ventricles in series and the ventricles pump blood in parallel into the systemic and pulmonary circulation;
2) The pulmonary region is where de-oxygenated blood from the heart is re-oxygenated by the exchange of CO_2 and O_2.
3) The systemic region is where the oxygenated blood is distributed to the body tissues. This region can be divided into abdomen, legs, head and arms, etc, depending on the required level of detail. For the particular models, it may be necessary to have other regions.

3.2.6 Mathematical Representations

The equations developed below have been derived from the general ones described by Leaning et al. (1983a). These are based upon equations originally derived by Pullen (1976) based on the work of Beneken and De Wit (1967) and others.

3.2.6.1 Pressures

(i) Arteries

The static pressure-volume curve of a typical lumped parameter segment is approximated as being linear in the normal operating range and is represented by the tangent at the operating point (Fig. 3.10).

As mentioned earlier, in modelling the arteries, inertial effects, wall visco-elasticity (K_i), and geometric and elastic taper are taken into account (See Section 3.2.5.1 part (i)). Thus, with constant compliance (C_i) the equation relating transmural pressure (P_i) and volume (V_i) for all arteries, i is then

$$P_i = (V_i - V_{ui})/C_i + (K_i/C_i)dV_i/dt \quad \text{if } V_i \geq V_{ui} \qquad (3.1)$$

where V_{ui} is the unstressed volume and K_i is wall visco-elasticity. The pressure drop across the arterial wall is therefore proportional to the stretched volume $(V_i - V_{ui})$. As the compliance C_i (reciprocal of wall elasticity) increases, the pressure drop becomes less. The second term in equation 3.1 relates to wall visco-elasticity. The shape of the wall can change only at a limited rate, summarized by the 'visco-elasticity' constant K_i. It is possible for the transmural pressure to become negative if $V_i < V_{ui}$. In this situation, the effective compliance C_i is likely to change, especially if the vessel is collapsible (as in certain venous segments). The constraint $V_i \geq V_{ui}$ has to be applied, which is equivalent to specifying that the compliance becomes zero at zero volume which corresponds to the real physical situation.

The arterial pressure is constantly changing throughout the cardiac cycle and the mean pressure throughout the cycle is not merely the value halfway between systolic and diastolic pressure, because diastole usually lasts longer than systole. The approach used here is based on Pullen's (1976) derivation.

For all arteries i;

For t=0 to t=T_H (over one cardiac cycle);

$$\text{mean arterial pressure (MAP)} = (1/T_H) \int_0^{T_H} P_i(t)\,dt$$

where $P_i(t)$ is the pressure in arterial compartment i.

The mean arterial pressure is actually the most important of the pressure because it is the <u>average</u> pressure driving blood into the tissues throughout the cardiac cycle. In other words, if the pulsatile pressure (Systolic pressure - Diastolic pressure) changes were eliminated and the pressure throughout the cardiac cycle was always equal to the mean pressure, the total flow would be unchanged.

(ii) **Veins**

As mentioned earlier, the inertial effect and wall viscosity effects are not considered in modelling the veins, thus the pressure-volume equation for all veins, i is

$$P_i = (V_i - V_{ui})/C_i \qquad (3.2)$$

where

$$C_i = C_{oi} \qquad \text{if } V_i > V_{ui}$$
$$C_i = \alpha_i C_{oi} \qquad \text{if } V_i \leq V_{ui}$$
$$(3.3)$$

where C_{oi} is the normal value of compliance, α_i is a factor by which the normal compliance changes when the transmural pressure becomes negative.

When a venous segment collapses, the effective compliance (C_i) changes, (Snyder & Rideout (1969)) and increases to $\alpha_i C_{oi}$ (where $\alpha_i \leq 1$). This occurs when transmural pressure becomes negative, i.e. when the volume becomes less than the unstressed volume (Fig. 3.11).

(iii) **Ventricles**

Beneken (1965) has pointed out that the actual shape of the systolic portion of the elastance waveform in the representation of the pumping action of the heart is not critical provided that the maximum and minimum elastance values remain unchanged. Pullen (1976) has confirmed this assertion in his validation tests by investigating the steady state using triangular, rectangular, half-sinusoidal and parabolic results but, reasonably enough, there are more substantial changes if grossly unrealistic waveforms are used as in the triangular and rectangular cases. His final decision was half-sinusoidal waveform as to parabolic waveform for programming convenience. The generic equations derived here are based on Pullen's (1976) approach. The time-varying elastances ($a_i(t)$) depicted in Figure 3.5 are given by the following general equation.

$$a_i(t) = x(t)(a_{is}-a_{id})+a_{id} \qquad (3.4)$$

(Using the formula $Y=A\sin(2\pi ft)$ as the displacement equation of a waveform where A, the amplitude, is equal to $(a_{is}-a_{id})$ and $\sin(2\pi ft)$ is equal to $x(t)$ as explained in Equation 3.5. A constant a_{id} is added to the wave formula in this case, effectively being a phase lag).
The suffixes d and s for elastance values denote minimum diastolic and maximum systolic values respectively.

The elastance waveforms for the heart are generated by introducing t_c, where t_c is the elapsed time during each cardiac cycle $(0 \le t_c \le T_H)$ and thus for all ventricles:

$$\begin{aligned} x(t) &= 0 && \text{if } t_c < T_{AV} \text{ or } t_c > T_{AV} + T_{VS} \\ x(t) &= \sin[(\pi/T_{VS})(t_c - T_{AV})] && \text{otherwise} \end{aligned} \qquad (3.5)$$

(Using the wave formula $\sin(2\pi ft)$ explained above, but only half the cycle is being considered here, and $ft = (t_c - T_{AV})/T_{VS}$. T_{AV} is the time between the onset of arterial systole and the onset of ventricular systole and T_{VS} is the duration of the ventricular systole. (Both ventricles are assumed to contract simultaneously).

Beneken & De Wit (1967) have made a linear approximation of T_{AV} and T_{VS}, where $T_{AV} = T_{AS}-\lambda_1$ and $T_{VS} = \lambda_2+\lambda_3 T_H$.

The pumping action of the heart in both ventricles is described by the equation relating pressure and volume.

$$P_i = a_i(t)(V_i - V_{ui}) \qquad (3.6)$$

where P_i and V_i are pressure and volume respectively, and V_{ui} is the unstressed volume.

(iv) Atria

The elastances of the atria have the same general equation as the ventricles (equation 3.5), but with for all atria:

$x(t) = 0$ if $t_c > T_{AS}$

$x(t) = \sin(\pi t_c/T_{AS})$ if $t_c \leq T_{AS}$ (3.7)

Beneken & De Wit (1967) have made a linear approximation of T_{AS}, where $T_{AS} = \lambda_1 + \lambda_2 T_H$ and where T_{AS} is the duration of the arterial systole (note that λ_1 and λ_2 are different to the parameters in (iii)).

The pumping action of the heart in both atria is represented by the equation relating pressure and volume (equation 3.6).

3.2.6.2 Flows

The effects of gravity on the columns of blood in the cardiovascular system are included in the flow equations so that it is possible to simulate the dynamics which occur during a passive tilt from a previously resting, recumbent position. The generic equation for the hydrostatic pressure is based on Snyder and Rideout (1969).

(i) From arteries

For all arteries i and all connections j, F_{ji} is the flow from i to j. These correspond to ArtArt, ArtVein and ArtAtr connection types (section 3.2.5.2). The hydrostatic pressure G_{ji} is given by

$$G_{ji} = n.b.g.len_{ji}.\sin(angl_{ji}) \tag{3.8}$$

where b is the density of blood, g is gravitational acceleration, n is the number of g of acceleration, len_{ji} is the effective segment length, $angl_{ji}$ is the angle between the axis of the segment.

For the *systemic circulation* if inertance $L_{ji}>0$ (inertance is inertia of the blood), then

$$dF_{ji}/dt = (P_i+P_{regi}-P_j-P_{regj}-G_{ji}-R_{ji}F_{ji})/L_{ji} \tag{3.9}$$

Else (inertance $L_{ji} = 0$)

$$F_{ji} = (P_i + P_{regi} - P_j - P_{regj} - G_{ji})/R_{ji} \qquad (3.10)$$

where R_{ji} is resistance to flow, and P_{regi}, P_{regj} are the pressures of region. For the *pulmonary circulation,*

$$F_{ji} = (P_i - P_j)/R_{ji} \qquad \text{if } P_j > P_{cc}$$
$$F_{ji} = (P_i - P_{cc})/R_{ji} \qquad \text{otherwise}$$
$$(3.11)$$

(since inertance and hydrostatic pressure are neglected). P_{cc} is the critical closing pressure introduced by Beneken (1967) in the model of pulmonary circulation.

An estimated total systemic resistance (ETSR) can be obtained on a beat-by-beat basis from Pullen (1976):

ETSR = MAP/CO ; where CO is cardiac output (explained later Section 3.3.1.2) and MAP is mean arterial pressure (explained earlier Section 3.2.6.1 part (i)).

(ii) From veins

This includes VeinVein and VeinAtr connections (Section 3.2.5.2). For all veins, i, and all connections, j, the laminar Poiseuille equation is

$$(P_i - P_j)/F_{ji} = (8q\,len_{ji})/(\pi r^4) \qquad (3.12.1)$$

(see Section 3.2.4 for definition of these quantities).

Non-linearity has to be considered (refer to Beneken & De Wit, 1967). If the volume of a vessel of fixed length decreases, the effective radius will decrease (V is proportional to r^4) and the resistance to flow will increase (R is proportional to $1/r^4$). Considering a straight vessel with a circular cross section of radius r connecting segment i and j, the laminar flow will be:

$$F_{ji} = K_A r^4 (P_i - P_j) \qquad (3.12.2)$$

by rearranging equation (3.12.1), where K_A is constant.

Due to the lumped parameter representation, the volume of the vessel is given by

$$V_i = K_B r^2 \tag{3.12.3}$$

Thus from (3.12.2) & (3.12.3)

$$F_{ji} = (K_A)(V_i/K_B)^2(P_i-P_j) \tag{3.12.4}$$

If $K_C=K_A/K_B^{\ 2}$ then from (3.12.4)

$$F_{ji} = K_C V_i^2(P_i-P_j) \tag{3.12.5}$$

K_C relates to the fixed fluidic resistance (R_{ji}) assumed by Beneken & De Wit (1967) for each of their linear venous segments.
If it is assumed that when $V_i=V_{ui}$,

$$F_{ji} = (P_i-P_j)/R_{ji} \tag{3.12.6}$$

then from (3.12.5) & (3.12.6):

$$K_C=1/(R_{ji}V_{ui}^2) \tag{3.12.7}$$

Substituting in equation (3.12.5) and considering the pressure of the regions, the equation for the flow is:

$$\begin{aligned}
F_{ji} &= ((P_i+P_{regi}-P_j-P_{regj}-G_{ji})V_i^2)/(R_{ji}V_{ui}^2) && \text{if } Pi \geq Pj \\
F_{ji} &= \beta_i[((P_i+P_{regi}-P_j-P_{regj}-G_{ji})V_i^2)/(R_{ji}V_{ui}^2)] && \text{if } Pi \leq Pj \text{ with no valve} \\
Fji &= 0 && \text{if } Pi \leq Pj \text{ with a valve}
\end{aligned} \tag{3.12.8}$$

where β_i is a constant which can effect the flow when there is no valve and the flow is less than zero (See Pullen (1976) for more details). P_{regi}, P_{regj} are pressures of region and G_{ji} is hydrostatic pressure in equation 3.8.

Note that equation (3.12.8) is non-linear in V_i. A physiological basis for this is that as the volume of a vein increases, the diameter becomes larger, and the resistance to flow decreases.

If there is a valve in the connection, there is no flow when the pressure across the valve ($P_i+P_{regi}-P_j-P_{regj}$) is negative. Not all VeinVein connection or any other connection representing a collection of veins have valves (Pullen, 1976). Under these circumstances the parameter β_i ($0 \leq \beta_i \leq 1$) indicates the fraction of veins in the collection which do <u>not</u> have valves.

(iii) From ventricles

This includes VentArt connections (Section 3.2.5.2). For all ventricles i, and connections j:

If $L_{ji}>0$ then

$$dF_{ji}/dt = (P_i-P_j-R_{ji}F_{ji}+(b/2)(F_{ji}/A_{ji})^2)/L_{ji} \qquad F_{ji} \geq 0 \qquad (3.13)$$

rearranging the above equation,

$$P_i-P_j = R_{ji}F_{ji} + (dF_{ji}/dt)(L_{ji}) + (b/2)(F_{ji}/A_{ji})^2 \qquad F_{ji} \geq 0$$

Where F_{ji} is the total inflow of j compartment, R_{ji} is resistance to flow, L_{ji} inertia of the blood and b is the density of blood.

The second term of the right hand side of the above equation is the acceleration term and, according to Jones and Kantrowiz (1965), is equivalent to the inertia of a column of blood having a length equal to the left ventricular inner radius and a diameter equal to the diameter of the outflow vessel.

The first term of the right hand side of the above equation indicates the pressure drop caused by the viscous properties of blood and is usually small in comparison with the last term which represents the pressure drop originating from the fact that the outflow vessel has a cross-sectional area A_{ji} that is much smaller than the cross-sectional area of the ventricles. Beneken & De Wit (1967) have applied Bernoulli's theorem (the law of variation of pressure along a stream line) to the efflux of a liquid through a small orifice in a large containing vessel (Newman & Searle, 1952). Steady state conditions are assumed to exist once the flow is established, blood is considered to be incompressible and viscous forces have been shown above to be insignificant. Thus the application of Bernoulli's theorem can be justified.

Else if the inertance $(L_{ji})=0$ <u>and</u> cross-sectional area $(A_{ji})=0$ then Equation (3.13) becomes,

$$F_{ji} = (P_i-P_j)/R_{ji} \qquad (3.14)$$

(iv) From atria

This includes AtrVent connections (Section 3.2.5.2). For all atria i and connections j

$$F_{ji} = (P_i-P_j)/R_{ji} \qquad (3.15)$$

3.2.6.3 Differential equations

(i) Volumes

Law of conservation of mass is applied here.

For every compartment i,

$$dV_i/dt= \sum_k F_{ik} - \sum_j F_{ji} \qquad Vi \geq 0 \qquad (3.16)$$

This says that the rate of change of volume in the compartment i is equal to the summation of all the flows from compartment k to compartment i minus the summation of all the flows from compartment i to compartment j subject to the constraint that the volume of the compartment i must be greater than zero.

(ii) Flows

The above equations (Section 3.2.6.2) are calculated explicitly, and flow needs no updating except when stroke volume (Section 3.3.1.2) is calculated. The rate of change of flow is updated whenever needed.

3.3 NEURAL CONTROL MODELLING

3.3.1 Physiological Background

Most of the physiological literature divides the nervous system into two parts. The first part is called the *Central Nervous System (CNS)* which consists of brain and spinal cord. The second part is called the *Peripheral Nervous System (PNS)*. The PNS consists of all the sensory receptors, the nerves that link receptors with the CNS, and the nerves that link the CNS with the effectors.

The PNS consists of two parts. The first part is the *somatic nervous system* which keeps the body in balance with the external environment. The second part is the *autonomic nervous system* which is designed to maintain internal homoeostasis (e.g. temperature, or heart rate).

The autonomic nervous system consists of two different type of neuron. The first are called the *afferent neurons* where sensory information from the viscera is transmitted to the CNS. The second are called the *efferent neurons* which comprise of the *sympathetic system* and the *parasympathetic system*.

The governing centre for the cardiovascular system is the so-called vasomotor centre in the medulla oblongata region of the brain. The efferent impulses are conveyed via the autonomic nervous system to the effectors which in this case are the cardiac muscle and smooth muscle of the arterioles and veins.

The sympathetic system is associated with mobilizing energy during stress situations. Its nerves increase blood pressure, speed the rate and force of the heart beat, increase blood sugar concentration, and re-route blood flow so that skeletal muscles receive the amounts of blood necessary to support their maximum effort.

The parasympathetic system works in the opposite way to the sympathetic system, acting to conserve and restore energy. Its nerves decrease the rate and force of the heart beat, and stimulate the digestive system to process food (Soloman and Davis, 1983).

3.3.1.1 Regulation of blood pressure

Specialized nerve cells called *baroreceptors* are located in the walls of the large arteries in the thoracic and neck regions. They are most abundant in the walls of the *aortic arch*, in the *carotid sinus*, in the *vena cava* and in the *right atrium*. These baroreceptors are sensitive to change in blood pressure. When an increase in blood pressure stretches their walls, they send impulses to the cardiac centres in the medulla. Parasympathetic nerves are stimulated and cause the heart rate to slow, bringing the blood pressure back to normal. Baroreceptors also send impulses to the vasomotor centre in the medulla, inhibiting sympathetic nerves that supply the arterioles and veins. The effect is to dilate the arterioles (and veins), thereby lowering blood pressure.

On the other hand, any slight decrease in blood pressure, causes the baroreceptors to decrease their steady rate of firing. As a result, sympathetic nerves send messages to the blood vessels, causing vasoconstriction. The cardiac centres in the medulla also slow their parasympathetic messages to the heart. This allows the sympathetic nerves to dominate so that the heart beats faster and blood pressure is increased. These normal reflexes act continuously to maintain a steady state of blood pressure (Soloman and Davis, 1983).

3.3.1.2 Regulation of stroke volume

Stroke volume, the volume of blood pumped by one ventricle during one contraction, also has a direct effect upon cardiac output (cardiac output = (stroke volume)×(heart rate). The ventricles do not eject all the blood within them when they contract. The more forcefully they contract, the greater the volume of blood ejected. Furthermore, the volume of blood delivered to the heart varies from time to time. Stroke volume is regulated mainly by venous return and by sympathetic stimulation.

The greater the amount of blood delivered to the heart by the veins, the more blood the heart pumps. This relationship, known as Starling's law of the heart, specifies that the heart pumps all the blood delivered to it within physiological limits (Soloman & Davis, 1983).

In terms of mathematical representation in this model the stroke volume (SV) can be defined as

For $t=0$ to $t=T_H$ (over one cardiac cycle);

$$SV_i = \int_0^{T_H} F_{ji}(t)\,dt \qquad \text{see Pullen (1976)}$$

where F_{ji} is ventricles flows from ventricle i to arterial compartment j. Cardiac output (CO_i, cardiac output is the volume of blood pumped by one ventricle in one minute) is;

$$CO_i = SV_i/T_H \qquad \text{see Pullen (1976)}$$

3.3.2 Mathematical Representation

3.3.2.1 Baroreceptor model

As mentioned earlier the baroreceptor reflexes provide important short-term negative feedback for the regulation of blood pressure. It is known that the baroreceptor reflex response depends on the net effect of all the baroreceptor impulses arriving at the brain and the firing frequency of single baroreceptor nerve fibres depends on both the time derivative and average value of the pressure. Katona et al (1967) developed a simple model in which the output of the baroreceptors is characterized by a single quantity, the 'input function' (B_1) for each heart beat, where B1 is:

$$B_0 = (P_S + P_D)/2 + \sigma(P_S - P_D) - P_T$$
$$B_1 = B_0 \qquad \text{if } B_0 > 0$$
$$B_1 = 0 \qquad \text{if } B_0 \leq 0$$

(3.17)

where P_S is systolic pressure, P_D is diastolic pressure, and P_T is the steady state threshold pressure bellow which neural firing cannot occur.

The first term of the equation (3.17), $(P_S + P_D)/2$, is a measure of the general pressure level and gives an approximation for the mean arterial pressure. The second term of the equation (3.17), $\sigma(P_S - P_D)$, is the integral of the positive pressure derivative for each beat and is proportional to the pulsatile pressure (See Section 3.2.6.1 (i) for definition of pulsatile pressure).

In Katona's model, B is a constant for each cardiac cycle. Pullen (1976) modified Katona's model to include the high speed dynamics occurring within the cardiac cycle. Also separate representations of the two baroreceptors, aortic arch and carotid sinus, are incorporated to take account for the large intra-thoracic pressure changes which may cause the blood pressure in these region to be different.

The neural control model described here is based on Katona's model with Pullen's modification. The generic equation for the neural control follows:

(i) **Baroreceptors**

For all baroreceptors, b

$$dx_1/dt = (P_i - x_1)/t_1 \tag{3.18}$$

The dynamic mean pressure estimate (x_1) is obtained by passing the sensed pressure (P_i) in compartment i through a first-order low-pass filter with a long time constant t_1.

$$dx_2/dt = ((dP_i/dt)^+ - x_2)/t_2 \tag{3.19}$$

where $(dP_i/dt)^+$ is a positive pressure derivative.

The dynamic estimate of the positive pressure derivative (x_2) is obtained by passing the positive time-derivative of the sensed pressure through a first order low-pass filter with a very short time constant t_2.

The output function for the individual baroreceptor is:

$$\text{output} = a_1[(x_1 - P_{Ti}) + a_2 x_2]$$
$$B_b = \text{output} \qquad \text{if output} > 0$$
$$B_b = 0 \qquad \text{if output} \leq 0 \tag{3.20}$$

where a_1 is the contribution of each baroreceptor over one cardiac cycle, a_2 is the average contribution of the positive pressure derivative term over one cardiac cycle and P_{Ti} is a threshold pressure. (Suffix i is arterial index).

(ii) Central nervous control of heart rate

The mathematical model of heart rate is based on work by Katona et al (1967). Katona used chloralose-anaesthetized dogs for his experiments. He inflated a balloon in the thoracic aorta and measured the changes in blood pressure and heart period (reciprocal of heart rate). From the study of his experimental results, a two-region dynamic empirical model was established and the model was simulated on a digital computer in order to find the parameters which resulted in the best least squares correspondence between the actual and predicted outputs.

There is only one heart rate controller. The CNS input function for heart rate is as follows:

For b=1 to nbar where nbar is a maximum number of baroreceptors input:

$$B_{HR} = \sum_{b=1}^{nbar} a_b B_b \tag{3.21}$$

where B_{HR} is a CNS input for heart rate, a_b is a constant (contribution of each baroreceptor) in the range of $0 \leq a_b \leq 1$ and B_b is a baroreceptor input. The term $\sum a_b B_b$ means the summation of all the baroreceptor inputs.

In region one when the blood pressure is above normal, the central nervous input function B_{HR} is greater than a threshold value B_{HRT}. The heart period is characterized by relatively large and fast responses to changes in blood pressure, which are predominantly of vagal origin. The dynamics of region one are approximated by a first order system and are described by the following equations:

$$x_1 = 0 \qquad \text{if } B_{HR} \leq B_{HRT}$$
$$x_1 = b_1(B_{HR}-B_{HRT}) \qquad \text{if } B_{HR} > B_{HRT} \tag{3.22}$$

where b_1 is a measure of relative steady-state gain in region one with respect to region two (i.e. is constant).

$$x_2 = t_1 \qquad \text{if } dx_1/dt \geq 0$$
$$x_2 = t_2 \qquad \text{if } dx_1/dt < 0 \tag{3.23}$$

$$dx3/dt = (x1-x3)/x2 \tag{3.24}$$

where t_1 and t_2 are time constants for increasing and decreasing pressures.

The above equation has been derived from the fact that for a first order filter the equation is

$sx_3 = (x_1-x_3)/x_2 \implies x_3(sx_2+1) = x_1 \implies x_3 = x_1(1/(sx_2+1))$

In region 2 when the blood pressure is below normal, the central nervous input function of heart rate B_{HR} is generally less than a threshold value B_{HRT}, the heart period is relatively small and slow, this is caused by the joint action of both sympathetic and vagus nerves. The dynamics of region 2 are approximated by a second order system and described by the following equations:

$$x_4 = B_{HRT} \qquad \text{if } B_{HR} \geq B_{HRT}$$
$$x4 = BHR \qquad \text{if } B_{HR} < B_{HRT} \tag{3.25}$$

$$dx_5/dt = (x_4-x_5)/t_3 \tag{3.26}$$

$$dx_6/dt = (x_5-x_6)/t_4 \tag{3.27}$$

where t_3 and t_4 are time constants.

The above equations have been derived from the fact that for a second order filter the equations are

$sx_5 = (x_4 - x_5)/t_3 \Longrightarrow x_5(st_3+1) = x_4 \Longrightarrow x_5 = x_4(1/(st_3+1))$

$sx_6 = (x_5 - x_6)/t_4 \Longrightarrow x_6(st_4+1) = x_5 \Longrightarrow x_6 = x_5(1/(st_4+1))$

$\Longrightarrow sx_6 = x_4(1/(st_3+1))(1/(st_3+1))$

The same argument is carried out in the following equations.

The overall response of the controller is obtained by linear combination of outputs in region 1 and 2:

$$x_7 = b_2(x_3+x_6) \qquad (3.28)$$

where b_2 is a constant.

For the heart period (T_H) the following constraint is added to Katona's basic model:

$$\begin{aligned} T_H &= T_{Hmax} && \text{if } x7 \geq T_{Hmax} \\ T_H &= x_7 && \text{if } T_{Hmin} < x7 < T_{Hmax} \\ T_H &= T_{Hmin} && \text{if } x7 \leq T_{Hmin} \end{aligned} \qquad (3.29)$$

where T_{Hmax} and T_{Hmin} are the maximum and minimum heart periods.

(iii) Central nervous control of peripheral resistance

Smooth muscle in the walls of the arterioles is normally in a state of partial contraction due to sympathetic tone originating from the medullary vasomotor centre. A reduction of sympathetic activity which results from a rise in blood pressure leads to vasodilation and a decrease of peripheral resistance (peripheral resistance is the impedance to blood caused by blood viscosity and by friction between the blood and the wall of the blood vessels). Conversely, an increase of sympathetic activity which results from a fall in blood pressure leads to vasoconstriction and an increase of peripheral resistance. A mathematical model of peripheral resistance control must approximate these effects satisfactorily.

To investigate the regulation of peripheral resistance, Scher and Young (1963) experimented with an isolated perfused carotid sinus of 35 cats and 13 dogs to maintain pressures at different levels, and measured the resultant systemic pressures. The experiments were conducted on various numbers of slightly anaesthetized cats and dogs. In most experiments, the vagus nerves were cut to eliminate effects from the aortic arch baroreceptors. The resulting changes in systemic arterial pressure were considered to be due to changes in total peripheral resistance.

Hyndman (1970) implemented a model based on the Scher & Young (1963) results which controlled the total peripheral resistance of the system. The controlling variable was not a mean pressure, as used in the experiments of Scher & Young (1963), but rather the 'input function B_1' defined in Section 3.3.2.1 (i). This function purports to be a more realistic representation of the effective baroreceptor activity.

Hyndman (1970) separated the system into two parts, a linear part and a non-linear part. With the non-linear block positioned between the baroreceptor and linear block a test was made to determine how critically the magnitude of the oscillations depended on the shape of the input (B) - output (peripheral resistance) curve. The characteristic curve chosen was an 'on-off' or 'bang-bang' function. For this curve the maximum and minimum resistance values in the steady state were 1.5s if the input function was less than its threshold value, and 0.5s if it exceeded its threshold value. In mathematical terms:

x1 = 1.5s if input function > threshold value

x1 = 0.5s if input function ≤ threshold value

Hyndman implemented the linear part as having two poles and one zero (shown below). He derived the actual differential equations describing the linear portion of the model as

Laplace(R)/Laplace(x1) = $[(1+7.95s)e^{(-1.5s)}]/(1+4s)(1+20s)$.

The poles correspond to time constants of 20s and 4s and the zero corresponds to a time constant of 7.95s. He obtained a time delay of 1.5s in the model from the experiments where the digital flow was observed by him after pressing the neck in the region of the carotid sinus in a conscious man.

In adapting Hyndman's representation to the large scale model Pullen (1976) made a number of changes. He omitted the time delay of 1.5s in

Hyndman's model because, firstly, such small changes will not be observable within the limits of experimental error, and secondly because it would add considerably to the computational time.

The mathematical model of peripheral resistance represented here is based on work by Hyndman (1970) and the assumptions made by Pullen (1976).

There is only one peripheral resistance controller. The CNS input function for peripheral resistance is:

For b=1 to nbar where nbar is a maximum number of baroreceptors input:

$$B_{PR} = \sum_{b=1}^{nbar} a_b B_b \qquad (3.30)$$

where B_{PR} is a CNS input for peripheral resistance, a_b is a constant in the range of $0 \leq a_b \leq 1$ and B_b is a baroreceptor input. The term $\sum a_b B_b$ means the summation of all the baroreceptor inputs.

Hyndman's peripheral resistance controller may be described by the following second order system equations:

$$\begin{aligned} x_1 &= B_{PRTmax} && \text{if } B_{PR} \leq B_{PRT} \\ x_1 &= B_{PRTmin} && \text{if } B_{PR} > B_{PRT} \end{aligned} \qquad (3.31)$$

The bang-bang (on-off) action represented by equation (3.31) occurs when the central nervous input function of peripheral resistance (B_{PR}) crosses the threshold value B_{PRT} which is used in the heart rate controller of Katona et al (1967) described in Section 3.3.2.1 part (ii); B_{PRTmax} is a maximum and B_{PRTmin} is a minimum value of resistance produced by reflex. The dynamics of the system are described by the following equations:

$$dx_2/dt = (x_1 - x_2)/t_1 \qquad (3.32)$$

$$dx_3/dt = (x_1 - x_3)/t_2 \qquad (3.33)$$

where t_1 and t_2 are time constants.

The overall response of the controller is obtained by linear combination of the above outputs:

$$x_4 = b_3 x_3 + (1-b_3) x_2 \tag{3.34}$$

thus CNSP = x_4. (CNSP used in Section 3.4)

where b_3 and $(1-b_3)$ control the percentage deviation of x_3 and x_2 from the normal value of 1.0.

The output x_4 in equation (3.34) is a dimensionless quantity which multiplies the normal value of resistance in the arterial compartments.

(iv) Central nervous control of myocardial contractility

The heart muscle has sympathetic innervation and increased sympathetic activity in the medullary centres (e.g. as a result of a fall in blood pressure) which results in a positive inotropic (force of contraction of the heart muscle) response i.e. more powerful contraction of the ventricular musculature.

Martin et al (1969) used the isolated, paced, isovolumetric, canine left ventricular preparation and separately perfused carotid sinuses, having provided parameter values for the simulation of this reflex.

Hyndman (1970) used these parameter values to implement his model. He assumed that the contractility model operated linearly in two regions; a region where the systolic elastance is under the control of sympathetic impulses (where the baroreceptor input is less than the threshold value) and a region in which vagal impulses controlled the systolic elastance (where the baroreceptor input is greater than the threshold value).

$x1 = 0.4$, if input function $>$ threshold value

$x1 = 1.0$, if input function $\leq$ threshold value

A first order system with a time constant of 10s was derived by Hyndman (1970). That model is adapted here.

There is only one myocardial contractility controller. The CNS input function for myocardial contractility is:

For b=1 to nbar where nbar is a maximum number of baroreceptors input:

$$B_{MC} = \sum_{b=1}^{nbar} a_b B_b \qquad (3.35)$$

where B_{MC} is a CNS input for myocardial contractility, a_b is a constant in the range of $0 \leq a_b \leq 1$ and B_b is a baroreceptor input. The term $\sum a_b B_b$ means the weighted summation of all the baroreceptor inputs.

Hyndman's myocardial contractility controller may be described by the following first order system equations:

$$x_1 = B_{MCTmax} \quad \text{if } B_{MC} \leq B_{MCT}$$
$$x_1 = B_{MCTmin} \quad \text{if } B_{MC} > B_{MCT}$$
$$(3.36)$$

The bang-bang (on-off) action represented by equation (3.36) occurs when the central nervous input function of myocardial contractility B_{MC} crosses the threshold value B_{MCT} which is used in the heart rate controller of Katona et al (1967) described in Section 3.3.2.1 part (ii); B_{MCTmax} is a maximum and B_{MCTmin} is a minimum value of contractility produced by reflex. The dynamics of the system are described by the following equation:

$$dx_2/dt = (x_1 - x_2)/t_1 \qquad (3.37)$$

thus CNSM = x_2. (CNSM used in Section 3.4)

where t_1 is time constant.

The output x_2 in equation (3.37) is a dimensionless quantity which multiplies the normal systolic elastances in the four heart chambers.

(v) Central nervous control of venous tone

The veins are supplied with vasomotor fibres for the resistance vessels. However the importance of the pressure-volume relationship in the veins, rather than changes in resistance to flow, has been pointed out by Bartelstone (1960).

Bartelstone (1960) provided quantitative evidence of the extent of a reflex in anaesthetized dogs. He occluded both the descending aorta and the thoracic vena cava simultaneously. Venous pressure was measured from the right femoral vein up into the abdominal portion of the inferior vena cava. Mean venous pressure was recorded. Venous outflow was measured by shunting the venous return (venous return is an amount of blood delivered to the heart by its veins) through an external circuit before it entered the right atrium.

These occlusions make it possible to investigate the responses of the venous system to vascular reflex activity. He showed in his results, during the 'resting' state, that blood returns to the heart because of the existence of a venous pressure gradient of 3 to 4 mmHg. Augmentation of sympathetic tone produces an increase in venous return.

As Bartelstone (1960) studied only the venous tone reflex in the lower region, Hyndman (1970) used the Bartelstone results and assumed a similar relationship for the rest of the circulation. Hyndman assumed 'on-off' or 'bang-bang' control for the venous tone as he did for peripheral resistance. He considered a first order system having a time constant of 14.0s.

The mathematical model of venous tone represented here is based on work by Hyndman (1970) and a suggestion of Snyder & Rideout (1969) which is that the changes of venous tone result in changes in the unstressed volume and compliances of lumped parameter venous segments.

There is only one venous tone controller. The CNS input function for venous tone is:

For b=1 to nbar where nbar is a maximum number of baroreceptors input:

$$B_{VT} = \sum_{b=1}^{nbar} a_b B_b \tag{3.38}$$

where B_{VT} is a CNS input for venous tone, a_b is a constant in the range of $0 \le a_b \le 1$ and B_b is a baroreceptor input. The term $\sum a_b B_b$ means the summation of all the baroreceptor inputs.

Hyndman's venous tone controller may be described by the following first order system equations:

$x_1 = B_{VTTmax}$ if $B_{VT} \leq B_{VTT}$
$x_1 = B_{VTTmin}$ if $B_{VT} > B_{VTT}$

(3.39)

The bang-bang (on-off) action represented by equation (3.39) occurs when the central nervous input function of venous tone (B_{VT}) crosses the threshold value B_{VTT} which is used in the heart rate controller of Katona et al (1967) described in Section 3.3.2.1 part (ii); B_{VTTmax} is a maximum and B_{VTTmin} is a minimum value of venous tone produced by reflex. The dynamics of the system are described by the following equation:

$$dx_2/dt = (x_1 - x_2)/t_1 \qquad (3.40)$$

where t_1 is time constant.

The overall responses of the controller are obtained by linear combination of the above output:

$$x_3 = 1 + c_{gain}(x_2 - 1) \qquad (3.41)$$

Thus CNSVC = x_3 (CNSVC used in Section 3.4)

$$x_4 = 1 + v_{ugain}(x_2 - 1) \qquad (3.42)$$

thus CNSVVu = x_4 (CNSVVu used in Section 3.4)

where c_{gain} and v_{ugain} control the percentage deviations of x_3 and x_4 from the normal value of 1.0.

3.4 EFFECT OF NEURAL CONTROL ON THE CIRCULATION

If neural control exists; the model is controlled, then there are four factors effecting the circulation.

1) If neural control exists, then it will have an effect on some unstressed volumes and compliances in venous compartments (e.g. in 19-segment Leaning et al. (1983) model, sympathetic innervation is assumed only in the head and arms veins, intestinal veins abdominal veins and leg veins

segments). The unstressed volume and compliance in equation 3.2 (venous pressure equation) will be as follows:

$V_{ui} = V_{ui}/CNSVVu$ if neural control is true
$V_{ui} = V_{ui}$ if neural control is false
(3.43)

The quantity CNSVVu is the dimensionless quantity in equation 3.42 from the previous Section (V).

The compliance will be as follows:

$C_i = C_i/CNSVC$ if neural control is true
$C_i = C_i$ if neural control is false
(3.44)

The quantity CNSVC is the dimensionless quantity in equation 3.41 from the previous Section (V).

2) If neural control exists then all the heart elastances will be effected. The $a_i(t)$ quantity in equation 3.4 (heart elastances equation) will be as follows:

$a_i(t) = x(t)((CNSM)(a_{is})-a_{id})-a_{id}$ if neural control is true
$a_i(t) = x(t)(a_{is}-a_{id})-a_{id}$ if neural control is false
(3.45)

The quantity CNSM is the dimensionless quantity in equation 3.37 from the previous Section (IV).

3) If neural control exists then it will have an effect on the arterio-venous resistances plus arterio-atrial resistance but not the arterial resistances (e.g. in the 19-segment Leaning et al. (1983) model, the dimensionless quantity which multiplies the normal values of arterio-venous and arterio-atrial resistances in the bronchial, intestinal, abdominal and leg vascular beds). The arterio-venous resistance Rji quantity in equation 3.9, 3.10, 3.11 (from arterial flow equations) will be as follows:

$$
\begin{aligned}
R_{ji}(t) &= (CNSP)R_{ji} && \text{if neural control is true} \\
R_{ji}(t) &= R_{ji} && \text{if neural control is false}
\end{aligned}
\qquad (3.46)
$$

The quantity CNSP is the dimensionless quantity in equation 3.34 from the previous Section (III).

4) The heart period T_H quantity is not constant, but is calculated using equation 3.29. For the uncontrolled model the heart period is the same as the initial value of the heart period.

3.5 CONCLUSION

This chapter has presented the basic elements of pulsatile circulatory and neural control modelling. An actual model will be generated in terms of a set of elastic compartments configured as a circulation network. Equations for pressures, volumes and flows must be written and solved for each compartment or connection. Also the neural equation equations must be written for each baroreceptor and for bang-bang control. Notice that the equations in this chapter are explicit and there is no algebraic loop. The modelling system described in the following chapters is based on these conceptual, structural and mathematical elements. In the next chapter the database-oriented architecture approach, and the representation of the modelling system discussed here in terms of a relational database, will be presented.

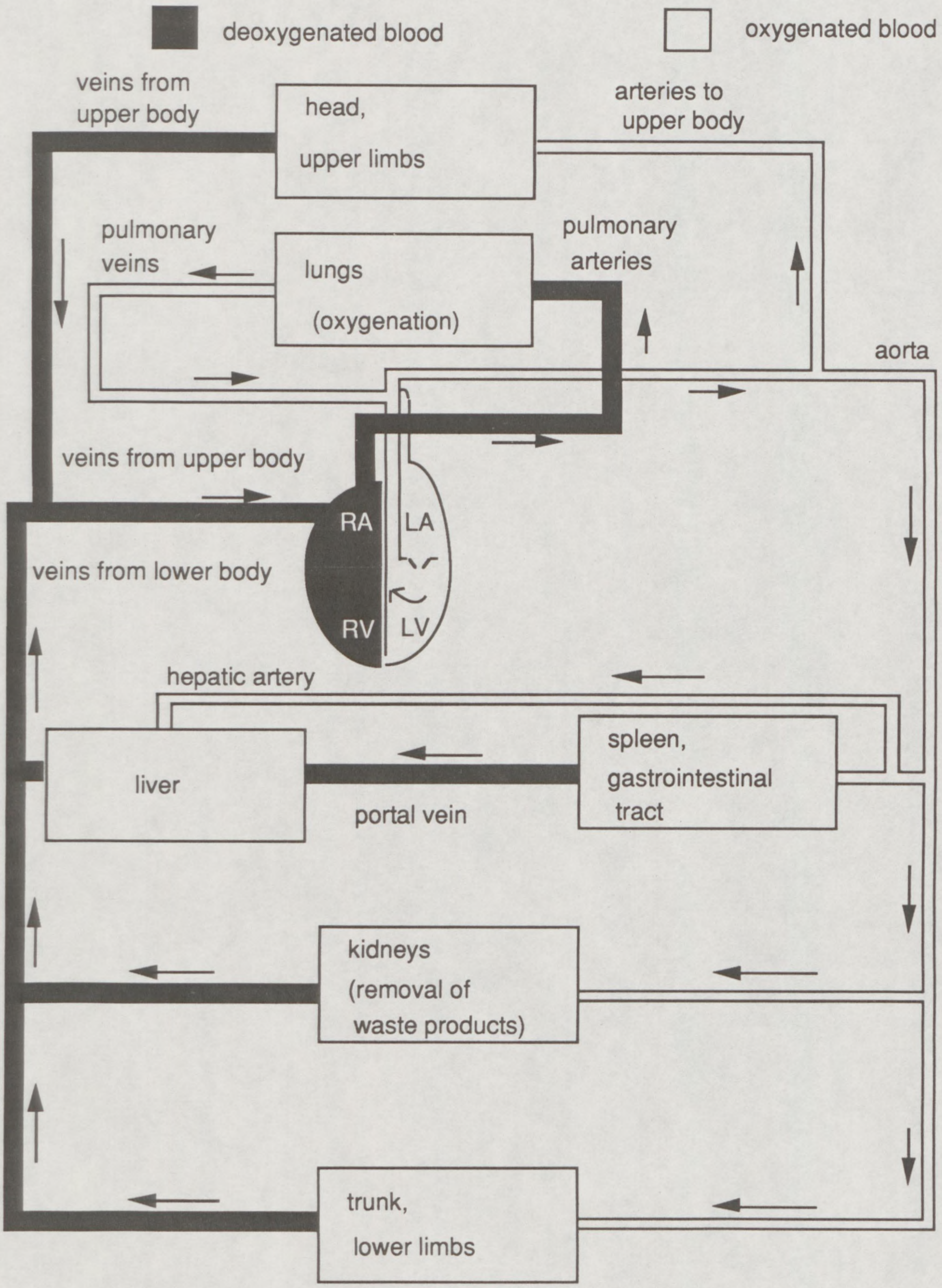


Figure 3.1 **Blood circulation**

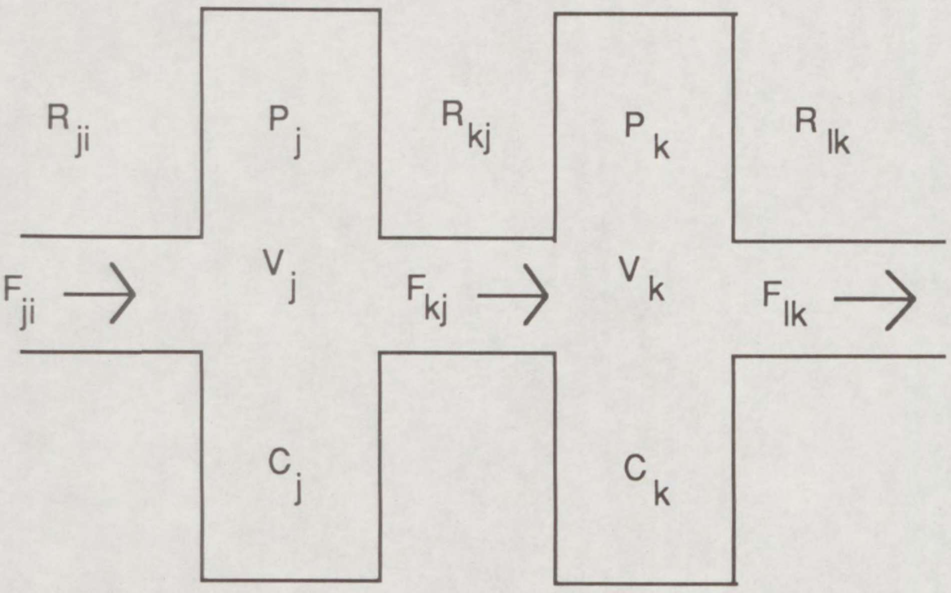


Figure 3.2 **Two typical adjoining elastic circulatory compartments**

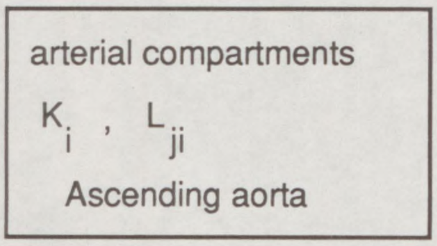


Figure 3.3 **A typical arterial compartment**

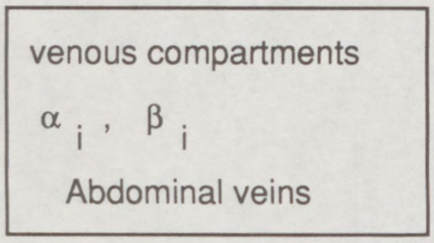


Figure 3.4 **A typical venous compartment**

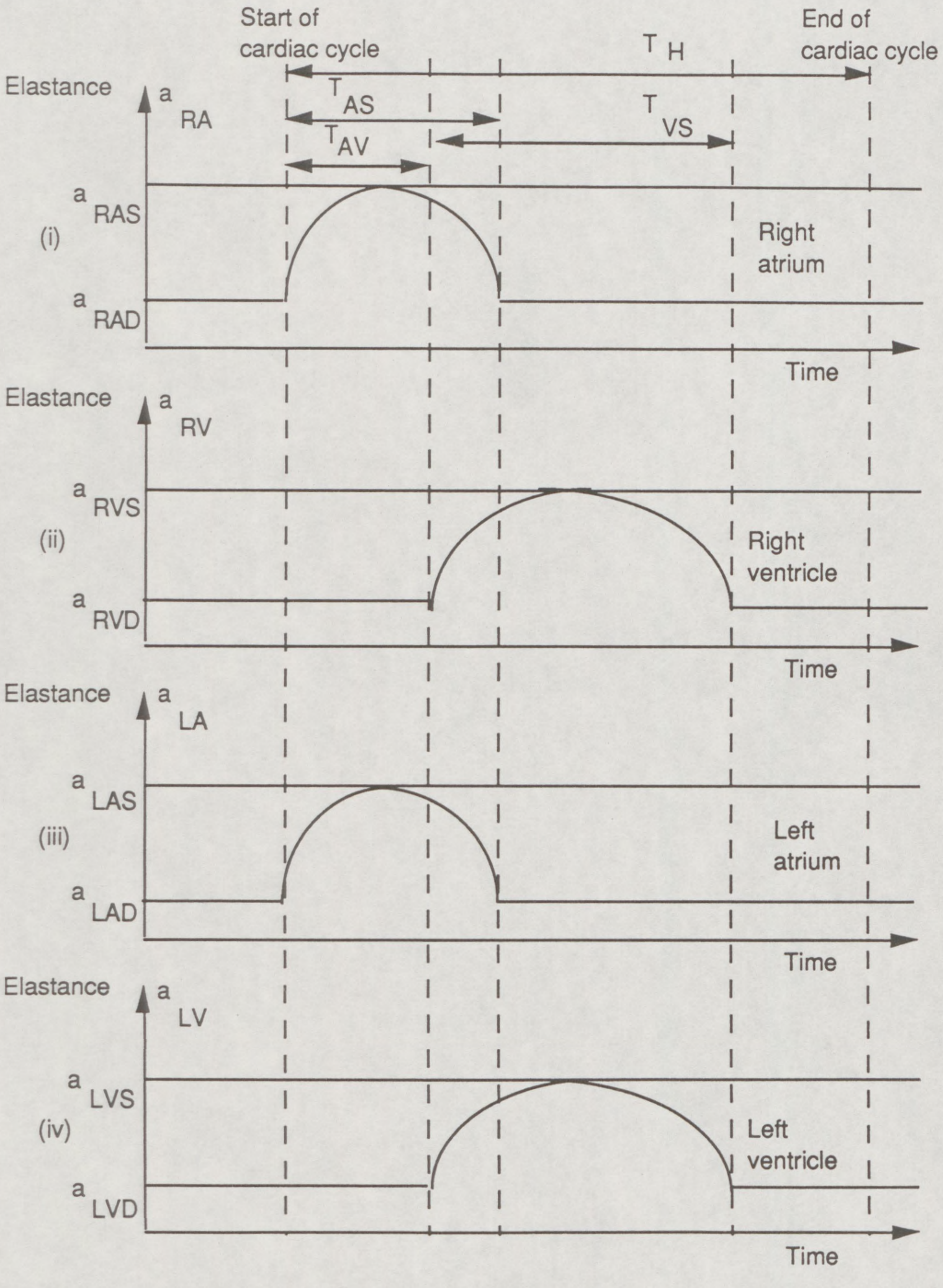


Figure 3.5 **Elastances of the four heart chambers**

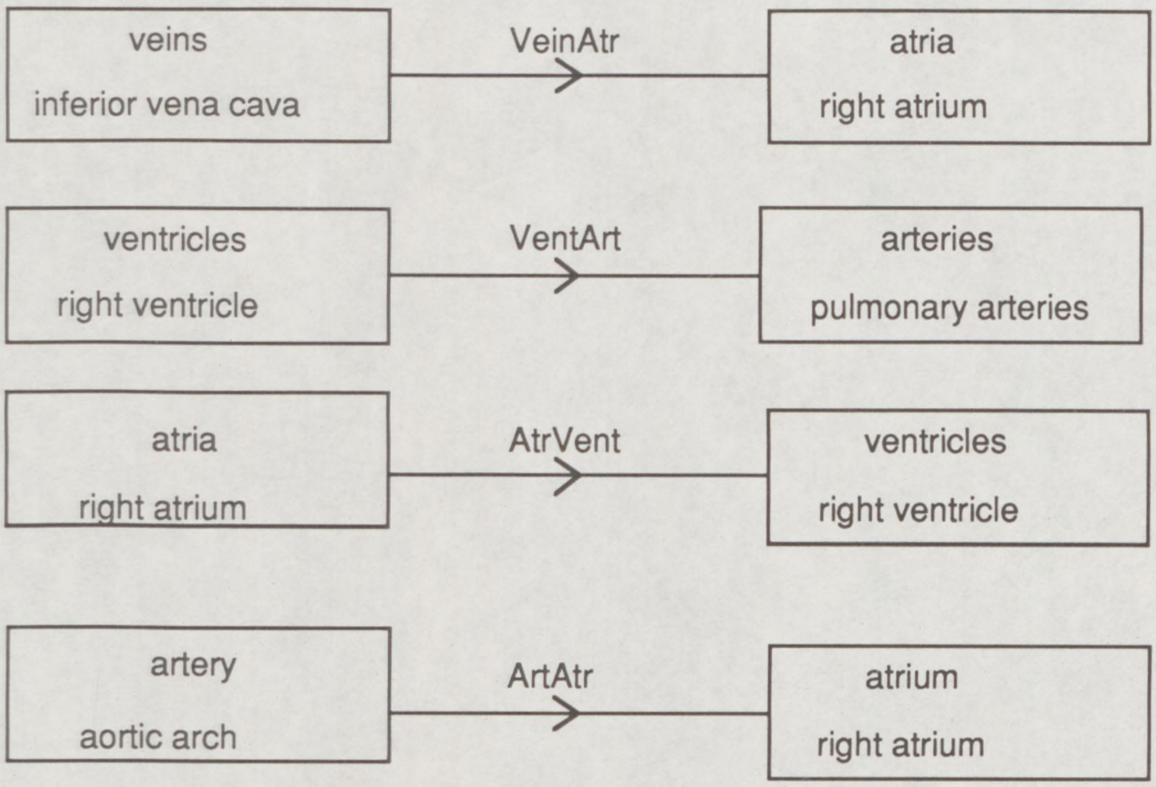


Figure 3.6 **The connection types associated with the heart**

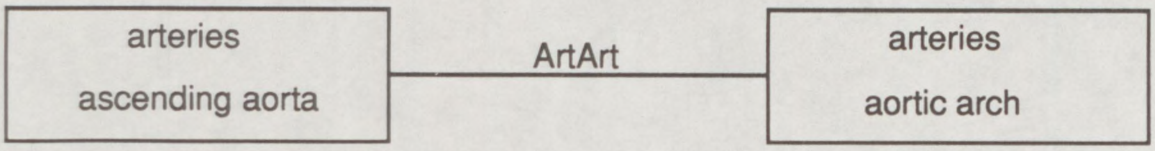


Figure3.7 **A typical arterial connection**

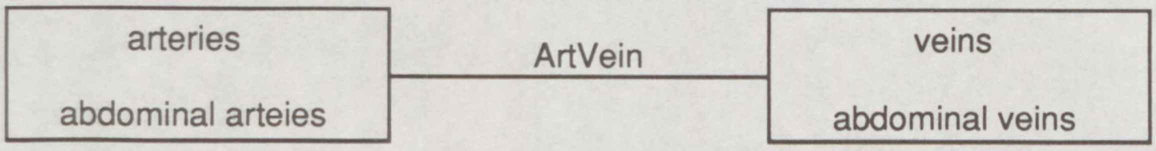


Figure 3.8 **A typical arteriovenous connection**

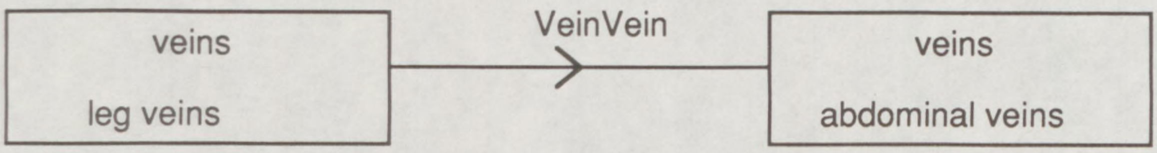


Figure 3.9 **A typical venous connection**

(where > indicates venous valve)

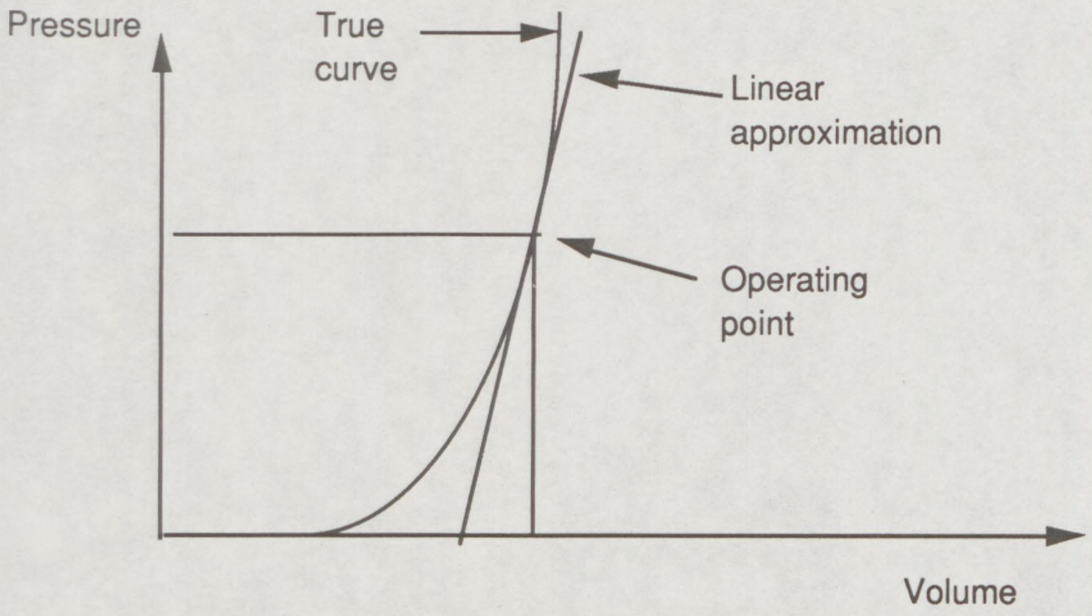


Figure 3.10 **Static pressure-volume curve**

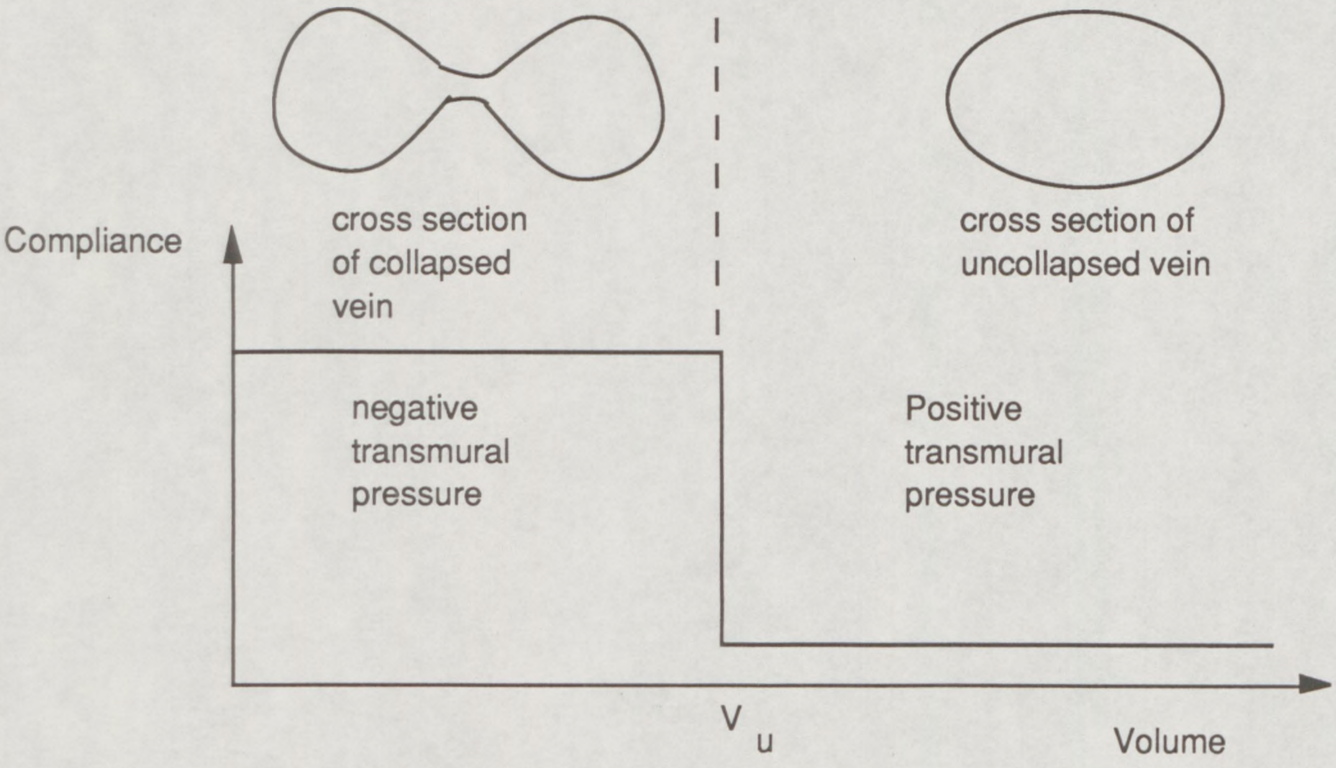

cross section of collapsed vein
cross section of uncollapsed vein
Compliance
negative transmural pressure
Positive transmural pressure
V_u
Volume

Figure 3.11 **Piecewise linear approximation for the compliance of a venous segment (Snyder & Rideout, 1969)**

CHAPTER 4

A DATABASE-ORIENTED ARCHITECTURE FOR CIRCULATORY MODELS

4.1 INTRODUCTION

A set of conceptual and structural foundations for pulsatile circulatory modelling was presented in the last chapter, together with the appropriate mathematical representations. In this chapter a new approach to computer realization of cardiovascular models based on relational databases (Date, 1986) is described. The chapter begins by outlining the design of the database-oriented architecture for computer simulation models, based on Leaning (1986). The relational schema for cardiovascular models is then developed based on the foundations laid down in Chapter 3. The chapter concludes with a description of the general algorithm for simulating cardiovascular models.

As discussed in Chapter 2 there are a number of important limitations to using general-purpose programming languages (e.g. FORTRAN):

(1) For each model a new program needs to be written;

(2) The model structure is difficult to change;

(3) the model is not explicit.

An artificial intelligence (AI) approach offers an appealing prospect for modelling but is not suitable for arithmetic problems (See Chapter 5 Section 5.2). The current approach is an intermediate solution, the worth of which is demonstrated by producing a functioning system. The chapter refers to a 19-segment model to illustrate some examples; this is the 19-segment model built by Beneken and De Wit (1967), reported by Pullen (1976) and Leaning (1983). Through this chapter for simplicity the references are omitted.

4.2 OVERALL DESIGN

As explained earlier, the aim of the current work is to adopt an intermediate approach in which the model is represented explicitly in a relational database. This will enable an explicit representation of the model and its parameter values to be made, as well as providing an easy facility to change both the model structure and parameters. Time simulations are performed by a general-purpose simulation engine. This approach has the advantage that the well-developed techniques of relational analysis (Date, 1986) can be used to structure a database, and that database management software is widely available and easy-to-use.

The principal features of the database-oriented architecture proposed by Leaning (1986) are shown in Fig. 4.1. The cardiovascular model is represented in a relational database. The "simulation engine" scans the database in order to produce a quantitative time simulation. There is also some conceptual overlap with the ideas of Yamamoto (1985).

4.2.1 The Database Environment

The database environment consists of a database management system (DBMS), its programming language and the various files. The model is represented as a set of related files - <u>the model relational database</u>. A structure for this database (the "relational schema") for cardiovascular models is described in Section 4.3 The <u>definition generator</u> converts the internal form of database storage into a set of independent <u>simulation definition files</u>. The latter are standard ASCII text files which are readable by the <u>simulation engine</u>. Both the user and system developer interact with the system through the standard front-end of the DBMS. The DBMS is also the channel for user interaction with the simulation. The model relational database section which consists of database files has been developed in dBASEIII+ (refer to Chapter 5 Section 5.2.1). After the database files have been developed an application program written in the DBMS programming language (Definition Generator Fig. 4.1) translates these files into ASCII code. These are called <u>simulation definition files</u>. Once the desired files are written in ASCII code (which will be illustrated in Chapter 5), they are then read by the simulation engine which performs both numerical and graphical simulations.

4.2.2 The Procedural Language Environment

The procedural language environment is that in which the system developer implements the simulation engine. The development of the simulation engine is one-off and does not have to be repeated for each cardiovascular model. The procedural language used here is Pascal (see Chapter 5 Section 5.2.1) and the user interacts with the simulation engine via the database environment. It is therefore not necessary for the end user to interact with the simulation engine directly. Thus, there exists only one simulation program, which is universal for the class of model.

4.3 RELATIONAL SCHEMA FOR CARDIOVASCULAR MODELS

A relational schema is the conceptual model underlying a relational database (Date, 1986). The term "schema" stands for a data structure which groups together information about one particular object or concept. It resembles a 'record' in Pascal. Each "schema" contains a number of slots, which represent the attributes of the characteristics of the object. Each of these slots may have one or more values. The links between schemata are specified using slots, which are known as "relations". The term schema is used here for "conceptual model" the more usual database term, to distinguish it from the cardiovascular model. The relational schema is divided into a number of sections as shown in Fig. 4.2.

Date (1986) defines a relational schema as a number of interconnected entities which reflect elements, processes or parameters in the real world. In our case the "real world" is the model of the cardiovascular system that is being built. Each entity possesses a set of attributes or properties. Here the entities and attributes of a relational schema map into the files and fields of the database. The database environment is organized by dividing the relational schema into two parts (Fig. 4.2), the <u>dictionary schema</u> and <u>cardiovascular schema</u>.

The dictionary schema is needed for various reasons, firstly it can be accessed for definitions of terms used in the relational schema, secondly for the structure of the schema such as the relationship between entities and attributes, key attributes, etc. Finally it is used to define the type and format of the terms used, whether they are character or numeric, and to how many decimal places they are represented. The dictionary holds the definitions of the entities and attributes of the entire relational schema while the cardiovascular schema holds the information and values of all the entities and attributes of circulation and neural control.

4.3.1 Entities in the Dictionary

The dictionary consists of three entities, `entity`, `attribute` and `ent_att`. Entities are all in the third normal form (3NF), defined as follows;

"An entity is in 3NF if and only if every non-key attribute is uniquely and non-transitively depended on the identifying key" (Date, 1981).

Entities are distinct objects, events or concepts, for example each blood compartment is an entity. The structure for the `entity` is illustrated below.

`entity(entname,entgroup,definition,occurrence)`
key: `entname`
The attributes for this entity are defined as:

`entname`	:	entity name
`entgroup`	:	entity group
`definition`	:	entity definition
`occurrence`	:	entity occurrence

The attribute `entname` is a unique identifying key (since it defines the entity uniquely and all the other attributes are dependent on it).

Once each entity is defined, the dependent attributes will be defined. The structure for the `attribute` is illustrated below.

`attribute(attname,type,definition)`
key: `attname`
The attributes for this entity are defined as:

`attname`	:	attribute name
`type`	:	attribute type (e.g. character, numeric logical, memo)
`definition`	:	attribute definition

The attribute `attname` is a unique identifying key (since it defines the attribute uniquely and all the other attributes are dependent on it).

The `ent_att` entity links attributes to the entities in which they occur; its structure is illustrated below.

`ent_att(entname,attname,key)`
key: `entname,attname`
The attributes for this entity are defined as:

`entname`	: entity name
`attname`	: attribute name
`key`	: entity/attribute relation (logical type)

The attributes `entname,attname` are a unique identifying key (since it defines the relation between the `ent_att` entity uniquely). For example the above `attribute` entity can be shown in the `ent_att` entity as:

entname	**attname**	**key**
attribute	attname	T
attribute	type	F
attribute	definition	F

where T stands for true and F stands for false.

Finally entity is shown below to illustrate the structure of entities in a dictionary schema in the database file (dBASEIII+ files);

Name	**type**	**decimal**
Entname	char8	
Entgroup	char16	
Definition	char40	
Occurrence	num3	0

where **type** is a type of entity used (e.g char stands for character and num stands for number) and **decimal** is number of decimal places (only for num type).

So far the structure of dictionary schema entries has been defined. The full lists of `entity,attribute` and `ent_att` are given in Appendix 1. In the next Section the circulatory schema will be defined.

4.3.2 Entities in the Cardiovascular Schema

Cardiovascular models are built up from entities representing the heart, circulation and neural control. The entities are of two types, where * is a wildcard:

*_d defines the model architecture-elements and connections.

*_p defines numerical parameter values for elements and connections, under a particular condition.

This section is divided into two parts: Section 4.3.2.1 describes the entities in the circulation schema whilst Section 4.3.2.2 describes entities in the neural control.

There are two points to mention here: (1) abbreviated names are used for each entity because this DBMS only allows a length of 8 characters for the name and thus meaningful names cannot be used; (2) there are two entities that need to be defined before defining the cardiovascular schema entities. One is the definition of the models entity `model` (model definition) and the other is the definition of the condition that the model is in (`conditio`). The `model` entity is illustrated below:

`model(model_code,name,reference)`

key: `model_code`

The attributes for this entity are defined as:

`model_code`	: code for the selected model
`name`	: name of the model
`reference`	: any reference to a selected model

The user inputs a model name and the program creates the code (code 1 has been used for the first model which was created and is the 19-segment model); the code is then incremented by one for each new model. The `conditio` (condition definition) is illustrated below:

`conditio(model_code,cond_code,cond_name)`

key: `model_code,cond_code`

The attributes for this entity are defined as:

`model_code`	: code for the selected model with this condition
`cond_code`	: code for the existing condition of the selected model
`cond_name`	: desired name for the defined condition

The user selects a desired model and then inputs a name for a created condition, the program generates a code for this condition (for the 19-segment model code 1 has been used for the normal condition since it was a first condition to be created) starting from one and incrementing for the next new condition.

4.3.2.1 Entities in the circulation

In Chapter 3 the circulatory model was developed in terms of 'arteries', 'veins', 'atria' and 'ventricles'; so the entities in the circulatory schema consist of two definition entities, four parameter entities and four flow entities (Fig. 4.3). The definition entities are compartment definition (comp_d) and flow compartment definition (f_comp_d). The parameter entities are arterial parameters (artery_p), venous parameters (vein_p), atrial parameters (atrium_p) and ventricular parameters (ventri_p). The flow parameter entities are arterial flow parameters (f_art_p), venous flow parameters (f_vein_p), atrial flow parameters (f_atri_p) and ventricular flow parameters (f_vent_p). An example of entities in the dictionary schema (entity entries) would be as follows:

Entname	artery_p
Entgroup	circulation
Definition	definition of all arterial parameters
Occurrence	10 (maximum of 10 arterial compartments)
Entname	f_art_p
Entgroup	circulation
Definition	definition of all arterial flow parameters
Occurrence	30 (maximum of 30 flow compartments)
Entname	comp_d
Entgroup	circulation
Definition	definition of all parameters
Occurrence	30 (maximum of 30 compartments)

The definition of each entity follows:

(i) `comp_d` entity. Defines all the *circulatory compartments*.

```
comp_d(model_code,comp_code,comp_type,region,index,name,anat_det,
     reference).
```

key: `model_code,comp_code`

The attributes for this entity are defined as:

`model_code`	: code for the selected model
`comp_code`	: code used for each compartment
`comp_type`	: type of the compartment
`region`	: where the compartment is located (e.g. abdomen, thorax or a freely chosen one)
`index`	: an index for the compartment
`name`	: the full name of the compartment (e.g. inferior vena cava)
`anat_det`	: holds the relevant anatomical details
`reference`	: literature reference

The attributes `Model_code` from `model` (model definition) and `comp_code` are a unique identifying codes (key). Keys are either generated by linkage between entities (e.g. `model_code` which is acquired by linkage to `model` entity), or internally by the DBMS programs (e.g. the `comp_code` is generated as A (arteries) or V(veins) or AT (atria) or VE (ventricles) followed by a number and incremented sequentially; so A1 is `comp_code` for the ascending aorta in 19-segment model), thus the user does not need to worry about them.

Each `comp_d` (compartment definition) will have a set of parameters (in *_p entities) for each condition modelled. The parameter set will depend upon the `comp_type` (artery, vein, atrium or ventricle) and the assumed mathematical equations (see Section 3.2.6 in the previous chapter). Figure 4.3 also shows the relation between `comp_d` (compartment definition) and its 4 associated *_p entities.

(ii) f_comp_d entity. Contains all the definitions of *flow from compartments.*

f_comp_d(model_code,link,from,to,descriptn)
key: model_code,link
The attributes for this entity are defined as:

model_code	: code for the selected model
link	: defines a relation between compartments
from	: from compartment
to	: to compartment
descriptn	: defines the relation that is entered by the user

Model_code and link are a unique identifying key. The model_code is extracted from the model (model definition) entity. The link is generated by the DBMS programs. The from attribute and the to attribute are extracted from the comp_d (compartment definition) entity by indexing to the comp_code attribute. For example in the 19-segment the f_comp_d (from compartment definition) entity, flow from ascending aorta can be defined as ;

f_comp_d(1,A1A2,A1,A2,ascending aorta to aortic arch link)

(iv) In the parameter entities, some of the entities are referred to equations. These are the equations in the previous chapter.

The parameter entities follows:

(iii) artery_p entity. This contains all the *arterial parameters.*

artery_p(model_code,cond_code,comp_code,i,ci,vui,ki,vio,pregion)
key: model_code,cond_code,comp_code
The attributes for this entity are defined as:

model_code	: code for the selected model
cond_code	: code for the existing condition of the selected model
comp_code	: code used for each compartment
i	: an index for the compartment
ci	: compliance
vui	: unstressed volume,
ki	: viscoelasticity
vio	: initial volume
pregion	: pressure of region

The keys are derived from the linkage to relative entities (model_code from model (model definition), cond_code from conditio (condition definition) and comp_code from comp_d (from compartment definition)). These are the keys throughout the rest of the parameter entities. The i attribute is linked to the index attribute in the comp_d (from compartment definition) and the value for it is extracted from this entity. This attribute is extracted the same way throughout all the rest of the parameter entities. The attributes ci, vui, ki, vio are extracted from Equation 3.1. The exception is pregion which comes from Equations 3.9 and 3.10. This quantity could not be defined in the flow from the arteries since it is not a pressure of the region between two compartments, but a pressure of the region of one compartment. For example

```
artery_p(1,A1,1,1,0.28,53.0,0.04,79.86140,4)
```

represents the ascending aorta in the 19-segment model with a normal condition. The program deals with the key and i attributes (explained above how this is achieved), then the user inputs the rest of the attributes (five attributes in this case).

(iv) vein_p entity. This contains all the *venous parameters*.

```
vein_p(model_code,cond_code,comp_code,i,cio,vui,alphai,vio,pregion,
     neural)
```

key: model_code, cond_code, comp_code

The attributes for this entity are defined as:

model_code	: code for the selected model
cond_code	: code for the existing condition of the selected model
comp_code	: code used for each compartment
i	: an index for the compartment
cio	: compliance
vui	: unstressed volume
alphai	: vein constant
vio	: initial volume
pregion	: pressure of region
neural	: neural effect

The attributes `cio, vui, alphai , vio` are extracted from Equations 3.2 and 3.3 in Chapter 3. The exception is `pregion` which comes from Equation 3.12.8. The `neural` attribute toggles between true and false and is reflected in Equations 3.43, 3.44. The user enters attributes `cio, vui, alphai, vio, pregion` and `neural`.

(v) `atrium_p` entity. This contains all the *atrial parameters.*

```
atrium_p(model_code,cond_code,comp_code,i,vui,ais,aid,lambda1,
         lambda2,vio, pregion)
```

key: `model_code,cond_code,comp_code`

The attributes for this entity are defined as:

`model_code`	: code for the selected model
`cond_code`	: code for the existing condition of the selected model
`comp_code`	: code used for each compartment
`i`	: an index for the compartment
`vui`	: unstressed volume
`ais`	: systolic elastance
`aid`	: diastolic elastance
`lambda1`	: constant
`lambda2`	: constant
`vio`	: initial volume
`pregion`	: pressure of region

The attributes `ais,aid` are from Equation (3.4), `lambda1, lambda2` are factors effecting T_{AS} from Section 3.2.6.1 (Chapter 3 part iv, corresponds to λ_1, λ_2). Other attributes `vio` and `vui` are from Equation (3.6) in the previous chapter. Notice from these equations that t_c (cardiac time) and T_H (heart period) are not included in this entity since these are implemented in the simulation engine (the initial T_H will be entered by the user before the simulation). The `pregion` attribute is used in the flow from arteries and flow from veins (Equations 3.10, 3.12.8). The user enters the last seven attributes.

(vi) `ventri_p` entity. This contains all the *ventricular parameters.*

```
ventri_p(model_code,cond_code,comp_code,i,vui,ais,aid,lambda1,
        lambda2,lambda3,vio,pregion)
```

key: `model_code,cond_code,comp_code`

The attributes for this entity are defined as:

`model_code`	: code for the selected model
`cond_code`	: code for the existing condition of the selected model
`comp_code`	: code used for each compartment
`i`	: an index for the compartment
`vui`	: unstressed volume,
`ais`	: systolic elastance
`aid`	: diastolic elastance
`lambda1`	: constant
`lambda2`	: constant
`lambda3`	: constant
`vio`	: initial volume
`pregion`	: pressure of region

The attributes `ais,aid` are from Equation (3.4), `lambda1` is factor effecting T_{AV} and `lambda2, lambda3` are factors effecting T_{VS} from Section 3.2.6.1 (Chapter 3 part iii corresponds to $\lambda_1, \lambda_2, \lambda_3$). Other attributes `vio` and `vui` are from Equation (3.6). Again t_c (cardiac time) and T_H (heart period) are not included in this entity since these are implemented in the simulation engine (the initial T_H will be entered by the user before the simulation). The `pregion` attribute is used in the flow from arteries and flow from veins (Equations 3.10, 3.12.8). The user enters the last eight attributes.

The parameter entities follows:

(vii) `f_art_p` entity. This contains all the *from arterial parameters.*

```
f_art_p(model_code,cond_code,link,j,i,Rji,lenji,anglji,Lji,Fji,pcc,
      neural)
```

key: `model_code,cond_code,link`

The attributes for this entity are defined as:

`model_code`	: code for the selected model
`cond_code`	: code for the existing condition of the selected model
`link`	: defines a relation between compartments
`j`	: an index for the compartment
`i`	: an index for the compartment
`Rji`	: resistance to flow
`lenji`	: effective segment length
`anglji`	: angle between the axis of the segment
`Lji`	: inertance
`Fji`	: flow
`Pcc`	: critical closing pressure
`neural`	: neural effect

`Model_code`, `cond_code` and `link` are unique identifying codes (key). These keys are extracted by the linkage to `model` (model definition), `conditio` (condition definition) and `f_comp_d` (from compartment definition) entities and are the keys used throughout the rest of flow parameter entities. The `i`, `j` are extracted by linkage to `index` attribute in `comp_d` entity. These attributes are extracted the same way throughout the rest of flow parameter entities. The attributes `lenji`, `anglji` are from Equation 3.8, `Rji`, `Lji`, `Fji` are from Equations 3.9 and 3.10. The `neural` attribute toggles between true and false and is reflected in Equations 3.46, P_{cc} is derived from Equation 3.11. The user enters the last seven attributes.

(viii) `f_vein_p` entity. This contains all the *from venous parameters*.

```
f_vein_p(model_code,cond_code,link,j,i,Rji,lenji,anglji,vui,betai,
      Fji,valve)
```

key: `model_code,cond_code,link`

The attributes for this entity are defined as:

`model_code`	: code for the selected model
`cond_code`	: code for the existing condition of the selected model
`link`	: defines a relation between compartments
`j`	: an index for the compartment
`i`	: an index for the compartment
`Rji`	: resistance to flow
`lenji`	: effective segment length
`anglji`	: angle between the axis of the segment
`vui`	: unstressed volume
`batai`	: a constant affecting the flow when flow is less that zero and no valve exists
`Fji`	: flow
`valve`	: veins valve

The attributes `lenji`, `anglji` are from Equation 3.8, `Rji`, `Lji`, `Fji`, `betai` and `valve` are from Equation 3.12.8. The attribute `valve` toggles between true and false. The user enters the last seven attributes.

(ix) `f_atri_p` entity. This contains all the *from atrial parameters*.

```
f_atri_p(model_code,cond_code,link,j,i,Rji,Fji)
```

key: `model_code,cond_code,link`

The attributes for this entity are defined as:

`model_code`	: code for the selected model
`cond_code`	: code for the existing condition of the selected model
`link`	: defines a relation between compartments
`j`	: an index for the compartment
`i`	: an index for the compartment
`Rji`	: resistance to flow
`Fji`	: flow

The attributes `Rji` and `Fji` are from Equation 3.15. The user enters the last two attributes.

(x) `f_vent_p` entity. This contains all the *from ventricular parameters.*

`f_vent_p(model_code,cond_code,link,j,i,Rji,Fji,Lji,Aji)`
key: `model_code,cond_code,link`
The attributes for this entity are defined as:

`model_code`	: code for the selected model
`cond_code`	: code for the existing condition of the selected model
`link`	: defines a relation between compartments
`j`	: an index for the compartment
`i`	: an index for the compartment
`Rji`	: resistance to flow
`Fji`	: flow
`Lji`	: inertance
`Aji`	: cross sectional area

The attributes `Rji`, `Lji`, `Fji` and `Aji` are from Equation 3.13. The user enters the last four attributes.

So far it has been shown how the model relational schema has been built for the circulatory model. If the model is uncontrolled, then values for neural control will be set to one by the simulation engine. All these relational database files are based on generic equations. However, these files are not ready to be read by the simulation engine. Programs are written in the DBMS language to transfer this information into a form of an ASCII file, which can then be read by the simulation engine.

4.3.2.2 Entities in the neural control

The Neural control schema consists of six definition entities and six parameter entities (Fig. 4.4). The six definition entities are baroreceptor definition (`baro_d`), CNS input definition (`cns_in_d`), heart rate controller definition (`heart_d`), peripheral resistance controller definition (`periph_d`), myocardial contractility controller definition (`contrc_d`), venous tone controller definition (venous_d) and the 6 parameter entities are baroreceptor parameter (baro_p), CNS input parameter (cns_in_p), heart rate controller parameter (heart_p), peripheral resistance controller parameter (periph_p), myocardial contractility controller parameter (contrc_p), venous tone controller parameter (venous_p).

Thus the total number of entities in the dictionary schema is 25 (3 dictionary, 10 circulation and 12 neural control). An example of entities in the dictionary schema (entity entries) follows:

Entname	baro_d
Entgroup	neural control
Definition	definition of all baroreceptors
Occurrence	4 (maximum of four baroreceptors)
Entname	baro_p
Entgroup	neural control
Definition	definition of all baroreceptor parameters
Occurrence	4 (maximum of four baroreceptors)

The definition entities follows:

(i) baro_d entity. This contains all the *baroreceptors definitions*.

baro_d(model_code,baro_code,comp_code,description)

key: model_code,baro_code,comp_code

The attributes for this entity are defined as:

model_code	: code for the selected model
baro_code	: code for the existing baroreceptor
comp_code	: code used for sensed compartment
descriptn	: description of the baroreceptors

Model_code (from linkage to model (model definition) entity), baro_code (generated by DBMS programs starting with B and a number and generating sequentially e.g. B1 for aortic arch baroreceptor in 19-segment model) and comp_code (from linkage to comp_d (compartment definition) entity) are unique identifying codes (key). The user enters the last attribute.

(ii) `cns_in_d` entity. This contains all the *CNS input definitions*.

`cns_in_d(model_code,baro_code,cns_in_cod,descriptn)`
key: `model_code,baro_code,cns_in_cod`
The attributes for this entity are defined as:

`model_code`	: code for the selected model
`baro_code`	: code for the existing baroreceptor
`cns_in_cod`	: code used for existing CNS input
`descriptn`	: description of the CNS input

`Model_code` (from linkage to `model` (model definition) entity), `baro_code` (from linkage to `baro_d` (baroreceptor definition) entity) and `cns_in_cod` (generated by DBMS programs starting with C and a number and generating sequentially e.g. C1 for aortic arch CNS input in 19-segment model) are unique identifying codes (key). The user enters the last attribute.

(iii) `heart_d` entity. This contains all the *heart rate control definition.*

`heart_d(model_code,descriptn)`
key: `model_code`
The attributes for this entity are defined as:

`model_code`	: code for the selected model
`descriptn`	: description of the heart rate controller

`Model_code` (from linkage to `model` (model definition) entity), is a unique identifying code (key) and is used as a key throughout the rest of controller definitions. The user enters the last attribute throughout the rest of the controller definitions.

(iv) `periph_d` entity. This contains the *peripheral resistance control definition.*

`periph_d(model_code,descriptn)`
key: `model_code`
The attributes for this entity are defined as:

`model_code`	: code for the selected model
`descriptn`	: description of the peripheral resistance controller

(v) `contrc_d` entity. This contains the *myocardial contractility control definition.*

`contrc_d(model_code,descriptn)`
key: `model_code`
The attributes for this entity are defined as:

`model_code`	: code for the selected model
`descriptn`	: description of the myocardial contractility controller

(vi) `venous_d` entity. This contains the *venous tone control definition.*

`venous_d(model_code,descriptn)`
key: `model_code`
The attributes for this entity are defined as:

`model_code`	: code for the selected model
`descriptn`	: description of the venous tone controller

So far the definition entities are defined. At this stage the parameter entities will be defined.

(vii) baro_p entity. This contains all the *baroreceptor parameters*.

```
baro_p(model_code,cond_code,baro_code,i,x1,x2,slow_time,
      fast_time,gain1,gain2,thrsh_pres)
```

key: model_code,cond_code,baro_code

The attributes for this entity are defined as:

model_code	: code for the selected model
cond_code	: code for the existing condition of the selected model
baro_code	: code for the selected baroreceptor
i	: an index for the sensed compartment
x1	: state variable
x2	: state variable
slow_time	: time through the first filter with a long time constant
fast_time	: time through the second filter with a short time constant
gain1	: contribution of each baroreceptor over one cardiac cycle
gain2	: average contribution of the positive pressure derivative over one cardiac cycle
thrsh_pres	: threshold pressure

Model_code (from linkage to model (model definition) entity) cond_code (from linkage to conditio (condition definition) entity) and baro_code (from linkage to baro_d (baroreceptor definition) entity) are unique identifying codes (key). The attributes i is extracted by linkage to index attribute in comp_d (compartment definition) entity, slow_time and fast_time correspond to t_1 and t_2 in Equations 3.18 and 3.19, gain1 and gain2 correspond to a_1 and a_2 in Equation 3.20, thrsh_pres corresponds to P_{Ti} in Equation 3.20. The user enters the last seven attributes.

(viii) cns_in_p entity. This contains all the *CNS input parameters*.

cns_in_p(model_code,cond_code,cns_in_cod,control,contributn)
key: model_code,cond_code,cns_in_cod
The attributes for this entity are defined as:

model_code	: code for the selected model
cond_code	: code for the existing condition of the selected model
cns_in_cod	: code for the selected CNS input
control	: defines which control is used (e.g. heart, peripheral, myocardial, venous)
contributn	: contribution value of any of baroreceptors

Model_code (from linkage to model (model definition) entity) cond_code (from linkage to conditio (condition definition) entity) and cns_in_cod (from linkage to cns_in_d (CNS input definition) entity) are unique identifying codes (key). The first two will be used for the rest of controller parameter entities. The control attribute is generated by DBMS for relevant controller. The contributn related to a_b in Equations 3.21, 3.30, 3.35, 3.38 for relevant control (e.g the 19-segment model has two baroreceptors thus the heart rate controller will have two contributions for CNS input a_1=0.3 and a_2=0.7 where a_1 is the contribution of aortic arch and a_2 is the contribution of carotid sinus baroreceptors). The user enters the last attribute.

(ix) `heart_p` entity. This contains all the *heart rate parameters*.

```
heart_p(model_code,cond_code,BHRT,x3,x5,x6,T1,T2,slow_time,fast_time,
        THmax,THmin,gain1,gain2)
```

key: `model_code,cond_code`

The attributes for this entity are defined as:

`model_code`	: code for the selected model
`cond_code`	: code for the existing condition of the selected model
`BHRT`	: threshold value
`x3`	: state variables
`x5`	: state variables
`x6`	: state variables
`T1`	: time constant for increasing pressures
`T2`	: time constant for decreasing pressures
`slow_time`	: time through the first filter with a long time constant
`fast_time`	: time through the second filter with a short time constant
`THmax`	: maximum value for heart period
`THmin`	: minimum value for heart period
`gain1`	: constant
`gain2`	: constant

The attribute `BHRT` is used in Equations 3.22, 3.25; `x3, x5, x6` are used in Equations 3.24, 3.26, 3.27; `T1,T2` correspond to t_1, t_2 in Equation 3.23; `slow_time, fast_time` correspond to t_3 and t_4 in Equation 3.26, 3.27; `THmax, THmin` are from Equation 3.29, finally `gain1` and `gain2` relate to b_1 and b_2 in Equations 3.22 and 3.28 respectively. The user enters the last 12 attributes.

(x) periph_p entity. This contains all the *peripheral resistance parameters.*

periph_p(model_code,cond_code,BPRT,X2,X3,slow_time,fast_time,PRTmax,
PRTmin,gain1)

key: model_code,cond_code

The attributes for this entity are defined as:

model_code	: code for the selected model
cond_code	: code for the existing condition of the selected model
BPRT	: threshold value
x2	: state variables
x3	: state variables
slow_time	: time through the first filter with a long time constant
fast_time	: time through the second filter with a short time constant
PRTmax	: maximum resistances produced by reflex
PRTmin	: minimum resistances produced by reflex
gain1	: constant

The attribute BPRT is used in Equation 3.31; x2, x3 are used in Equations 3.32, 3.33; slow_time, fast_time correspond to t_1 and t_2 in Equation 3.32, 3.33, PRTmax, PRTmin are from Equation 3.31 and finally gain1 relates to b_3 in Equation 3.34. The user enters the last eight attributes.

(xi) contrc_p entity. This contains all the *myocardial contractility parameters.*

contrc_p(model_code,cond_code,BMCT,X2,T,MCTmax,MCTmin)

key: model_code,cond_code

The attributes for this entity are defined as:

model_code	: code for the selected model
cond_code	: code for the existing condition of the selected model
BMCT	: threshold value
x2	: state variables
T	: time constant
MCTmax	: maximum contractility produce by reflex
MCTmin	: minimum contractility produce by reflex

The attribute BMCT used in Equation 3.36; x2 is used in Equation 3.36; T corresponds to t_1 in Equation 3.37, finally MCTmax, MCTmin are from Equation 3.37. The user enters the last five attributes.

(xii) venous_p entity. This contains all the *venous tone parameters*.

venous_p(model_code,cond_code,BVVT,X2,T,VTTmax,VTTmin,gain1, gain2)
key: model_code,cond_code
The attributes for this entity are defined as:

model_code	: code for the selected model
cond_code	: code for the existing condition of the selected model
BVVT	: threshold value
x2	: state variables
T	: time constant
VTTmax	: maximum venous tone produce by reflex
VTTmin	: minimum venous tone produce by reflex
gain1	: constant
gain2	: constant

The attribute BVVT is used in Equations 3.39; x2 is used in Equation 3.40; T corresponds to t_1 in Equation 3.40; VTTmax, VTTmin are from Equation 3.39, finally gain1 and gain2 relate to cgain and vugain in Equations 3.41, 3.42 respectively. The user enters the last seven attributes.

Thus it has been shown how the relational schema was created, on what basis and how it was transferred into simulation definition files.

4.4 SIMULATION ENGINE

The simulation engine is written in Pascal. One of the advantages of the Pascal language is its ability to define and use complex data structures, as explained in Chapter 2. The general approach to simulation is depicted in Fig 4.5. The entities could be put in a record, and so for each entity there exists a record in the Pascal program. Thus there are eight circulation (eight *_p entities) records and six neural control (six *-p entities) records.

Two important points to mention here. One is in the arterial pressure equation, where the rate of change of volume is required and thus the pressures in other compartments (atrium, ventricle, vein) are calculated before calculating the arterial pressure. This is because the rate of change of volume from ventricular flow is needed. The second point is that the neural control is calculated after the circulation since we need the arterial pressure and the rate of change of pressure for the baroreceptors.

To design a simulation engine, three steps are considered (Fig. 4.5). These steps are as follows:

(1) The first step is the **model procedure** which states that:

dX_1/dt = gains-losses (law of conservation of mass).

In this step the state variables are implemented as follows:

(i) Representing dV_i/dt into the simulation engine;

For the rate of change of volume (dV_i/dt) (Equation 3.16 in Chapter 3)

```
DV[i] := DV[i]-F[j,i]          (for compartment i)
DV[j] := DV[j]+F[j,i]          (for compartment j).
```

(ii) Representing dP_i/dt into the simulation engine;

The rate of change of arterial pressure (dP_i/dt) (Equation 3.19 in Chapter 3) can be calculated by the following procedures:

From Equation 3.1

$dP_i/dt = (dV_i/dt)/C_i+(K_i/C_i)d^2V_i/dt^2/dt$ if $V_i \geq V_{ui}$

For the second rate of change of volume (d^2V_i/dt^2) which is needed in the above equation, the following statements are derived from Equation 3.16 Chapter 3:

$$d^2V_i/dt^2 = \sum_k dFik/dt - \sum_j dFji/dt \qquad Vi \geq 0$$

and thus d^2V_i/dt^2 is calculated as follows in the simulation:

```
D2V[i] := D2V[i]-DF[j,i]          (for compartment i)
D2V[j] := D2V[j]+DF[j,i]          (for compartment j)
```

where DF[j,i] is the rate of change of flow from compartment i to j and is available.

(iii) Representing dx_1/dt in the simulation engine;

For the rate of change of state variable (dx1/dt) which is needed in Equation 3.23 Chapter 3, the following statements are given:

From Equation 3.22 Chapter 3,

$dx_1/dt = 0$ if $B_{HR} \leq B_{HRT}$

$dx_1/dt = b_1(dB_{HR}/dt)$ if $B_{HR} > B_{HRT}$

Now, from Equation 3.21 Chapter 3,

$$dB_{HR}/dt = \sum_{b=1}^{nbar} a_b dB_b/dt$$

where dB_b/dt can be calculated from Equation 3.20 Chapter 3,

$doutput/dt = a_1[(dx_1/dt)+a_2 dx_2/dt]$

$dB_b/dt = doutput/dt$ if output > 0

$dB_b/dt = 0$ if output ≤ 0

Then representing these quantities in the simulation engine,

`DB[b] := a1*[DX1+a2*DX2]` (for baroreceptor b)

where `DX1` and `DX2` correspond to dx_1/dt and dx_2/dt in Equations 3.18 and 3.19 in Chapter 3.

```
DBHR:=0;
For b:=1 to nbar do  (where nbar is number of baroreceptors used)
    DBHR := a[b]B[b]+DBHR;
```

Notice that a[b] is contribution of each baroreceptor in Equation 3.20 Chapter 3.

(2) The second step is the **update procedure** which states that

$X(t+h) := X(t) + g(dX/dt,h)$

Euler's method (will be explained in Section 4.5) is used for the integration of state variables.

In this step all the state variables get updated.

For volume (V)

`V[i] := V[i]+tstep*DV[i]` (for all compartments i)

Where `tstep` is the integration step shown as **h** in the above statement.

For flow (F)

`F[j,i] := F[j,i]+tstep*DF[j,i]` (for all flow from i to j)

Since the integration of flow is needed for stroke volume in Chapter 3 Section 3.3.1.2:

```
    integF[j,i] := integF[j,i]+tstep*F[j,i]  (for all flow from i to j)
    For all neural control state variables
    x1 := x1+tstep*Dx1
    x2 := x2+tstep*Dx2
    etc
For the cardiac time
    tcardiac := tcardiac+tstep
```

(3) The third step is the **main procedure**;

```
Set cardiac time and real time to zero (tcardiac := 0;treal := 0)
Set THlast to the heart period value (THlast := TH)
Call the model procedure explained above (step 1) (modelprocedure)
Repeat
    Repeat
    Call update procedure explained above (step 2) (updateprocedure)
    Call model procedure explained above (step 1) (modelprocedure)
    increase real time by integration step unit (treal := treal+integstep)
    Write numerical results (writenumerialresults)
    Display graphical results (displaygraphicalresults)
    UNTIL cardiac time > heart period (UNTIL tcardiac > THlast)
Reset cardiac time for the next cycle (tcardiac := 0)
Until real time = simulation time (UNTIL treal:=simulationtime)
```

4.5 EULER'S METHOD

Euler's method (see Mccann, R. (1982) states that if the slope of a polynomial is equal to f, then the slope of the polynomial through the $\{y_{n+i}\}$ will be equal to $f(y_n,t_n)$, where y_n is the computed approximation to $y(t_n)$.

If a straight line approximation through points y_n and y_{n+1} is used, the approximation equation becomes

$$(y_{n+1}-y_n)/(t_{n+1}-t_n) = f(y_n,t_n). \qquad (4.1)$$

This method is called Euler's method. It is usually written as

$(y_{n+1}) = y_n + hf(y_n,t_n)$, Where $h = t_{n+1} - t_n$ (4.2)

An alternative approach to numerical integration is discussed in Chapter 8.

4.6 CONCLUSION

This chapter has presented the motivation for, and the design of, a database-oriented architecture for the implementation of computer models. A relational schema for circulatory and neural control models has been described extensively, making use of the basic elements of Chapter 3. Finally, the general form for the simulation engine is described as well as the integration routine in the simulation. The next chapter describes the 'PULSE' software which is based on the work presented here.

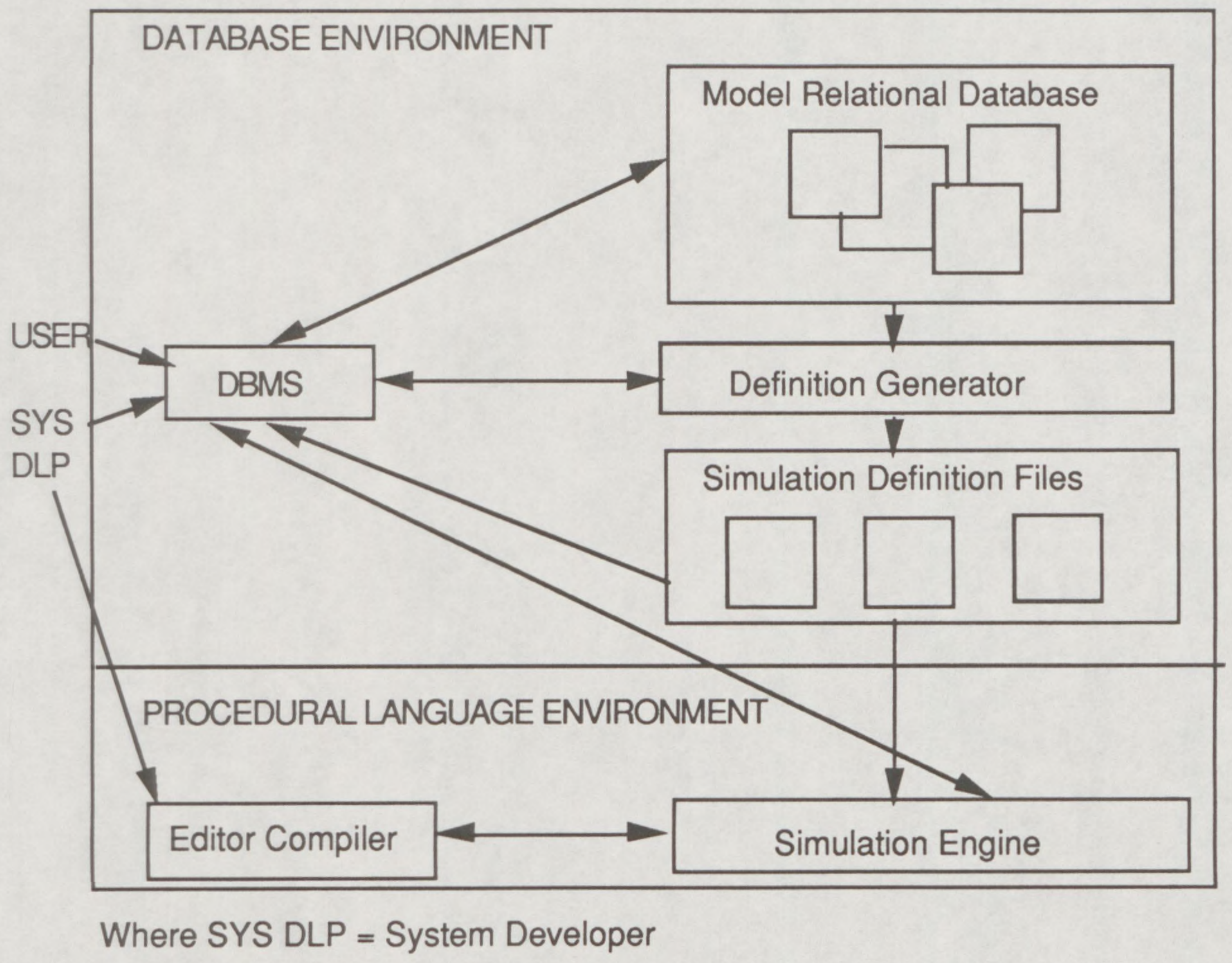

DATABASE ENVIRONMENT
Model Relational Database
USER
SYS
DLP
DBMS
Definition Generator
Simulation Definition Files
PROCEDURAL LANGUAGE ENVIRONMENT
Editor Compiler
Simulation Engine
Where SYS DLP = System Developer

Figure 4.1 **Database-oriented architecture for computer models**

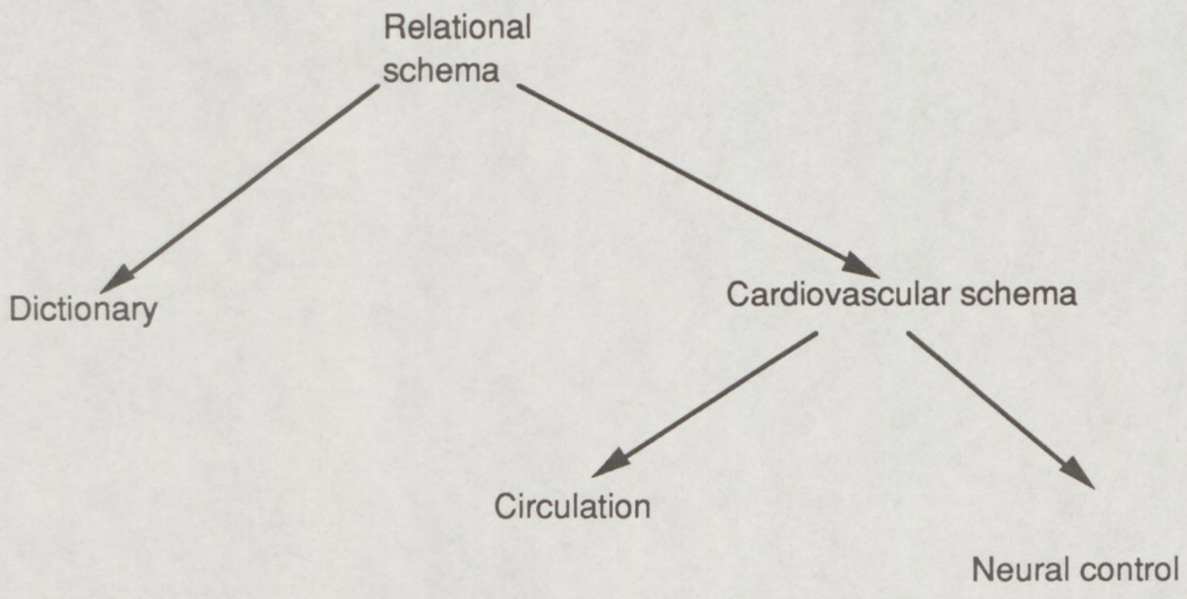


Figure 4.2 **Organisation of the relational schema**

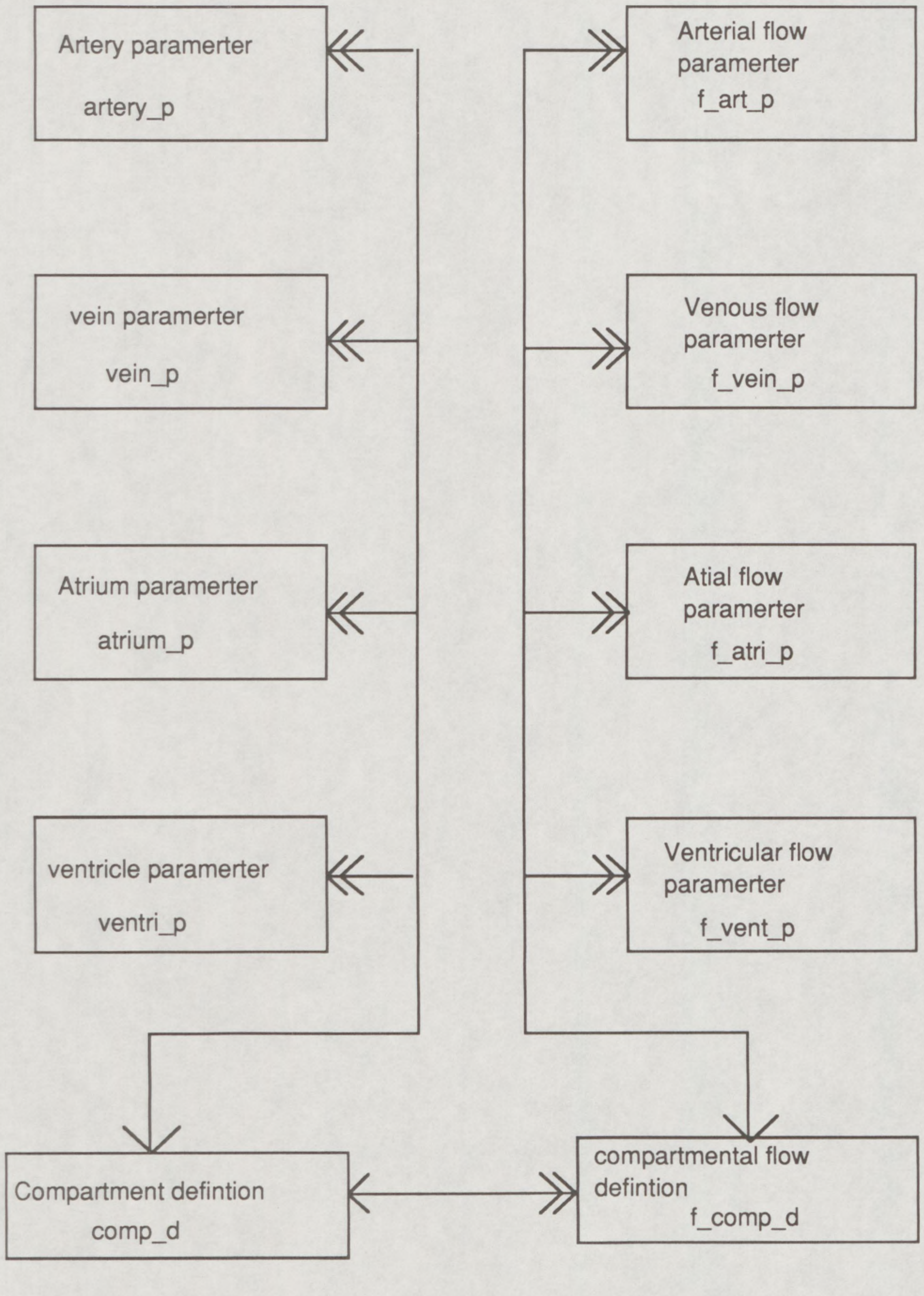


Figure 4.3 **Relational schema for circulatory models**

Notice that A $\twoheadleftarrow\!\!\rightarrow$ B is read as A has one B, but B has many As

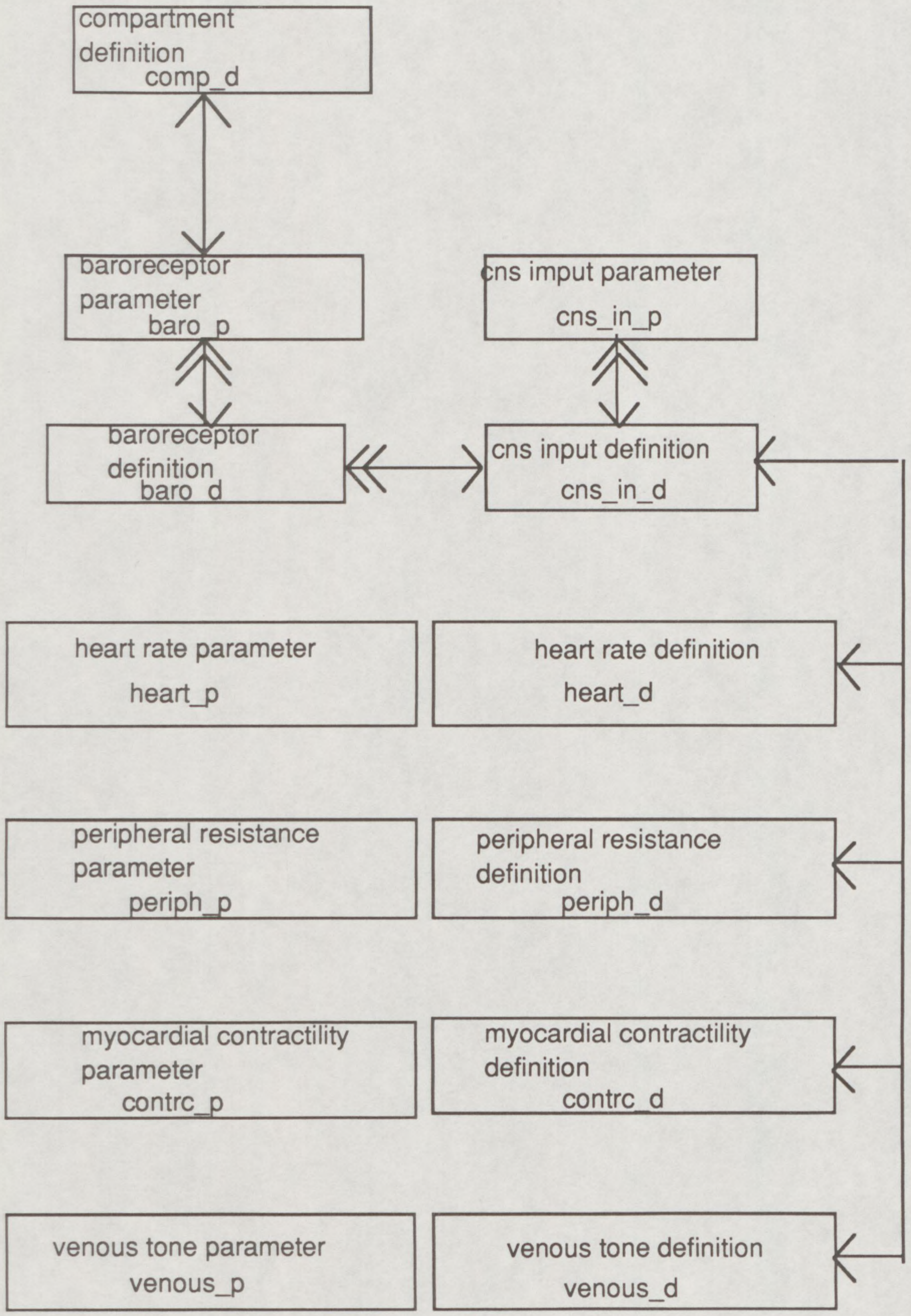

compartment definition comp_d
baroreceptor parameter baro_p
baroreceptor definition baro_d
cns imput parameter cns_in_p
cns input definition cns_in_d
heart rate parameter heart_p
heart rate definition heart_d
peripheral resistance parameter periph_p
peripheral resistance definition periph_d
myocardial contractility parameter contrc_p
myocardial contractility definition contrc_d
venous tone parameter venous_p
venous tone definition venous_d

Figure 4.4 **Relational schema for neural control models**

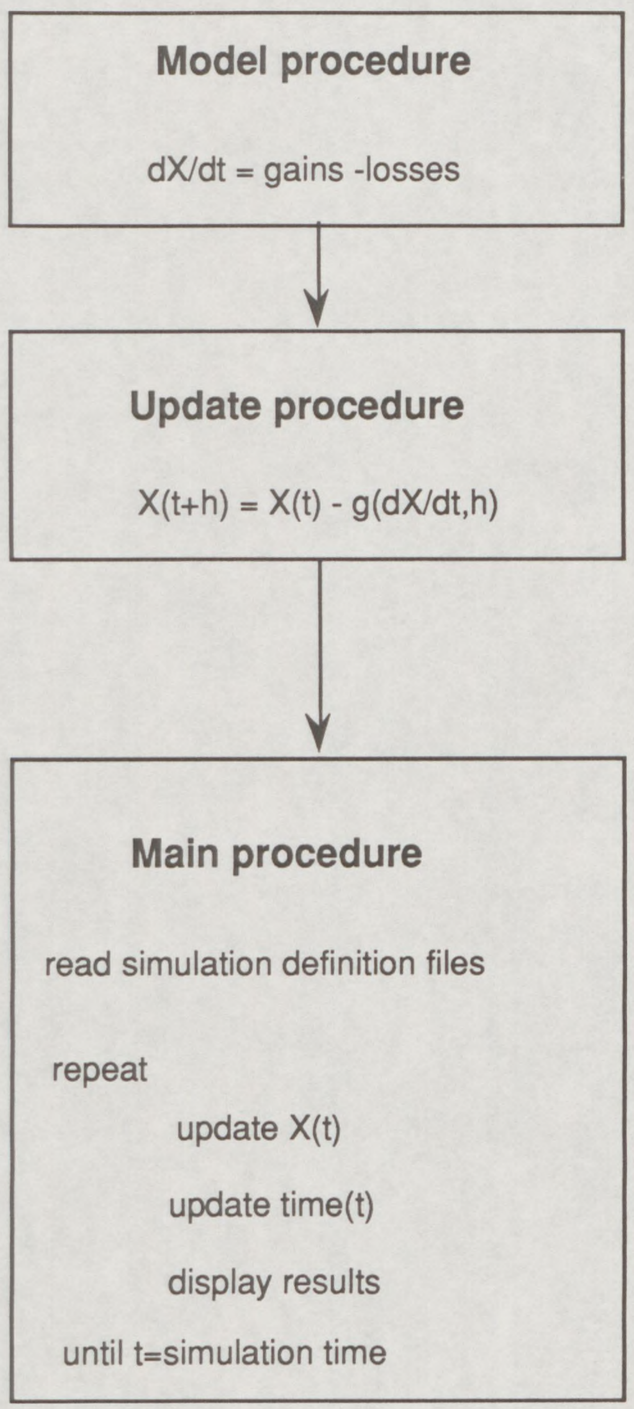


Figure 4.5 **General form of the simulation engine**

CHAPTER 5

DEVELOPMENT OF THE SYSTEM

5.1 INTRODUCTION

In the previous chapter, a database-oriented architecture for computer models of the pulsatile circulation and neural control was presented. This chapter discusses how the software 'PULSE' was developed. It begins by discussing the approach to software development which was taken. The choice of the software and hardware for implementation will then be discussed.

The 'PULSE' package will then be described in detail, followed by the function of each database program. The 19-segment model is used as an example throughout this chapter. This is a 19-segment model built by Beneken and De Wit (1967) and reported by Pullen (1976) and Leaning et al. (1983). For sake of avoiding repetition the references will not be quoted.

5.2 APPROACH TO SOFTWARE DEVELOPMENT

As mentioned earlier, the scientific language FORTRAN which is used in conventional simulation was not an ideal choice as a single software environment because of its limitation in terms of lack of flexible data structures, and the slow compilation time of PC compilers (discussed in detail in Chapter 2). An artificial intelligence approach using logic programming (PROLOG) was not chosen since these symbolic languages are not recommended for arithmetic problems because they are too slow for numerical calculation (e.g. Prolog was adopted by Leaning and Nicolosi (1986) in their MODEL system, which can represent compartmental physiological systems at conceptual, and symbolic, quantitative levels). So the final choice was to use an intermediate approach in which the emphasis would be on model-building, rather than knowledge representation. The choice was to use a database system coupled to a good procedural language with graphics support and a fast compiler. As mentioned earlier Pascal was the best choice for the latter.

5.2.1 Choice of Software

The database management system is Ashton-Tate's dBASEIII+. This is an upgraded version of dBASEIII with improved performance. It is easier to use and at the same time more powerful and secure (i.e. it protects data to an impressive degree). It is also a well-established user-friendly package and very widely used. Furthermore it fits adequately into relational analysis (Date, 1986). It is, by no means perfect, but it suited the purpose.

The procedural language is Turbo Pascal by Borland (Version 5). This has the advantages of being compatible with dBASEIII+ and has a very fast and efficient compiler and it compiles rapidly to a compact, fast object-code. It also has full-screen editing facilities backed by a complete range of "tool boxes" (e.g. graphics) that give most of the programming tools needed. As well as these advantages, it has a hierarchical record type structure which fits into dBASEIII+ entities perfectly and thus makes the simulation engine more compact and efficient.

5.2.2 Choice of Hardware

These programs have been designed to run on microcomputers; specifically IBM PCs and compatibles. It was decided to use microcomputers in preference to mini- or main-frame computers. This decision was based on 4 factors:

(i) Cost - Micro computers cost far less than larger computers, and they are far more easily obtained.

(ii) Portability - Floppy discs allow for the programs to be transferred easily to computers at different locations.

(iii) Speed - Even though microcomputers generally execute less instructions per second, they are usually used by a single user at any one time. Thus, speed problems associated with time-sharing characteristic of the use of Mainframe computers does not arise.

(iv) Intimidation factor - Since microcomputers are commonplace more people without a background in computing have used them, or at least seen them; basically they are familiar and potentially user-friendly.

These programs have been designed to be as user-friendly as possible.

5.3 'PULSE'

'PULSE' is a link between dBASEIII+ and the Pascal program. Several programs are written in DBMS to develop a user friendly environment for the user. First 'PULSE' is discussed in terms of its menus. Each menu has a number on the right hand side of the screen. The first menu is a ***copyright*** menu, the second menu is a ***main*** menu and the third menu is a ***closing*** menu warning the user to backup their disks. The ***main*** menu will be discussed here.

The ***main*** menu of 'PULSE' is shown in Fig. 5.1 (menu number 2). It consists of 8 entries plus one exit option. The bottom menu bar indicates the current model with its condition. If the model has not been selected, the user gets a prompt indicating that the model and related condition should be selected.

Option 1 is the ***set-up*** menu (Fig. 5.2, menu number 2.1). This option allows the user to <u>select</u> a model or a condition; <u>create</u> a model or a condition and <u>delete</u> a model or a condition. The data will be added or updated by the user in the `model` (model definition) and the `conditio` (condition definition) files (explained in the Chapter 4).

Option 2 is the ***update/display structure*** menu (Fig 5.3, menu number 2.2). For this option the user at least should have selected a desired model. The option consists of 12 entries plus one exit option. The entries are divided into 3 classes (compartments, flow, neural control). For each class 4 options are given, such as <u>list all</u> (to list the desired entry), <u>add</u> (to add a desired entry), <u>edit</u> (to edit a desired entry) and <u>delete</u> (to delete a desired entry). Figures 5.4, 5.5 and 5.6 are examples of <u>list all compartments</u> , <u>list all flows</u> and <u>list all baroreceptors,</u> definition option for the 19-segment model.

Option 3 is the ***update/display parameter*** menu (Fig 5.7, menu number 2.3). For this option the user should at least have selected a desired model with a condition. The option consists of 9 entries plus one exit option. The entries are divided into 3 classes (compartments, flow, neural control). For each class 3 options are given, such as <u>list all</u> (to list the desired entry), <u>add</u> (to add a desired entry), <u>edit</u> (to edit a desired entry). Notice that the user cannot delete a parameter from a file since to delete an entry the user has to select the ***update/display structure*** option. This is because it is important to delete the compartment and flows from the structure file first and then to delete the entry from parameter files. This option cannot be used if the structure of the entry is not defined. Figures 5.8, 5.9 and 5.10 are examples of list all compartments, list all flows and list all baroreceptors, parameter option for the 19-segment model.

Option 4 is the ***display equations*** menu (Fig. 5.11, menu number 2.4). The option displays the circulatory and the neural control equations. These equations are explicit and defined in Chapter 3. The equations are saved in two text files called `result1.equ` and `result2.equ` respectively (See Appendix 2 for a listing of these files).

Option 5 is the ***simulate*** menu (menu number 2.5). This option is used to simulate the selected model and condition. The option checks whether any floating compartment exists (any compartment with no link, flow coming to or flow going from; discontinuity in the model). If there are any floating compartments then it will produce a message indicating that "Floating compartment(s) exist, do you want to continue with the simulation?". If the answer is no then it stops the simulation, otherwise it continues. After an acceptable reply the program continues and prompts the user for the desired requirement (Fig. 5.12 top). If i) some variables have already been defined then the option of ***select***, ***delete*** and ***add*** a variable will be given to the user for any changes (Fig. 5.12 bottom). If the option is select then it selects it for the simulation (with * in front to indicate this, Fig. 5.12). If the option is ***add*** then the incorporated variable menu will be displayed (Fig. 5.13 top) followed by the limit menu (Fig. 5.13 bottom). However, if ii) no output has been defined then the incorporated variable menu will be displayed (Fig. 5.13). The program then scans the database and extracts the required data and saves it into ASCII files (Figs. 5.14 (circulation), 5.15 (neural control top) and 5.15 (desired outputs bottom) show samples of an ASCII file for the 19-segment model) which are then read by the simulation engine to produce graphical and numerical simulations.

Option 6 is the ***dictionary*** menu (Fig. 5.16 top, menu number 2.6). This options lists `entity`, `attribute` and `ent_att` files in dictionary, circulation and neural control (explained extensively in Chapter 4). Figures 5.16 bottom, 5.17 top and 5.17 bottom show an example of each class.

Option 7 is the ***sample*** menu (Fig. 5.18, menu number 2.7). This options provides a sample of results of the package for the 19-segment model. The model is in a steady state and its features are explained in detail in the next chapter. The results are displayed in both graphical and numerical form (Fig. 5.19 shows the facility to scan the numerical result file).

Option 8 is the ***help*** menu (Fig. 5.20). This provides help facilities on how to use the package.

5.4 DATABASE PROGRAMS

'PULSE' consists of one directory called ***PULSE***. This directory contains 9 sub-directories shown in Fig. 5.21.

The sub-directories will be defined later but as well as the following sub-directories, the ***PULSE*** directory contains programs and files illustrated in Fig. 5.22.

The program <u>initial</u> is used to initialise all the variables in dBASEIII+ programs. The program <u>main</u> runs all the other programs in the entire package. The program <u>exit</u> exits the user from 'PULSE' and displays a message warning the user to back-up their works and reset the variables. The program <u>harvey.com</u> is a Pascal program which is compiled and simulates the package. The files `model1.def`, `model2.def`, `model3.def` are ASCII files (called Simulation Definition Files earlier) which are written by the 'PULSE' package. The files `result1.res`, `result2.res`, `result3.res` are text files which are written by Pascal programs containing initial values for circulation, neural control and simulation output respectively. The `result4.res` is a text file containing the numerical simulation result and `result5.res` is the text file containing the last values of all state variables such as volume, flows, neural control's state variables, integP (refer to Chapter 3, Section 3.2.6.1, Part i), integF (refer to Chapter 3, Section 3.3.1.2) for future simulation. The rest of the files in this directory are required for execution of the Pascal programs. At this stage the sub-directories of ***PULSE*** will be discussed.

The ***FILES*** directory contains all the database files with related indices. These are the files explained in Chapter 4. The main files are the `conditio` (condition definition) a file containing different conditions for different or the same model, `model` file containing different model. For the circulation `comp_d`, f_comp_d are used for the definition of compartments and flows; artery_p, vein_p, atrium_p and ventri_p for arterial, venous, atria and ventricular parameters; f_art_p, f_vein_p, f_atri_p and f_vent_p for from arterial, venous, atria and ventricular flow parameters. For the neural control baro_d, cns_in_d, heart_d, periph_d, contrac_d and venous_d are used for definition of baroreceptors, cns input, heart rate, peripheral resistance, myocardial contractility and venous tone; baro_p, cns_in_p, heart_p, periph_p, contrac_p and venous_p are used for parameters of baroreceptors, cns input, heart rate, peripheral resistance, myocardial contractility and venous tone. Most files are indexed by keys or by another attribute. As an example for indexed files, model is indexed by model_code attribute, comp_d is indexed by comptype, compcode and index attributes; etc. Thus files are indexed for the role they have in the programs.

The ***SET-UP*** directory contains all the programs written for the ***set-up*** menu. The flow of the main part of the directory chart is depicted in Fig. 5.23. The selectmo selects and displays the model on the screen. The selectco selects and displays the condition. The createmo will allow the users to create a new model; it will give them the option of copying from the previous model or to build it from scratch. If the choice is to copy from the previous model then the choice of the condition that the model is in will also be given. The createco will allow the users to create a new condition and will also give them the option of copying from an old condition. The deletemo will delete a model from a database. This is a drastic option since all the entries for the model will be deleted. The deleteco will delete a condition for the selected model from a database. This option will not work if the model is not selected. It is, however, very secure in terms of making sure that the user knows what condition of what model is being deleted.

The ***UPDATEST*** directory contains all the programs written for the ***update/display structure*** menu. The flow chart of the main part of the directory is depicted in Fig. 5.24. For this option to work the users need to select a model first (see above). The listallc program will list all the compartments of the selected model. The addcomp program will allow the user to add compartments to the selected model. The editcomp program will allow the user to edit compartments from the selected model. The delcomp program will allow the user to delete compartments from the selected model.

This option also deletes all the flows from and to the deleted compartments and consequently all parameters of the deleted compartments and flow parameters relating to the deleted compartments. The listallf program will list all the flows of the selected model. The addflow program will allow the user to add flows to the selected model. The editflow program will allow the user to edit flows from the selected model. The delflow program will allow the user to delete flows from the selected model. This option also deletes all the correspondent flow parameters of the deleted compartments. The listbaro program will list all the baroreceptors, cns inputs, heart rate, peripheral resistance, myocardial contractility and venous tone of the selected model. The addbaro program will allow the user to add baroreceptors, cns inputs, heart rate, peripheral resistance, myocardial contractility and venous tone to the selected model. The editbaro program will allow the user to edit baroreceptors, cns inputs, heart rate, peripheral resistance, myocardial contractility and venous tone from the selected model. The delbaro program will allow the user to delete baroreceptors, cns inputs, heart rate, peripheral resistance, myocardial contractility and venous tone to the selected model. This option also deletes all the corresponding parameters or related entities (e.g. if a baroreceptor is deleted then the cns input will be deleted also).

The ***UPDATEPA*** directory contains all the programs written for the ***update/display parameter*** menu. The flow chart of the main part of the directory is depicted in Fig. 5.25. For this option to work the users need to select a model and condition first (see above). The listpar program will list all the parameters for the selected option (artery, vein, atrium and ventricle) of the selected model with the condition. The addpar program will allow the user to add parameters for the chosen compartments to the selected model with the condition. The editpar program will allow the user to edit parameters of compartments from the selected model with the condition. The listfpar program will list all the flow parameters of the selected model with the condition. The addfpar program will allow the user to add flow parameters to the selected model with the condition. The editfpar program will allow the user to edit flow parameters from the selected model with the condition. The listbarp program will list all the baroreceptors, cns inputs, heart rate, peripheral resistance, myocardial contractility and venous tone parameters of the selected model with the condition. The addbarp program will allow the user to add baroreceptors, cns inputs, heart rate, peripheral resistance, myocardial contractility and venous tone flow parameters to the selected model with the condition. The editbarp program will allow the user to edit baroreceptors, cns inputs, heart rate, peripheral resistance, myocardial contractility and venous tone flow parameters from the selected model with the condition. To delete compartments, flows or neural control parameters the previous directory menu (***update/display structure***) should be chosen. Also this option will not work if the compartments, flows and neural control entries are not defined in ***update/display structure*** menu (see above).

The ***DISPLAY*** directory contains all the programs written for the ***display equations*** menu. The flow chart is depicted in Fig. 5.26. The aim of this directory is to list the generic equations of circulation and neural control. The bcom.com is a compiled program. The idea is generated from the IBM browsing facility and adapted here to browse the files `result1.equ` and `result2.equ`. The programs equa1.pas and equa2.pas are written in Pascal to produce the above files for circulatory and neural control equations.

The ***SIMULATE*** directory contains all the programs written for the ***simulate*** menu. The flow chart of the main part of the directory is depicted in Fig. 5.27. For this option to work the users need to select a model and condition first (see above). The execu1 program asks the user a selection of questions related to simulation (e.g. if neural control is true/false). The execu2 program asks the user for selected variables to be chosen for the simulation. The addvar program adds, deletes or selects already defined variables (Section 5.3 option 5). The execut1 program will execute the circulatory output to `model1.def` (ASCII file). The execut2 program will execute the neural control output to `model2.def` (ASCII file). The execut3 program will execute the simulation output to `model3.def` (ASCII file). The last three files are Simulation Definition Files. The floating program checks to see if there are any floating compartments in the selected model, (such as no flow from or to a compartment) and asks the user if they would like to continue or not.

The ***DICTION*** directory contains all the programs written for the ***diction*** menu. The flow chart of the main part of the directory is depicted in Fig. 5.28. The aim of this menu is to consult the dictionary for any entity or attribute which is not clear in the circulation or neural control to the user. It consists of three programs. The modelatt program will access a file `attribute` discussed in Chapter 4, defining all the attributes in the entire database. The <u>modelent</u> program will access a file `entity` discussed in Chapter 4, defining all the entities in the entire database. The <u>modelena</u> program will access a file `ent_att` (entity/attribute relation) discussed in Chapter 4, defining the relation between the attributes and the entities in the relational database.

The ***SAMPLE*** directory contains all the text files used for the ***sample*** menu. It contains the `result5.res` file (a file containing a steady state variables for 19-segment model) and the simulation definition files for the 19-segment model (`model1.def`, `model2.def` and `model3.def`).

The ***HELP*** directory contains the programs written for the ***help*** menu. This consists of several text menus written for the user in using the package 'PULSE'.

Finally, it is inconvenience to include the listing of all the DBMS programs (about 100 programs). Thus a listing of these programs are omitted from the thesis.

5.5 PASCAL PROGRAM

So far the database part of the package has been defined. As explained earlier, the simulation engine is written in Pascal (`harvey.pas` see Appendix 3 for listing of this program) where the compiled version is incorporated in the package for efficiency. The program consists of 17 records where 14 of these records are mentioned in Chapter 4 Section 4.4. The other 3 records are:

(1) `meascomp` which measures compartment quantity such as pressure.
(2) `measfcomp` which measures from compartment quantity such as flow.
(3) `measothers` which measures other quantities such as stroke volume.

The Pascal program consists of several procedures. The main ones are discussed below:

(i) `ReadNewState` procedure
Reads state variable values.

(ii) `OutputOptions` procedure

Reads what the desired user option for output display is and writes the values into the result4.txt file.

(iii) `InitialNeural` procedure

Reads all the neural files and initialises the state variables x_1, x_2, x_3, etc.

(iv) `InitialCirculation` procedure

Reads all the circulatory files and writes the initial values into a `result1.txt` file.

(v) `HeartControl` procedure

If neural control is chosen then writes the initial values to a `result2.txt` file else sets all the neural variables in the circulation to 1.

(vi) `UpdateHeartTime` procedure

Initialises & updates the heart chambers period through the program T_{AS}, T_{AV}, T_{VS}.

(vii) `ModelEquations` procedure

Reads all the circulatory files and resets all the state variables; volumes, etc also reads all the neural controls and resets all the state variables; x_1, etc.

(viii) `Integ` procedure

Integrates volumes, flow and neural control state variables using Euler's methods (**update procedure** explained earlier in Chapter 4).

(ix) `TextWritten` procedure

Writes text in windows.

(x) `MakeScreen` procedure

Draws window graphics.

(xi) `SimulateNumFile1, 2` procedures

Writes files for numerical simulation.

(xii) SimulGraphFile procedure

Draws graphical simulation.

(xiii) valuesaved procedure

Saves the last values for the volume and flow just before leaving the simulation so that it could be used for the next simulation. (e.g. Valsalva manoeuvre, see chapter 7 Section 7.3 for the definition of this term).

The rest of the procedures are self explanatory.

5.6 PROCEDURE FOR BUILDING A MODEL

The database environment and the procedural language environment have been discussed. At this stage the procedure for building a model will be outlined.

The simulation engine does not need user interaction.

(i) Make an entry in the model file.

(ii) Make an entry in the condition file.

(iii) Create a set of circulation compartments in the comp_d (compartment definition) file.

(iv) Make a link between the compartments in the f_comp_d (flow compartment definition) file.

(v) Fill in the parameters in artery_p (arterial parameter), venous_p (venous parameter), atrium_p (atrial parameter) and ventri_p (ventricular parameter) files.with one entry per condition.

(vi) Fill in the flow parameters in f_art_p (arterial flow parameter), f_vein_p (venous flow parameter), f_atri_p (atrial flow parameter) and f_vent_p (ventricular flow parameter) files with one entry per condition.

(vi) Define the baroreceptors in the baro_d (baroreceptor definition) file.

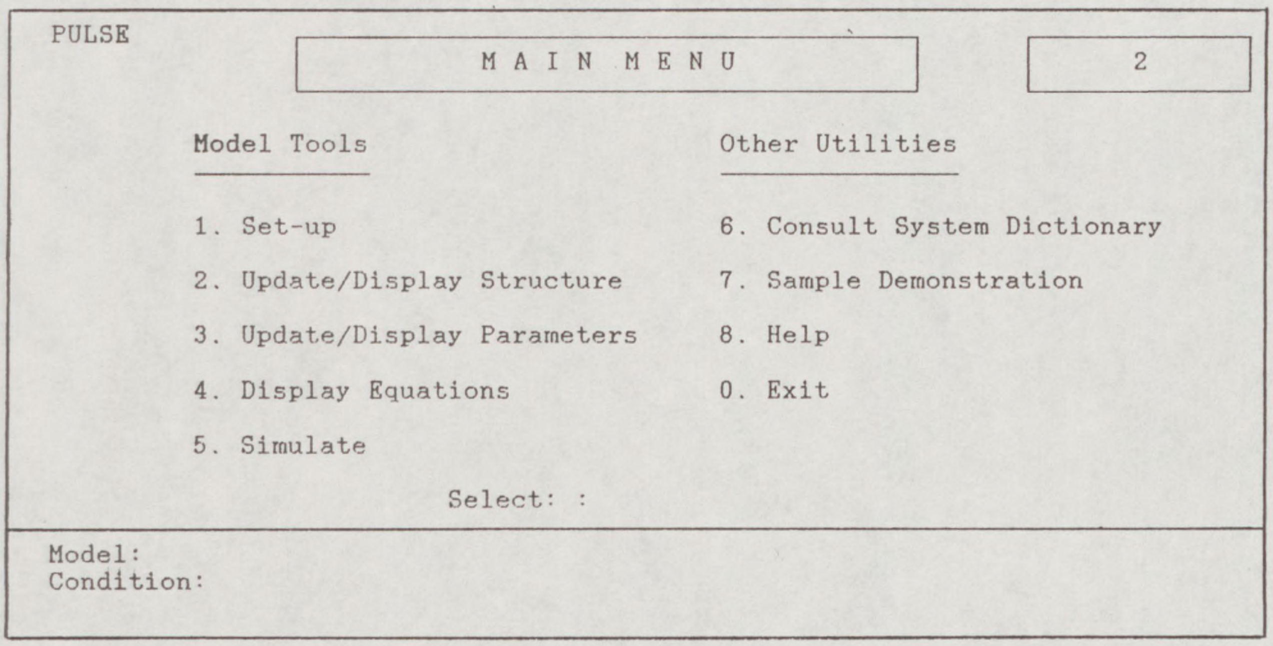
(viii) Define the cns inputs for these baroreceptors in the cns_in_d (cns input definition) file.

(ix) Define a set of controls in the `heart_d` (heart rate definition), `periph_d` (peripheral resistance definition), `contrc_d` (myocardial contractility definition) and `venous_d` (venous tone definition) files.

(x) Fill in the parameters for baroreceptors, cns input and controls in the `baro_p` (baroreceptor parameter), `cns_in_p` (cns input parameter), `heart_p` (heart rate parameter), `periph_p` (peripheral resistance parameter), `contrc_p` (myocardial contractility parameter) and `venous_p` (venous tone parameter) files.

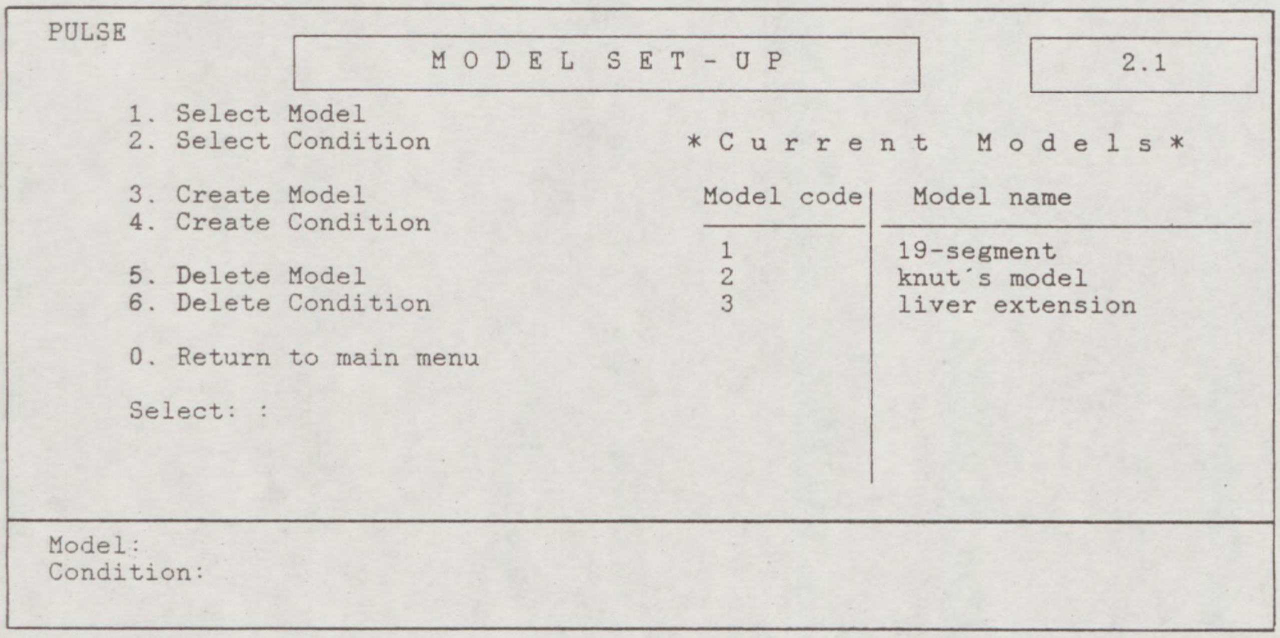
5.7 CONCLUSION

In this chapter the reasons for the choice of software and hardware have been given and the development of 'PULSE' has been described. The 'PULSE' package has been explained extensively and the aim of its directories has been explained in detail. The chapter has shown how an approach to developing a computer system for cardiovascular modelling based on relational analysis has been implemented and a full functioning system 'PULSE' is used in the following chapter to produce some cardiovascular models, each for different applications in system science and clinical research.

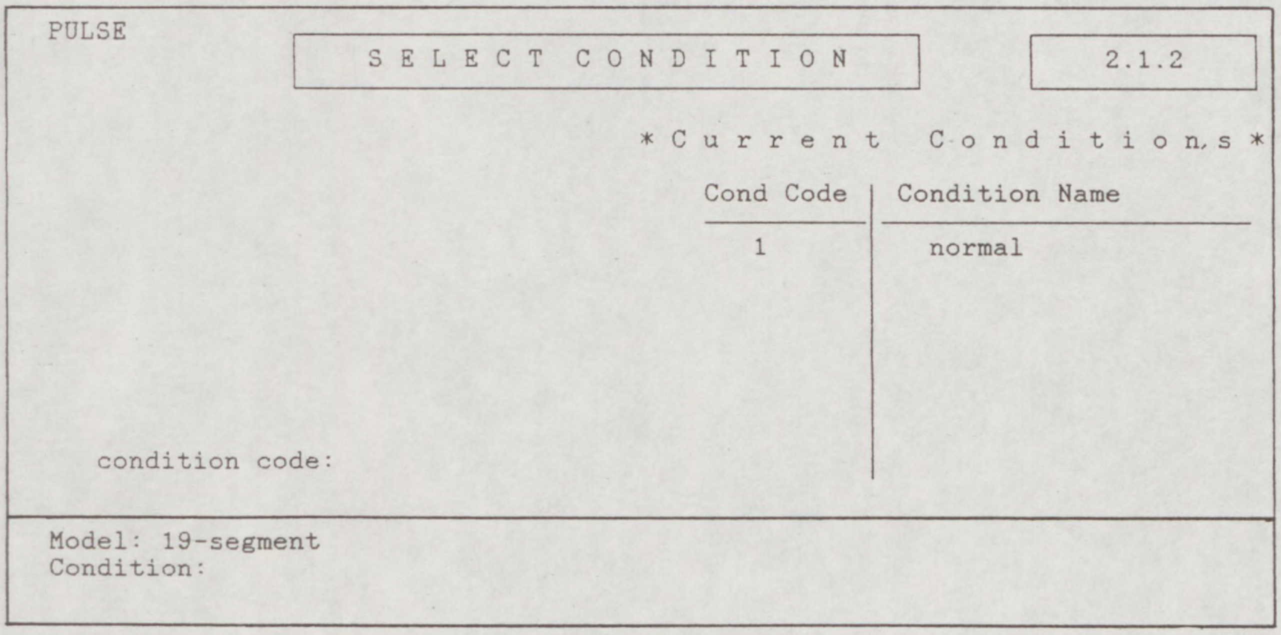


```
PULSE
                    M A I N   M E N U                           2

          Model Tools                     Other Utilities

          1. Set-up                       6. Consult System Dictionary

          2. Update/Display Structure     7. Sample Demonstration

          3. Update/Display Parameters    8. Help

          4. Display Equations            0. Exit

          5. Simulate
                          Select: :
Model:
Condition:
```

Figure 5.1 **MAIN menu**



```
PULSE
                 M O D E L   S E T - U P                    2.1

     1. Select Model
     2. Select Condition              * C u r r e n t   M o d e l s *

     3. Create Model                   Model code |  Model name
     4. Create Condition              ____________|______________________
                                          1       | 19-segment
     5. Delete Model                      2       | knut´s model
     6. Delete Condition                   3       | liver extension

     0. Return to main menu

     Select: :

Model:
Condition:
```



```
PULSE
               S E L E C T   C O N D I T I O N              2.1.2

                                   * C u r r e n t   C o n d i t i o n s *

                                      Cond Code | Condition Name
                                      __________|____________________
                                          1     |  normal



   condition code:

Model: 19-segment
Condition:
```

Figure 5.2 **SET-UP menu**

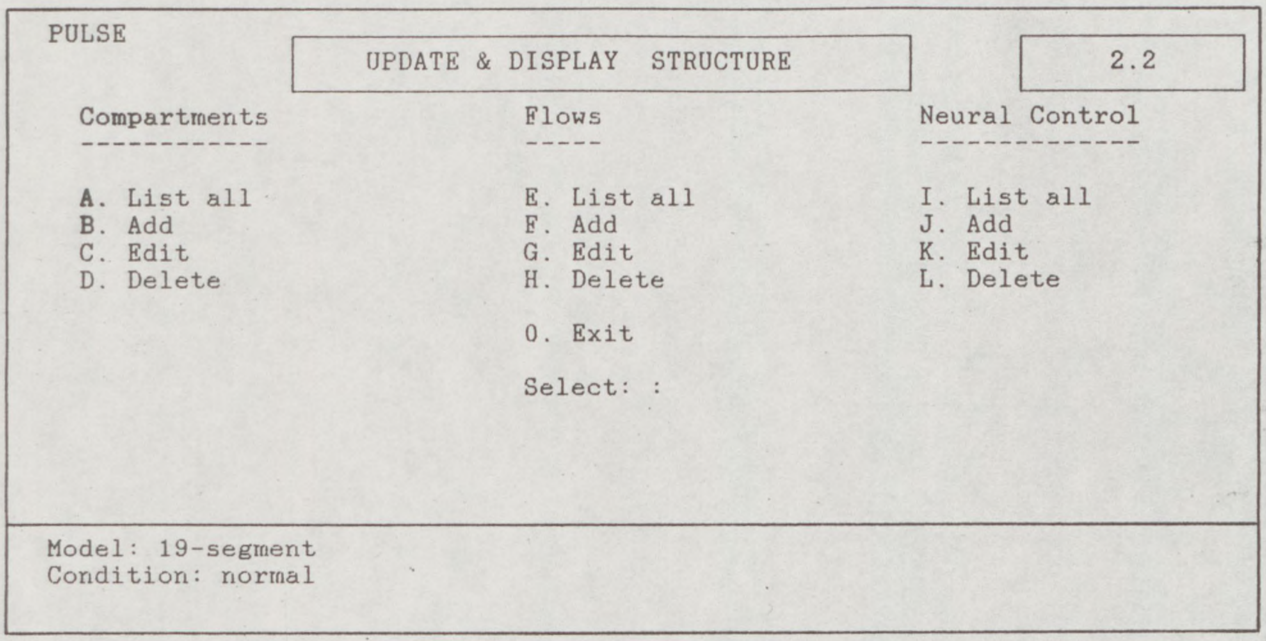
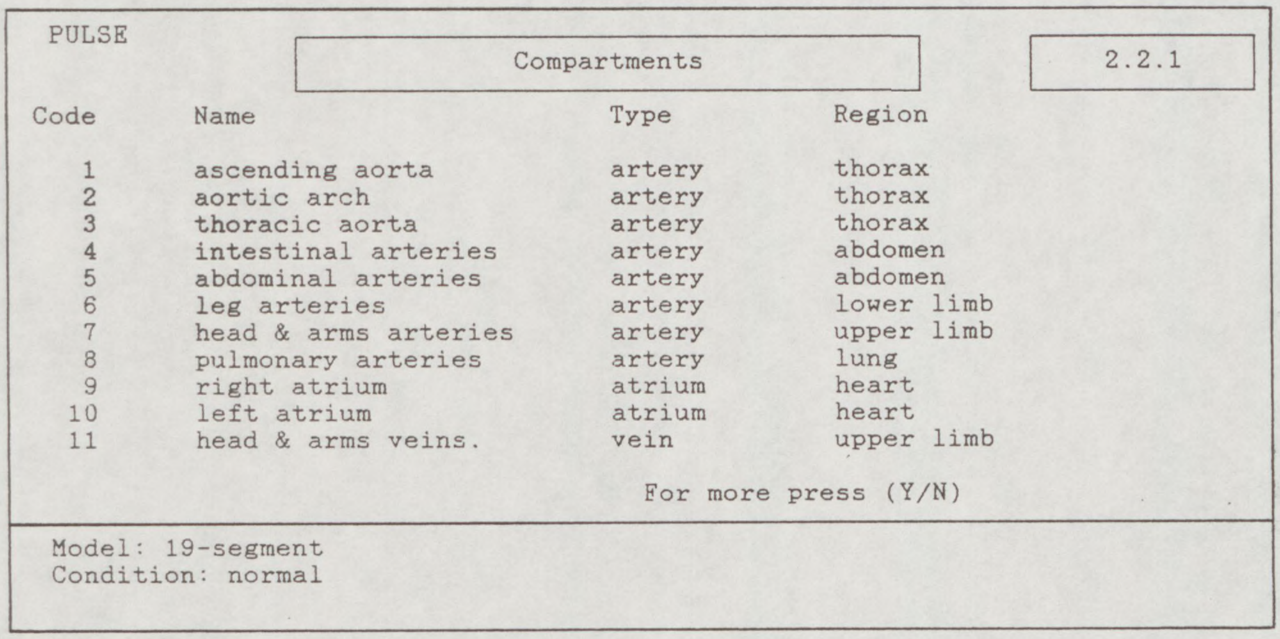
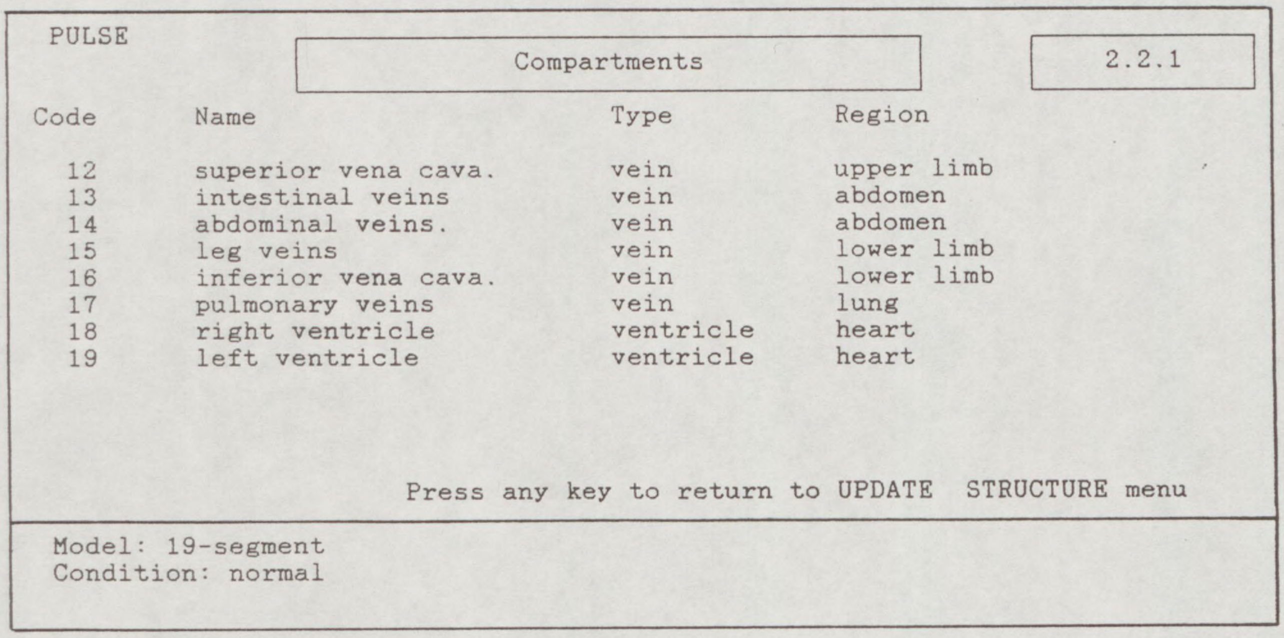

```
PULSE
                    UPDATE & DISPLAY  STRUCTURE                              2.2

  Compartments                  Flows                    Neural Control
  ------------                  -----                    --------------

  A. List all                   E. List all              I. List all
  B. Add                        F. Add                   J. Add
  C. Edit                       G. Edit                  K. Edit
  D. Delete                     H. Delete                L. Delete

                                0. Exit

                                Select: :

Model: 19-segment
Condition: normal
```

Figure 5.3 **UPDATE & DISPLAY STRUCTURE menu**

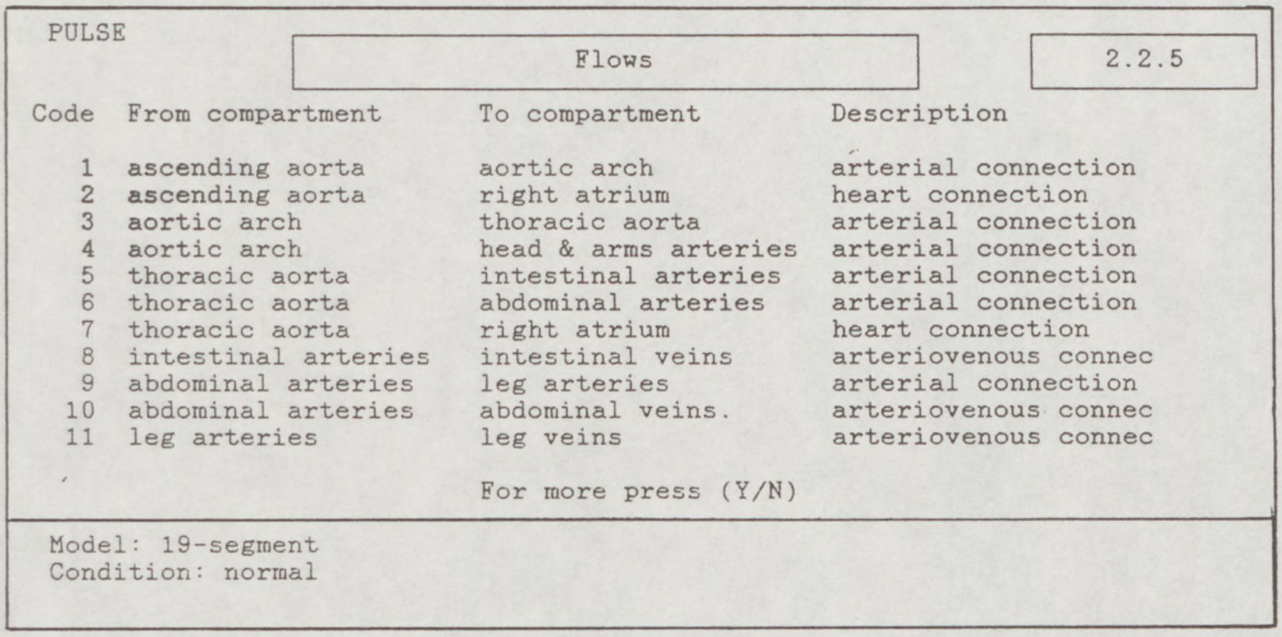


```
PULSE
                         Compartments                           2.2.1

Code      Name                          Type          Region

  1       ascending aorta               artery        thorax
  2       aortic arch                   artery        thorax
  3       thoracic aorta                artery        thorax
  4       intestinal arteries           artery        abdomen
  5       abdominal arteries            artery        abdomen
  6       leg arteries                  artery        lower limb
  7       head & arms arteries          artery        upper limb
  8       pulmonary arteries            artery        lung
  9       right atrium                  atrium        heart
 10       left atrium                   atrium        heart
 11       head & arms veins.            vein          upper limb

                                          For more press (Y/N)

Model: 19-segment
Condition: normal
```



```
PULSE
                         Compartments                           2.2.1

Code      Name                          Type          Region

 12       superior vena cava.           vein          upper limb
 13       intestinal veins              vein          abdomen
 14       abdominal veins.              vein          abdomen
 15       leg veins                     vein          lower limb
 16       inferior vena cava.           vein          lower limb
 17       pulmonary veins               vein          lung
 18       right ventricle               ventricle     heart
 19       left ventricle                ventricle     heart



                   Press any key to return to UPDATE  STRUCTURE menu

Model: 19-segment
Condition: normal
```

Figure 5.4 **List all compartments definition menu**


```
PULSE
                          Flows                               2.2.5

Code  From compartment         To compartment          Description

   1  ascending aorta          aortic arch             arterial connection
   2  ascending aorta          right atrium            heart connection
   3  aortic arch              thoracic aorta          arterial connection
   4  aortic arch              head & arms arteries    arterial connection
   5  thoracic aorta           intestinal arteries     arterial connection
   6  thoracic aorta           abdominal arteries      arterial connection
   7  thoracic aorta           right atrium            heart connection
   8  intestinal arteries      intestinal veins        arteriovenous connec
   9  abdominal arteries       leg arteries            arterial connection
  10  abdominal arteries       abdominal veins.        arteriovenous connec
  11  leg arteries             leg veins               arteriovenous connec

                               For more press (Y/N)

Model: 19-segment
Condition: normal
```

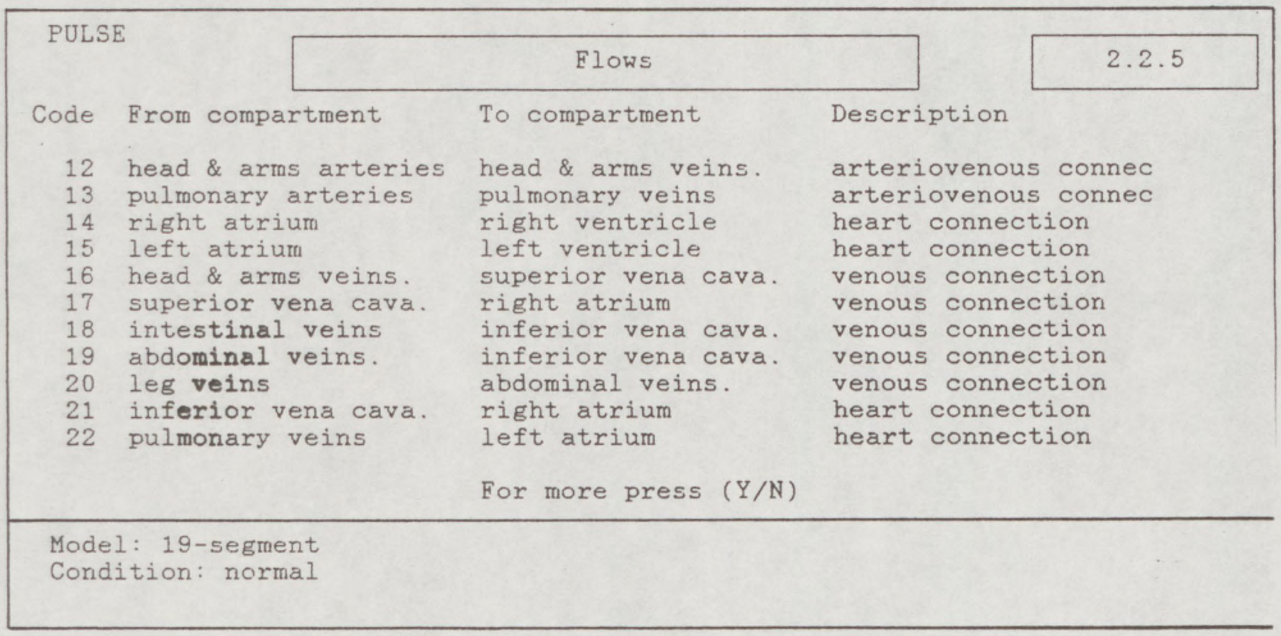
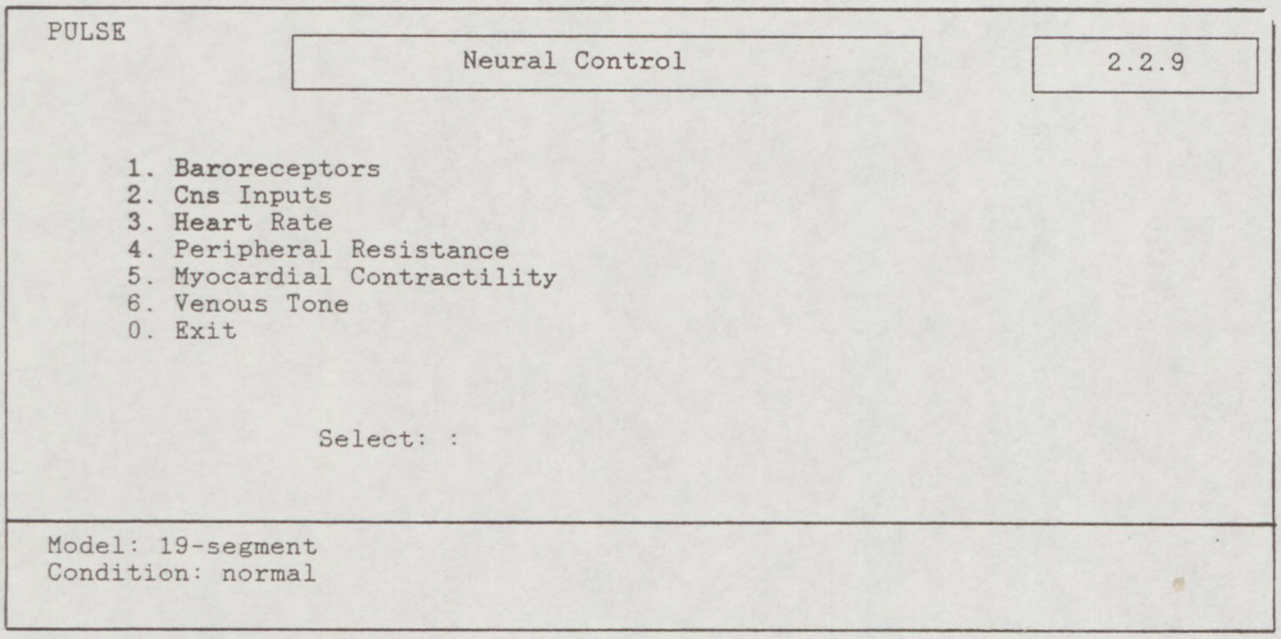

```
PULSE
                          Flows                               2.2.5

Code  From compartment         To compartment          Description

  12  head & arms arteries     head & arms veins.      arteriovenous connec
  13  pulmonary arteries       pulmonary veins         arteriovenous connec
  14  right atrium             right ventricle         heart connection
  15  left atrium              left ventricle          heart connection
  16  head & arms veins.       superior vena cava.     venous connection
  17  superior vena cava.      right atrium            venous connection
  18  intestinal veins         inferior vena cava.     venous connection
  19  abdominal veins.         inferior vena cava.     venous connection
  20  leg veins                abdominal veins.        venous connection
  21  inferior vena cava.      right atrium            heart connection
  22  pulmonary veins          left atrium             heart connection

                               For more press (Y/N)

Model: 19-segment
Condition: normal
```

Figure 5.5 **List all flows definition menu**


```
PULSE
                    Neural Control                          2.2.9

     1. Baroreceptors
     2. Cns Inputs
     3. Heart Rate
     4. Peripheral Resistance
     5. Myocardial Contractility
     6. Venous Tone
     0. Exit

                  Select: :

Model: 19-segment
Condition: normal
```

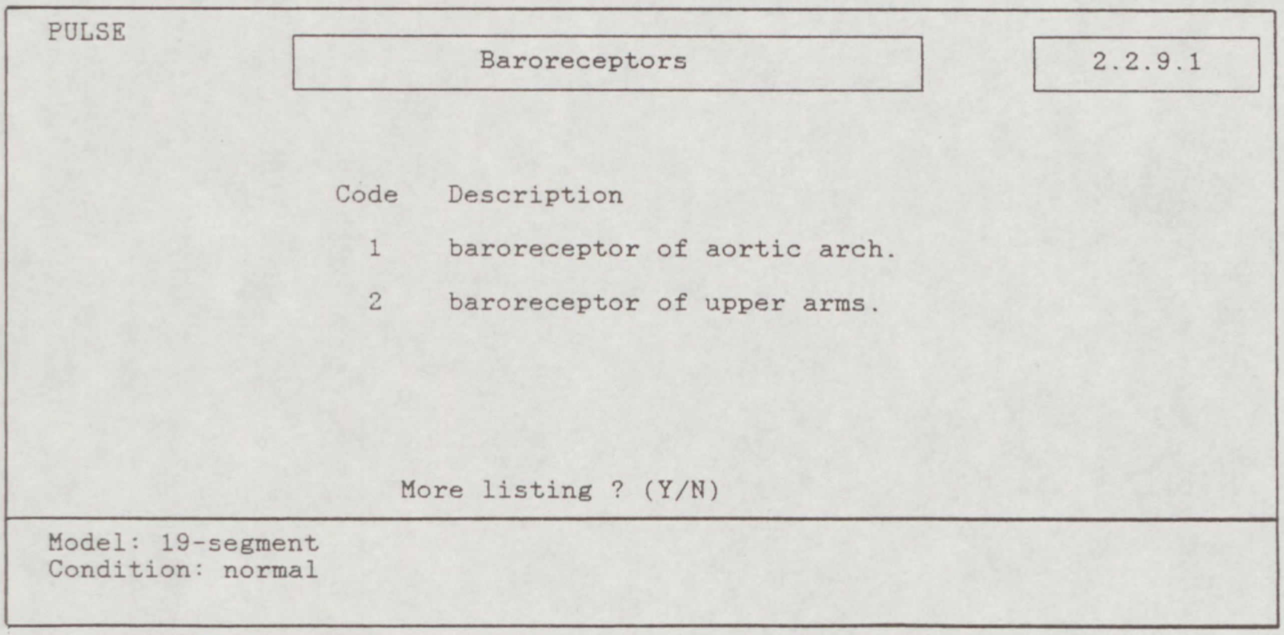
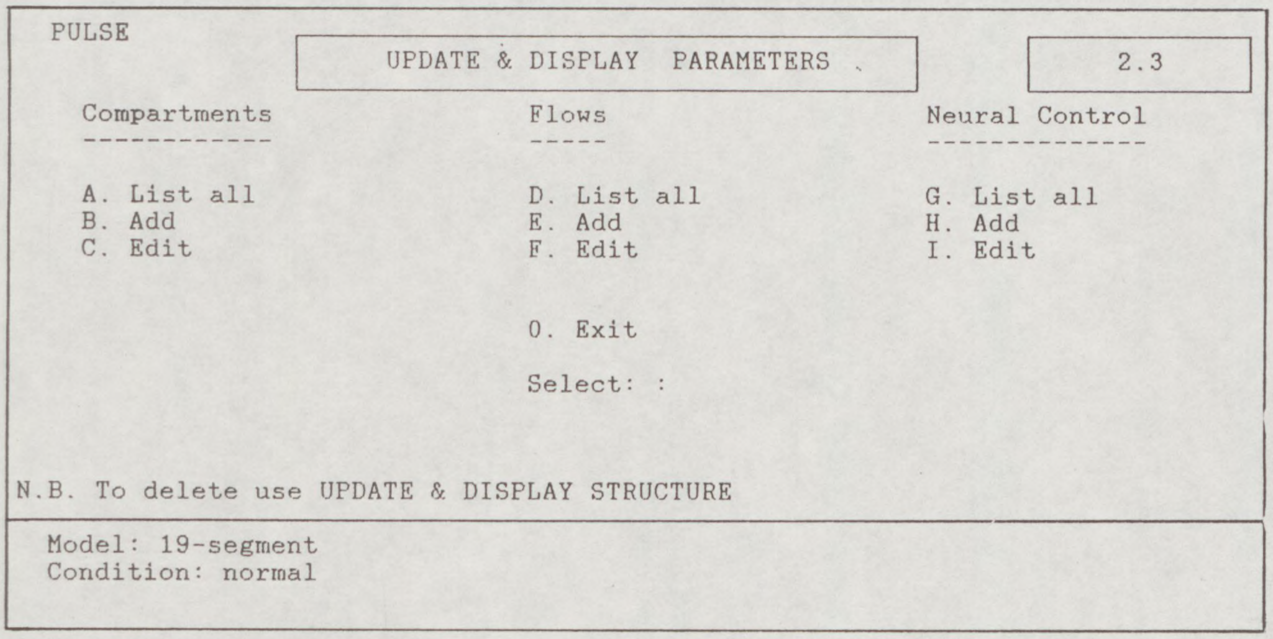
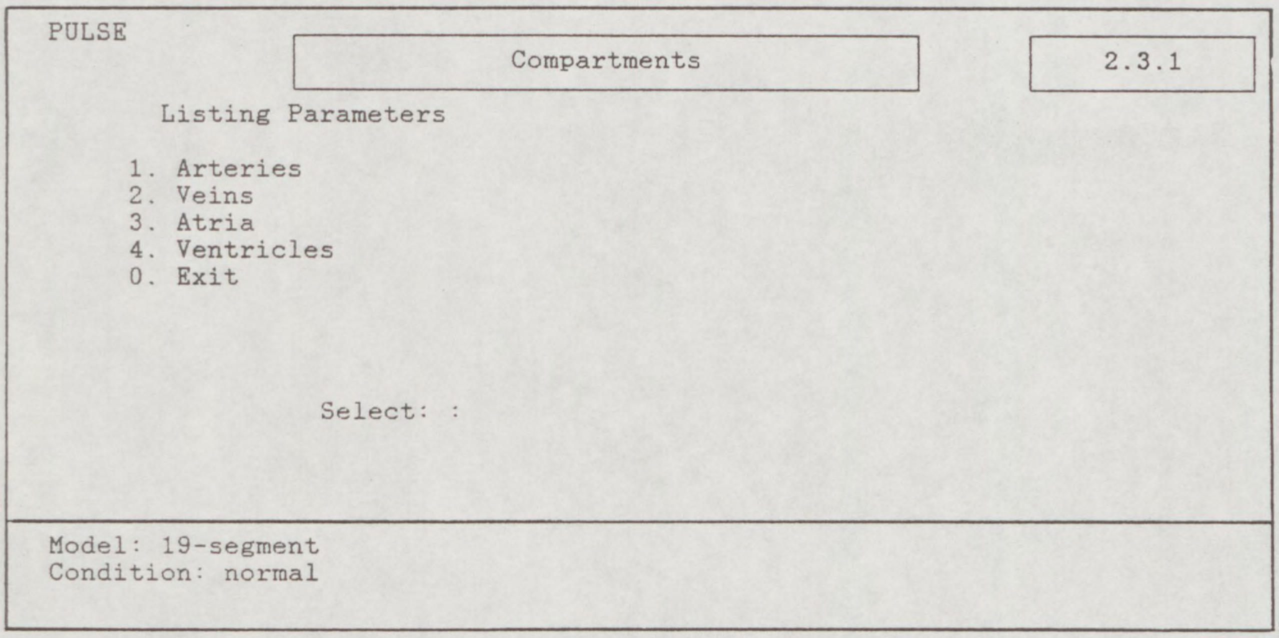

```
PULSE
                    Baroreceptors                           2.2.9.1

                   Code   Description

                     1    baroreceptor of aortic arch.

                     2    baroreceptor of upper arms.

                        More listing ? (Y/N)

Model: 19-segment
Condition: normal
```

Figure 5.6 **List all baroreceptors definition menu**



```
 PULSE
                     UPDATE & DISPLAY  PARAMETERS                 2.3

   Compartments                 Flows                    Neural Control
   ------------                 -----                    --------------

   A. List all                  D. List all              G. List all
   B. Add                       E. Add                   H. Add
   C. Edit                      F. Edit                  I. Edit


                                0. Exit

                                Select: :



N.B. To delete use UPDATE & DISPLAY STRUCTURE

 Model: 19-segment
 Condition: normal
```

Figure 5.7 **UPDATE & DISPLAY PARAMETER menu**


```
PULSE
                    Compartments                         2.3.1

     Listing Parameters

    1. Arteries
    2. Veins
    3. Atria
    4. Ventricles
    0. Exit



                 Select: :


Model: 19-segment
Condition: normal
```

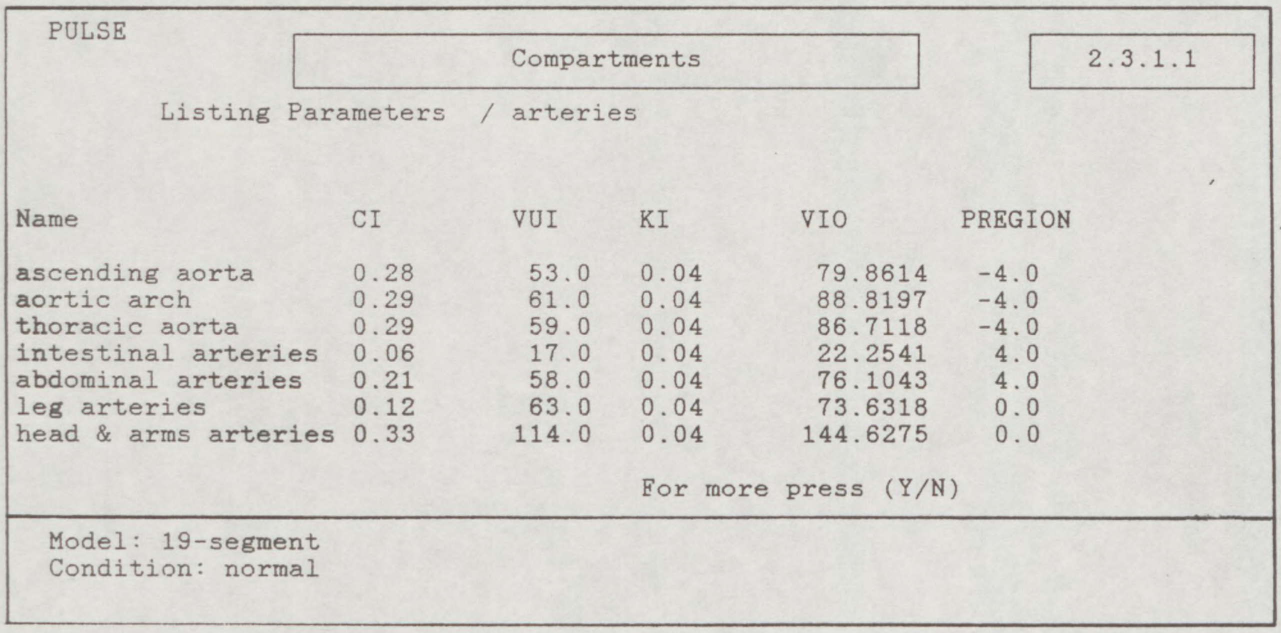
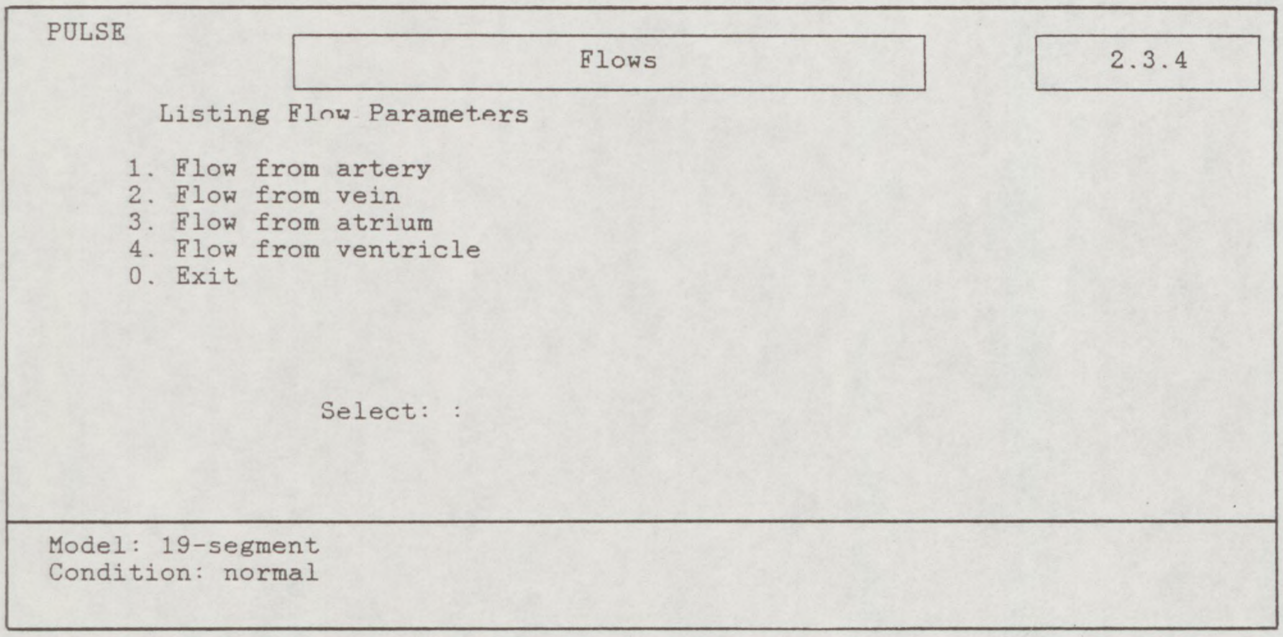
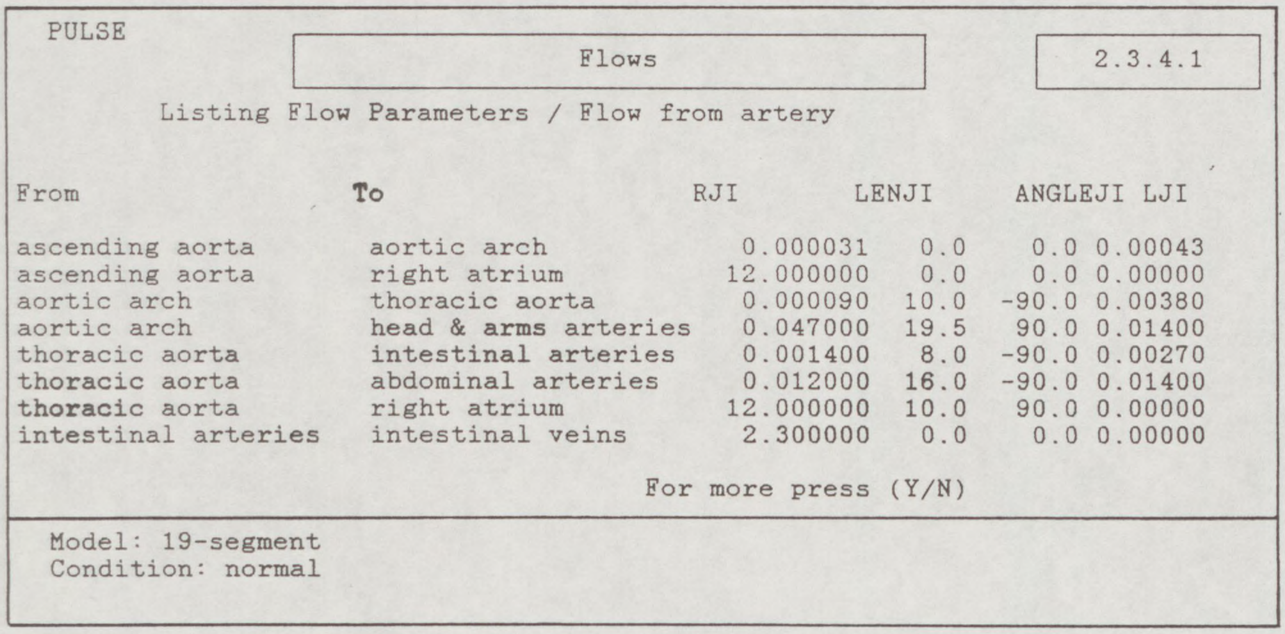

```
PULSE
                    Compartments                         2.3.1.1

     Listing Parameters   / arteries


Name                   CI          VUI      KI          VIO        PREGION

ascending aorta        0.28         53.0    0.04         79.8614    -4.0
aortic arch            0.29         61.0    0.04         88.8197    -4.0
thoracic aorta         0.29         59.0    0.04         86.7118    -4.0
intestinal arteries    0.06         17.0    0.04         22.2541     4.0
abdominal arteries     0.21         58.0    0.04         76.1043     4.0
leg arteries           0.12         63.0    0.04         73.6318     0.0
head & arms arteries   0.33        114.0    0.04        144.6275     0.0

                                         For more press (Y/N)

Model: 19-segment
Condition: normal
```

Figure 5.8 **List all compartments parameter menu**

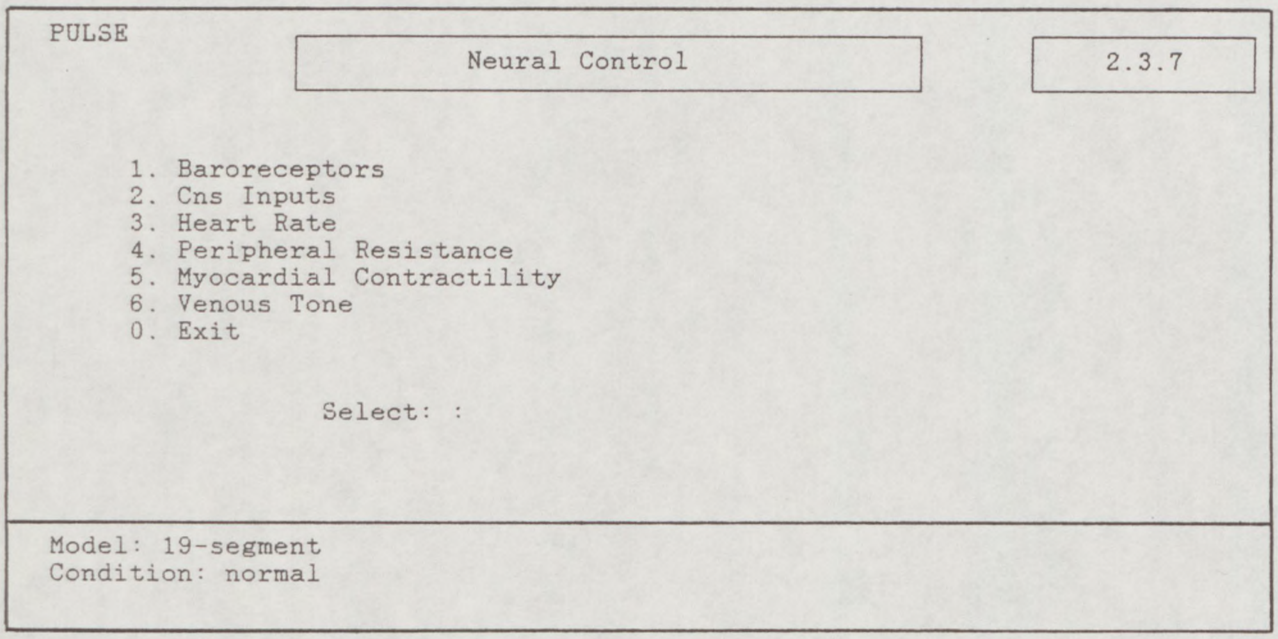


```
PULSE
                      Flows                              2.3.4

      Listing Flow Parameters

        1. Flow from artery
        2. Flow from vein
        3. Flow from atrium
        4. Flow from ventricle
        0. Exit


                  Select: :

Model: 19-segment
Condition: normal
```

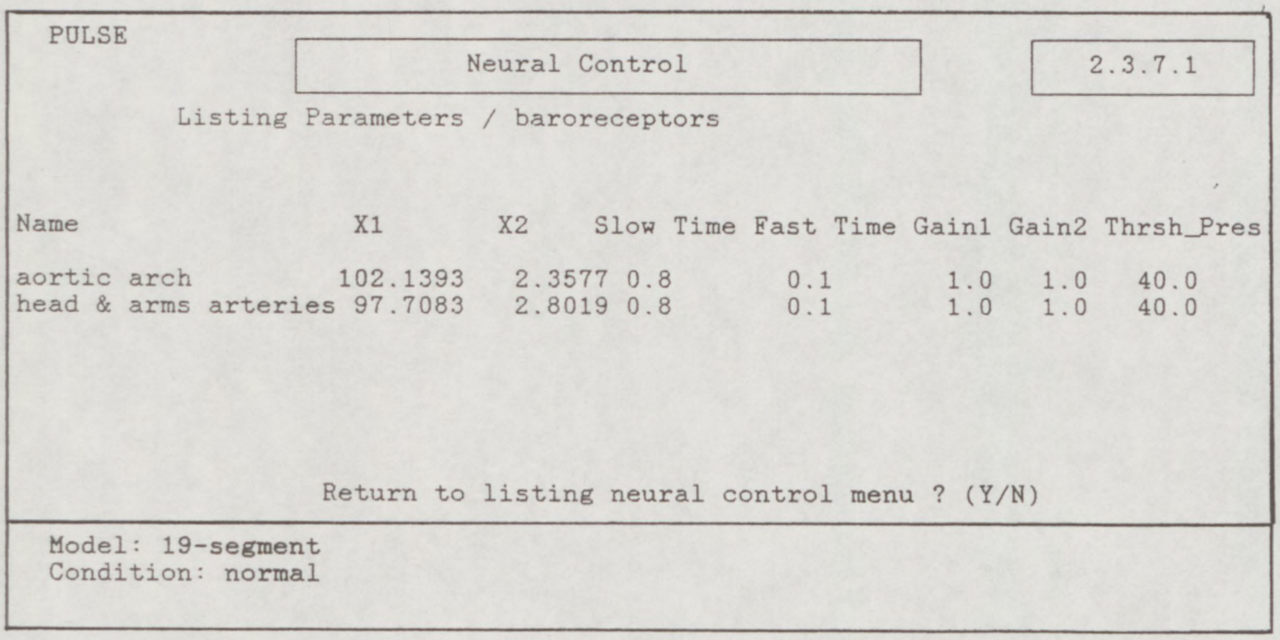


```
PULSE
                      Flows                              2.3.4.1

      Listing Flow Parameters / Flow from artery

From                 To                       RJI      LENJI     ANGLEJI LJI

ascending aorta      aortic arch             0.000031    0.0     0.0 0.00043
ascending aorta      right atrium           12.000000    0.0     0.0 0.00000
aortic arch          thoracic aorta          0.000090   10.0   -90.0 0.00380
aortic arch          head & arms arteries    0.047000   19.5    90.0 0.01400
thoracic aorta       intestinal arteries     0.001400    8.0   -90.0 0.00270
thoracic aorta       abdominal arteries      0.012000   16.0   -90.0 0.01400
thoracic aorta       right atrium           12.000000   10.0    90.0 0.00000
intestinal arteries  intestinal veins        2.300000    0.0     0.0 0.00000

                                        For more press (Y/N)
Model: 19-segment
Condition: normal
```

Figure 5.9 **List all flows parameter menu**

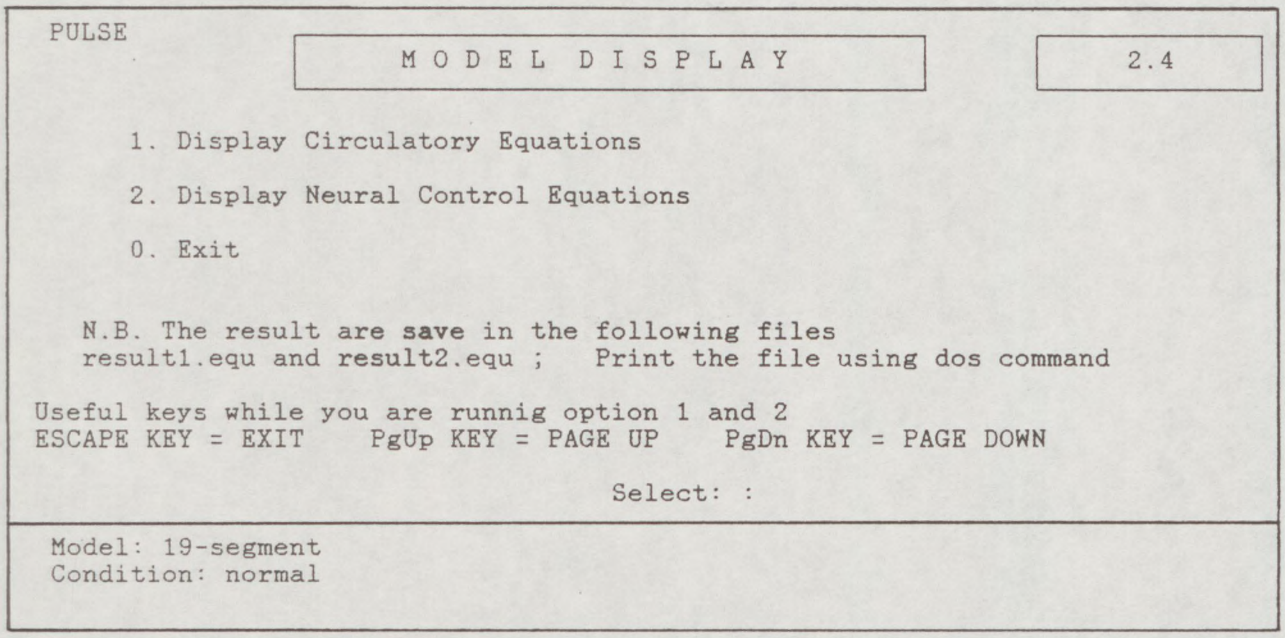


```
PULSE
                  Neural Control                     2.3.7


      1. Baroreceptors
      2. Cns Inputs
      3. Heart Rate
      4. Peripheral Resistance
      5. Myocardial Contractility
      6. Venous Tone
      0. Exit

                  Select: :

Model: 19-segment
Condition: normal
```



```
PULSE
                  Neural Control                     2.3.7.1
     Listing Parameters / baroreceptors


Name                    X1       X2     Slow Time Fast Time Gain1 Gain2 Thrsh_Pres

aortic arch            102.1393   2.3577 0.8       0.1        1.0   1.0   40.0
head & arms arteries  97.7083   2.8019 0.8       0.1        1.0   1.0   40.0



                  Return to listing neural control menu ? (Y/N)

Model: 19-segment
Condition: normal
```

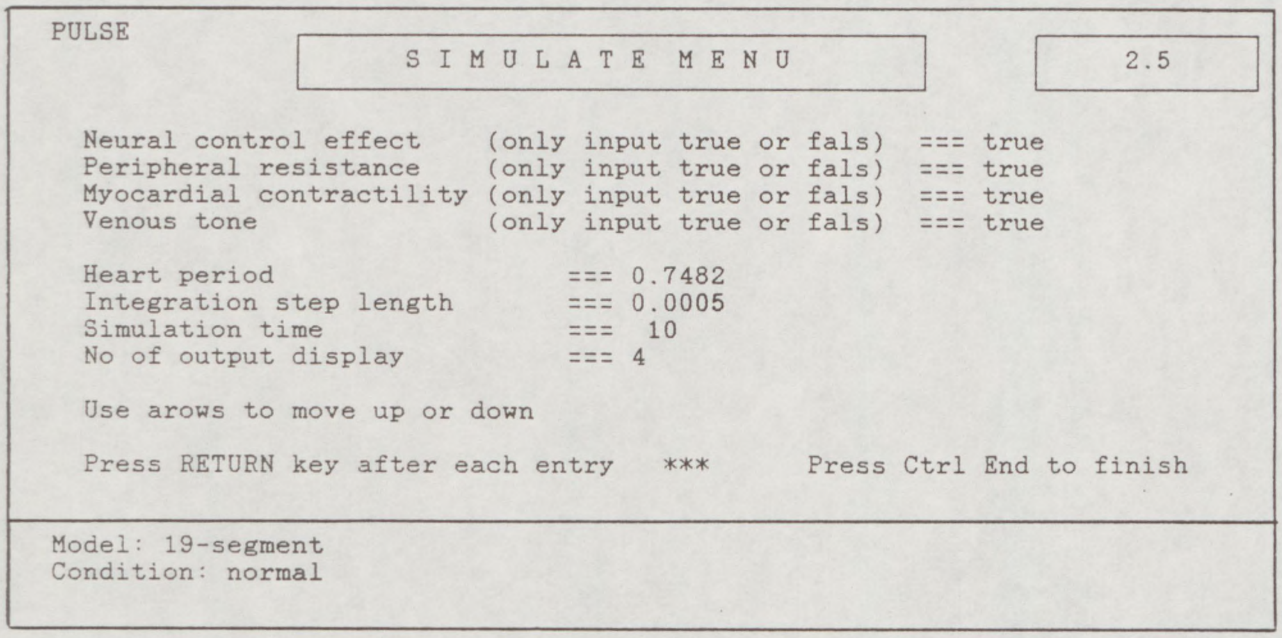
Figure 5.10 **List all baroreceptors parameter menu**



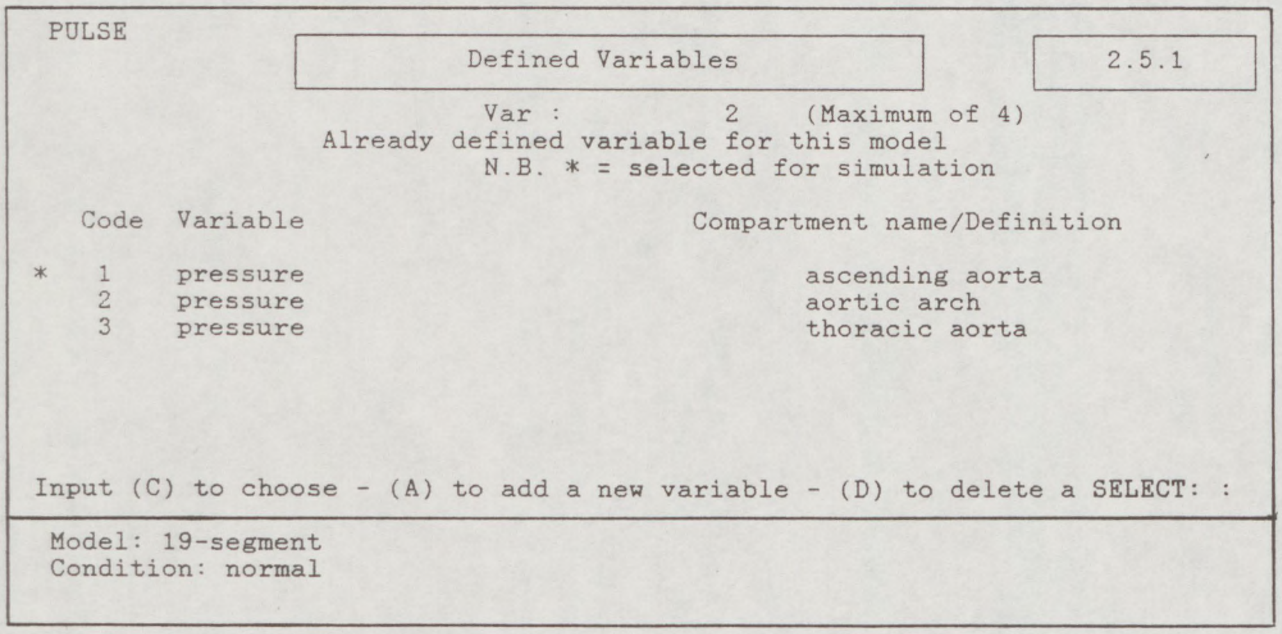
```
PULSE
                      M O D E L   D I S P L A Y                     2.4

      1. Display Circulatory Equations

      2. Display Neural Control Equations

      0. Exit


   N.B. The result are save in the following files
   result1.equ and result2.equ ;   Print the file using dos command

 Useful keys while you are runnig option 1 and 2
 ESCAPE KEY = EXIT    PgUp KEY = PAGE UP    PgDn KEY = PAGE DOWN

                                         Select: :

 Model: 19-segment
 Condition: normal
```

```
***************************************************************************
                               2.  Pressures
***************************************************************************


                               2.1 Arteries
                               ------------

For all arteries i;

P[i] = ((V[i]-VU[i])/C[i]) + ((K[i]/C[i])dV[i]/dt)

££££££££££££££££££££££££££££££££ END OF ARTERIES ££££££££££££££££££££££££££££££££££


                               2.2 Veins
                               ---------

For all veins i;

P[i] = (V[i]-sigmaVU[i]) (sigmaC[i])


   sigmaVU[i] = VU[i]/CnsVVu                       if neural is true
   sigmaVU[i] = VU[i]                              if neural is false

   C[i] = (initial compliance Cio)                 if V[i] >  sigmaVU[i]
   C[i] = (initial compliance Cio)(alfa[i])        if V[i] <= sigmaVU[i]

   sigmaC[i] = CnsVC/C[i]                          if neural is true
   sigmaC[i] = 1/C[i]                              if neural is false
```

Figure 5.11 **DISPLAY EQUATIONS menu**

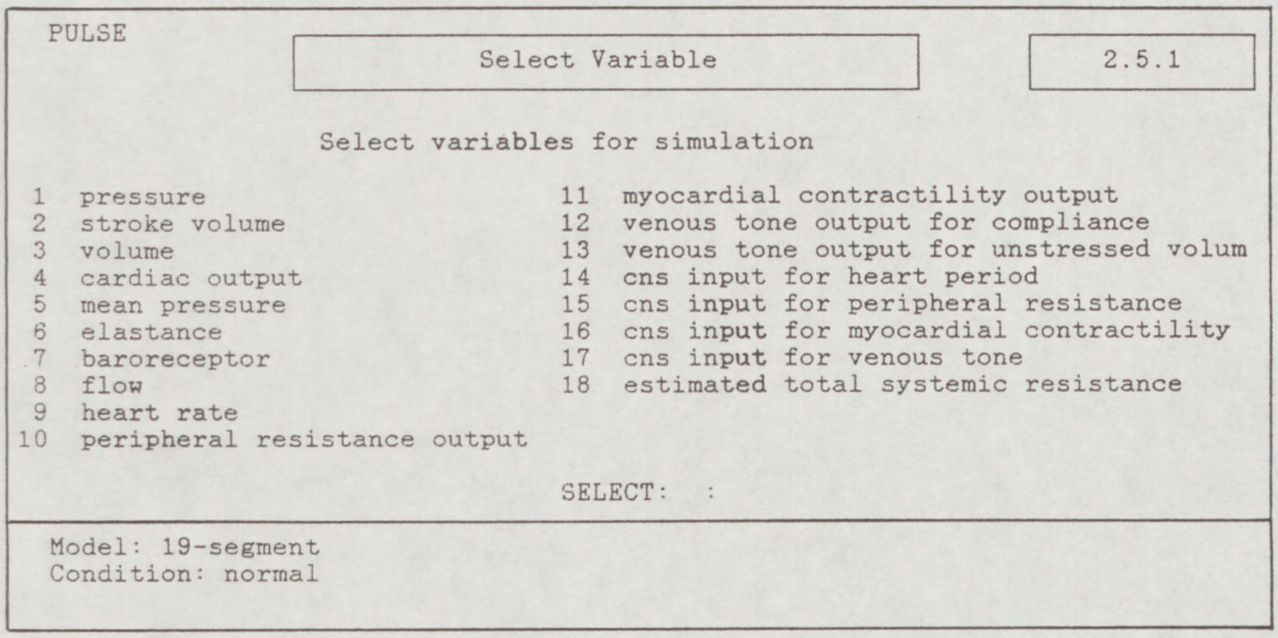


```
PULSE
                    S I M U L A T E  M E N U                       2.5

   Neural control effect     (only input true or fals)  === true
   Peripheral resistance     (only input true or fals)  === true
   Myocardial contractility (only input true or fals)  === true
   Venous tone               (only input true or fals)  === true

   Heart period                         === 0.7482
   Integration step length              === 0.0005
   Simulation time                      ===  10
   No of output display                 === 4

   Use arows to move up or down

   Press RETURN key after each entry   ***      Press Ctrl End to finish

Model: 19-segment
Condition: normal
```

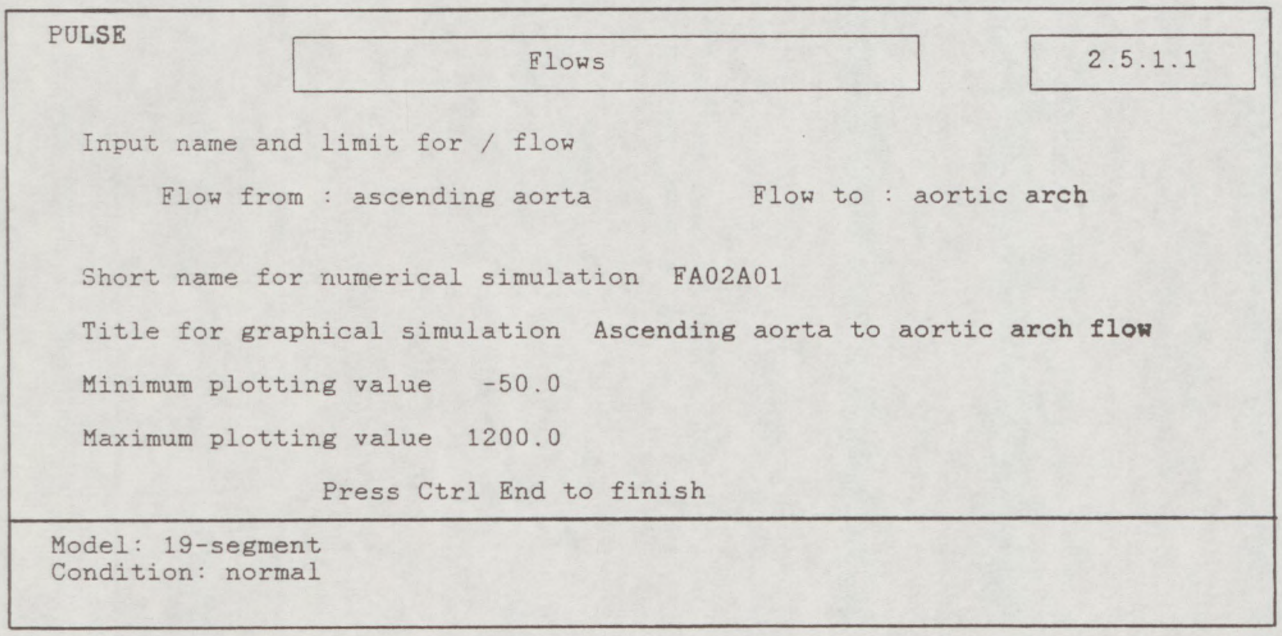


```
PULSE
                           Defined Variables                      2.5.1
                         Var :             2    (Maximum of 4)
                   Already defined variable for this model
                         N.B. * = selected for simulation

   Code  Variable                            Compartment name/Definition

*   1    pressure                                   ascending aorta
    2    pressure                                   aortic arch
    3    pressure                                   thoracic aorta

  Input (C) to choose - (A) to add a new variable - (D) to delete a SELECT: :

Model: 19-segment
Condition: normal
```

Figure 5.12 **Initial requirement (top) and already defined variables (bottom) menus**



```
PULSE
                         Select Variable                          2.5.1

                      Select variables for simulation

 1  pressure                          11  myocardial contractility output
 2  stroke volume                     12  venous tone output for compliance
 3  volume                            13  venous tone output for unstressed volum
 4  cardiac output                    14  cns input for heart period
 5  mean pressure                     15  cns input for peripheral resistance
 6  elastance                         16  cns input for myocardial contractility
 7  baroreceptor                      17  cns input for venous tone
 8  flow                              18  estimated total systemic resistance
 9  heart rate
10  peripheral resistance output

                                  SELECT:  :

Model: 19-segment
Condition: normal
```



```
PULSE
                          Flows                                   2.5.1.1

  Input name and limit for / flow

      Flow from : ascending aorta         Flow to : aortic arch

  Short name for numerical simulation  FA02A01

  Title for graphical simulation  Ascending aorta to aortic arch flow

  Minimum plotting value   -50.0

  Maximum plotting value  1200.0

                  Press Ctrl End to finish

Model: 19-segment
Condition: normal
```

Figure 5.13 **Select variables (top) and limit of variables (bottom) menus**

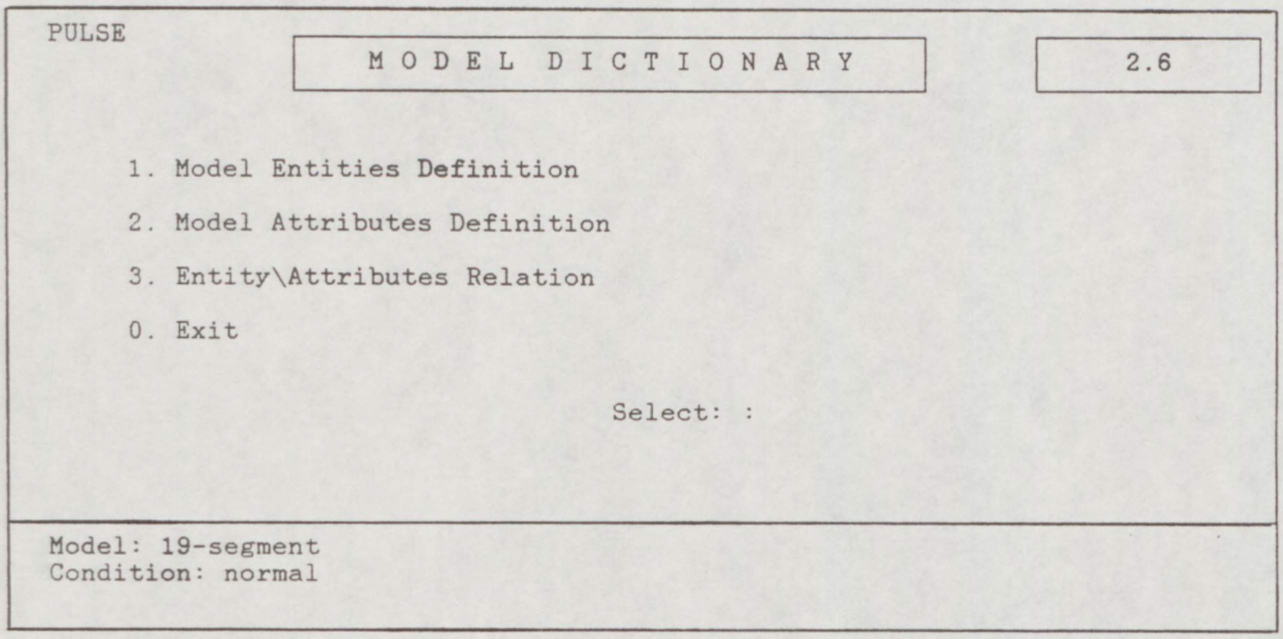
```
 8
 1 0.28  53.0 0.04  -4.0
 2 0.29  61.0 0.04  -4.0
 3 0.29  59.0 0.04  -4.0
 4 0.06  17.0 0.04   4.0
 5 0.21  58.0 0.04   4.0
 6 0.12  63.0 0.04   0.0
 7 0.33 114.0 0.04   0.0
 8 4.30  50.0 0.00  -4.0
 7
 9  9.4 552.0 20   0.0 true
10  8.3 488.0 20  -4.0 fals
11 10.6 607.0 20   4.0 true
12  5.1 305.0 20   4.0 true
13  4.8 257.0 20   0.0 true
14  8.3 488.0 20  -4.0 fals
15  8.4 460.0 20  -4.0 fals
 2
16  0.0 0.30 0.046 0.04 0.16 0.20  -4.0
17  0.0 1.50 0.067 0.04 0.16 0.20  -4.0
 2
18 30.0 0.15 0.050 0.10 0.09  -4.0
19 30.0 0.28 0.120 0.10 0.09  -4.0
13
 2  1  0.000031  0.0   0.0 0.00043 0.0 fals
18  1 12.000000  0.0   0.0 0.00000 0.0 fals
 7  2  0.047000 19.5  90.0 0.01400 0.0 fals
 3  2  0.000090 10.0 -90.0 0.00380 0.0 fals
18  3 12.000000 10.0  90.0 0.00000 0.0 true
 4  3  0.001400  8.0 -90.0 0.00270 0.0 fals
 5  3  0.012000 16.0 -90.0 0.01400 0.0 fals
11  4  2.300000  0.0   0.0 0.00000 0.0 true
 6  5  0.180000 48.0 -90.0 0.03100 0.0 fals
12  5 57.000000  0.0   0.0 0.00000 0.0 true
13  6 15.000000  0.0   0.0 0.00000 0.0 true
 9  7  6.000000  0.0   0.0 0.00000 0.0 fals
15  8  0.110000  0.0   0.0 0.00000 7.0 fals
 7
10  9 0.226 18.0 -90.0 552.0  0.667 fals
18 10 0.060  1.5 -90.0 488.0  0.100 fals
14 11 0.166  8.0  90.0 607.0  1.000 true
14 12 0.595 16.0  90.0 305.0  1.000 true
12 13 0.300 48.0  90.0 257.0  0.000 true
18 14 0.015 10.0  90.0 488.0  0.100 fals
19 15 0.007  0.0   0.0 460.0  0.100 fals
 2
 8 16 0.003 0.00018 1.539 fals
 1 17 0.003 0.00022 1.539 true
 2
16 18 0.003
17 19 0.003
```

Figure 5.14 **ASCII file for circulation for the 19-segment model**

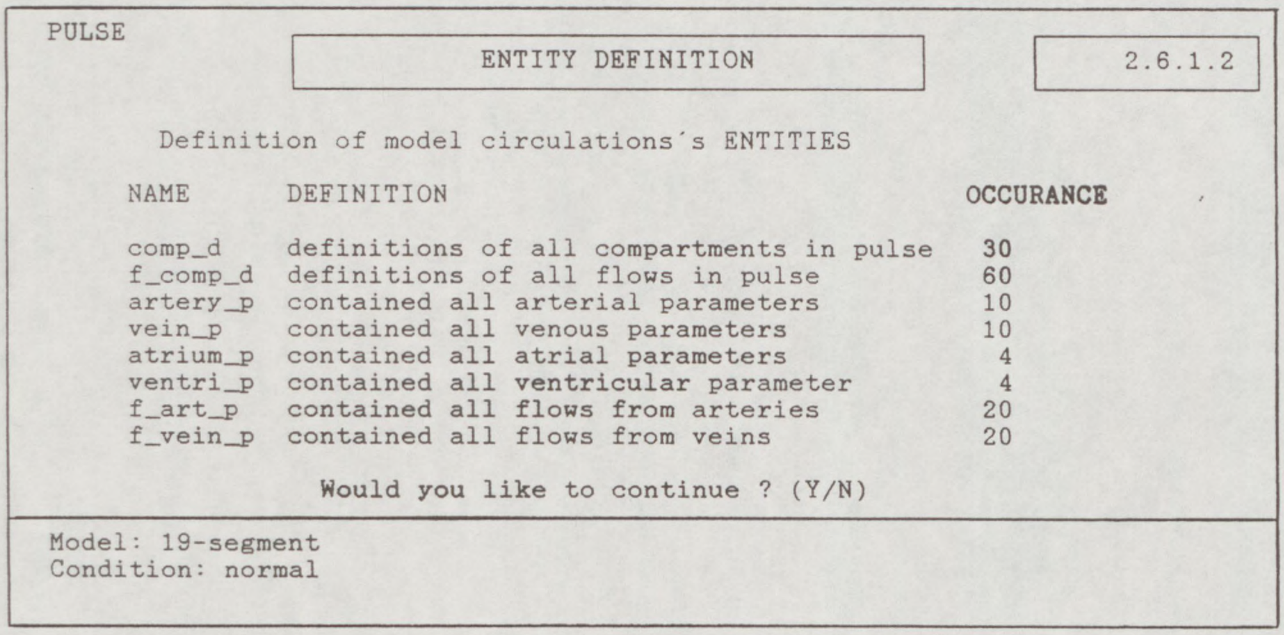
```
2
 2 0.8 0.1 1.0 1.0 40.0
 7 0.8 0.1 1.0 1.0 40.0
 2 0.3
 7 0.7
80.0 1.5 4.5 1.0 2.0 2.0 0.3 1.0 0.006
true
 2 0.3
 7 0.7
80.0 4.0 20.0 0.6 1.4 0.75
true
 2 0.3
 7 0.7
80.0 10.0 0.6 1.4
true
 2 0.3
 7 0.7
80.0 14.0 0.7 1.6 1.0 1.0
```

```
0.0005
  1
4
others
inputh
BHR      Cns input for heart period
others
cnsper
CnsP     Peripheral resistance output
others
cnsmyo
CnsM     Myocardial contractility output
others
cnsvvu
CnsVVu   Venous tone output effecting Vu
50.0  300.0
0.0   10.0
0.0   10.0
0.0   10.0
```

Figure 5.15 **ASCII file for neural control (top) and desired output (bottom) for the 19-segment model**

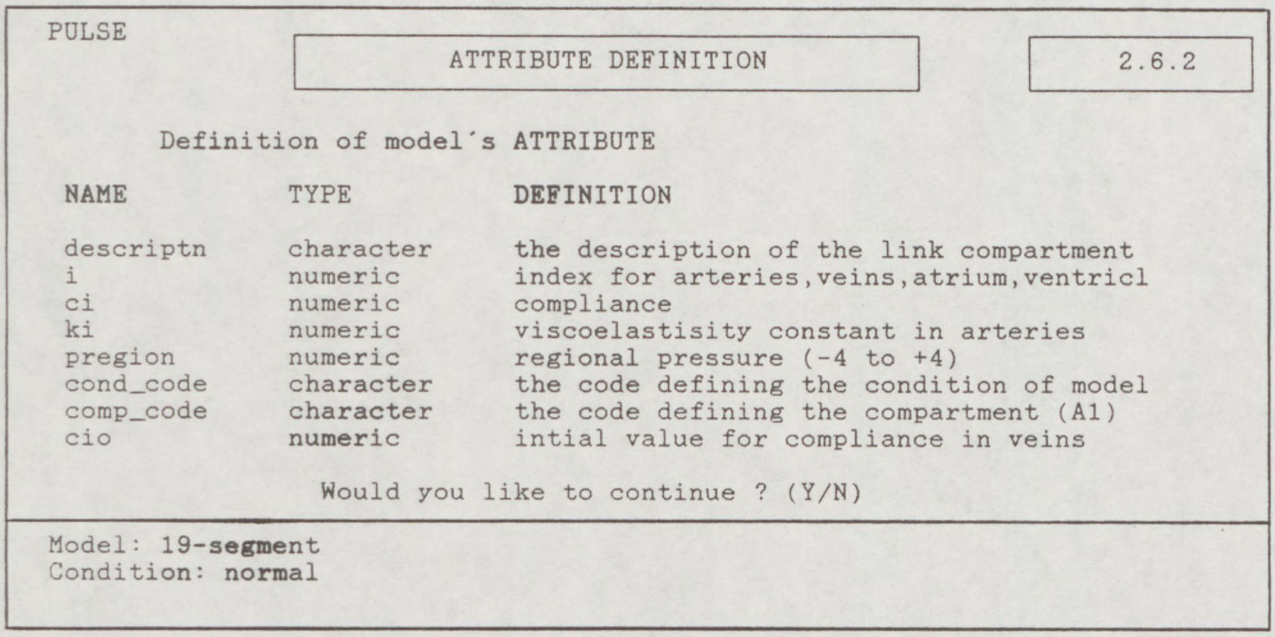


```
PULSE
                    M O D E L   D I C T I O N A R Y                 2.6

        1. Model Entities Definition

        2. Model Attributes Definition

        3. Entity\Attributes Relation

        0. Exit

                                          Select: :

Model: 19-segment
Condition: normal
```



```
PULSE
                          ENTITY DEFINITION                         2.6.1.2

          Definition of model circulations´s ENTITIES

        NAME      DEFINITION                                    OCCURANCE

        comp_d    definitions of all compartments in pulse       30
        f_comp_d  definitions of all flows in pulse              60
        artery_p  contained all arterial parameters              10
        vein_p    contained all venous parameters                10
        atrium_p  contained all atrial parameters                 4
        ventri_p  contained all ventricular parameter             4
        f_art_p   contained all flows from arteries              20
        f_vein_p  contained all flows from veins                 20

                    Would you like to continue ? (Y/N)

Model: 19-segment
Condition: normal
```

Figure 5.16 **DICTIONARY menu (top), entity menu (bottom)**


```
PULSE
                    ATTRIBUTE DEFINITION                     2.6.2

       Definition of model's ATTRIBUTE

  NAME          TYPE          DEFINITION

  descriptn     character     the description of the link compartment
  i             numeric       index for arteries,veins,atrium,ventricl
  ci            numeric       compliance
  ki            numeric       viscoelastisity constant in arteries
  pregion       numeric       regional pressure (-4 to +4)
  cond_code     character     the code defining the condition of model
  comp_code     character     the code defining the compartment (A1)
  cio           numeric       intial value for compliance in veins

                   Would you like to continue ? (Y/N)

Model: 19-segment
Condition: normal
```

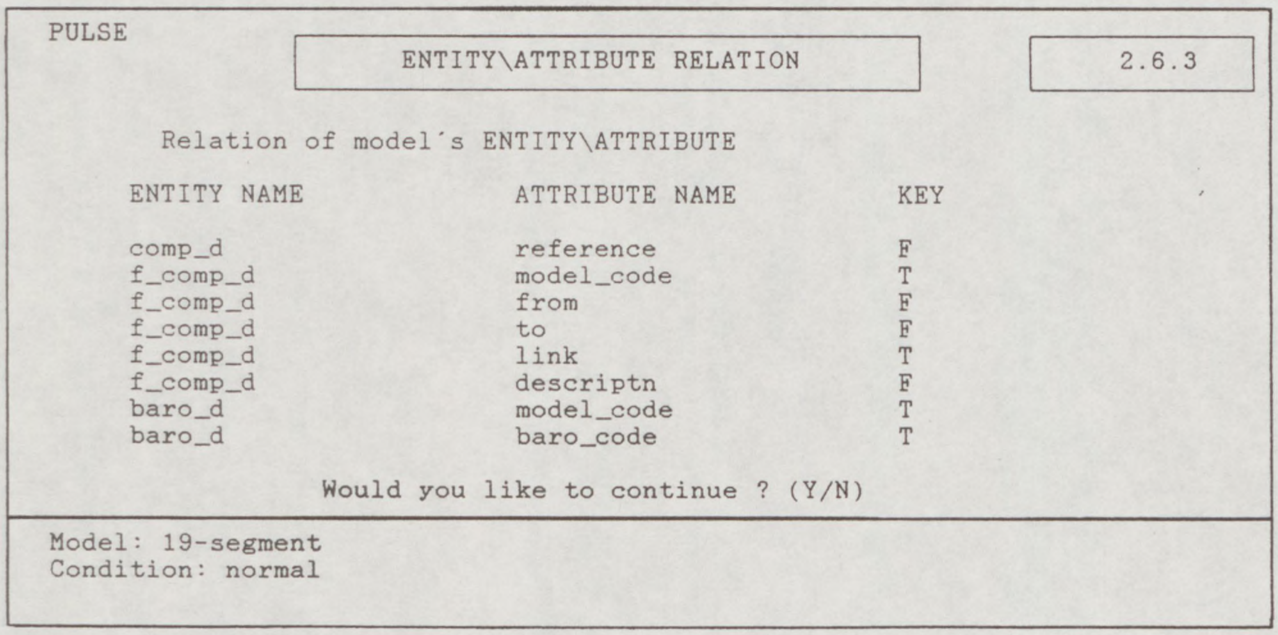
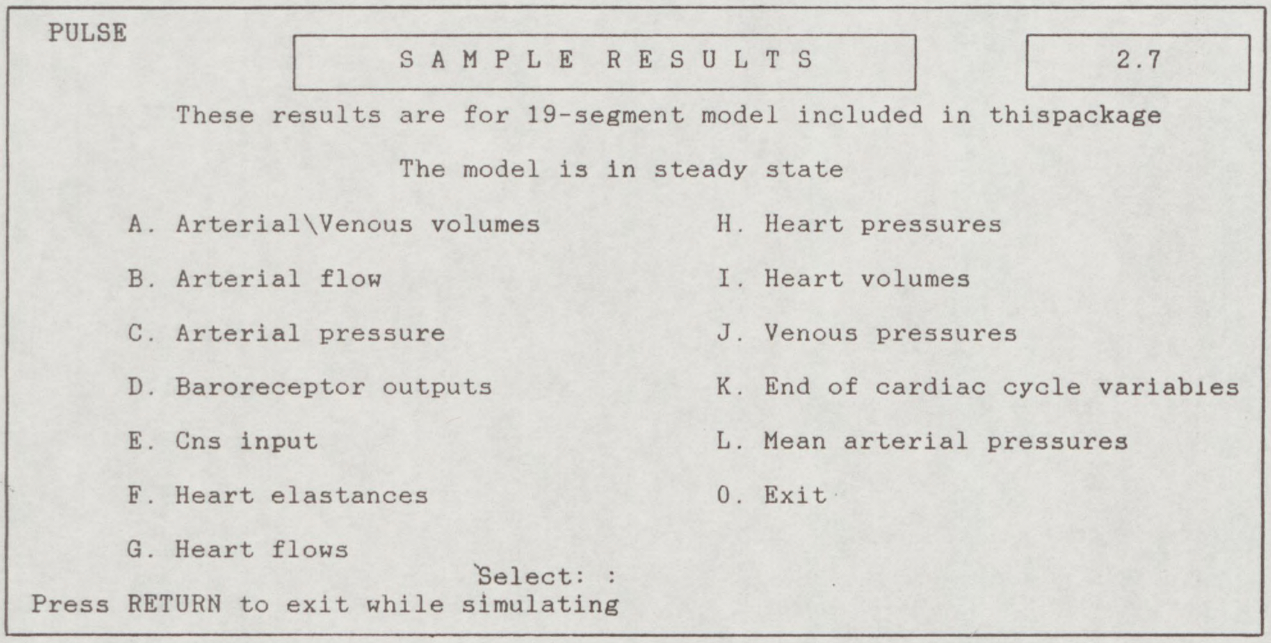

```
PULSE
                    ENTITY\ATTRIBUTE RELATION                2.6.3

       Relation of model's ENTITY\ATTRIBUTE

      ENTITY NAME              ATTRIBUTE NAME           KEY

      comp_d                   reference                F
      f_comp_d                 model_code               T
      f_comp_d                 from                     F
      f_comp_d                 to                       F
      f_comp_d                 link                     T
      f_comp_d                 descriptn                F
      baro_d                   model_code               T
      baro_d                   baro_code                T

                   Would you like to continue ? (Y/N)

Model: 19-segment
Condition: normal
```

Figure 5.17 **attribute menu (top), ent_att menu (bottom)**



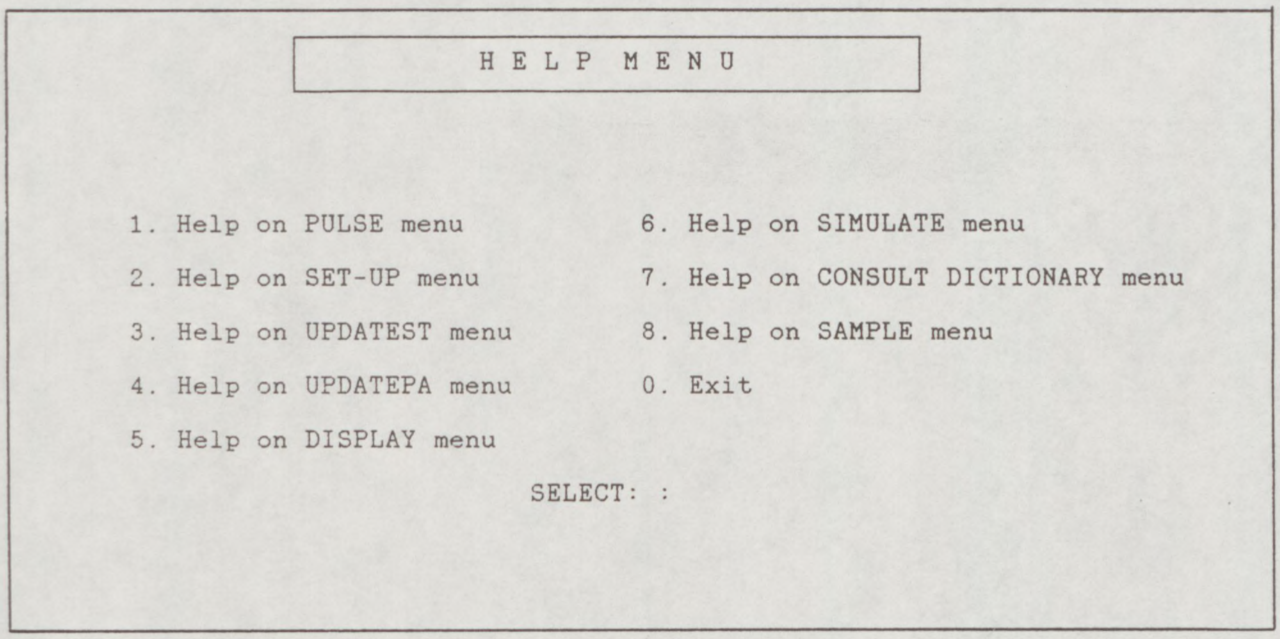
```
PULSE
                   S A M P L E   R E S U L T S                 2.7
        These results are for 19-segment model included in thispackage

                        The model is in steady state

     A. Arterial\Venous volumes              H. Heart pressures

     B. Arterial flow                        I. Heart volumes

     C. Arterial pressure                    J. Venous pressures

     D. Baroreceptor outputs                 K. End of cardiac cycle variables

     E. Cns input                            L. Mean arterial pressures

     F. Heart elastances                     0. Exit

     G. Heart flows
                             Select: :
Press RETURN to exit while simulating
```

Figure 5.18 **SAMPLE menu**

```
                 NUMERICAL  SIMULATION RESULTS

             Commands for listing numerical simulation results
             -------------------------------------------------

             Use the following keys
             ----------------------

                  ESCAPE = Exit

                  PgUp   = to go up

                  PgDn   = to go down

   The result is saved in \pulse\result3.res file.
   This is a text file which may be printed with the DOS command print

        Would you like to view numerical simulation results (Y/N)
```

```
MPA01      Mean arterial pressure

MPA03      Mean thoracic aorta pressure

MPAA       Mean abdominal arteries pressure

MPCA       Mean Leg arteries pressure


      ***** During Cardiac Cycle *****

Time         MPA01         MPA03         MPAA          MPCA
 0.0000        0.0000        0.0000       0.0000        0.0000
 0.0505        6.1032        6.5716       6.8399        9.2715
 0.1005       12.0593       13.0049      13.5674       18.4583
 0.1505       17.8727       19.2988      20.1665       27.5372
 0.2005       24.0849       25.5475      26.6310       36.4850
 0.2505       31.9353       32.9916      33.3763       45.3629
 0.3005       40.1495       41.6248      41.2569       54.6462
 0.3505       48.0287       50.4810      50.2725       64.9680
 0.4005       55.0523       58.7088      59.6717       76.4407
 0.4505       62.1573       66.3858      68.6260       88.4587
 0.5005       69.3582       74.0597      77.0280      100.2072
 0.5505       76.5429       81.7390      85.0722      111.2702
 0.6005       83.6278       89.3112      92.8570      121.6620
```

Figure 5.19 **Numerical result file from SAMPLE menu**



```
                    H E L P   M E N U

   1. Help on PULSE menu        6. Help on SIMULATE menu

   2. Help on SET-UP menu       7. Help on CONSULT DICTIONARY menu

   3. Help on UPDATEST menu     8. Help on SAMPLE menu

   4. Help on UPDATEPA menu     0. Exit

   5. Help on DISPLAY menu

                        SELECT: :
```

All the work will be automaticaly saved by the package and the user need not worry about saving their work. However it is advisable to backup ones work after exiting from PULSE.

Ctrl End prompt is dBASEIII+ command to save the current work and it is advisable to use it whenever prompted. However the work will be saved if you use another key by mistake and thus you need not worry about it.

The PULSE contains of 3 main menu.

The first menu (1) is the COPYRIGHT menu.

The second menu (2) is the MAIN menu and this menu consists of several submenus.

The third menu (3) is the EXIT menu which gives the user a warning to backup their disks.

Press any key to continue

Figure 5.20 **HELP menu**

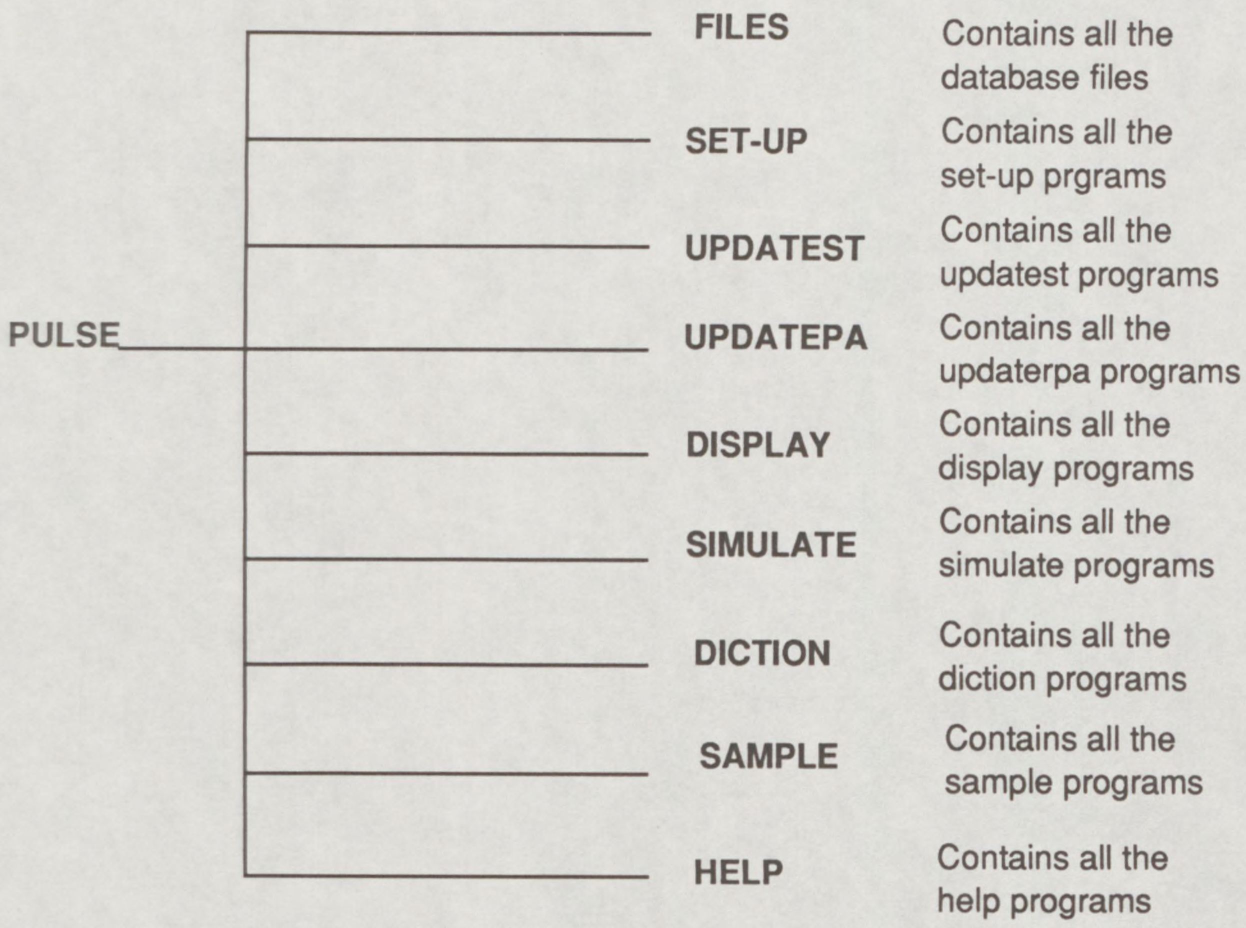

PULSE
FILES
Contains all the database files
SET-UP
Contains all the set-up prgrams
UPDATEST
Contains all the updatest programs
UPDATEPA
Contains all the updaterpa programs
DISPLAY
Contains all the display programs
SIMULATE
Contains all the simulate programs
DICTION
Contains all the diction programs
SAMPLE
Contains all the sample programs
HELP
Contains all the help programs

Figure 5.21 **Structure of the PULSE directory**

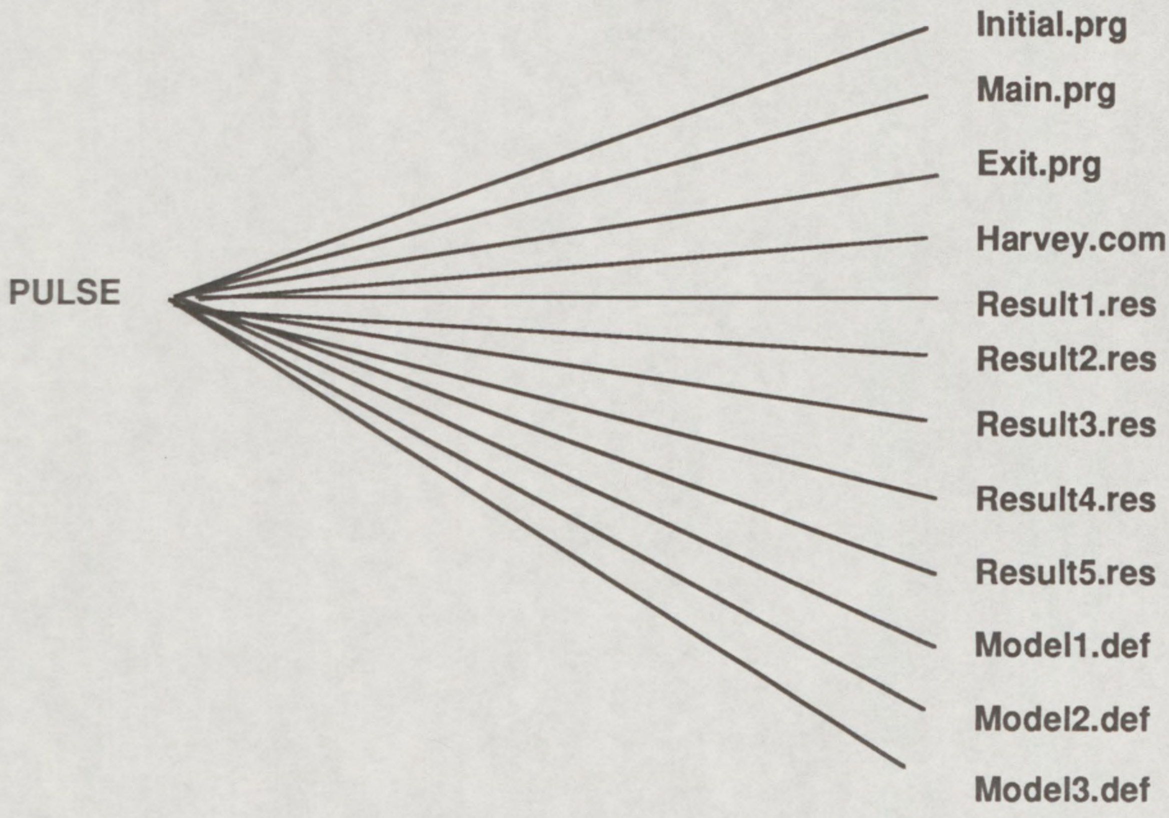


Figure 5.22 **Files in the PULSE**

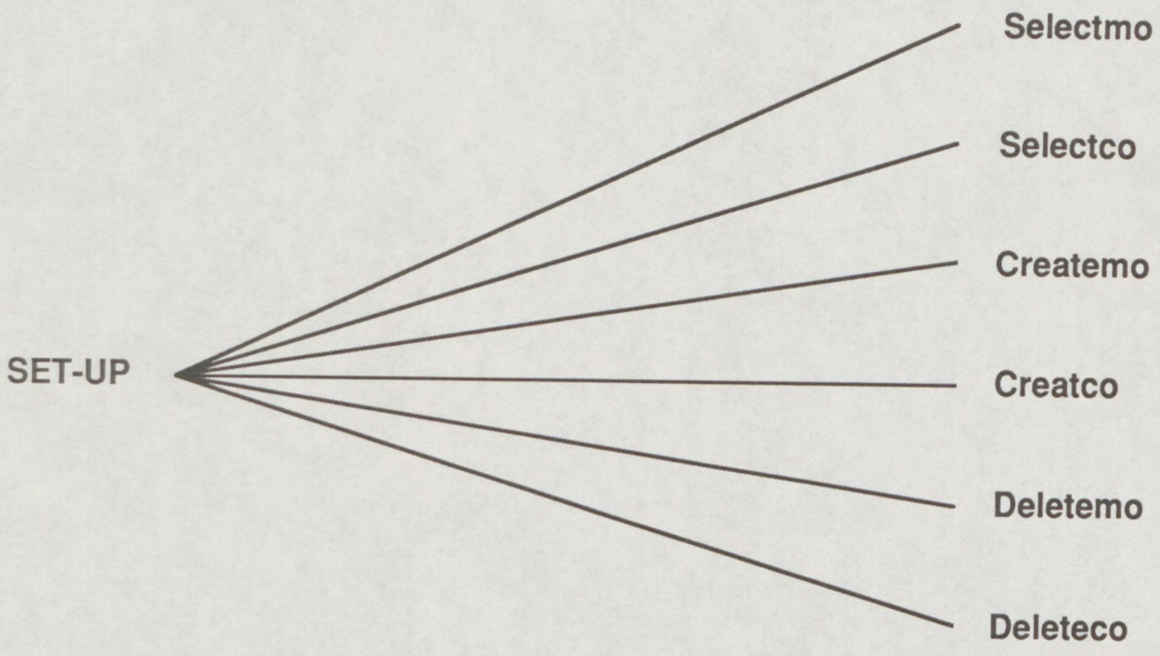


Figure 5.23 **Programs in SET-UP directory**

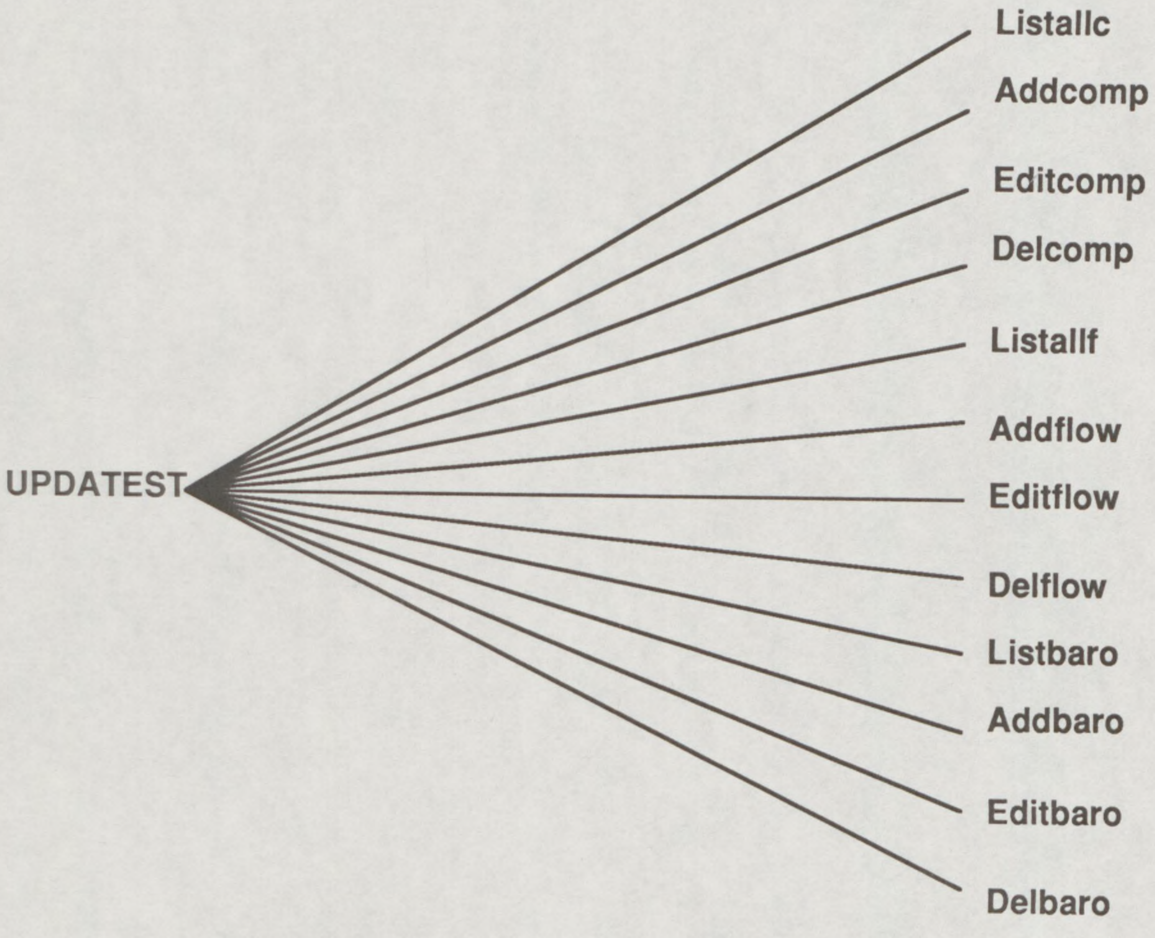


Figure 5.24 **Programs in UPDATEST directory**

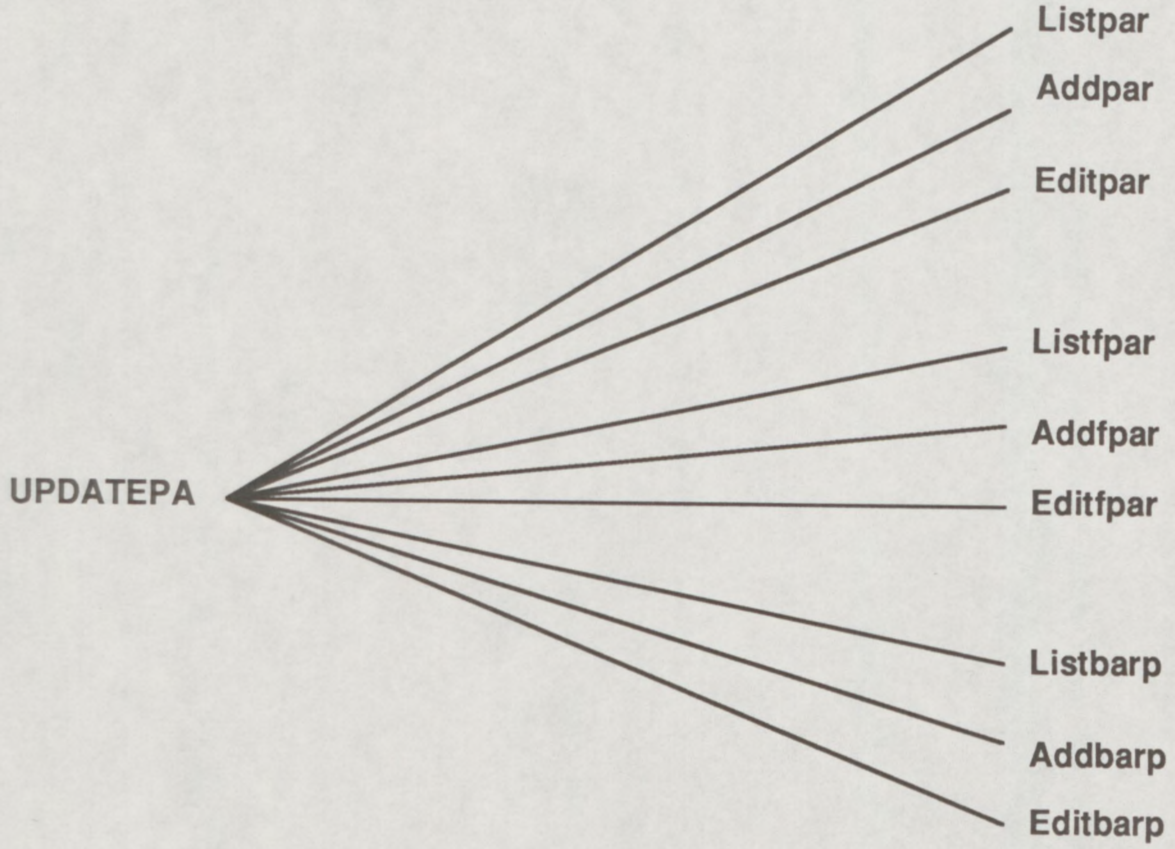


Figure 5.25 **Programs in UPDATEPA directory**

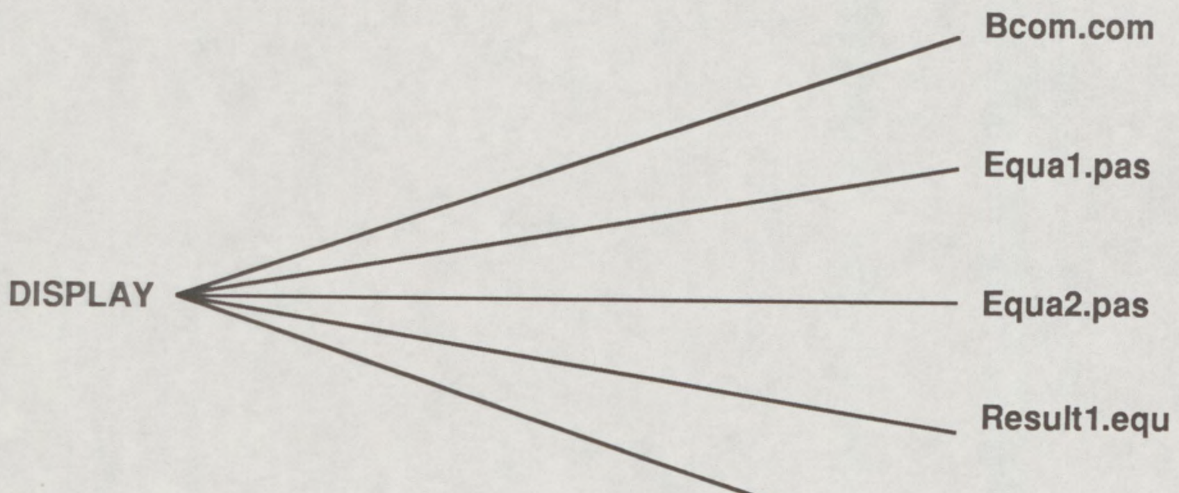


Figure 5.26 **Programs in DISPLAY directory**

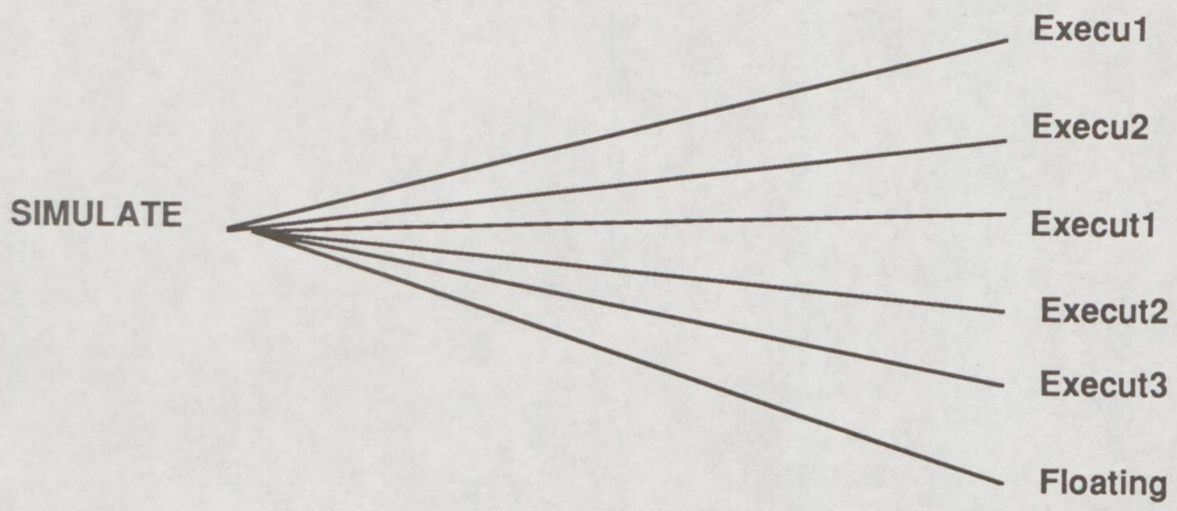


Figure 5.27 **Programs in SIMULATE directory**

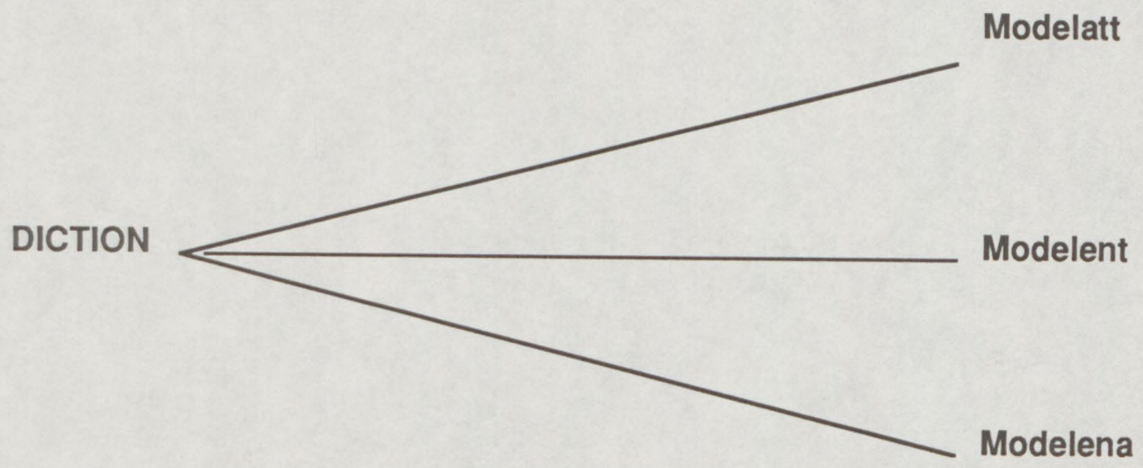


Figure 5.28 **Programs in DICTION directory**

CHAPTER 6

VALIDATION OF 'PULSE'

6.1 INTRODUCTION

In the previous chapter, the approach to software development was discussed. The chapter also included the choice of the software and hardware for implementation of 'PULSE' as well as the development of 'PULSE'.

Leaning (1980) and Carson et al. (1983) define model validation as an integral component of the modelling process and the criteria in terms of which validity is assessed reflect the several stages of the modelling process, being joined with all stages of model development. Validity, from their point of view, is not performed solely as a final step in modelling, but as a part of the overall process. Leaning et al. (1983b) class the validity criteria as <u>internal criteria</u> and <u>external criteria</u>. The <u>internal criteria</u> consist of tests which do not require reference to theories and as such they are prerequisite criteria. The important aspects of internal validity are the consistency and completeness of a model. The <u>external criteria</u> refer to aspects external to the model itself, and divide into several stages. *Empirical* validity (that the model should correspond to the available data), *theoretical* validity (that the model should be consistent with accepted theories or models), *pragmatic* validity (to what extent the model satisfies the objectives of the user) and finally *heuristic* validity (that assesses the potential of the model for scientific explanation, discovery, hypothesis testing etc).

Here, the aim of the validation studies was not to validate any specific model, but rather to validate the methodology underlying the approach and also to verify the software 'PULSE'. Thus, the chapter was divided into two parts. The first part of the chapter deals with the internal validity of the software and the approach, that is to verify the consistency and completeness of 'PULSE', to verify the accuracy of the simulation results, and to illustrate the usability of 'PULSE'.

For internal validity, the existing 19-segment model (Beneken and De Wit, 1967; Pullen, 1976; Leaning et al., 1983) was used. The reason for the choice of the 19-segment model lay in the fact that the model was already well-validated by Leaning (1980) and Al-Dahan (1984). It is a large model and was thought to be a good example to verify the software package at the challenging level of complexity. For the sake of avoiding repetition, the references for this model will not be quoted throughout this chapter.

For the external validity, two models were built. One of these models is the four segment model built by an independent scientist (Dr Knut Lande) and the other is an extension to the 19-segment model (referred to as the 23-segment model). Also the validity of the results obtained for the 19-segment model will be discussed.

At this stage it is important to mention that it is not expected that all the new models built with 'PULSE' will have such complexity as the 19-segment model. Indeed, a goal of 'PULSE' is to allow the building of sets of small models each addressing a specific problem (see Chapter 7).

6.2 INTERNAL VALIDITY

As has been explained in Section 6.1, to ensure the internal validity of the approach, the package 'PULSE' was first verified by implementing the well-validated 19-segment model (See Appendix 4 for the list of database files for this model). Notes were taken of any inconsistency (i.e. where the representation of the model was not consistent, such as duplicate entry for the same variables, inconsistent connections for the flows, etc) or incompleteness (i.e. where the model could not be represented completely, due to lack of presentation of parameters, flows, etc) due to the limitations of the approach. This section is divided into three parts:

(1) Consistency and completeness of representation of the 19-segment model in 'PULSE';

(2) The accuracy of simulation results for the 19-segment model in 'PULSE';

(3) The usability of 'PULSE' using the 19-segment model.

6.2.1 Consistency and Completeness of 'PULSE'

The 19-segment model represented in Fig. 2.1, consists of 8 arteries, 7 veins, 2 atria and 2 ventricles. The flow between the compartments is also depicted in this figure. There are 13 arterial flows, 7 venous flows, 2 atrial flows and 2 ventricular flows. Using the tools described in Chapter 5, this model was reproduced in 'PULSE'. First, the model name was entered (using the ***set-up*** menu). Then, the ***update/display structure*** menu was selected. For each compartment an entry had to be defined. So for the 19-segment model, 19 compartments were defined using the option ***add compartment*** in this menu (e.g. for the ascending aorta compartment the entries were:

name :	ascending aorta
region :	thorax
comptype :	artery
anat_det :	first part of the great aorta
reference :	any anatomical text book

see Chapter 4 for the definition of these entities).

Once the compartment definitions were finished, the flow definition had to be defined. For the 19-segment model, 24 flows were defined using the option ***add flow*** from the above menu (e.g. for the inferior vena cava to the right atrium flow, the entries were:

from :	inferior vena cava
to :	right atrium
descriptn :	venous/atrial connection

see Chapter 4 for the definition of these entities).

Now, the <u>circulatory schema</u> has been defined in 'PULSE'. To define the <u>neural control schema</u>, the entries in baroreceptors had to be defined first. For the 19-segment model, 2 baroreceptors were defined using the option ***add neural control*** followed by ***baroreceptor*** selection from the above menu (e.g. for the aortic arch baroreceptor, 'PULSE' displays the list of arteries in this model and prompt the user for the selection, then the following entry will be displayed:

artery name :	*aortic arch*
descriptn :	aortic arch baroreceptor).

Once the baroreceptors have been defined the central nervous system (CNS) input for these baroreceptors have to be defined. For the 19-segment model, 2 CNS were defined using the option ***add neural control*** followed by ***CNS input*** selection from the above menu (e.g. for the aortic arch CNS input, 'PULSE' displays the list of baroreceptors in this model and prompts the user for selection, then the following entry will be displayed:

baroreceptor: aortic arch baroreceptor

`descriptn :` CNS input for the aortic arch).

Finally, the controllers have to be defined. For the 19-segment model, 4 controllers were defined using the option ***add neural control*** followed by ***controller*** selection from the above menu (e.g. for the heart rate controller, the following entry will be displayed:

`descriptn :` heart rate controller).

The following points ensures the consistency of 'PULSE' and hence the approach:

(1) The order in which the definitions are carried out in 'PULSE' should be the same as the above order (compartments, flows, neural control). This is because one cannot define the flow for an undefined compartment, and also the neural control cannot be defined unless the arterial compartments are defined.

(2) If a compartment, flow or neural control is already defined, the package 'PULSE' does not allow these quantities to be added again. However, these entries can be edited for any mistakes that might have occurred.

(3) 'PULSE' does not allow the parameters (will be explained later) to be entered without the structure being defined first.

(4) The structure of the model is only defined once while the parameter entries (which will be explained later) are entered under various conditions. (each condition has one set of parameters).

(5) deleting a structure entry (e.g. ascending aorta) will delete all the parameters relating to the structure entry (e.g. all the parameters for the ascending aorta will be deleted, also all the flows relating to ascending aorta will be deleted too).

Once the structure of the model is defined, the ***set-up*** menu is reselected so that the entry for a condition of the model can be entered. For the 19-segment model the normal condition was entered. Now, the parameters for the 19-segment model can be entered. This is achieved by selecting the ***update/display parameter*** menu. For each compartment an entry for the parameter was entered. So for the 19-segment model, 8 arterial parameters, 7 venous parameters, 2 atrial parameters and 2 ventricular parameters were entered by selecting the ***add compartment*** in this menu. 'PULSE' displays the list of compartments in the selected model and the desired compartment was chosen (e.g. after selecting the ascending aorta from the displayed lists the following entries are prompted:

Name : ascending aorta

Ci	:	0.28
Vui	:	53.0
Ki	:	0.04
ViO	:	79.86140
Pregion	:	-4.0

see Chapter 4 for the definition of these entities).

Once the compartment parameters were entered, the flow parameter had to be input. For the 19-segment model, 24 flow parameters were input using the option ***add flow*** from the above menu. 'PULSE' displays the list of flows in the selected model and the desired flow was chosen (e.g. after selecting the ascending aorta to aortic arch flow from the displayed lists the following entries are prompted:

From : ascending aorta *To : aoric arch*

Rji	:	0.000031
lenji	:	0.0
angIji	:	0.0
Lji	:	0.00043
F_{ji}	:	6.5532
Pcc	:	0.0
neural	:	fals

see Chapter 4 for the definition of these entities).

Now, the circulatory parameters have been entered in 'PULSE'. To enter the neural control parameter, for the 19-segment model, 2 baroreceptors parameter were entered using the option ***add neural control*** followed by ***baroreceptor*** selection from the above menu. 'PULSE' displays the list of baroreceptors in the selected model and the desired baroreceptor was chosen (e.g. after selecting the aortic arch baroreceptor from the displayed lists the following entries are prompted:

Name : aortic arch baroreceptor

x1	:	102.1393
x2	:	92.3577
slow_time	:	0.8
fast_time	:	0.1
gain1	:	1.0
gain2	:	1.0
thrsh_pres	:	40.0

see Chapter 4 for the definition of these entities).

Once the baroreceptors parameters have been entered the central nervous system (CNS) input parameter for these baroreceptors have to be enter. For the 19-segment model, 2 CNS parameters were entered for each controller, using the option ***add neural control*** followed by ***CNS input*** selection from the above menu. 'PULSE' displays the list of CNS inputs in the selected model and the desired CNS input was chosen (e.g. after selecting the CNS input for the aortic arch baroreceptor from the displayed lists the following entries are prompted:

name : CNS input for the aoric arch baroreceptor

control	contributn
heart	0.3
periph	0.3
myocar	0.3
venous	0.3

see Chapter 4 for the definition of these entities.).

Finally, the controller parameters have to be entered. For the 19-segment model, 4 controller parameters were entered. using the option ***add neural control*** followed by ***controller*** selection from the above menu. 'PULSE' displays the list of controllers in the selected model and the desired controller was chosen (e.g. after selecting the heart rate controller from the displayed lists the following entries are prompted:

name: heart rate controller

bhrt	:	80.0
x3	:	52.3483
x5	:	72.3248
x6	:	72.1437
t1	:	1.5
t2	:	4.5
fast_time	:	1.0
slow_time	:	2.0
thmax	:	2.0
thmin	:	0.3
gain1	:	1.0
gain2	:	0.006

see Chapter 4 for the definition of these entities).

The following points ensure the consistency of 'PULSE'.

(1) 'PULSE' displays the listing of compartments, flow and neural control to help the user and also to stop entry under wrong selection.

(2) If a parameter for a compartment, flow or neural control is already entered, the package 'PULSE' does not allow these quantities to be added under the same condition again. However, these entries can be edited for any mistakes that might have occurred.

(3) 'PULSE' does not allow the parameters to be entered without the structure and the condition being defined first (mentioned in point 3 above).

(4) 'PULSE' generates some of the entities (e.g. `index` in compartment definition, refer to Chapter 4 for the definition of this entity) to avoid any mistake and maintain consistency.

The model was completely reproduced in 'PULSE'. No variables were omitted.

Finally the model was ready to be simulated. But, the final consistency test was carried out automatically by 'PULSE' by checking if there are any floating compartments (any compartment that does not have any flow going to it or coming from it). Once this test is passed successfully, 'PULSE' prompts for the simulation variable values shown below (see Chapter 4 for more details):

```
For the 19-segment model :
        is model controlled ? Y
if Y then
                is peripheral resistance controller used ? Y
                is myocardial contractility controller used ? Y
                is venous tone controller used ? Y
endif
step length ? 0.0005
initial heart period ? 0.7482
simulation time ? 14
```

Finally 'PULSE' prompts for four variables to be selected for the simulation.

So far, the consistency and completeness of 'PULSE' has been shown. At this stage the accuracy of the simulation results will be discussed.

6.2.2 Accuracy of the Simulation Results in 'PULSE'

To verify the simulation, two aspects of comparison are considered. 1) the qualitative shape of the simulated variables, 2) the range of limits of the simulated variables. The actual comparison of numerical values was not possible since these values are not quoted by Pullen (1976) (but these results could be read from the graph and should be satisfactory). However, the end of cardiac cycle variables were compared quantitatively.

Fig. 6.1 shows the result obtained by Pullen (1976) for the 19-segment model when it is in a steady state. This figure is used as a reference guide throughout this section. The subscripts used here for this model are derived from the earlier work on this model.

The simulation results for the 19-segment model are achieved by computing the solution from t=0 for a sufficiently a long time until the constraints in the approach have decayed to a negligible level (dynamic equilibrium, about 14 seconds real time). The accuracy of the simulation results was verified by observing the variables such as pressure, volume, flow, etc and verifying them against the already produced results in Fig. 6.1 (Pullen, 1976). The following results are obtained when the 19-segment model was in the steady state.

6.2.2.1 Pressure verification in the simulation

The waveforms of selected transmural pressures during one cardiac cycle are shown in Fig. 6.2 and Fig. 6.3. A dichrotic notch (incisura) occurs in the ascending aortic pressure waveform. The changes of the pressure waveform with distance from the heart can be seen by computing P_{A01} (ascending aorta pressure), P_{A03} (thoracic aorta pressure), P_{AA}(abdominal arteries pressure), and P_{CA} (leg arteries pressure), as in Fig. 6.2. The further from the heart, the longer until peak (systolic) pressure is reached. The systolic pressure increases with distance from the heart owing to variation in wave velocity in the arterial tree. Transmural pressures in the heart chambers are depicted in Fig. 6.3. It is quite clear that when atria are contracting, the ventricles are filling. From comparison of Figs. 6.1, 6.2 and 6.3; it can be seen that the shapes of the waveform do agree and also the ranges of limits are satisfactory.

The waveforms of pulmonary circulation (P_{PA}, P_{PV}) and intestinal circulation (P_{IA}, P_{IV}) are shown in Fig. 6.4. As the right ventricle is filling, the pulmonary arteries pressure start increasing but for the same reason explained above the right ventricular peak is 46mmHg whereas the pulmonary arteries' peak is 25mmHg. The intestinal circulation was not shown by Pullen (1976), but the waveform of the pulmonary arteries in Figs. 6.1 and 6.4 are matching in both the above aspects.

The waveforms of venous compartments' pressures (abdominal veins, P_{AV}, leg veins, P_{CV}, inferior vena cava, P_{IVC}, superior vena cava, P_{SVC}) are more or less constant since the blood pressure in the veins is not very high. The inferior vena cava's pressure is much higher than that of the superior vena cava. This is due to the fact that it has to overcome the gravitational pull (see Chapter 3). These waveforms are not shown by Pullen (1976), however, it can be deduced that most probably they would agree, since the variables in the cardiovascular modelling are highly inter-connected, and since arterial and heart pressures match there is no reason why venous pressure should not be in agreement.

6.2.2.2 Elastance verification in the simulation

The waveforms of time-varying elastances are shown in Fig. 6.6. These are modelled by a half-sinusoid. These will correspond to Fig. 3.5 in Chapter 3. The elastances are based on Beneken's model (1965) (the approach presented here is also the same) and are considered constant. Later (in Chapter 8) a new approach to modelling the heart elastances will be discussed. These waveforms are not shown by Pullen (1976).

6.2.2.3 Volume verification in the simulation

The waveforms of the heart chamber volumes are shown in Fig. 6.7. As the volume of the atria decreases (atrial contraction), ventricular volumes increase until the point is reached when both the atrial and ventricular volumes plateau then the ventricular volumes start decreasing (ventricular contraction, blood flows into the arteries) while the atria are filling. Three phases are shown, atrial contraction (ventricular filling), ventricular contraction (atrial filling), and the resting period. From comparison of Figs. 6.1 and 6.7, the results are acceptable in both above aspects.

The waveforms of arterial and venous volumes (V_{PA}, V_{A01}, V_{SVC}, V_{IVC}) are shown in Fig. 6.8. The venous volumes (V_{SVC}, V_{IVC}) are very high and this is expected since the venous compartments are acting as a reservoir of blood. Veins, unlike arteries, do not have great elasticity (Section 2.2.3 Chapter 3) so they do not stretch as much as the arteries and thus the volumes do not change much. For the arterial volumes in Figs. 6.1 and 6.8, the results are satisfactory in both above aspects, however the venous volumes are not shown by Pullen (1976).

6.2.2.4 Flow verification in the simulation

The waveforms of the heart chamber flows (F_{LVA01}, F_{RVPA}, F_{RARV}, R_{LALV}) are shown in Fig. 6.9. In the simulation the state variables for ventricular outflows are applied with the constraint that $F_{ji}>=0$ to represent the pulmonary and aortic valve actions. From Fig. 6.9 it is seen that the left and right ventricular outflows (F_{LVA01}, F_{RVPA}) decrease rapidly to zero at the end of systole but do not become negative. The volume of the right ventricle is shown to be greater than the volume of the left ventricle (Fig. 6.7) and this is because the right ventricle does not contract as strongly as the left ventricle (Fig. 6.9); this may be the reason behind the pulmonary arteries volume being larger than ascending aorta. Another reason could be that pulmonary arteries carry deoxygenated blood (like veins) and thus act as a reservoir. From the comparison of Fig. 6.1 and ventricular outflow in Fig. 6.9, the results are in agreement in both above aspects.

The waveforms of arterial flows (F_{A01A02}, F_{A02A03}, F_{A03AA}, F_{AACA}) are shown in Fig. 6.10. The arterial flow goes negative which is a backflow from the compartment to which it is going. As the flow moves downwards in the model, it decreases in magnitude. Thus the abdominal arteries flow is very small. From the comparison of Figs. 6.1 and 6.10, the results are acceptable in both the above aspects.

6.2.2.5 Baroreceptor verification in the simulation

The waveforms of aortic arch pressure, upper arteries pressure, aortic arch baroreceptor output and carotid sinus baroreceptor output (P_{A02}, P_{UA}, B_{A02}, B_{UA}) are shown in Fig. 6.11. These two pressures correspond to the input for aortic arch and carotid sinus baroreceptors. The rapid rise at t=0.18s reflects the sensitivity of the baroreceptors to the positive pressure derivative. This is due to the fact that the aortic arch pressure is higher than upper arms pressure and thus the derivative will be higher. From the comparison of Figs. 6.1 and Fig. 6.10, the results match in both above aspects.

6.2.2.6 CNS input verification in the simulation

The waveforms of CNS input for heart rate control, output from peripheral resistance, myocardial contractility and venous tone are shown in Fig. 6.12. Only one CNS input is shown here since for this model the CNS inputs for heart rate control, peripheral resistance control, myocardial contractility control and venous tone control are all the same (Eqns. 3.21, 3.30, 3.35, 3.38 Chapter 3) and are a summation of both baroreceptors. These variables are not shown by Pullen (1976), but since the baroreceptor output was verified (Section 6.2.2.5), there is no reason why these quantities should not agree.

The waveforms of mean arterial pressures are shown in Fig. 6.13 (the numerical results are shown in Fig. 6.14). Pullen (1976) quoted mean ascending aorta pressure (MP_{A01}) and thus this quantity could be compared using the numerical simulation results shown in Fig. 6.14. The accuracy of this quantity is shown in Section 6.2.2.7.

The waveforms of the end of the cardiac cycle calculations (heart rate, estimated total systemic resistance, cardiac output and stroke volume) are shown in Fig. 6.15 (the numerical results are shown in Fig. 6.16). These quantities are quoted by Pullen (1976) and have been compared in Section 6.2.2.7.

6.2.2.7 Verification of the overall behaviour

These are the variables calculated at the end of the cardiac cycle. Table 6.1 shows the comparison of 'PULSE' results against Pullen's (1976; P. 111) results.

T_H	=	(0.823/0.837)(100)	=	%98.327 accuracy
SV	=	(69.6/75.7398)(100)	=	%91.89 accuracy
CO	=	(84.6/90.5112)(100)	=	%93.47 accuracy
P_{A01max}	=	(130.9/136.4080)(100)	=	%95.96 accuracy
P_{A01min}	=	(90.8/96.1234)(100)	=	%94.46 accuracy
ETSR	=	(1.29/1.2182)(100)	=	%105.89 accuracy
MAP	=	(109.1/110.2593)(100)	=	%98.95 accuracy

These results are not exactly the same, but the difference is within an acceptable tolerance. The stroke volume (SV) accuracy is not so good. This is because the ventricular outflow is very fast and since a constant step length integration routine is used here, the ejection flow will cause an error. This effects the cardiac output (CO) and estimated total systemic resistance (ETSR) results.

6.2.3 Usability of 'PULSE'

(1) Simulation time issue:

Pullen has run his model on a CDC 7600 digital computer with the FTN extended FORTRAN IV language operating on that machine. His program was designed to start from the steady state position but he stated that with the pharmacokinetics switched off the time taken (execution time) was 39s to reach t=100 (the machine used by Pullen is very large, and this 39s of CPU time would probably take a very much longer on a PC). The program was run in a batch model and the user was not able to interact with the simulation. The output was produced as a numerical simulation. The model was also reproduced by Leaning and Fardipour (1987, CVS1), for IBM PC using FORTRAN 77 (the output was produced in numerical simulation)..CVS1 reached the steady state in 150s (14 seconds of real time on an XT compatible), while 'PULSE' reached the steady state for the same model with the simulation time (the output was produced in numerical and graphical simulations) of 66s on the same machine and with the same real time.

It can be seen from Table 6.2, that 'PULSE' is faster and thus presumably more efficient than the previous two approaches.

6.3 EXTERNAL VALIDITY

So far the internal validity of 'PULSE' and the approach has been demonstrated using the 19-segment model. For the external validity of the approach, the methodology underlying it has been validated both by a user and by building an extended model to the above 19-segment model.

6.3.1 A Small Model Built by Knut Lande

Knut Lande has developed a highly simplified 4 compartment model illustrated in Fig. 6.17. His main objectives were to challenge the validity of 'PULSE' using a small model. He achieved his goals and his comments are noted in Chapter 7. The pressures in all four compartments are illustrated in Fig. 6.18. The pressures in the heart compartments reproduce the haemodynamic behaviour of the heart. However the model was uncontrolled and needs more parameter adjustment to investigate the arterial and venous compartments. The time taken by Lande to build the model was approximately half a day.

6.3.2 Validity of the Results Obtained from 'PULSE' for the 19-segment model

From the numerical results of arterial and heart pressures (Figs. 6.19, 6.20, refer to Figs. 6.2 and 6.3), it can be seen that the left ventricular pressure waveform has a peak of 189mmHg whereas the ascending aorta pressure has a maximum value of 137mmHg. The large difference between these pressures was reported by Pullen (1976) as being due to the lumped parameter approximation and the way the ejection dynamics have been modelled. In the approach designed here, the Bernoulli term contributes significantly at peak flow since the ejected blood is constrained to flow in a curved elastic artery. (Pullen, 1976; Appendix 5). The peak of the left ventricular pressure is normally quoted as being approximately the same as the peak of ascending aorta pressure (Green, 1972; P.28), but this will only be true when the pressure drop across the opened aortic valve is due to viscous resistance alone, but as explained above because of the way the ejection dynamic is modelled here, the pressure drop is higher. Thus, this could be classified as a criticism of the approach adopted here.

The constant step length does seem to cause some problems with ventricular outflow. The way to overcome it is to use a variable step length based on ventricular outflow.

The transmural left atrial pressure waveform exhibits some correct features but the effects of ventricular contraction on the left atrium do not seem to appear (Figs. 6.3 and 6.20). This is reported by Pullen (1976) as a result of the left atrial segment being mechanically isolated in the 19-segment model. This also applies to the approach here.

These are the negative features of 'PULSE' as discussed above. However, the advantages of this approach over previous ones are that the user is allowed to plot any variables which thus makes the validation more critical. Furthermore, it is easy to use and is very fast.

6.3.3 An Extension to the 19-Segment Model

Hepatic and splenic circulations play an important role in clinical research and in the circulation of blood. It is here that the diseases related to the liver and spleen are investigated and thus interests from clinicians working in this field made it a major task to develop a model including hepatic and splenic circulation. The aim of building such a model is to show the ease of building a model in 'PULSE' as well as highlighting the pragmatic validity (to satisfy the user's objective, which is to build a model of hepatic and splenic circulation) and the heuristic validity (the potential of the approach for scientific explanation, discovery, hypothesis testing etc).

6.3.3.1 Background anatomy

Below the diaphragm the descending aorta is termed the abdominal aorta (Fig. 6.21). In the lower abdominal cavity the aorta itself divides to form the left and right common iliac arteries, which deliver blood to the lower extremities and pelvic structures. Thus, in the extended model here, the abdominal aorta is an arterial compartment and the iliac arteries are lumped as leg arteries, in the same fashion as in the Benken and De Wit (1967), Pullen (1976) and Leaning et al. (1983) models. The abdominal aorta branches off into the coeliac trunk which, in term, has three branches, left gastric (which supplies blood to the stomach), hepatic (supplies blood to the liver and gall bladder and branches into the right gastric and gastro duodenal arteries) and splenic (which is a largest branch of coeliac and supplies blood to the spleen, pancreas and stomach). The superior mesentric branch of the abdominal aorta supplies blood to the small intestine and part of the large intestine. The suprarenal branch of the abdominal aorta supplies the blood to adrenal glands and the renal branch supplies blood to the kidneys. The testicular or ovarian branch of abdominal aorta supplies blood to the testes or ovaries. The inferior mesentric branch of the abdominal aorta supplies blood to the large intestine including the rectum (Soloman & Davis, 1983).

6.3.3.2 Lumped representation of hepatic and splenic circulation

In order to develop a model of these circulations, a research study of these phenomena was undertaken. After discussion with clinicians from the Royal Free Hospital (Dr. A. Hilson) and Dr Akiwumi (TUDU Clinic, Accra), it was agreed that the major arterial branches of the abdominal blood circulation are as depicted in Fig. 6.22.

In Fig. 6.22, some arteries are lumped together, since these circulations are currently thought to have less importance than others. The renal and mesentric circulations were not of primary interest at this time, thus these two are lumped with the lower abdominal branch. If, however, the renal circulation was of interest then it could have been separated out. In this figure (Fig. 6.22) the portal vein empties into the hepatic vein and the mesentrics are emptied into the inferior vena cava (since they are lumped with the lower abdominal arteries), but in fact the mesentrics empty into the portal vein in reality. In the case of a liver disease such as cirrhosis, the alcohol destroys the tissues on the mouth of the portal vein where it enters the hepatic vein and thus the surgeon operating on a patient with cirrhosis has to open a shunt between the portal vein and the inferior vena cava so that the blood empties into the inferior vena cava. In the case of a liver disease such as hepatitis B, the pressure in the portal vein and lower extremities increases since the liver is not functioning properly and again the surgeon has to open a shunt to reduce the pressure. Modelling a circulation such as the hepatic is very interesting from the clinician's point of view since both the normal and diseased states can be studied.

6.3.3.3 Derivation of parameter values for the 23-segment model

The 23-segment model was compared with the 19-segment model. The difference between the gut circulation of the 19-segment model and the 23-segment model lies in the abdominal section, and thus the heart, large arteries leaving the heart, pulmonary model and upper section of the model have not been changed. The approximation is as follows :

thoracic aorta (A03) = thoracic aorta
hepatic artery (HA) + spleen artery (SA) = intestinal artery
leg arteries (CA) = leg arteries
lower abdominal & mesentric arteries + abdominal aorta (LAA)= abdominal arteries
hepatic vein (HV) + spleen vein (SV) + portal vein (PV) = intestinal vein (IV)
lower abdominal and mesentric veins = abdominal veins
leg veins (CV) = leg veins (CV)

By comparison of the gut circulation of the 19-segment model and the 23-segment model, the parameter values for the 23-segment model (see the above approximation) were calculated (the rest of the 23-segment model is the same as the 19-segment model).

The unknown parameters for the arterial class (refer to Chapter 3 for definitions of these parameters) are $\mathbf{C_i}, \mathbf{V_{ui}}, \mathbf{K_i}, \mathbf{V_{io}}, \mathbf{P_{region}}$. The unknown parameters for the venous class are $\mathbf{C_{io}}, \mathbf{V_{ui}}, \alpha_i, \mathbf{V_{io}}, \mathbf{P_{region}}$, **neural**. The unknown flow parameters for the flow from arterial class are **Rji**, $\mathbf{len_{ji}}$, $\mathbf{angl_{ji}}$, $\mathbf{L_{ji}}$, $\mathbf{F_{ji}}$, $\mathbf{P_{cc}}$, neural. The unknown flow parameters for the flow from the venous class are $\mathbf{R_{ji}}$, $\mathbf{len_{ji}}$, $\mathbf{angl_{ji}}$, β_i, $\mathbf{F_{ji}}$, **valve**. Pregion which is a regional pressure, has been explained in Chapter 3. Benken & De Wit (1967) in their circulatory fluid mechanics model assume a constant intra-thoracic pressure (P_{TH}) of -4 mmHg and a constant intra-abdominal pressure (P_{ABD}) of +4 mmHg, both pressures being measured relative to atmospheric pressure. These values were incorporated in the 23-segment model built here. So the values are :

Pregion thoracic aorta (A03)	=	- 4
Pregion hepatic artery (HA)	=	+4
Pregion abdominal aorta (A04)	=	+4
Pregion spleen artery (SA)	=	+4
Pregion lower abdominal arteries (LAA)	=	+4
Pregion hepatic vein (HV)	=	+4
Pregion portal vein (PV)	=	+4
Pregion spleen vein (SV)	=	+4
Pregion lower abdominal vein (LAV)	=	+4

In flow parameters all the $\mathbf{F_{ji}}$ values are assumed to be zero since these are the initial flows.

All the $\mathbf{K_i}$ values are set to 0.04 since $\mathbf{K_i}$ is a visco-elasticity time constant and is calculated by Beneken and De Wit (1967) to be 0.04s. Pullen (1976) has tested a 19-segment model for the effect of changing the visco-elastic time constant for systemic arterial walls on the steady state of the model. He observed that below the normal value of 0.04s, the control mechanisms are unable to maintain the blood pressure and stroke volume at normal values. He also discovered that the effect of removing arterial wall visco-elasticity altogether on the input impedance of the systemic circulation model results in the modulus and phase angle becoming highly oscillatory and thus unrealistic. Hence a normal values of 0.04s is incorporated in the 23-segment model.

As has been shown in Chapter 3, the effect of the venous segment collapsing will result in a change in the compliance and resistance of the segment. Snyder & Rideout (1969) considered the compliance to be a piecewise linear function of volume with the compliance increasing to 20 times its normal value when the transmural pressure becomes negative ($V_i<V_{ui}$). Thus, following Snyder & Rideout (1969), the compliance constant α_i is set here to 20 for the venous compartments in the 23-segment model.

Pullen (1976) has incorporated the effect of neural control on R_{BRONC}, R_{INT}, R_{ABD} and R_{LEG} connections while the rest he assumed to be uninfluenced by the central nervous control. He also adapted the Rushmer (1976) approach which showed that the pulmonary vasculature was highly non-reactive to both neural and hormonal control, and thus it was excluded from peripheral resistance control in the 19-segment model.

Pullen's approach was used in the thoracic aorta to right atrium connection (R_{BRONC}), hepatic artery to hepatic vein connection ($R_{HEPATIC}$), spleen artery to spleen vein connection (R_{SPLEEN}), lower abdominal arteries to lower abdominal veins connection (R_{LABD}) and leg arteries to leg veins connection (R_{LEG}) while the others were uneffected (in the flow from arterial class connection). Thus **neural** parameter is set to true for these flows and set to false for the rest.

In the venous compartment, the compliance and unstressed volume are divided by dimensionless neural control variables. This division is required to ensure that the control system exhibits negative feedback, so that if the blood pressure rises, venous tone will decrease and the venous compliance and unstressed volume will increase, thus lowering the venous pressure. This results in reduced venous return and reduced cardiac output so that the original blood pressure rise will be limited. Thus **neural** control is true for all the venous compartments except pulmonary veins (Rushmer 1976), superior vena cava and inferior vena cava (in the parameters for the venous class).

The P_{cc} parameter in arterial connections only applies to pulmonary circulation (refer to Chapter 3), and thus the value is set to zero for the rest.

In the circulatory fluid mechanics of Beneken & De Wit (1967), a venous valve is included between the segments representing leg veins and abdominal veins and this valve obstructs the backflow completely. In the 23-segment model the **valve** parameter was only incorporated between the leg veins and the lower abdominal veins and is set to false for the rest. The β_i parameter is a constant used in multiplying the flow when a venous segment is collapsing. In the 19-segment model, Pullen (1976) incorporated the following values :

β_{IV}	(intestinal vein)	=1.0
β_{AV}	(abdominal veins)	=1.0
β_{CV}	(leg veins)	=0.0

which meant that there exists a valve between the leg veins and abdominal veins (so for the leg section the value is set to zero), while for the rest the valve does not exists (the values are set to one).

Here, adapting the same approach, the following values are incorporated into the 23-segment model :

β_{PV}	(portal veins)	=1.0
β_{SV}	(spleen vein)	=1.0
β_{LAV}	(lower abdominal veins)	=1.0
β_{HV}	(hepatic vein)	=1.0
β_{CV}	(leg veins)	=0.0

In arterial connections, **anglji** is an angle between the axis of the segment and a perpendicular to the direction of gravitational force (there is a maximum of three sets of values; zero, 90, -90 degrees). So all the ArtArt connections are as follows :

$angl_{A03RA}$	(thoracic aorta to right atrium))	= 90
$angl_{A03A04}$	(thoracic aorta to abdominal aorta)	= -90
$angl_{A04SA}$	(abdominal aorta to spleen artery)	= -90
$angl_{A04HA}$	(abdominal aorta to hepatic artery)	= -90
$angl_{A04LAA}$	(abdominal aorta to lower abd. arteries)	= -90
$angl_{LAACA}$	(lower abd. arteries to leg arteries)	= -90

All the ArtVein connections are zero angle since they are not perpendicular. Thus :

$angl_{HAHV}$	(hepatic artery to hepatic vein)	= 0
$angl_{SASV}$	(spleen artery to spleen vein)	= 0
$angl_{LAALAV}$	(lower abd. arteries to lower abd. veins)	= 0
$angl_{CACV}$	(leg arteries to leg veins)	= 0

All the VeinVein connections are as follows :

$angl_{HVIVC}$	(hepatic vein to inferior vena cava)	= 90
$angl_{LAVIVC}$	(lower abd. veins to inferior vena cava)	= 90
$angl_{SVPV}$	(spleen vein to portal vein)	= 90
$angl_{PVHV}$	(portal vein to hepatic vein)	= 90

Al-Dahan (1984) has run extensive validation tests on the 19-segment model and also validated the parameter values and variables incorporated in that model. She compared the parameter value for len_{ji} used in the 19-segment model and the data acquired by literature survey. Al-Dahan (1984) shows that these values differ, and problems relating to the measurement of blood vessel length have also been pin-pointed. She reported a length of 15.9×10^{-2}m from the end of the thoracic aorta to the end of the abdominal aorta according to her literature survey.

She reported the following data: the effective length from thoracic aorta to the abdominal aorta (the end of the thoracic aorta to the end of the abdominal aorta) to be equal to 15.9×10^{-2}m; the length for coeliac artery is 1×10^{-2}m which means that the hepatic artery and splenic artery could be approximated as being 1×10^{-2}m; the distance from the beginning of the abdominal aorta to the beginning of the femoral artery (the artery of the lower limb) could be 12×10^{-2}m and the length for inferior mesentric was reported to 5×10^{-2}m and for superior mesentric to be 5.9×10^{-2}m.

In this thesis using Al-Dahan's survey results:

length from lower abdominal arteries to leg arteries =1.1m (which is derived from 12×10^{-2}m - 5×10^{-2}m - 5.9×10^{-2}m) and similarly

length from leg veins to lower abdominal veins = 1.1m

length from lower abdominal veins to inferior vena cava = 27.9×10^{-2}m (which is derived from 12×10^{-2}m + 15.9×10^{-2}m)

length for hepatic vein to inferior vena cava = 16.9×10^{-2}m which is derived from 1×10^{-2}m + 15.9×10^{-2}m)

Finally the rest are as follows (the same as the approach adopted in 19-segment model by Pullen, 1976):

the length for hepatic artery to hepatic vein = 0

the length for spleen artery to spleen vein = 0

the length for spleen artery to portal vein = 0

the length for portal vein to hepatic vein = 0

the length for leg arteries to leg veins segments = zero

The following values are the same as the corresponding values in the 19-segment model:

$C_{LAA} = 0.21$	$V_{LAA} = 76.104$	$V_{uLAA} = 58.0$
$C_{CA} = 0.12$	$V_{CA} = 73.632$	$V_{uCA} = 63.0$
$C_{CV} = 4.8$	$V_{CV} = 257.857$	$V_{uCV} = 257.0$
$C_{IVC} = 8.3$	$V_{IVC} = 533.075$	$V_{uIVC} = 488.0$
$C_{LAV} = 5.1$	$V_{LAV} = 276.191$	$V_{uLAV} = 305.0$

$R_{LAAA04} = 0.012$	$L_{LAAA04} = 0.014$
$R_{CALAA} = 0.18$	$L_{CALAA} = 0.031$
$R_{CVCA} = 15.0$	$L_{CVCA} = 0$
$R_{LAVLAA} = 57.0$	$L_{LAVLAA} = 0$
$R_{IVCLAV} = 0.595$	
$R_{LAVCV} = 0.3$	

Finally, the following algorithm was adopted to calculate the remainder of unknown compliances, volumes, unstressed volumes, resistances and inertances.

The splenic artery is the largest branch of the coeliac, and the liver receives one quarter of its blood from the hepatic artery and three quarters from the portal veins. According to most physiological textbooks the hepatic artery not only supplies the liver with blood which is fully oxygenated, but may also play an important part in maintaining the pressure of blood in the liver capillaries, for the blood in the portal vein is at a pressure of 8-10 mmHg while that in the hepatic artery is at the systemic arterial pressure.

The derived parameters for hepatic and splenic circulation are as follows :

Arteries

where A03 is thoracic aorta, A04 is abdominal aorta, HA is hepatic artery, SA spleen artery, LAA lower abdominal arteries.

Using 1/3 of the value for LAA connection and 2/3 for A04 connection:

$L_{A04A03} + L_{LAAA04} = L_{AAA03} = 0.014$ $L_{A04A03} = 0.0093$, $L_{LAAA04} = 0.00467$

$R_{A04A03} + R_{AAA04} = R_{AAA03} = 0.012$ $R_{A04A03} = 0.008$, $R_{LAAA04} = 0.004$

$C_{LAA} + C_{A04} = C_{AA} = 0.21$ $C_{A04} = 0.14$, $C_{LAA} = 0.07$

$V_{LAA} + V_{A04} = V_{AA} = 76.1043$ $V_{A04} = 50.736$, $V_{LAA} = 25.368$

$V_{uLAA} + V_{uA04} = V_{uAA} = 58.0$ $V_{uA04} = 38.7$, $V_{uLAA} = 19.3$

Using 1/3 of the value for SA connection and 2/3 for HA connection:

$L_{HAA04} + L_{SAA04} = L_{IAA03} = 0.0027$ $L_{HAA04} = 0.0009$, $L_{SAA04} = 0.0018$

$R_{HAA04} + R_{SAA04} = R_{IAA03} = 0.0014$ $R_{HAA04} = 0.00047$, $R_{SAA04} = 0.00093$

$C_{HA} + C_{SA} = C_{IA} = 0.06$ $C_{HA} = 0.02$, $C_{SA} = 0.04$

$V_{HA} + V_{SA} = V_{IA} = 22.254$ $V_{HA} = 7.418$, $V_{SA} = 14.836$

$V_{uHA} + V_{uSA} = V_{uIA} = 17.0$ $V_{uHA} = 5.67$, $V_{uSA} = 11.33$

Arterio-Venous

where HV is hepatic vein, SV spleen vein, LAV lower abdominal veins.

Using 1/3 of the value for SV connection and 2/3 for HV connection:

$R_{HVHA} + R_{SVSA} = R_{IVIA} = 2.3$ $L_{HVHA} = 0.767$, $L_{SVSA} = 1.53$

$L_{LAVLAA} = 0$

Venous

Using 3/6 of the value for IVCHV connection, 1/6 for PVSV connection and 2/6 for HVPV connection:

$R_{IVCHV} + R_{PVSV} + R_{HVPV} = R_{IVCIV} = 0.166$

$R_{IVCHV} = 0.083, R_{PVSV} = 0.02767, R_{HVPV} = 0.0553$

Using 1/6 of the value for HV connection, 2/6 for SV connection and 3/6 for PV connection:

$C_{HV} + C_{SV} + C_{PV} = C_{IV} = 0.0027 = 10.6$

$C_{HV} = 1.8, C_{SV} = 3.5, C_{PV} = 5.3$

$V_{HV} + V_{SV} + V_{PV} = V_{IV} = 568.38$

$V_{HV} = 94.73, V_{SV} = 189.46, V_{PV} = 284.19$

$V_{uHV} + V_{uSV} + V_{uPV} = V_{uIV} = 607.0$

$V_{uHV} = 101.2, V_{uSV} = 202.33, V_{uPV} = 303.5$

6.4 DISCUSSION OF RESULTS OBTAINED FOR THE 23-SEGMENT MODEL

The model has the same controllers as the 19-segment model and was run for 3 minutes. The results obtained are illustrated in Figs. 6.23, 6.24 and 6.25. The results in Fig. 6.23 show the features of the haemodynamics of the heart. The model takes a few seconds before it starts to stabilize. The heart period has been reduced to 0.67 (tachycardia, fast heart rate).

Figure 6.24 shows the pressure waveforms for the arterial and venous pressures. The arterial pressure is higher than the venous, as expected. The portal pressure is higher than expected (about 8-10mmHg is reported by most physiological textbooks).

Figure 6.25 shows the flow in the abdominal circulation. The arterial flow waveform shows some haemodynamic features, but the venous flow starts with negative flow (portal hypertension) but after a few seconds stabilizes to a positive value.

The results cannot be studied further since the aim of producing this model was to illustrate the pragmatic and heuristic validity of the approach, and since the parameter estimation is not accurate the results could not have been argued against the real life results. However, the model contains some acceptable haemodynamic features and satisfies the user objective. This model can be studied further and the parameter values can be provided from clinical data to provide better results and understanding.

The time taken to extend the 19-segment mode is one and a half hours provided the parameter values were known.

6.5 LONG TERM EFFECT

The objective of the intended range of application for the approach designed here is to study the short-term cardiovascular control mechanisms. However, the approach is invalid for predicting over a time scale greater than 2-3 minutes, such as hormonal effects. In the case of renal circulation, a simplified model of the kidney, together with a model of renin-angiotensin-aldosterone-ADH system, can be built. However, the testing of the hypothesis concerning blood volume regulation and the role of the kidney in blood pressure regulation is not possible with this approach since the regulation is over a long period of time and the approach is not designed for such a time scale.

6.6 CONCLUSION

The validation of any modelling approach designed for a large-scale non-linear system such as the cardiovascular system can never be totally satisfactory because there are only a finite number of validation tests that can be performed. The approach cannot be exhaustively validated because there is an infinite number of validation tests.

The chapter has defined the validation criteria and has then considered the objective of the approach, which relates to short-term haemodynamics cardiovascular models. In this context, the approach has been validated. The validity tests were carried out based on two criteria:

A) Internal validity which consists of the following;

1) Validate the consistency and completeness of the package 'PULSE' and hence the approach;
2) Validate the accuracy of the simulation results against the already obtained results of Pullen (1976);
3) The usability of the approach;

B) External validity which is the validation of the methodology underlying the approach as tested by the following:

1) Validity of the result obtained for the well-validated 19-segment model;
1) Validity from the user point of view (development of 4-segment model);
2) Heuristic and pragmatic validity of the approach by building a comprehensive 23-segment model.

The chapter also describes the negative features of the approach for instance in relations to long-term effects. In the next chapter the results obtained by the user will be discussed.

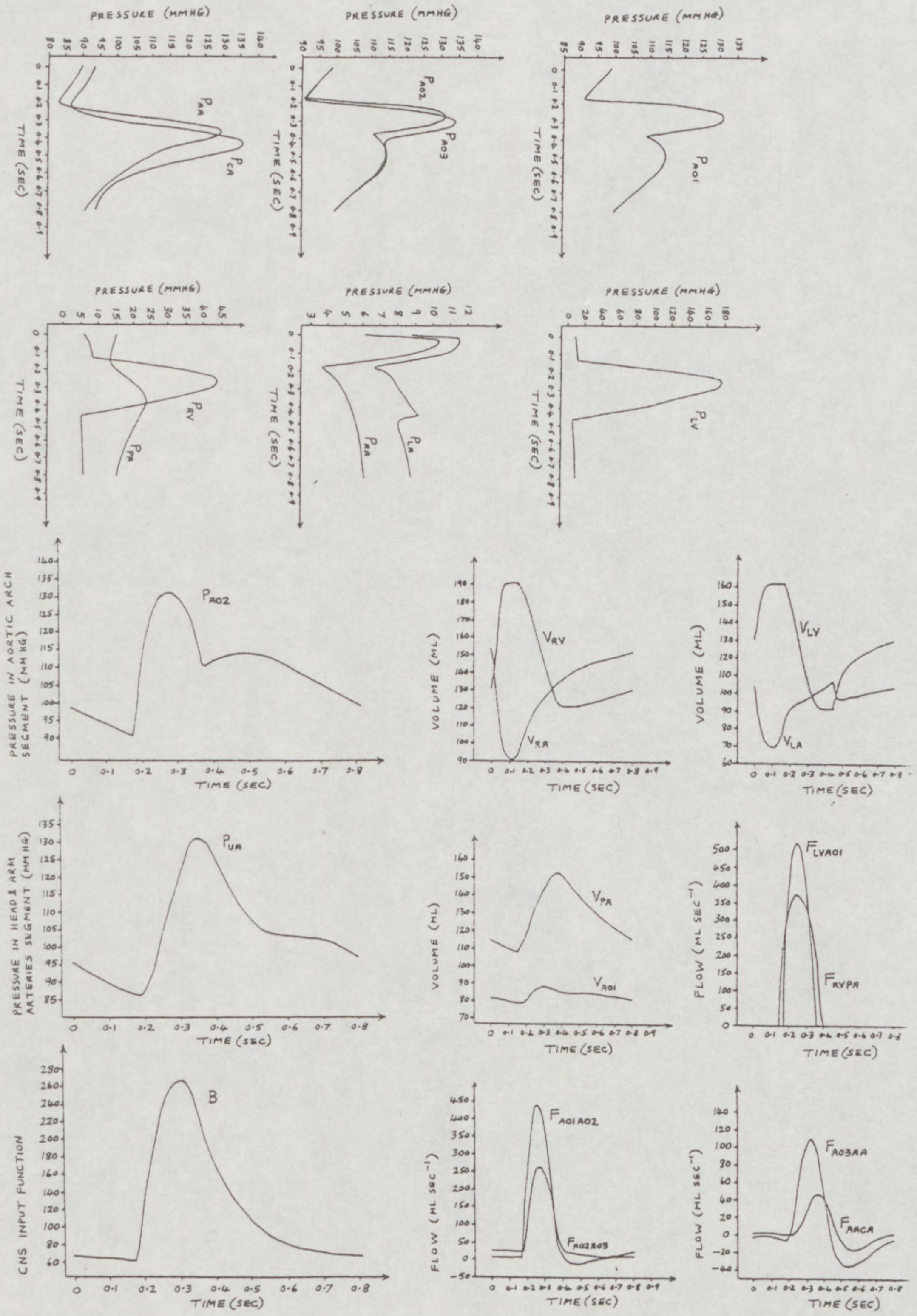


Figure 6.1 **Results for the 19-segment model produced by Pullen (1976)**

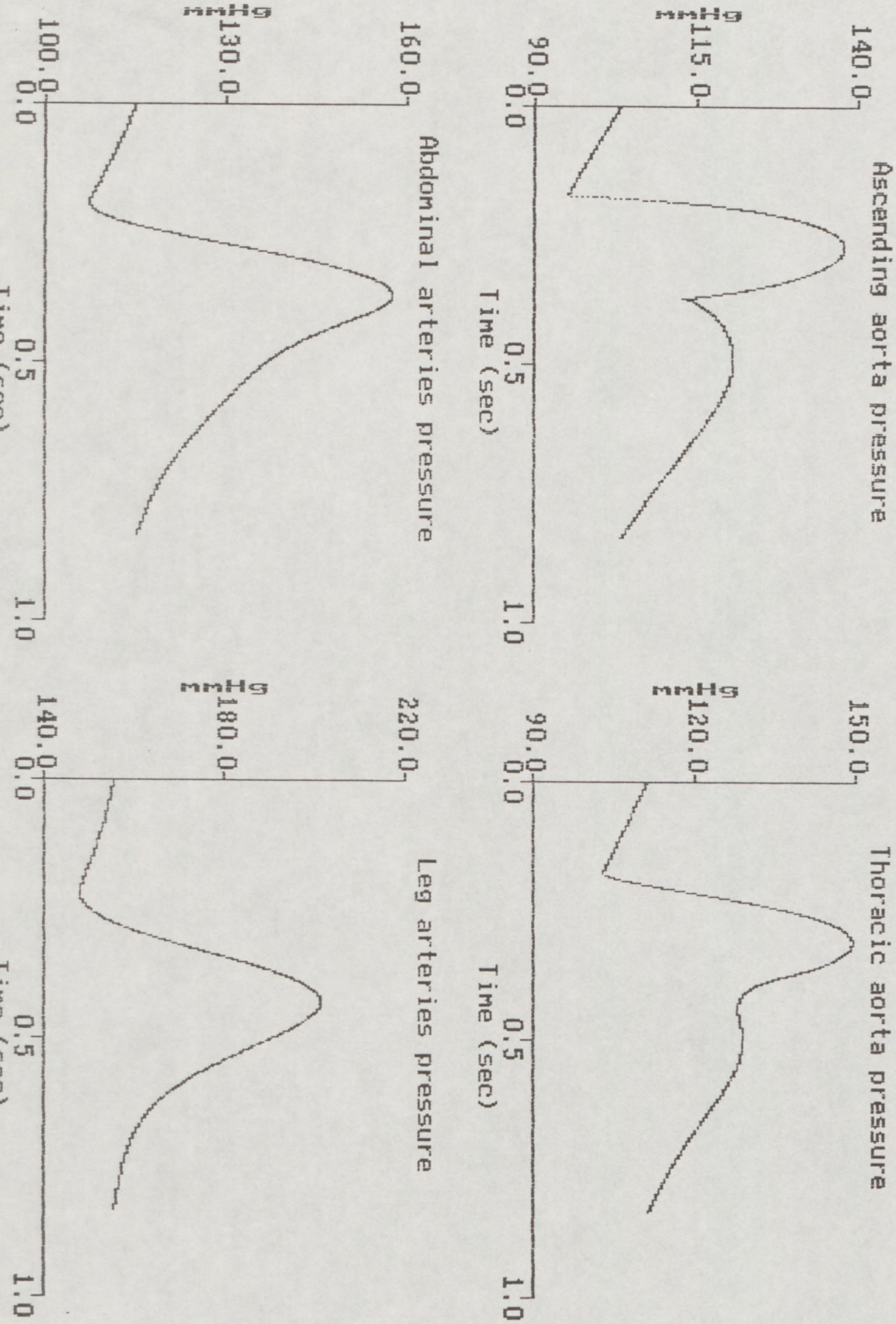


Figure 6.2 **Waveforms of arterial pressure during one cycle in the steady state**

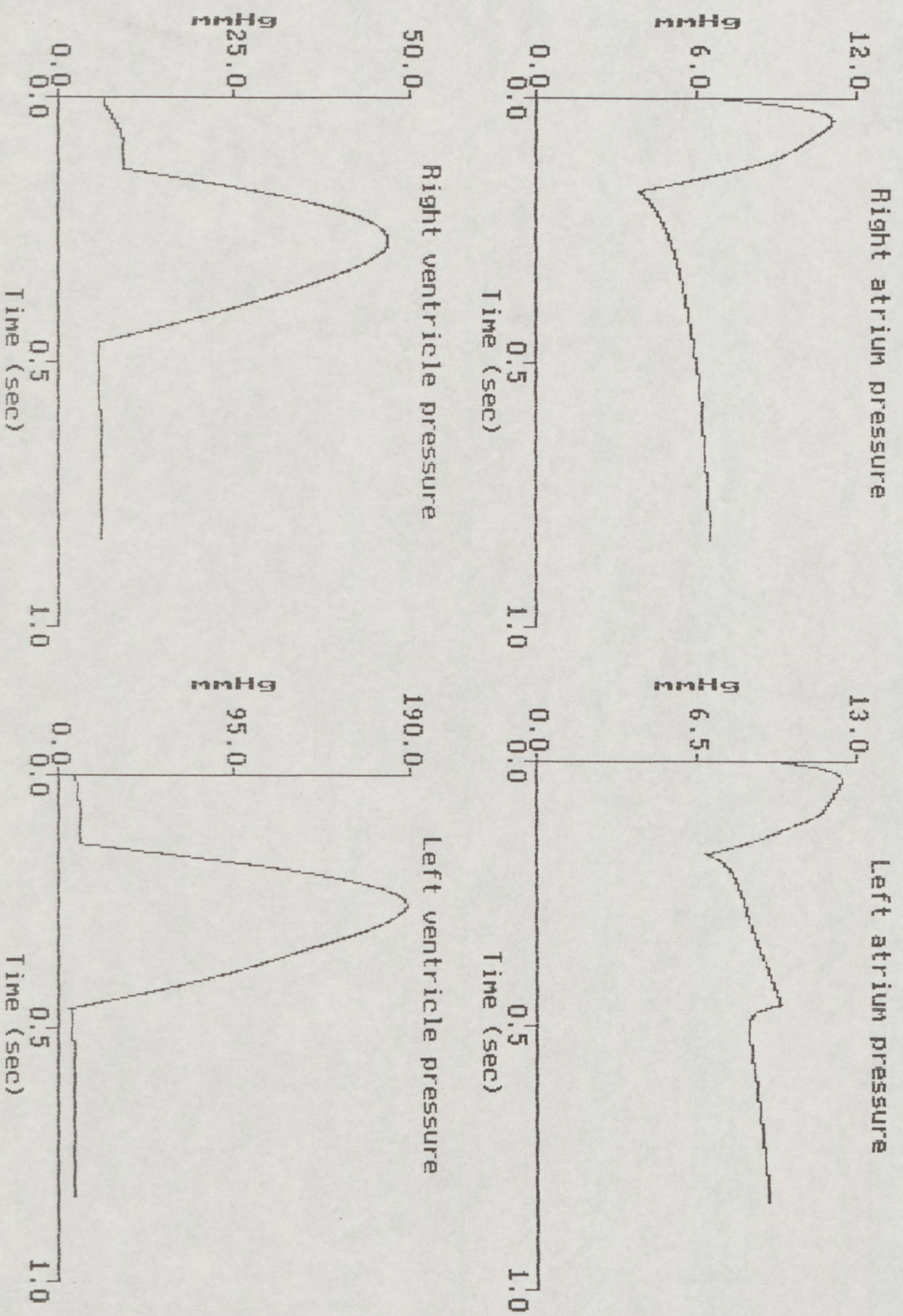


Figure 6.3 **Waveforms of heart pressure during one cycle in the steady state**

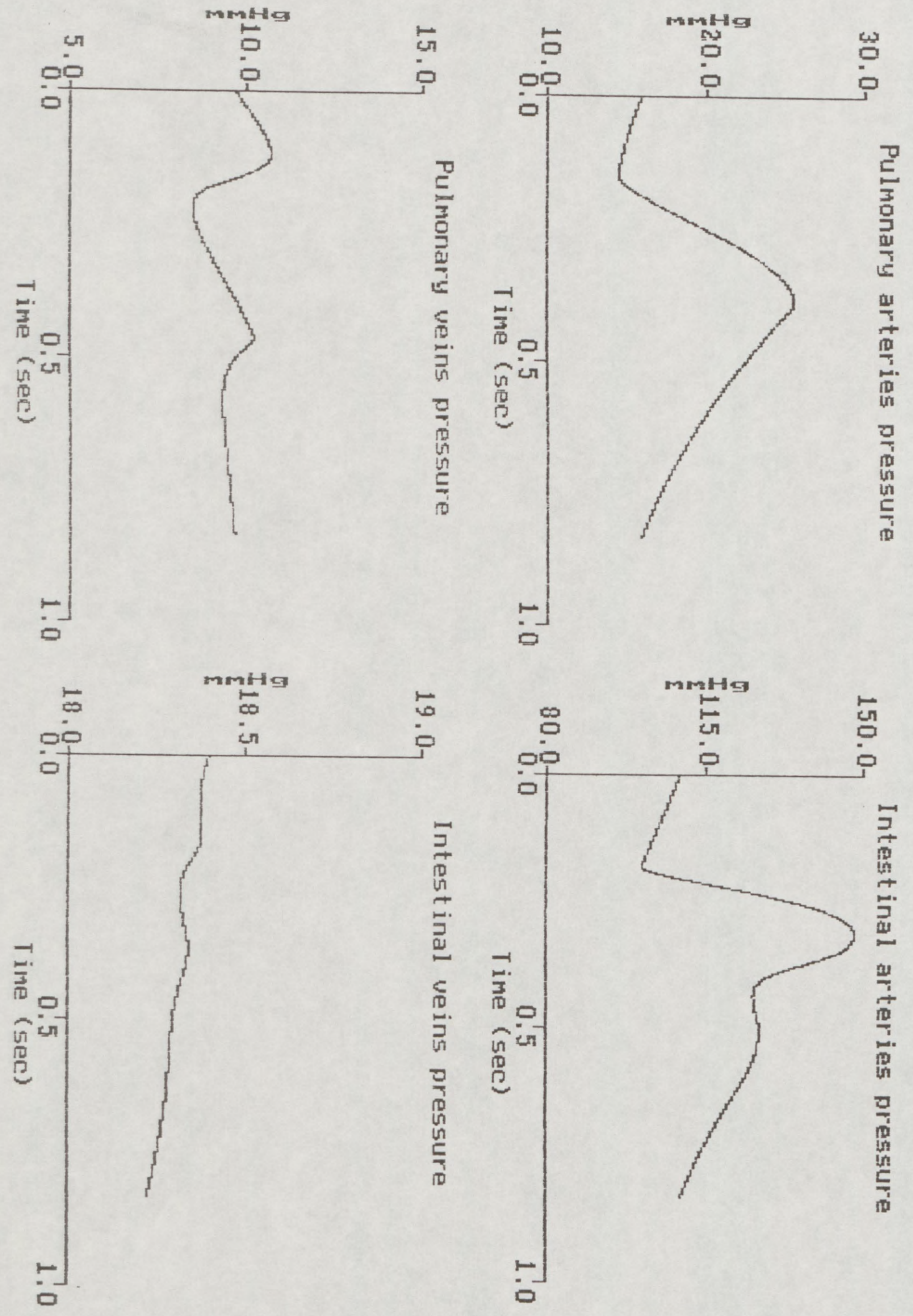


Figure 6.4 **Waveforms of pulmonary and intestinal circulation pressures during one cycle in the steady state**

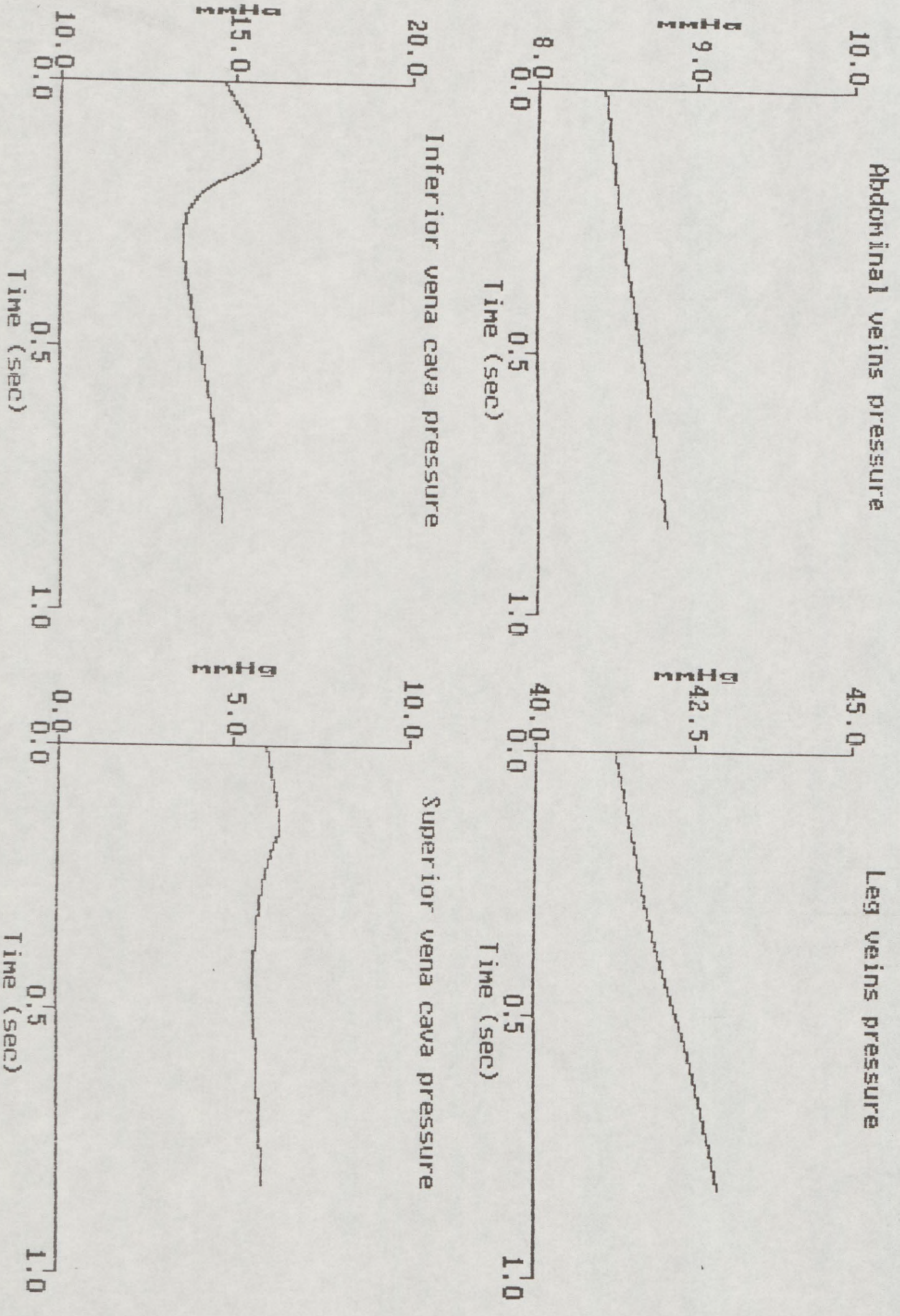


Figure 6.5 **Waveforms of venous pressure during one cycle in the steady state**

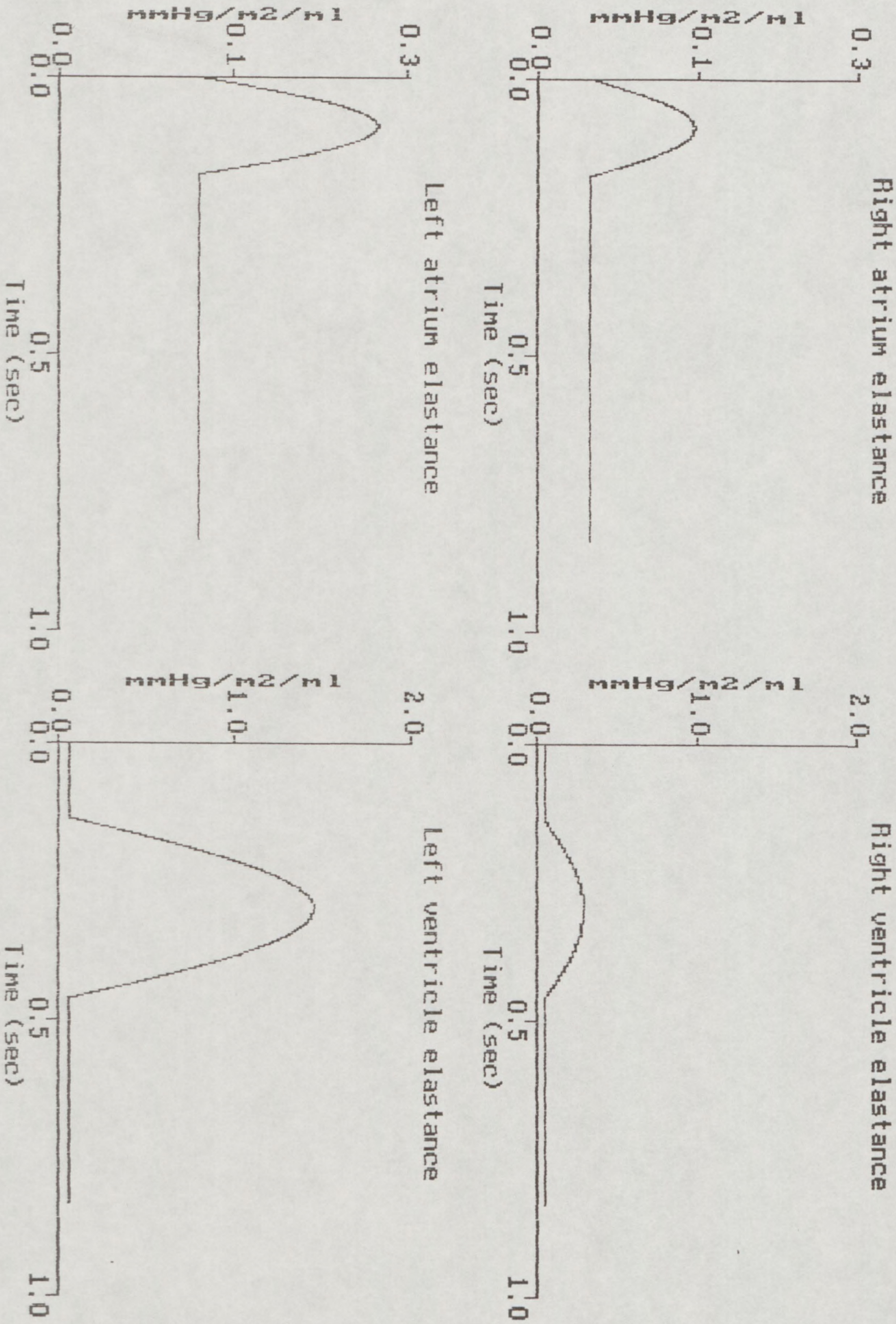


Figure 6.6 **Waveforms of heart elastances during one cycle in the steady state**

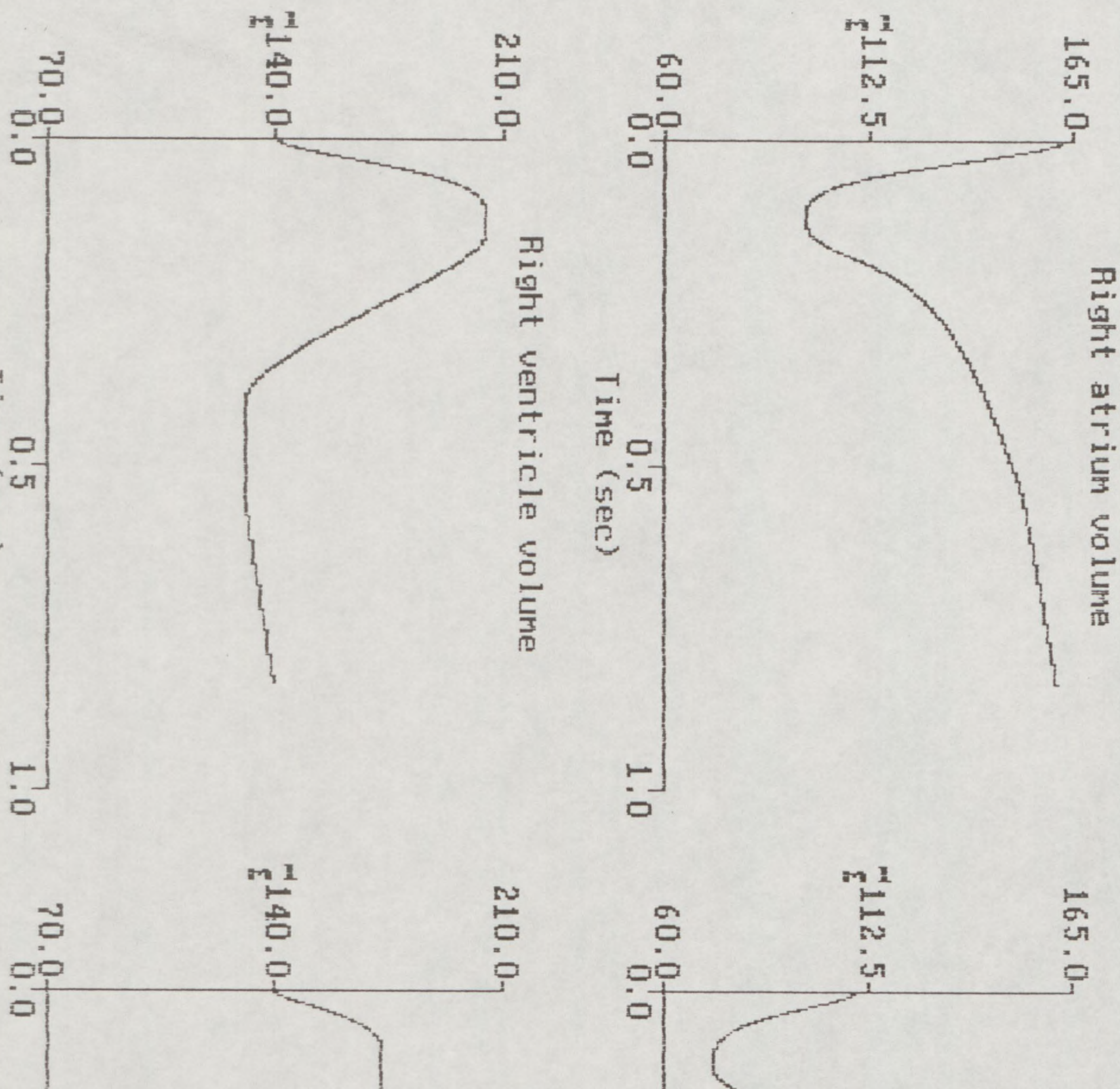


Figure 6.7 **Waveforms of heart volumes during one cycle in the steady state**

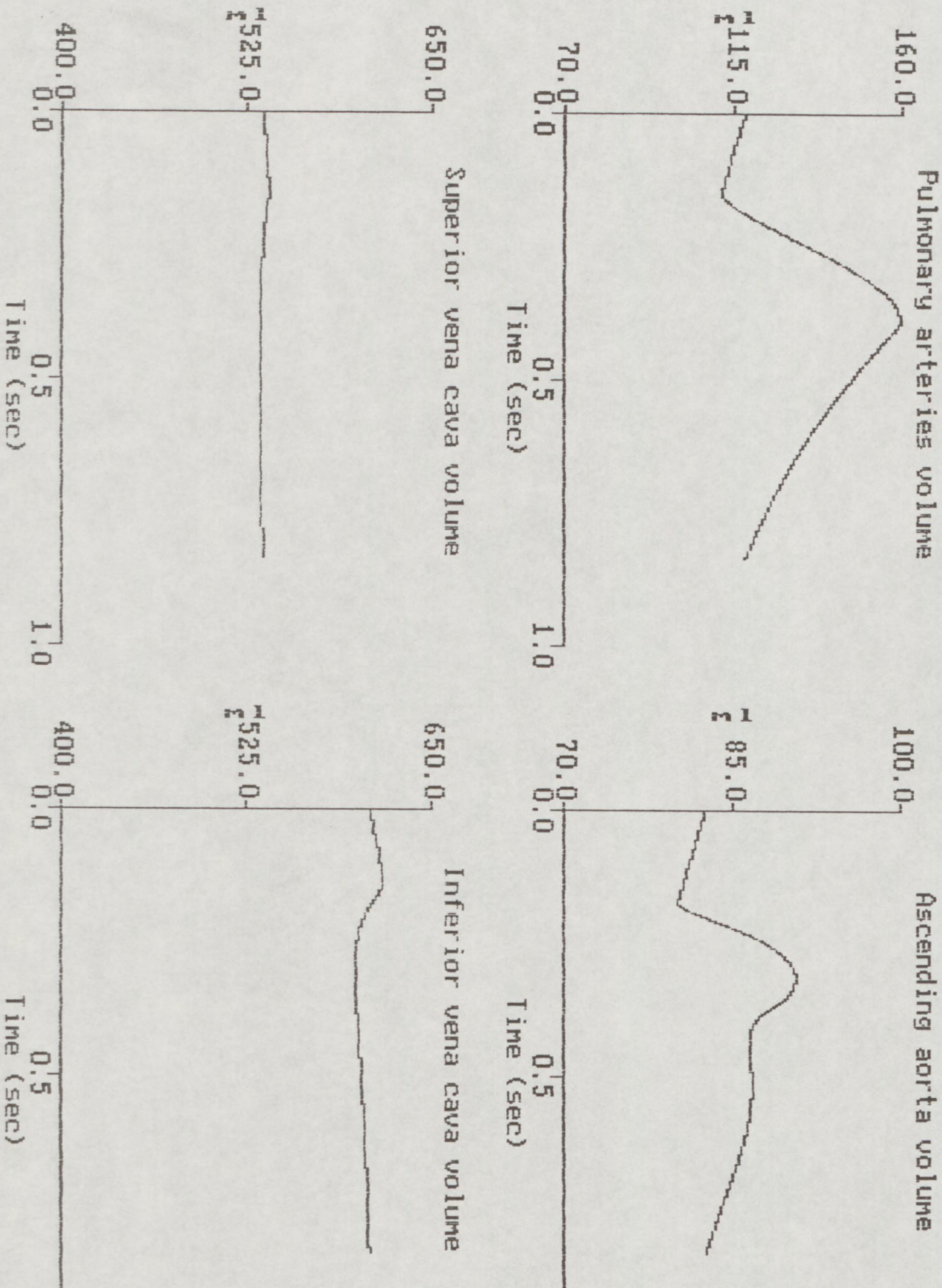


Figure 6.8 **Waveforms of arterial and venous volumes during one cycle in the steady state**

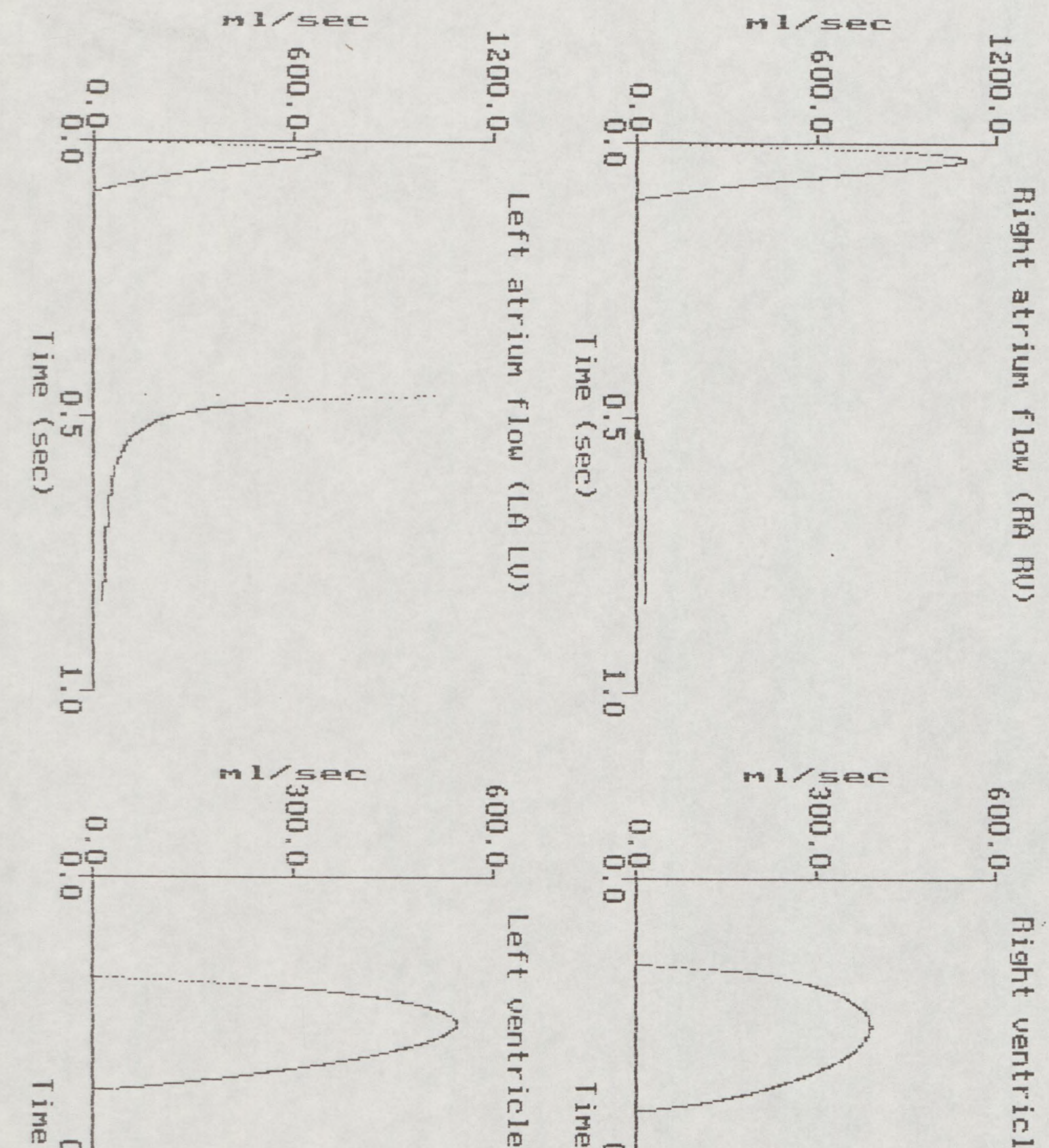


Figure 6.9 **Waveforms of heart flows during one cycle in the steady state**

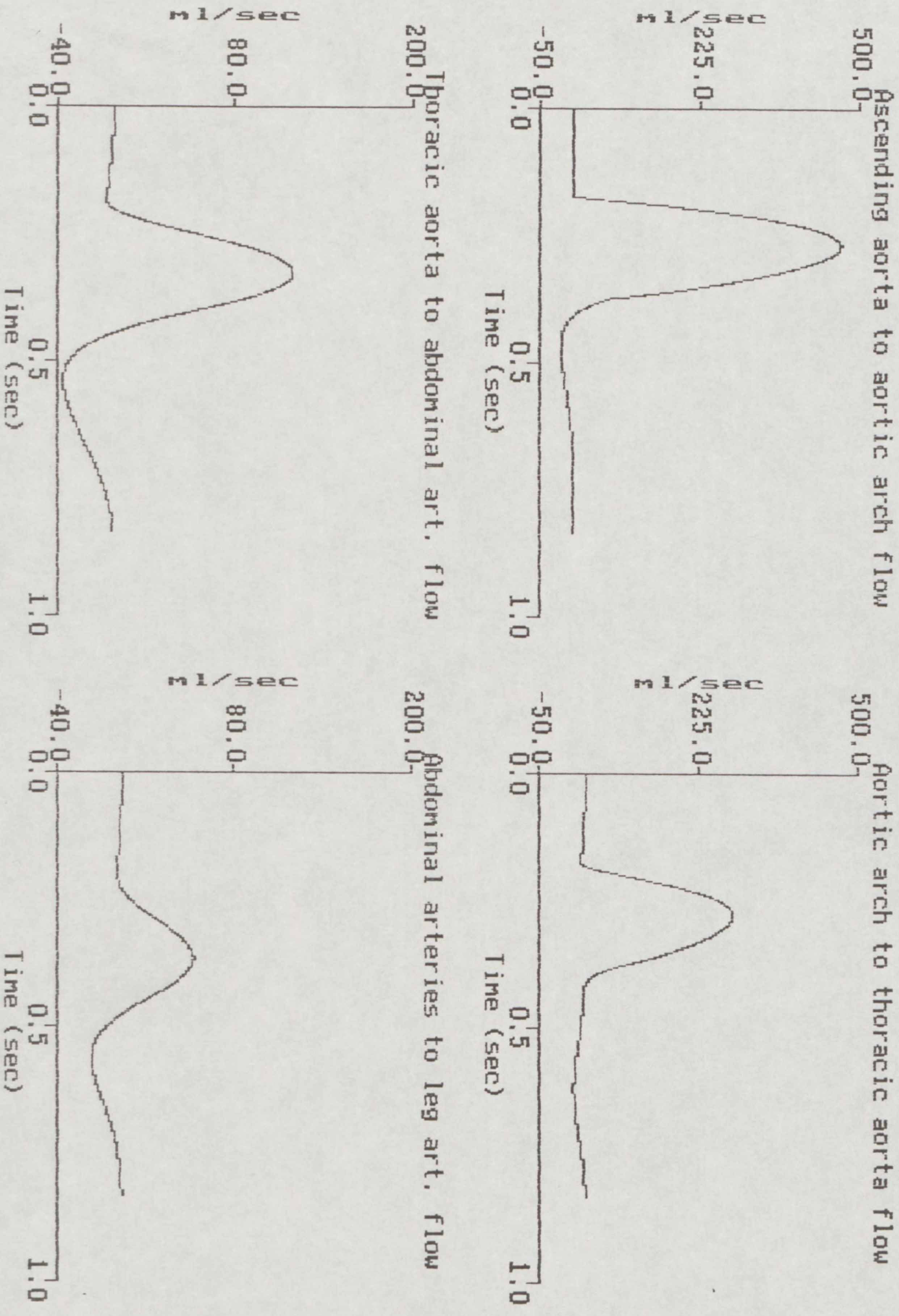


Figure 6.10 **Waveforms of arterial flows during one cycle in the steady state**

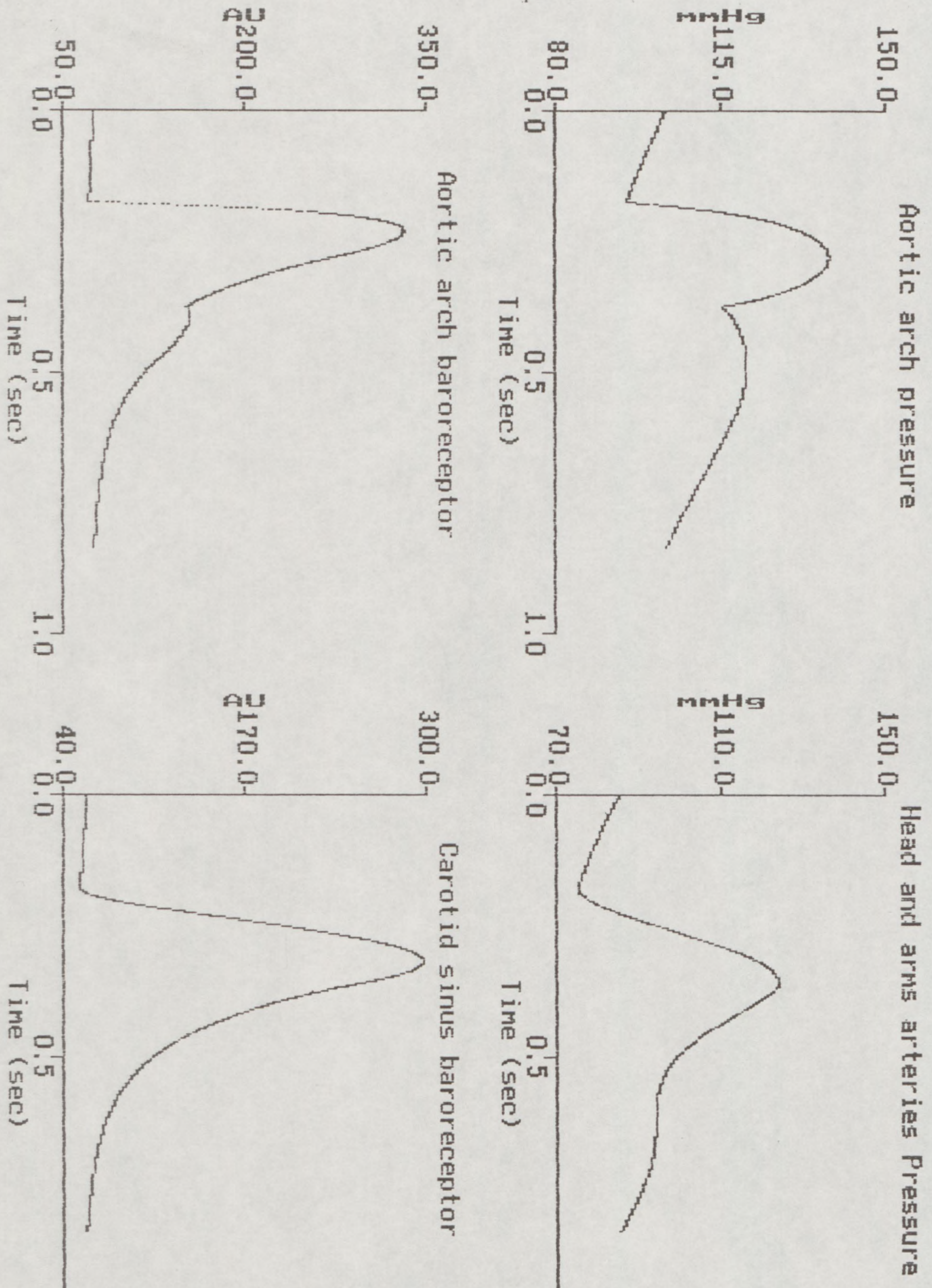


Figure 6.11 **Waveforms of baroreceptor outputs during one cycle in the steady state**

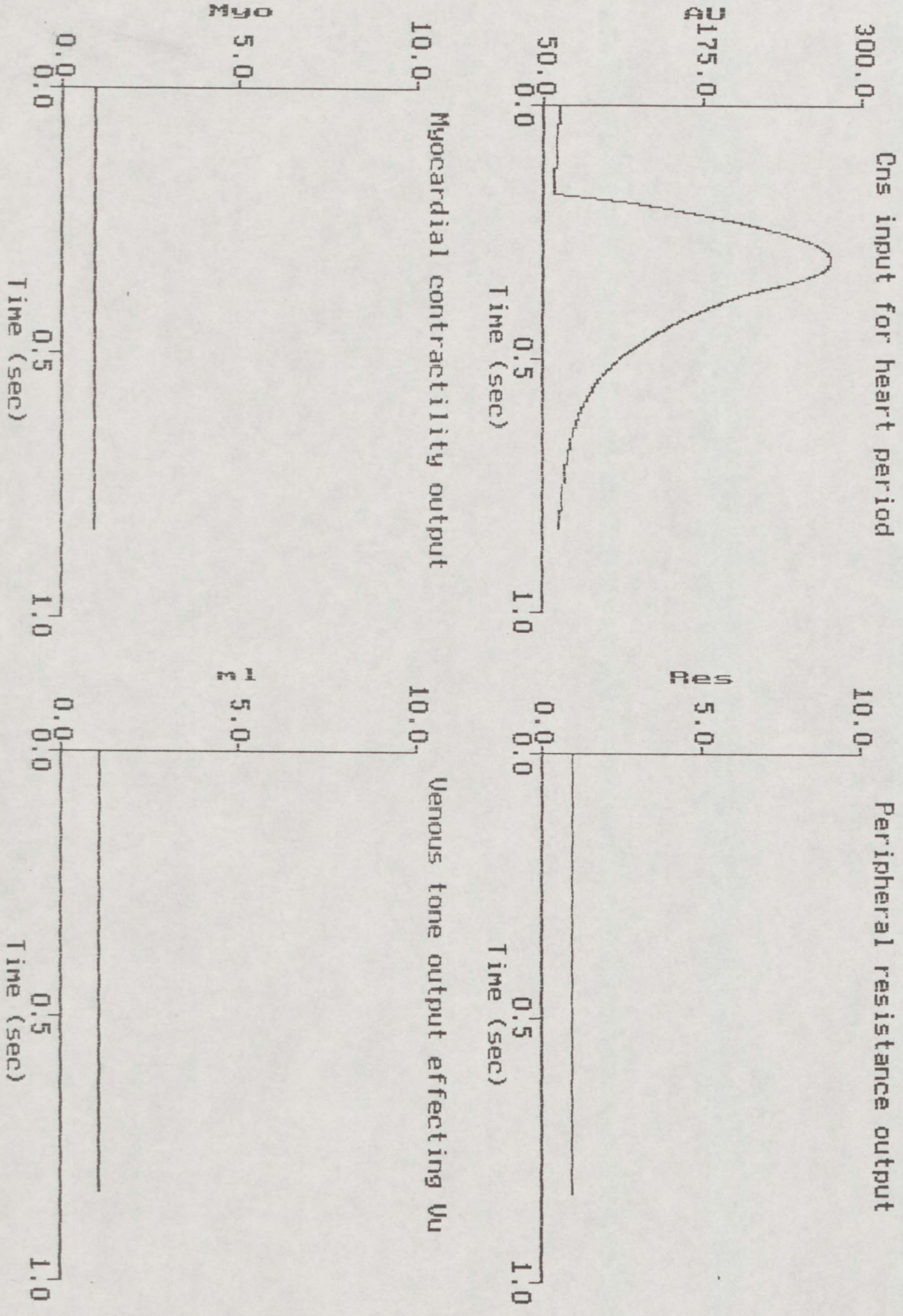


Figure 6.12 **Waveforms of cns input variables during one cycle in the steady state**

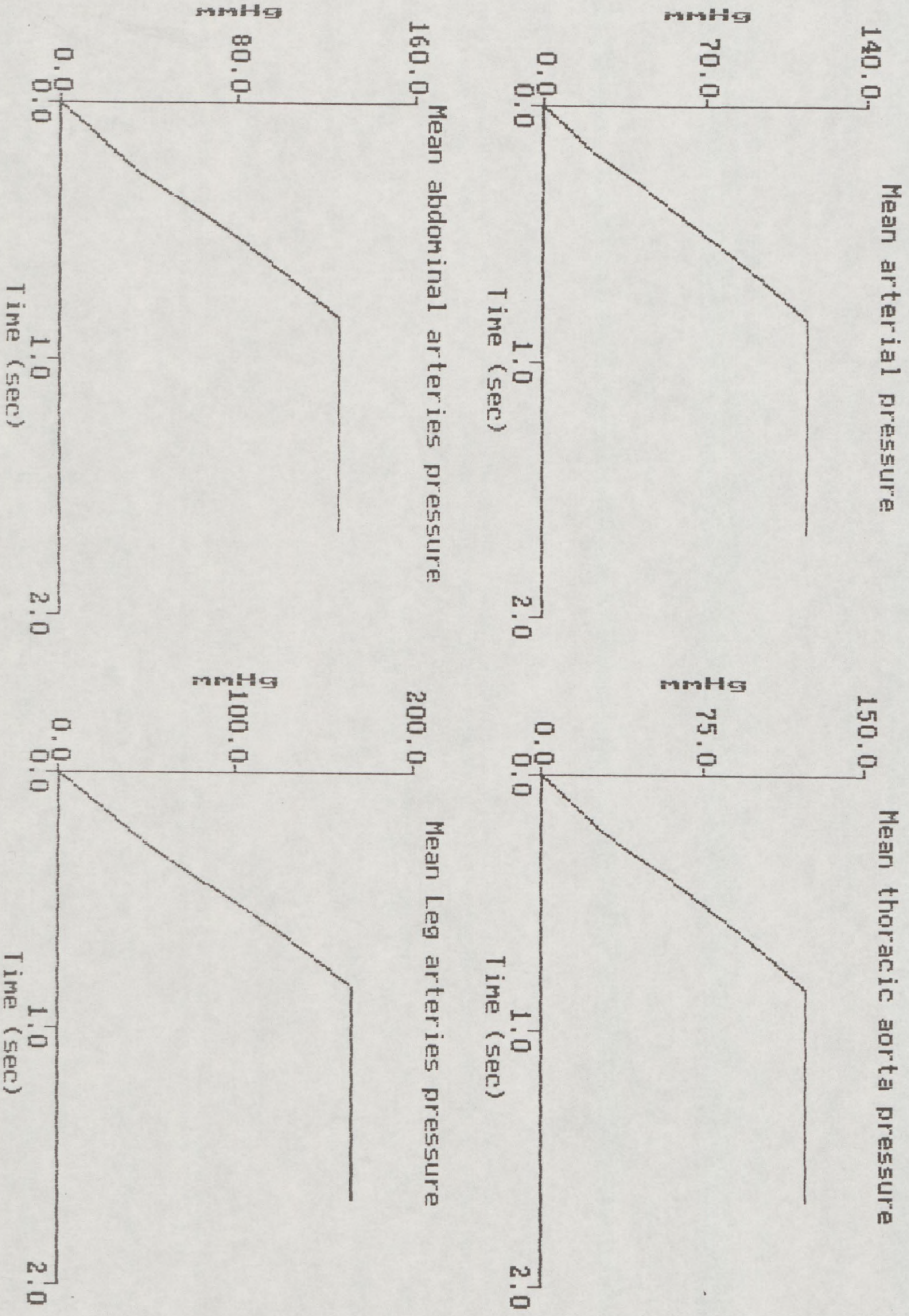


Figure 6.13 **Waveforms of mean arterial pressures during two cycle in the steady state**

```
      MPA01        Mean arterial pressure

      MPA03        Mean thoracic aorta pressure

      MPAA         Mean abdominal arteries pressure

      MPCA         Mean Leg arteries pressure


          ***** During Cardiac Cycle *****

    Time          MPA01          MPA03          MPAA           MPCA
     0.0000        0.0000         0.0000         0.0000         0.0000
     0.0505        6.1032         6.5716         6.8399         9.2715
     0.1005       12.0593        13.0049        13.5674        18.4583
     0.1505       17.8727        19.2988        20.1665        27.5372
     0.2005       24.0849        25.5475        26.6310        36.4850
     0.2505       31.9353        32.9916        33.3763        45.3629
     0.3005       40.1495        41.6248        41.2569        54.6462
     0.3505       48.0287        50.4810        50.2725        64.9680
     0.4005       55.0523        58.7088        59.6717        76.4407
     0.4505       62.1573        66.3858        68.6260        88.4587
     0.5005       69.3582        74.0597        77.0280       100.2072
     0.5505       76.5429        81.7390        85.0722       111.2702
     0.6005       83.6278        89.3112        92.8570       121.6620
     0.6505       90.5567        96.7024       100.4029       131.5811
     0.7005       97.3047       103.8919       107.7334       141.2212
     0.7505      103.8695       110.8918       114.8917       150.7076
     0.8005      110.2593       117.7241       121.9198       160.1033

          *****  End Cardiac Cycle  *****
      MPA01        Mean arterial pressure

      MPA03        Mean thoracic aorta pressure

      MPAA         Mean abdominal arteries pressure

      MPCA         Mean Leg arteries pressure


          ***** During Cardiac Cycle *****

    Time          MPA01          MPA03          MPAA           MPCA
     0.0000      114.8816       122.6828       127.0515       167.0117
     0.0505      114.8816       122.6828       127.0515       167.0117
     0.1005      114.8816       122.6828       127.0515       167.0117
     0.1505      114.8816       122.6828       127.0515       167.0117
     0.2005      114.8816       122.6828       127.0515       167.0117
     0.2505      114.8816       122.6828       127.0515       167.0117
     0.3005      114.8816       122.6828       127.0515       167.0117
     0.3505      114.8816       122.6828       127.0515       167.0117
     0.4005      114.8816       122.6828       127.0515       167.0117
     0.4505      114.8816       122.6828       127.0515       167.0117
     0.5005      114.8816       122.6828       127.0515       167.0117
     0.5505      114.8816       122.6828       127.0515       167.0117
     0.6005      114.8816       122.6828       127.0515       167.0117
     0.6505      114.8816       122.6828       127.0515       167.0117
     0.7005      114.8816       122.6828       127.0515       167.0117
     0.7505      114.8816       122.6828       127.0515       167.0117
     0.8005      114.8816       122.6828       127.0515       167.0117

          *****  End Cardiac Cycle  *****
```

Figure 6.14 **Numerical results for the mean arterial pressures during two cycles in the steady state**

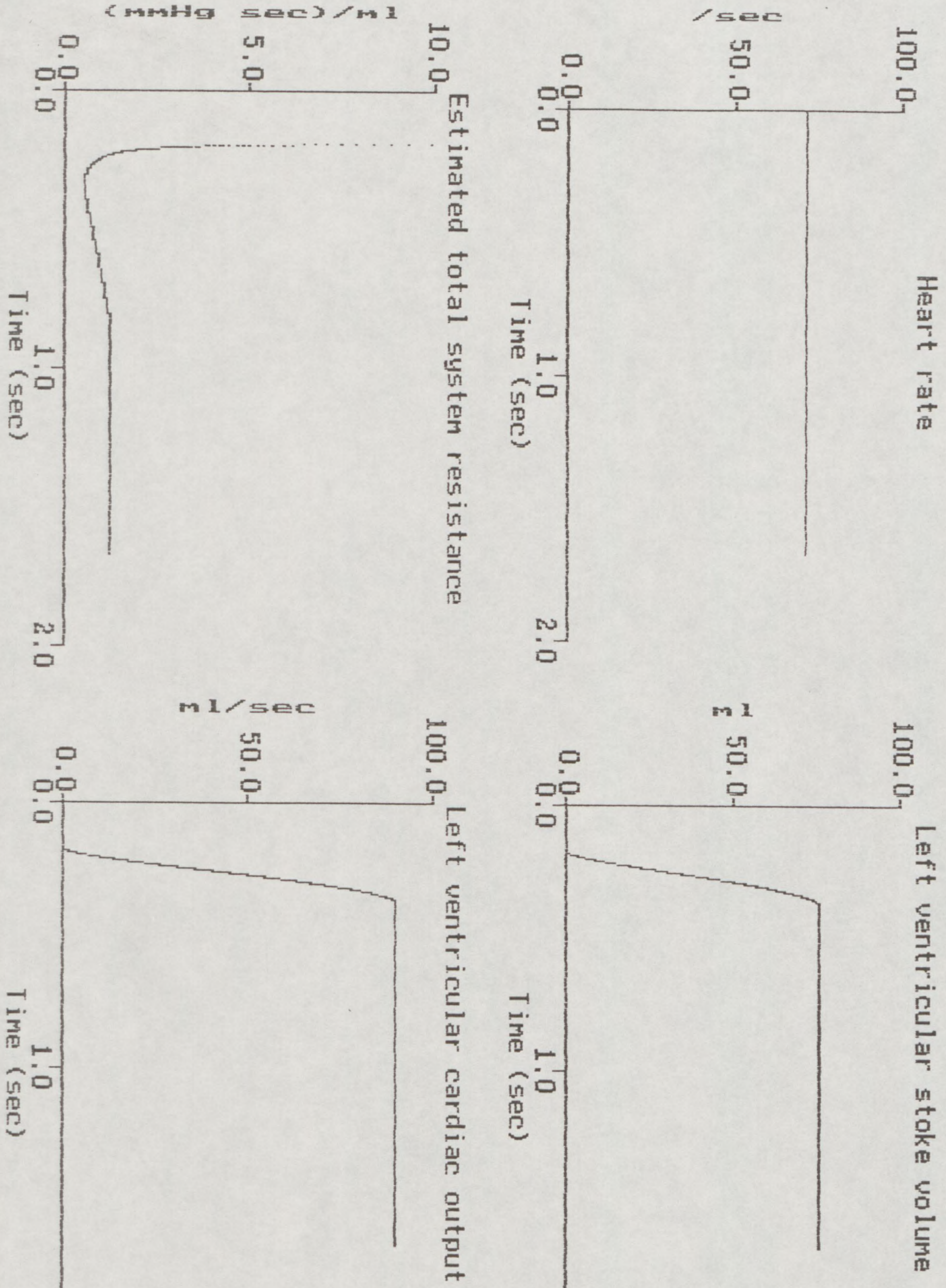


Figure 6.15 **Waveforms of end of cardiac cycle variables during two cycles in the steady state**

```
   HR          Heart rate

   SVLV        Left ventricular stoke volume

   ETSR        Estimated total system resistance

   COLV        Left ventricular cardiac output


        ***** During Cardiac Cycle *****

Time          HR            SVLV          ETSR           COLV
 0.0000        71.7017        0.0000       0.0000         0.0000
 0.0505        71.7017        0.0000       0.0000         0.0000
 0.1005        71.7017        0.0000       0.0000         0.0000
 0.1505        71.7017        0.0000       0.0000         0.0000
 0.2005        71.7017        5.7566       3.5010         6.8793
 0.2505        71.7017       30.1950       0.8850        36.0839
 0.3005        71.7017       56.1927       0.5979        67.1519
 0.3505        71.7017       73.3324       0.5481        87.6344
 0.4005        71.7017       75.7398       0.6082        90.5112
 0.4505        71.7017       75.7398       0.6867        90.5112
 0.5005        71.7017       75.7398       0.7663        90.5112
 0.5505        71.7017       75.7398       0.8457        90.5112
 0.6005        71.7017       75.7398       0.9240        90.5112
 0.6505        71.7017       75.7398       1.0005        90.5112
 0.7005        71.7017       75.7398       1.0751        90.5112
 0.7505        71.7017       75.7398       1.1476        90.5112
 0.8005        71.7017       75.7398       1.2182        90.5112

        *****  End Cardiac Cycle  *****
   HR          Heart rate

   SVLV        Left ventricular stoke volume

   ETSR        Estimated total system resistance

   COLV        Left ventricular cardiac output


        ***** During Cardiac Cycle *****

Time          HR            SVLV          ETSR           COLV
 0.0000        71.7017       75.7398       1.2693        90.5112
 0.0505        71.7017       75.7398       1.2693        90.5112
 0.1005        71.7017       75.7398       1.2693        90.5112
 0.1505        71.7017       75.7398       1.2693        90.5112
 0.2005        71.7017       75.7398       1.2693        90.5112
 0.2505        71.7017       75.7398       1.2693        90.5112
 0.3005        71.7017       75.7398       1.2693        90.5112
 0.3505        71.7017       75.7398       1.2693        90.5112
 0.4005        71.7017       75.7398       1.2693        90.5112
 0.4505        71.7017       75.7398       1.2693        90.5112
 0.5005        71.7017       75.7398       1.2693        90.5112
 0.5505        71.7017       75.7398       1.2693        90.5112
 0.6005        71.7017       75.7398       1.2693        90.5112
 0.6505        71.7017       75.7398       1.2693        90.5112
 0.7005        71.7017       75.7398       1.2693        90.5112
 0.7505        71.7017       75.7398       1.2693        90.5112
 0.8005        71.7017       75.7398       1.2693        90.5112

        *****  End Cardiac Cycle  *****
```

Figure 6.16 **Numerical results of the end of cycle variables during two cycles in the steady state**

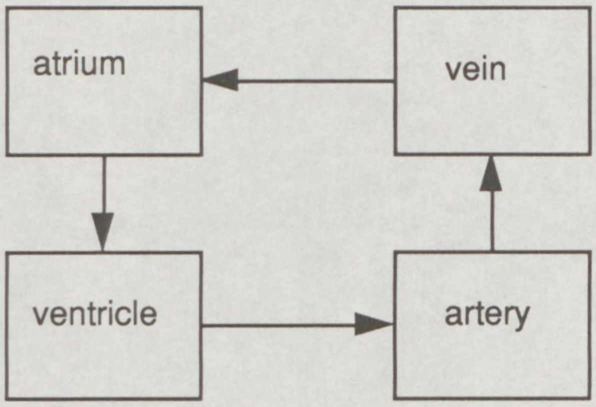


Figure 6.17 **Lande's 4 Compartment Model**

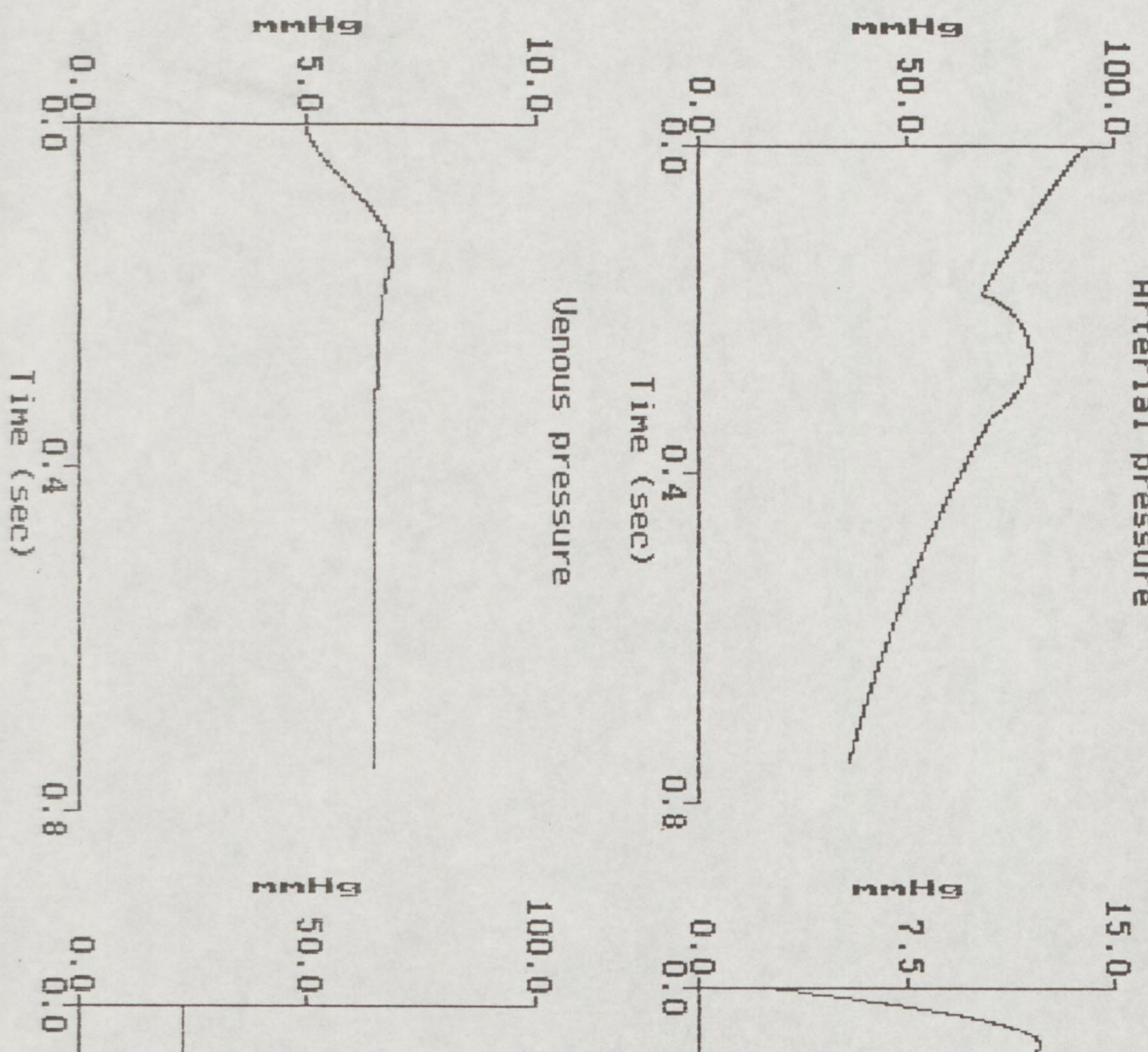


Figure 6.18 **Waveforms of pressure in the 4-segment model (Lande, 1989)**

```
PA01        Ascending aorta pressure

PA03        Thoracic aorta pressure

PAA         Abdominal arteries pressure

PCA         Leg arteries pressure

      ***** During Cardiac Cycle *****

Time        PA01         PA03          PAA          PCA
 0.0000     103.3842     111.1160     115.3366     155.7702
 0.0505     100.8634     108.7938     113.5453     154.4959
 0.1005      98.4388     106.4702     111.5301     152.8891
 0.1505      96.0885     104.1262     109.2484     150.8711
 0.2005     122.8145     111.1347     107.8981     148.5655
 0.2505     136.8192     137.4945     121.2057     150.1050
 0.3005     136.3514     148.9511     143.0512     162.9961
 0.3505     124.8792     145.0564     156.9005     183.4439
 0.4005     117.2370     130.3980     154.8483     199.1608
 0.4505     120.1188     127.9735     144.5703     200.6895
 0.5005     120.6343     128.7856     137.0730     191.1470
 0.5505     119.5850     127.8827     132.2860     178.9227
 0.6005     117.3380     125.2917     128.1927     169.2310
 0.6505     114.4329     121.9589     124.3316     163.1512
 0.7005     111.3409     118.6296     121.0665     159.7630
 0.7505     108.3304     115.6449     118.5827     157.8758
 0.8005     105.4969     113.0170     116.6710     156.6349

      *****  End Cardiac Cycle  *****
```

Figure 6.19 **Numerical results for the arterial pressures over one cardiac cycle in the steady state (refer to Fig. 6.2)**

```
        PRA          Right atrium pressure

        PLA          Left atrium pressure

        PRV          Right ventricle pressure

        PLV          Left ventricle pressure

            ***** During Cardiac Cycle *****

      Time          PRA          PLA          PRV          PLV
       0.0000        6.6034       9.5125       6.5086        9.4219
       0.0505       11.0459      12.2908       8.4991       11.1850
       0.1005        9.5500      11.4801       9.4379       11.6537
       0.1505        6.2737       8.8805      16.7887       46.9188
       0.2005        4.2988       7.7425      36.8379      148.7516
       0.2505        4.8668       8.2881      46.5939      188.9710
       0.3005        5.2111       8.6295      45.5955      172.2715
       0.3505        5.4577       9.0145      36.8236      130.4075
       0.4005        5.6547       9.4384      24.2379       83.6288
       0.4505        5.8274       9.8527       9.8077       22.5629
       0.5005        5.9861       8.6136       6.0364        7.9058
       0.5505        6.1160       8.7303       6.0547        8.4207
       0.6005        6.2018       8.8907       6.1141        8.6936
       0.6505        6.2782       9.0379       6.1843        8.8857
       0.7005        6.3523       9.1700       6.2570        9.0415
       0.7505        6.4256       9.2879       6.3302        9.1756
       0.8005        6.4981       9.3929       6.4032        9.2936

            *****  End Cardiac Cycle  *****
```

Figure 6.20 **Numerical results for the heart pressures over one cardiac cycle in steady state (refer to Fig. 6.3)**

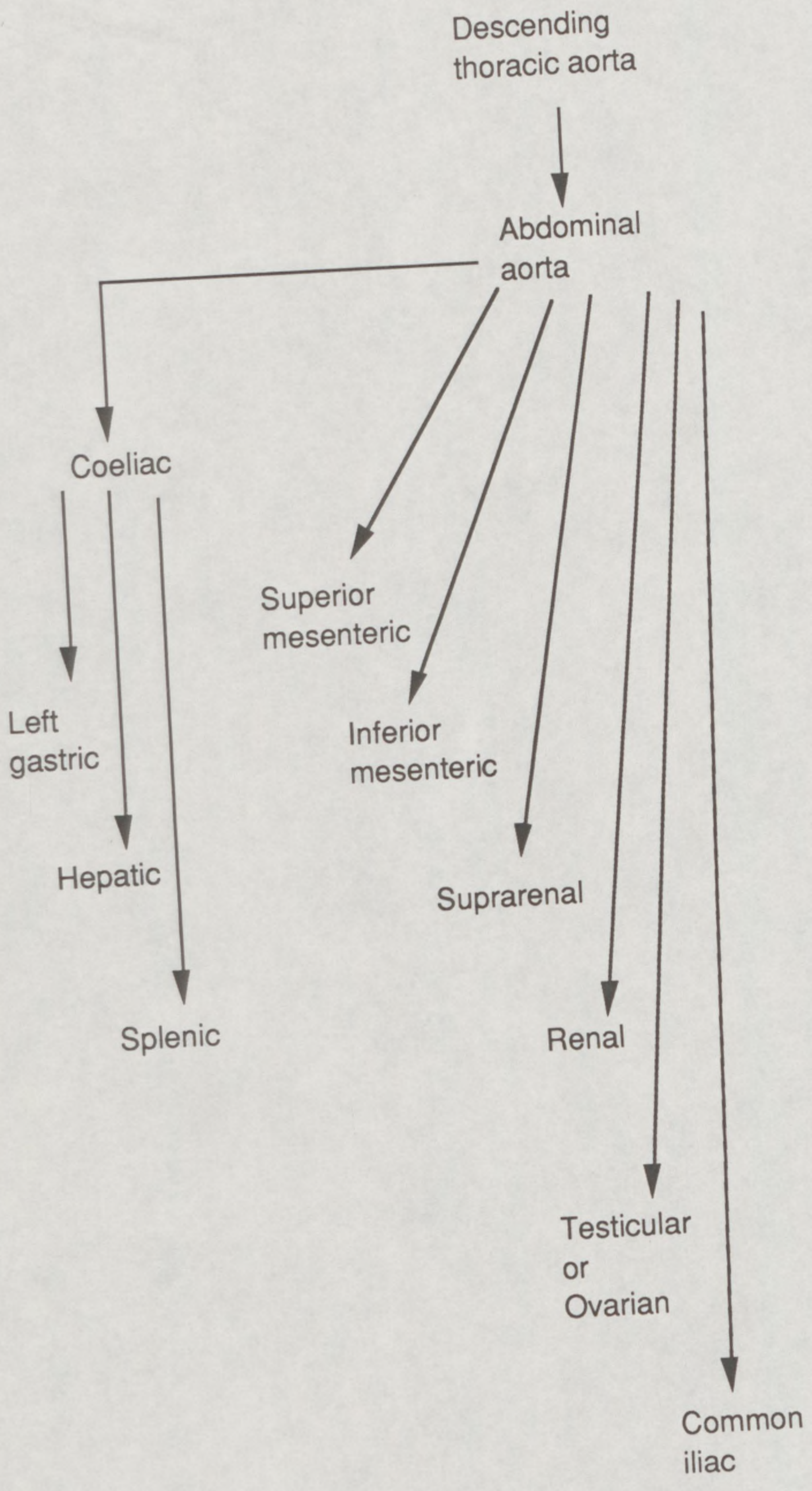


Figure 6.21 **Major arterial branches of the abdominal circulation**

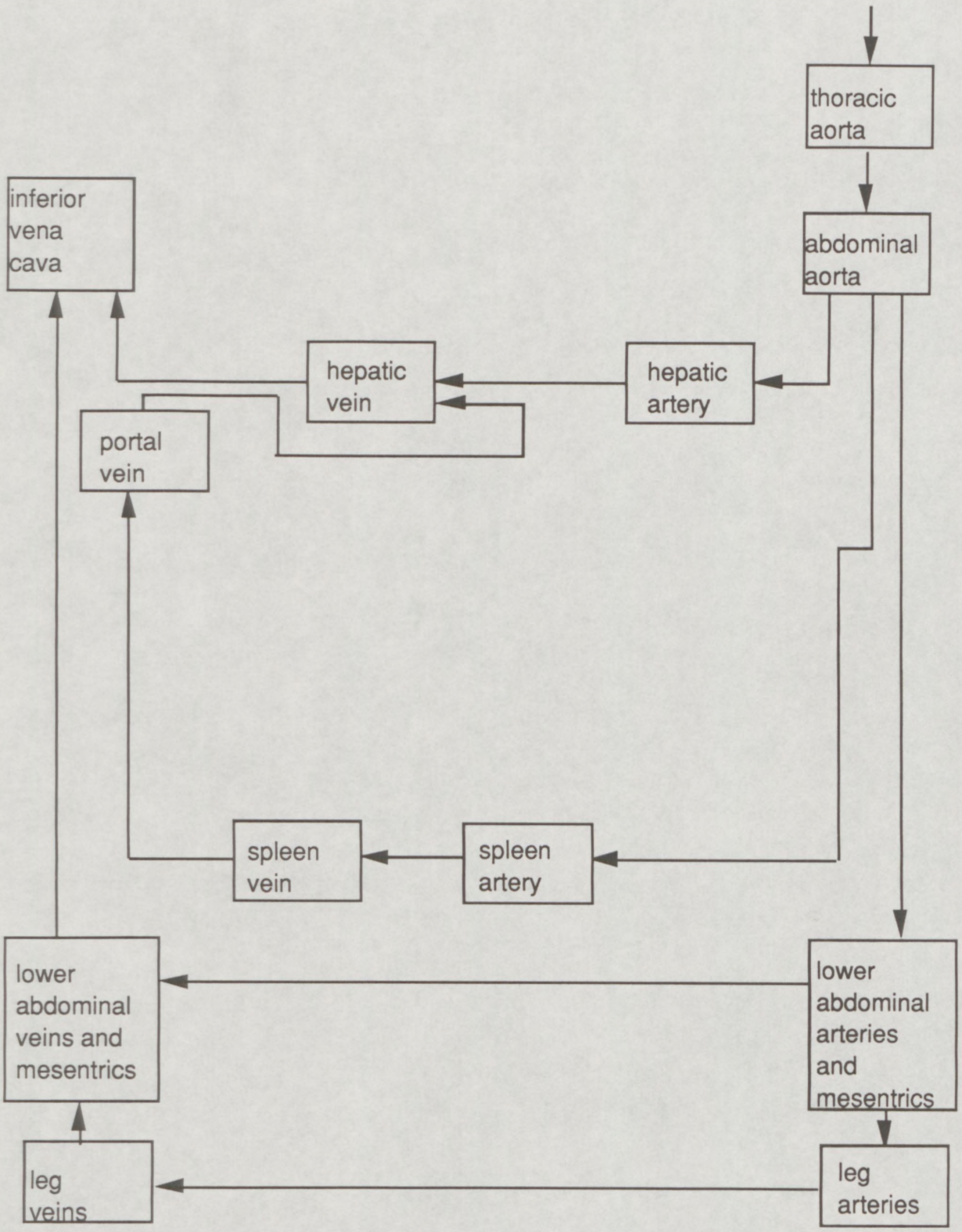


Figure 6.22 **23-Segment Model**

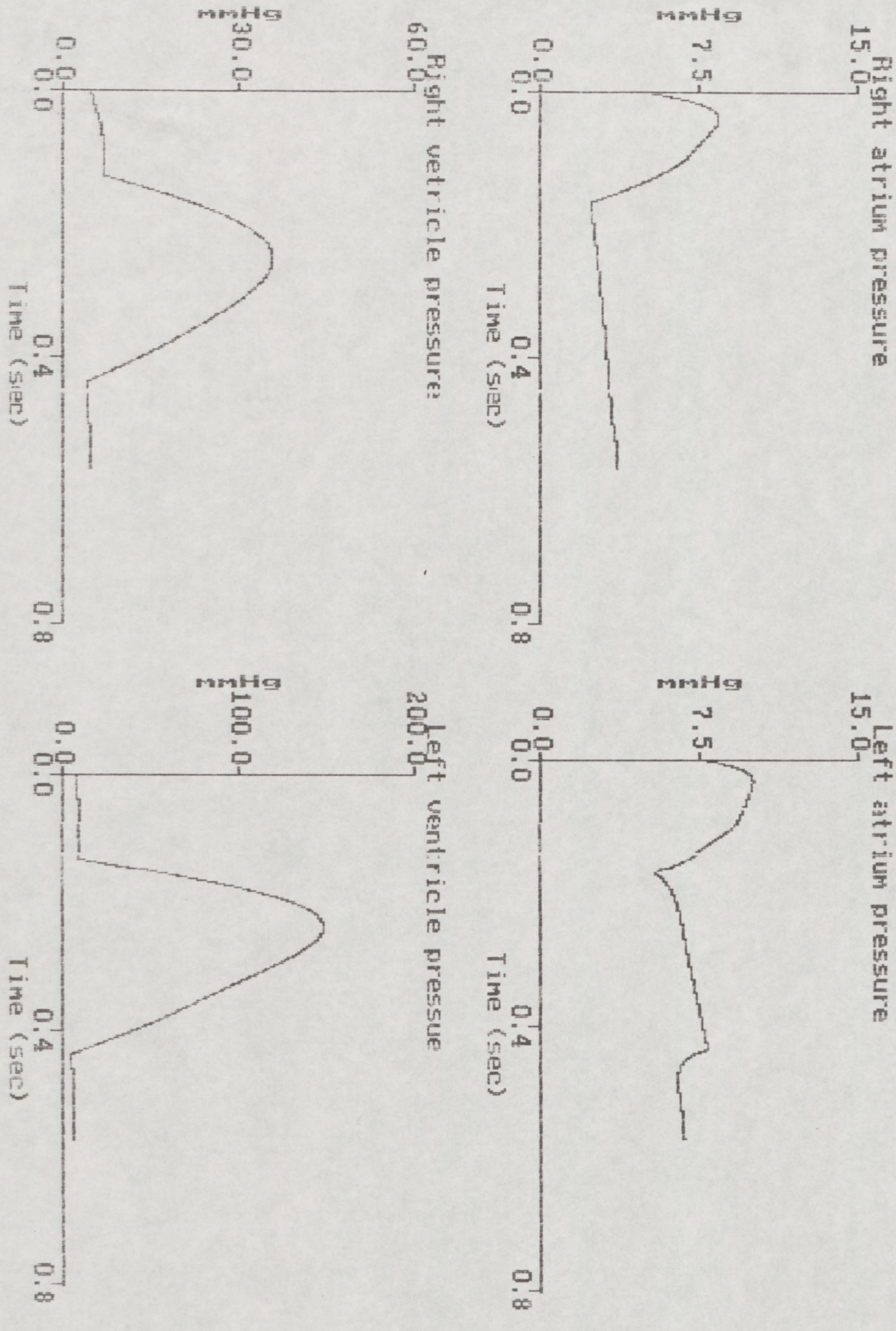


Figure 6.23 **Waveforms of heart pressures in the 23-segment model**

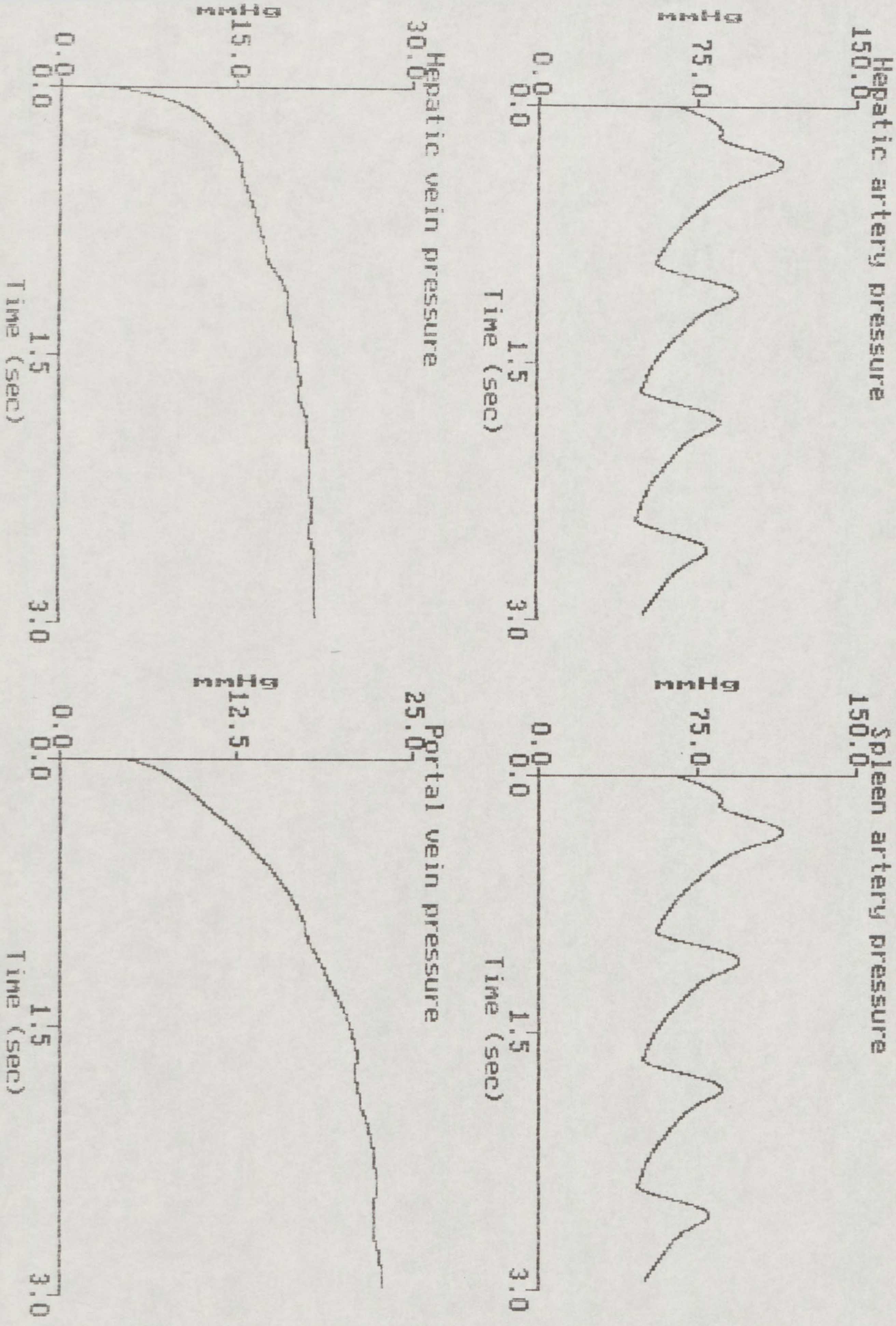


Figure 6.24 **Waveforms of arterial and venous pressures for the 23-segment model**

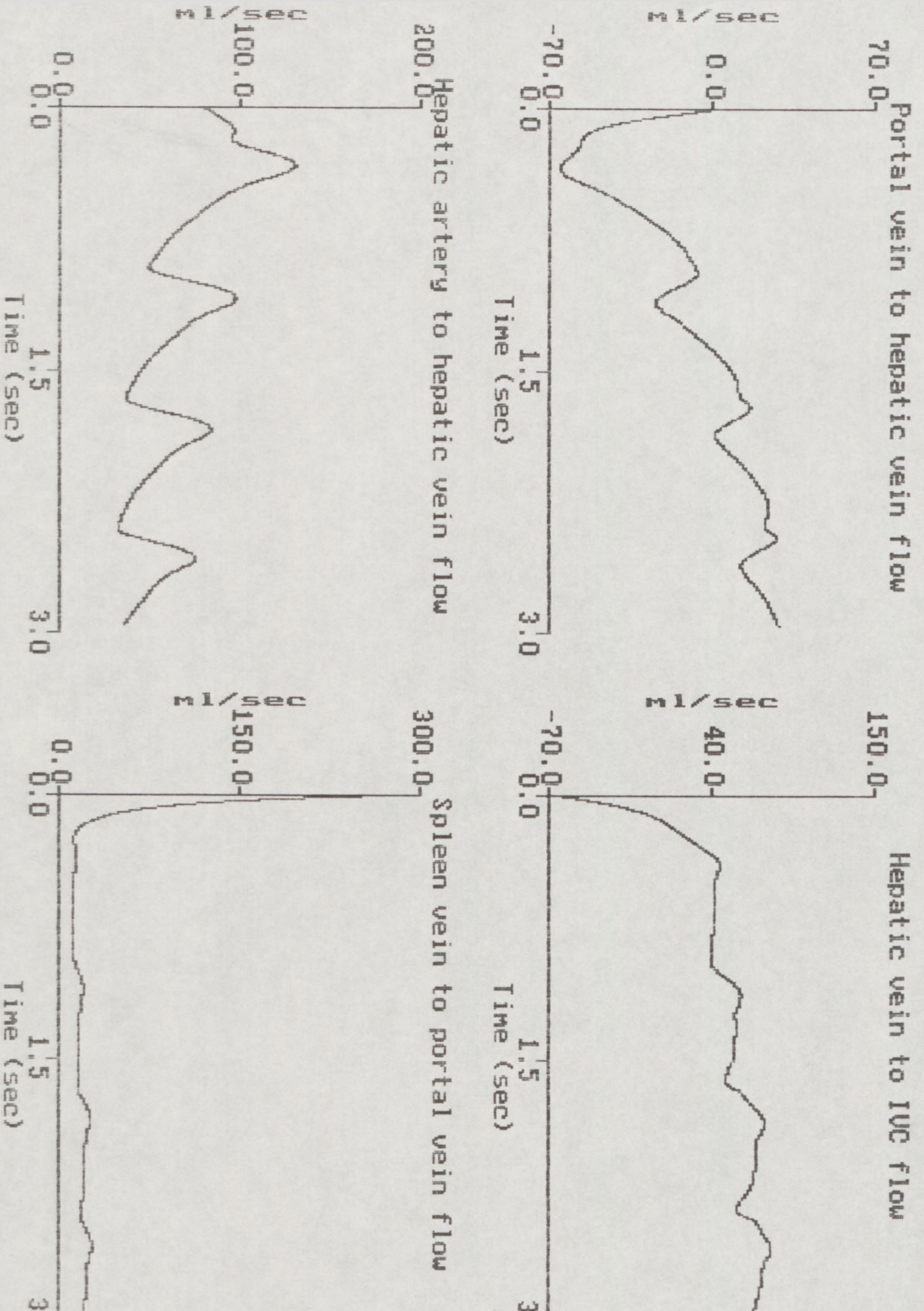


Figure 6.25 **Waveforms of arterial and venous flows for the 23-segment model**

Variable	PULSE (Fardipour 1989)	Pullen (1979)
T_H	0.837	0.823
SV	75.7398	69.6
CO	90.5112	84.6
P_{A01max}	136.4080	130.9
P_{A01min}	96.1234	90.8
ESTR	1.2182	1.29
MAP	114.8816	109.1

Table 6.1 **Comparison of the end of cycle variables between PULSE and Pullen (1976)**

	PULSE	CVS1	Pullen (1979)
Time	1:06 hr	2:30 hr	39 s
Output	Numerical/ Graphical	Numerical	Numerical
Version	Intractive	Interactive	Batch
Language	Pascal	FORTRAN 77	FORTRAN IV
Machine	PC's	PC's	CDC 7600

Table 6.2 **Comparison of time and output between PULSE, CVS1 and Pullen (1976)**

CHAPTER 7

USER EVALUATION OF 'PULSE'

7.1 INTRODUCTION

In the previous chapter, the validity of the approach was tested. The chapter emphasized two important criteria of validity; internal validity which was defined in terms of verification of the approach, for example the consistency and completeness of the approach, and external validity which relates to the methodology underlying the approach.

In this chapter, the term alpha testing was used for the tests carried out 'on site' in the laboratory and beta testing is used for tests carried out by the user (Dr Knut Lande). Thus, the tests described in the previous chapter carried out in the laboratory constitute the alpha testing.

Following alpha testing, the approach and the package needed to be evaluated from the user view point. The user should be someone who is familiar with cardiovascular modelling and has an ability to build a model. The package was sent out to collaborators at The Institute of Experimental Medical Research at the University of Oslo, Norway for their use. It was tested there by Dr Knut Lande who is both an experimental scientist and cardiovascular modeller and has been working in this field for several years. His background in medicine made it possible to analyse the results from the package in terms of relevance to clinical problems and he was thus an ideal candidate for 'PULSE' evaluation. Dr Lande has built a small 4-segment model, illustrated in the previous chapter. He divided the validity of 'PULSE' into three categories. These were as follows:

Verification (to verify the package in terms of user input, output. In other words the consistency of the package);

Usability (the usefulness of the package, its limitation and the ease of building a model in 'PULSE');

Validity (the validity of the approach compared to real life expectation. In other words the methodology underlying the approach).

Dr Lande's comments on these issues are presented here, plus the discussion of the results.

7.2 VERIFICATION OF 'PULSE'

On verification of 'PULSE' he has commented that:

1 - "Index files fall out of line with the database files. This happens when structural components are added."

2 - "After copying a condition, the simulation crashes."

RESPONSE - These two remarks have been checked and corrective action taken. The problems were due to the limitation of dBASEIII+ programming language, where the indices (define the relation between database files) have to be constantly stated which is truly not a 4GL programming language unlike Oracle. The problem has arisen because the Side-Kick (SK) editor, which was used instead of dBASEIII+ (since dBASEIII+ cannot handle very large programs), did wrap around long lines and thus caused problems when a large number of index files had to be called. This problem has been solved by using Turbo Pascal editor instead of SK.

3 "The simulation crashes if the number of variables chosen for display is changed from 4."

RESPONSE - The problem has been overcome by not allowing the user to choose the number of displays (fixed at 4). The user has to select which four variables are to be displayed. This is easier since more than four displays might look clumsy. This can be classified as a usability problem since the user is not allowed to display more than four variables at one time.

7.3 USABILITY OF 'PULSE'

On the usability of 'PULSE' Knut Lande has commented that:

1 - "It is necessary to be able to stop the simulation, change parameters and then restart, displaying the results before and after the change. This would mean being able to save the conditions at a certain step of the simulation. Generally it would be very convenient to be able to save many sets of initial values for a specific model and choose them without having to go through defining each variable separately."

RESPONSE - This feature is now incorporated in 'PULSE'. This is necessary as Knut Lande explained for various reasons. An important example is the Valsalva manoeuvre which is a forced expiration against a closed glottis. The effect is to increase the intra-thoracic pressure to a large positive value which temporarily prevents venous blood from entering the thorax. Before the manoeuvre, the state variables could be saved and then the intra-thoracic pressures could be increased for a few seconds and the simulation can be continued from the previous state and stopped after the manoeuvre is over (this test can be used for testing the asymmetry in the heart rate controller). After every simulation, results are saved in a file called `result5.res`. When the model is run again it will ask the user if they would like to run it with the previously saved state variables. If so then the saved state variables are used, otherwise the initial state variables are used.

2 - "The restriction to four output variables is unsatisfactory. The screen cannot handle more than four plots at a time. However, often you would like to follow five or ten variables when exploring the importance of some intervention. It would be very nice if the user could define for instance ten relevant variables for a simulation which could then be readily picked from a many with a number identifying them and without having to re-define scaling, heading etc. Also it should be possible to use less than four output variables."

RESPONSE - This has been changed slightly. Now users can define as many variables as they wish and the select, delete and add options which were shown in Chapter 5 have been incorporated.

3 - "The length of the condition field is too short."

RESPONSE - This was dealt with by increasing the length to 20 characters.

4 - "The "999" prompt that appears in some numeric fields is confusing."

RESPONSE - This was dealt with by changing the prompt to a null string all through the package.

5 - "The menu numbering system is confusing."

RESPONSE - This is explained thoroughly in the HELP facility of the package.

6 - "The meaning of the "previous results exist" message is unclear."

RESPONSE - This is no longer valid since the layout of the simulation has been changed.

7 - "The use of "Ctrl-End" is not obvious to non-dBASE users."

RESPONSE - This is explained thoroughly in the HELP facility of the package.

8 - "During the simulation, it is not clear that "Return key" interrupts."

RESPONSE - This is dealt with by adding another menu just before the simulation indicating how to interrupt the simulation.

9 - "The reason for the "short title" are not clear."

RESPONSE - The short title (such as "PLV" for the left ventricular pressure) is used to title the columns of the numerical simulation. This is now explained thoroughly in the HELP facility of the package.

10 - "The user is responsible for the title of a graph. It is possible to give the wrong title (e.g. volume instead of pressure). This should be clear to the user."

RESPONSE - This is dealt with by giving the user the name of the variable (e.g. pressure) and the compartment (s) (e.g. ascending aorta) on the top of the screen when the user is entering the title. This should help the user in entering the title of the graph.

11 - "How can the user examine the numerical simulation results?"

RESPONSE - This is dealt with by adding another menu just after the graphical simulation which says that the results are saved in `result3.res` (see Chapter 5 for more details), "would you like to view it?". However the file can also be accessed via DOS.

7.4 VALIDITY OF 'PULSE'

Knut Lande developed a small model which he describes as follows: "This is a highly simplified model of a two-chamber heart coupled to systemic arteries and systemic veins" (Fig. 6.17). He monitored the following changes:

(i) - Initially, there was only one phase of ventricle filling from the atrium, corresponding to active atrial contraction. The flow which occurs during passive filling of the atrium (the atrial "V" wave) was eventually reproduced by adjusting parameter values.

(ii) - It was necessary to look at the atrial - ventricular pressure difference to see whether the atrial pressure would reach high enough values during the V-wave. Eventually this was achieved by decreasing ventricular diastolic elastance (lowering diastolic ventricular pressure), increasing the resistance from atrium to ventricle and decreasing the venous return resistance.

(iii) - It was surprising how well such a small model could reproduce features of haemodynamic behaviour. In fact, this model could be reduced to fewer compartments or expanded to more compartments and thus be very convenient for studying the primitive heart and circulation in early embryonic development." (Fig. 6.18 from the previous chapter).

In his remarks on the validity of 'PULSE', Knut Lande has noted that:

1 - "'PULSE' has ventricles, atria, veins and arteries but no pericardium."

RESPONSE - The Piene 1983 model does have a pericardium and the need for a pericardium is increasing, since this encloses the heart and in the case of pericardium diseases such as tuberculosis (infection of lungs which eventually causes pericardium infection), or chronic bacterial infection (pericarditis), it is very difficult for the heart to contract or relax. In cases like this a needle is injected into the pericardium by a surgeon to extract inflamed material. In the long run, diseases of the pericardium can cause the heart to stop and thus it is necessary to be able to model the heart complete with pericardium. However, this is beyond the scope of the work presented here, but there is no reason why the pericardium should not be added to 'PULSE' at a later date (see Future Work).

2 - "The parameters in 'PULSE' may sometimes be more complex than is necessary (e.g. visco-elasticity time constants)."

RESPONSE - It is agreed that 'PULSE' is very complex but cardiovascular modelling is complex and 'PULSE' is built to cope both with a small model such as that built at the beta testing site, and a large model, such as the 19-segment model which was verified in alpha testing. The complexity of 'PULSE' might deter the scientist and clinician who are interested in small models (since not many variables are known), but it is hoped that the user can always input a null value whenever a variable is not required.

3 - "The time - varying elastances of the heart chambers are modelled by a half-sinusoid, which is questionable. First there is evidence that the diastolic elastance is not constant, but rather is a non-linear function of volume (as in the Piene model). Secondly, recent data (Lande et al., unpublished) suggest that the shape of the elastance function during systole is not sinusoidal but more peaked to the end of systole. It would be very interesting if the effect of different elastance function curves could be studied in 'PULSE'."

RESPONSE - The work which will be published by Knut Lande in future will stimulate very interesting discussions, but at the moment it has been decided to leave the elastances as they are (see future work). Since it has been validated by Pullen (1976), the effect on behaviour may be small (see Chapter 3, Section 3.2.6.1 part (iii)).

4 - "The main problem for the model validity will be in determining model parameters experimentally. It would be interesting to see if a model could be fitted to data in pig experiments. If so, the doors would then be open for developing models for matching to patients. This would be important clinically."

RESPONSE - This question has raised two interesting situations.

1) The question arose by Dr Lande can be illustrated diagrammatically (Figure 7.1). An input is entered into a **SYSTEM** and a **MODEL** (in our case 'PULSE') and the output is passed to the **OBJECTIVE FUNCTION** (likelihood function or squared error function) and the output from the OBJECTIVE FUNCTION is passed into **OPTIMISATION ALGORITHM** (such maximization or minimization). The output from OPTIMISATION ALGORITHM (parameter estimate) is passed back to the MODEL. The question is whether 'PULSE' can achieve this request. At the moment, the parameter values are input into 'PULSE' and the raw data are displayed numerically and graphically. This has been shown in the previous chapter (19-segment model, data are taken from studies on dogs and pigs), but the parameter estimation is not possible since none of the algorithms are incorporated in 'PULSE'.

However, it might be possible to produce this feature in future, although further research is needed.

2) In reality, the following question was brought up by Dr Ian Fore (Wyeth Research, UK). "Would it be possible to estimate the parameter values, given the raw data from the experiment? The raw data are usually analysed by statistical tests (such as the T test) to find the significance of the probability."

Dr Fore referred to Guyton's (1980) work on estimating a mean circulatory filling pressure. Guyton (1980) has measured this parameter for his model by stopping the heart and then pumping blood from the arteries to the veins very rapidly, so that an equilibrium pressure is established between the venous and the arterial trees (the experiment was carried out on anaesthetized dog).

RESPONSE - To estimate the parameter values given the raw data is beyond the scope of this research. Also, the T test is not incorporated in 'PULSE' and thus the query cannot be tackled using 'PULSE'.

Finally Knut Lande's overall verdict on 'PULSE' is:

"As well as the points I have mentioned above, I would have more demands on 'PULSE' in future once I have spent more time on it. However, I think it is very important that such wishes do not stop you from releasing it soon, because 'PULSE' is very good. Congratulations."

7.5 CONCLUSION

'PULSE' has been validated in beta testing by an independent source and has been passed successfully. The points raised in the verification of the package have been corrected. On the usability issues, the aim was to make 'PULSE' as user friendly as possible and yet produce a powerful modelling package. Thus, some of the limitations which were mentioned by Dr Lande have not been dealt with although most of them were overcome. On validity, the points Dr Lande has raised are very interesting, of which some could be dealt in with future work, some need more thought and some were not accepted. However, the reasons for every decision have been discussed in detail. It is hoped that 'PULSE' will in the future satisfy clinical research needs as well as those of scientific research.

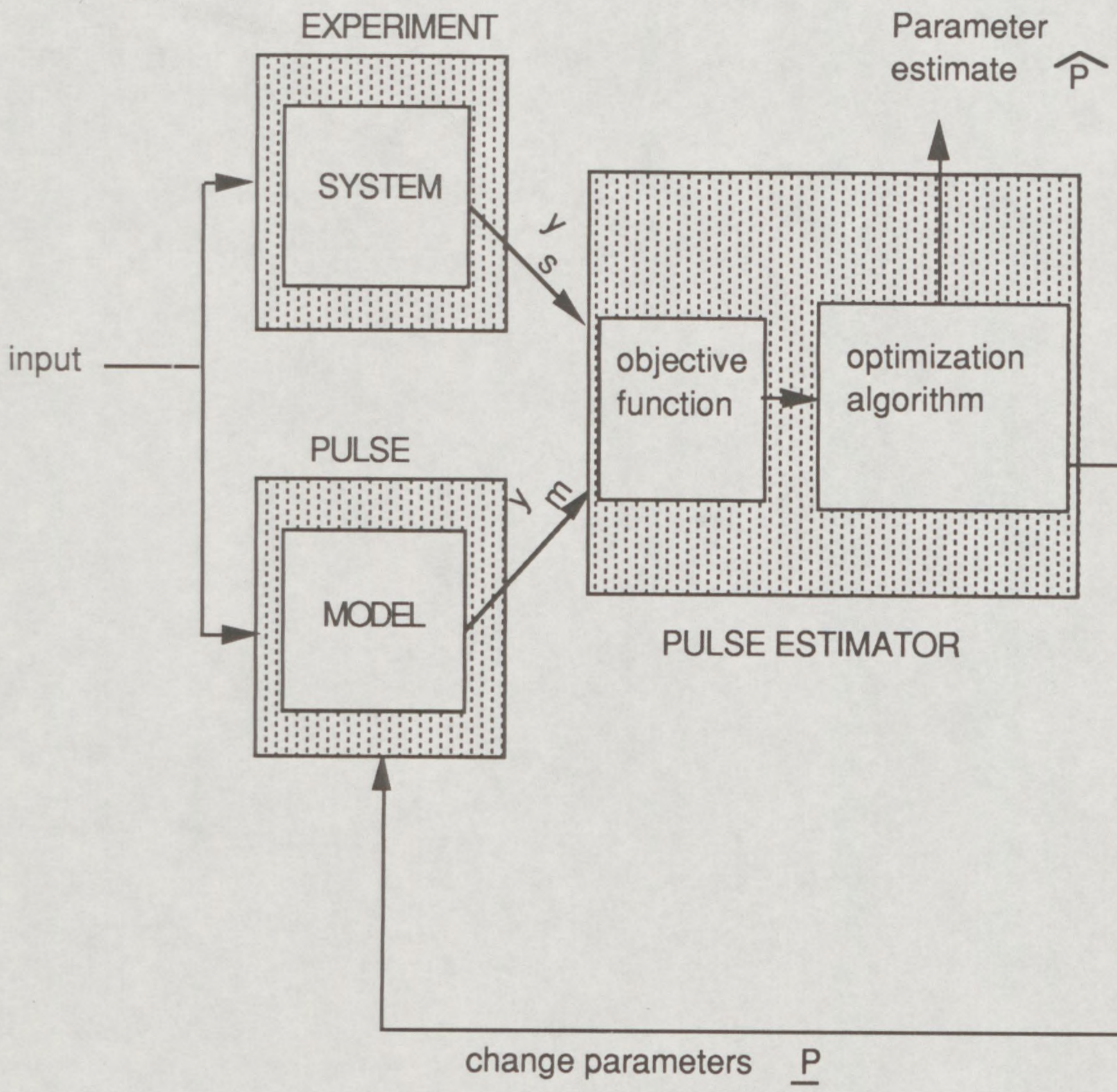


Figure 7.1 **A possible arrangement for future modelling**

CHAPTER 8

FUTURE WORK

8.1 INTRODUCTION

The conceptual and mathematical framework of cardiovascular modelling has been described in earlier chapters. This thesis has shown how a new approach to computer modelling alleviates some of the difficulty in developing a model. 'PULSE' was developed, based on a circulatory model with neural control. The software system has passed the alpha and beta testing and is ready to function both in clinical research and in system science. However, the following points should be considered in development.

1) The possibility of including local effects of drugs as well as pharmacokinetics.

2) Changing the integration routine to Runge-Kutta for greater accuracy.

3) Considering the approach used by Knut Lande for time-varying elastances of the heart chambers.

4) The need for a representation of the pericardium is desirable for the package.

5) The model of hepatic circulation developed by alpha testing could be tested further with clinical data.

6) In 'PULSE' a fixed step length is used for integration. This could be changed to a variable step length.

8.2 DRUG EFFECT EXTENSION

Drugs that affect the function of the heart and blood vessels are among the most widely used in medicine. Although these drugs may exert their primary effect either on the blood vessels or on the heart itself, the cardiovascular system functions as an integral unit. Thus, drugs that effect blood vessels are often useful in treating primary disorders such as hypertension (high blood pressure), angina pectoris (pain resulting from inadequate blood flow through the coronary vessels to the muscular wall of the heart), heart failure (inadequacy of the output of the heart in relation to the needs of the rest of the body), and arrhythmias (disturbances of cardiac rhythm).

Drugs affect the function of the heart in three main ways. They can affect the force of contraction of the heart muscle (inotropic effects), they can affect the frequency of the heart beat, or heart rate (choronotropic effects), or they can affect the regularity of the heart beat (rhythmic effects).

Drugs affect blood vessels by altering the state of contraction of the smooth muscle in the vessel wall, altering its calibre, or diameter, thereby regulating the volume of blood flow. Such drugs are classified as vasoconstrictors if they cause the smooth muscle lining to contract, and vasodilators if they cause it to relax. Drugs may act indirectly, for example by altering the activity of nerves of the autonomic nervous system that regulate vasoconstriction or vasodilation. Some drugs mainly affect arteries, which control the resistance to blood flow in the vascular system, an important determinant of the arterial blood pressure. Others mainly affect the veins, which control the pressure of blood flowing back to the heart, and hence the cardiac output (i.e. the volume of blood pumped out by the heart per minute).

Beta-blockers, (Beta-Adrenergic blocking agents), are a group of synthetic drugs used to treat a wide range of diseases and conditions of the sympathetic nervous system.

Stimulation by adrenalin of beta type receptor sites, which predominate in cells of the heart and are present in vascular and other smooth muscle, results in excitation of the sympathetic nervous system. The administering of a beta blocker diminishes the reaction at the beta receptor sites, preventing or decreasing excitation. These drugs are prescribed to control anxiety and hypertension and to treat a variety of cardiac conditions including anginal pain and cardiac arrythmias. They have proved to be successful in reducing a patient's risk of a second heart attack.

Pullen (1976) studied the effects of an injected drug in which the major dynamics are complete within two or three minutes. He derived a mathematical model of the injection, transport and action for a single drug which can be combined with other models (circulation and neural control). He assumed that the time taken to complete an injection is very short compared with the time constants of the transport dynamics in the model, and the volume of the injection is assumed to be negligible compared with the volume of the segment being injected.

Pullen's model of drug transport is based on the 'multiple modelling' techniques of Beneken and Rideout (1968). In this technique segments of a slave transport model are coupled to the corresponding segments of the master blood circulation model so that transport flow is proportional to concentration in the transport model multiplied by the flow in the circulation model.

As an injected drug moves around the circulatory system, the drug will be absorbed or transformed chemically. The effective concentration of the drug will decrease with time due to its breakdown or absorption decreasing because of dilution resulting from distribution throughout the blood volume.

Pullen (1976) assumed as an approximation that the rate of change of mass due to breakdown was proportional to the remaining mass of drug in a segment and that the time constant for breakdown is identical in all segments. Finally he applied his empirical approach to the specific location of drug action such as effect on heart rate (bradycardia (slow heart rate), tachycardia (fast heart rate)), peripheral resistance (vasoconstriction (narrowing blood vessels), vasodilation (widening blood vessels)), myocardial contractility (inotropic drug action, influencing muscular contractility) and venous properties (venoconstriction, venodilation).

Pullen's approach can be incorporated into the system 'PULSE' represented here. A possible relational schema for this work has been developed and is illustrated in Fig. 8.1.

i) `drug_input` entity. This contains *all the drugs that are input in the model.*

```
drug_input(model_code,drug_code,comp_code,input_type,dose,drug_units,
           start_time,stop_time)
```

key : `model_code,drug_code,comp_code,start_time`

The attributes are defined as:

`model_code`	: a code used for a selected model
`drug_code`	: a code use for a given drug
`comp_code`	: a code used for a compartment that was administered
`input_type`	: the type of the input, injection (intravenous, intramuscular), infusion
`dose`	: the dosage of the drug given
`drug_units`	: the unit used for the administered drug mm
`start_time`	: the starting time of the drug given
`stop_time`	: the stopping time of the drug given

`Model_code`, `drug_code`, `comp_code`, `start_time` are unique identifying code (key). `Model_code` is extracted from `model` (model definition), `drug_code` is generated by DBMS and `comp_code` is extracted from `comp_d` (compartment definition). The starting time (`start_time`) is a key because a drug is given to the system at a particular time. The user inputs the last five attributes. An example of `drug_input` will be:

```
    drug_input(1,aspirin,VU,injection,50,mg,0,0)
```

ii) `drug_d` entity. This contains *all the drugs definitons in the model.*

```
drug_d(model_code,drug_code,name,descriptn)
```

key : `model_code,drug_code`

The attributes are defined as:

`model_code`	: a code used for a selected model
`drug_code`	: a code use for a given drug
`name`	: a name of drug
`descriptn`	: a description of drug

Model_code, drug_code are unique identifying code (key). Model_code is extracted from model (model definition) and drug_code is extracted from drug_input (input drug). The user inputs the two five attributes.

iii) drug_e_d entity. This contains *all the drug effect definiton in the model.*

drug_d(model_code,drug_e_cod,drug_code,from,to,mod_par,dir_of_effect,
    descriptn)

key : model_code,drug_e_cod,drug_code

The attributes are defined as:

model_code = a code used for a selected model

drug_e_cod = a code use for a effect of given drug

drug_code = a code use for a given drug

from = from a compartment

to = to a compartment

mod_par = a modulated parameter

dir_of_effect = direction of effect of the drug

descriptn = a description of the drug

Model_code, drug_e_cod, drug_code are unique identifying code (key). Model_code is extracted from model (model definition), drug_code is extracted from drug_input (input drug) and drug_e_cod is generated by DBMS. The user inputs the last five attributes.

iv) drug_p entity. This contains *all the drug parameters in the model.*

drug_p(model_code,cond_code,drug_code,kio)

key : model_code,cond_code,drug_code

The attributes are defined as:

model_code = a code used for a selected model

cond_code = a code for the condition the model is in

drug_code = a code use for a given drug

kio = rate constant

Model_code, cond_code, drug_code are unique identifying code (key). Model_code is extracted from model (model definition), cond_code is extracted from conditio (condition definition) and drug_code is extracted from drug_d (input definition). The user inputs the last attribute.

v) drug_e_p entity. This contains *all the drug effect parameters in the model.*

drug_e_p(model_code,cond_code,drug_e_cod,sensitive)
key : model_code,cond_code,drug_e_cod
The attributes are defined as:

model_code	= a code used for a selected model
cond_code	= a code for the condition the model is in
drug_e_cod	= a code use for a effect of given drug
sensitive	= sensitivity of drug

Model_code, cond_code, drug_code are unique identifying code (key). Model_code is extracted from model (model definition), cond_code is extracted from conditio (condition definition) and drug_e_cod is extracted from drug_e_cod (drug effect definiton). The user inputs the last attribute.

The database files for the drugs effects of the 19-segment model of the above five entities are shown in Appendix 5.

For future work the generic equations have to be derived; this should not be difficult since from relational schema it can be seen that five types are needed to be incorporated into 'PULSE'. The entries into DBMS can be based on work presented here.

8.3 RUNGE-KUTTA APPROACH

To indicate this method due to the German mathematicians Carl David Tolme Runge (1895) and William Martin Kutta (1901) (see R Mccann; 1982), the initial value problem:

$$dy/dx = f(x,y) \tag{1}$$

where $y(a)=y_0$ is considered. In this method the Taylor expansion is used indirectly : y(a+h) is calculated in terms of y(a),

$$y(a+h) = y(a) + h \sum_{\kappa=0}^{p} a_k f(a+b_k h,\ y_0+c_k h) \tag{2}$$

The function $f(x,y)$ where $x = a+b_k h$ and $y = y_0+c_k h$ and constants $a_0 \ldots a_p$, $b_0 \ldots b_p$, $c_0 \ldots c_p$ which are chosen in y(a+h) in ascending power of h, the coefficients agree with those of the Taylor expansion of y(a+h).

For example if p=1 then y(a+h) can be derived as:

$$y_{n+1} = y_n+hk_2 \text{ where } k_2 = f(x_n+h/2,\ y_n+(hk_1)/2) \text{ and } k_1 = f(x_0,y_0) \tag{3}$$

This is known as the Runge-Kutta second order process.

Comparing this equation with Euler's method (Equation 4.2) an example is selected to illustrate the difference :

Suppose $y' = y$ with $y_0 = 1$ when $x_0 = 0$ and $h = 0.5$

using the Euler's method

$y_1 = y_0+hf(x_0,y_0) = 1+1.5(1) = \mathbf{1.5}$ where $f(x_0,y_0) = f(0,1)= 1$

using the Runge-Kutta method

$y_1 = y_0+hk_2$ where $k_2 = f(x_0+h/2,y_0+(hk_1)/2)$ and $k_1 = f(x_0,y_0)$

therefore

$k_1 = f(0,1) = 1$ and $k_2 = f(0.25,1.25) = 1.25$

and thus

$y_1 = 1+(0.5)(1.25) = \mathbf{1.625}$

The actual value is

$x = \int ydx$

$\Longrightarrow$ $x=\ln y$ $\Longrightarrow$ $y = e^x$

$\Longrightarrow$ $y = e^{0.5} = \mathbf{1.6487}$ when $x = 0.5$

(since h = 0.5 in the example illustrated here).

From the example above, the Runge-Kutta result is closer to the true value than Euler's result.

The error truncation of Runge-Kutta is less than Euler's method and thus Runge-Kutta is more accurate. In the current work the more accurate the method the better the output results. The adoption of a Runge-Kutta algorithm should be considered as an item of future work

8.4 NON-LINEAR ELASTANCES

As Knut Lande suggested in the previous chapter, the linear approximation of heart elastances, which has been adopted here could be re-assessed.

The elastance function is considered to be sinosoid in the present approach. This was based on the Beneken and De Wit (1967) approximation. However, it is a linear approximation and as such differs from real life. Also, the approximation gives the model a large ventricular outflow in the beginning which is not very appropriate. The approach Knut Lande suggests might suit the need of the clinician better. Piene (1983) implemented his model using this approach. The approach could be a subject of future research, involving the derivation of generic equations and their incorporation into 'PULSE'.

8.5 THE NEED FOR THE PERICARDIUM

The heart and the origins of the great blood vessels are enclosed in a loose - fitting sac called the pericardium. The Piene (1983) model, consists of the pericardium as well as the vascular interconnections of the active heart compartments. To add this feature should be trivial as a mathematical and computing task. Piene (1983) suggested that the pressure - volume characteristic of the pericardium to be exponential. The approach can be incorporated into the model described here.

8.6 THE HEPATIC CIRCULATION

The study of the splanchnic circulation and its change in progressive chronic diseases of the liver, for example cirrhosis caused by inadequate diet, alcohol or by chronic infection, has been developed by the alpha testing and explained in Chapter 6.

In the alpha testing, the values of the parameters are either estimated or approximated (explained in Chapter 6). However, these values are not accurate and, since this is a very important circulation, parameter changes for the degree of disease could be investigated. This requires clinical knowledge either through literature reference or collaboration with a clinician working in this field.

8.7 THE VARIABLE STEP LENGTH

The fixed step length can result in a slow integration process. A variable step length could be used to speed up the simulation. This may be achieved by increasing the step length when nothing interesting is happening such as when ventricular flow is zero and decreasing it when flow is not zero.

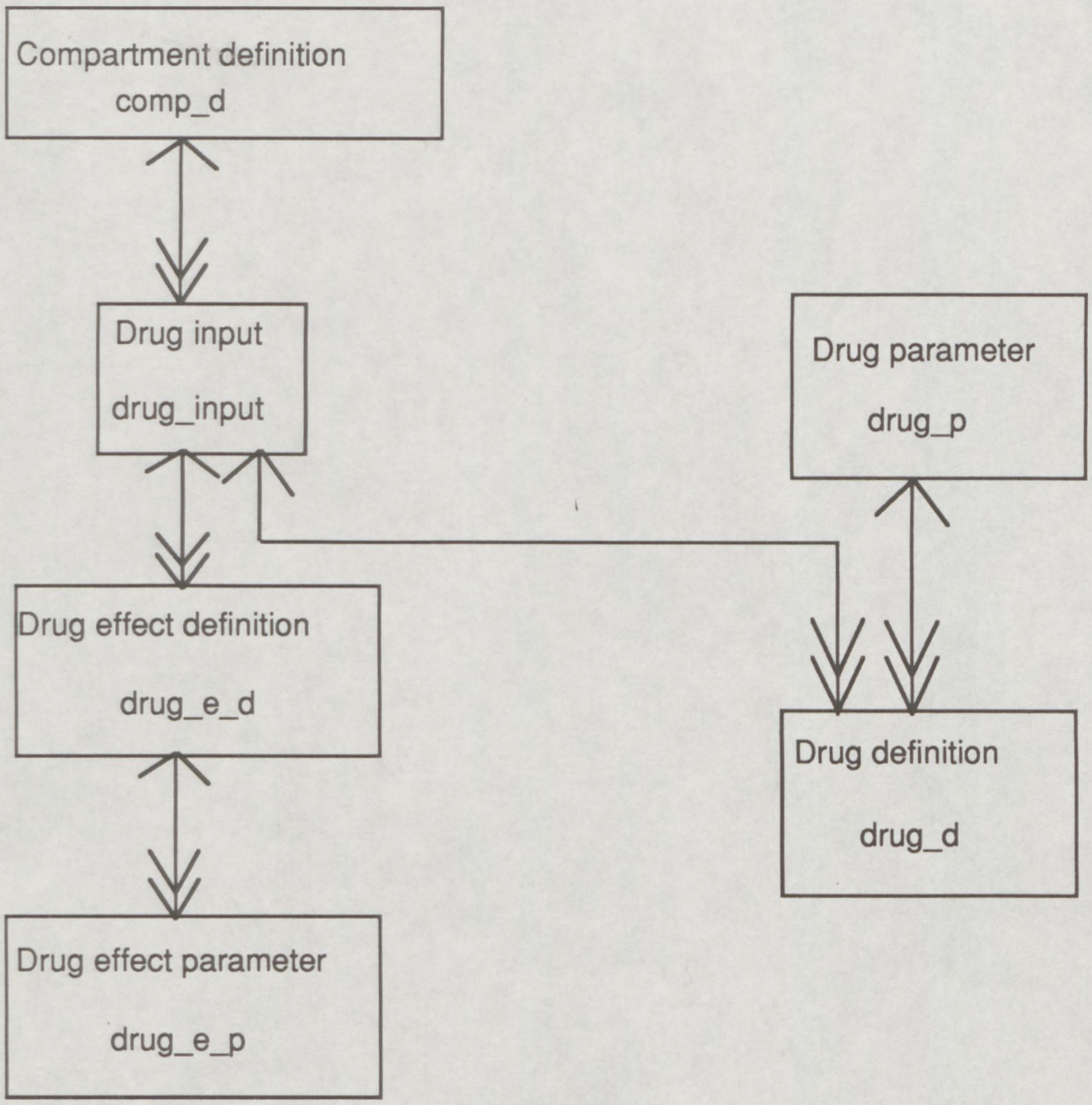


Figure 8.1 **A possible relational schema for drug extension**

CHAPTER 9

CONCLUSIONS

This thesis has described the development of a new approach to cardiovascular modelling and its implementation as $\overset{a}{\wedge}$software system (called 'PULSE') has been developed, where non-programmers can build, simulate, edit and test models easily. It has been shown that complex models of the cardiovascular system can be completely represented using a relational database. This has been achieved by designing the database to include dictionary files which contain the definitions of the entities and attributes of the entire relational schema. The database also includes the cardiovascular schema and consists of 1) the circulatory schema which contains entities representing the heart and the circulation and 2) the neural control schema which contains entities representing baroreceptors and cns input with four controllers. To successfully utilize the database to simulate a working heart under various conditions, a definition generator has been designed to extract from the database only those data which are required for simulation. These data are written into ASCII files by the definition generator. The files are then read by a simulation engine. The simulation engine then performs the simulation. Output is available in both numerical and graphical forms. The thesis has discussed in detail how the new approach functions, its achievements and its advantages.

This new approach represents one possible direction leading towards the evolution of complex models which will be more transparent and therefore more testable. The explicit representation and documentation of the model, with its greater usability, should increase the use of model-based studies in investigating cardiovascular dynamics. The user (researcher or clinician) can concentrate more on the designing of the model than in its programming.

The advantages of this approach are that models are represented explicitly, including assumptions, and model structure and parameters are easily changeable. The system is user friendly, and most clinicians and scientists working in this field will not be daunted by the thought of acquiring programming skills as has been the case with conventional approaches.

The new approach was well validated by tests at both alpha and beta sites, and the results from both sets of tests have confirmed the anticipated advantages of this novel approach. The functioning system 'PULSE' shows the worth of the approach. Three different models were built and simulated each showing a different application of cardiovascular modelling.

Many criticisms of large scale models are at a conceptual level and it is suggested that this approach goes further in representing model structure at a definition or conceptual level, which does not depend upon the precise form of mathematical equations which are used. Thus this feature will improve model testability and validity.

In summary, the work described within this thesis makes contributions to systems science, to physiology and to medicine. For systems science it represents a significant methodological advance in modelling complex dynamic systems. It provides means whereby users can have a clear understanding of the underlying dynamic processes, and have available a tool which is flexible and hence can assist the user in testing hypotheses regarding model structure and parameters and can make predictions.

Used in physiology, these methodological advances will enable the physiological research worker to have easier access to dynamic modelling approaches; approaches which are necessary if such research workers are to enhance their understanding of complex physiological mechanisms and their control. These advantages in physiology can equally be mirrored into the clinical areas. Models, providing that they are in a form which renders them acceptable to the clinician, can be used to quantify dynamic effects relating to disease processes and to make predictions of patient response to therapy. The work described in this thesis provides a basis for such clinical application.

REFERENCES

Al-Dahan, M. (1984), Development and Validation of Mathematical Models of the Human Cardiovascular System, Ph.D. Thesis. City University, London.

Bartelstone, H. J. (1960), "Role of veins in venous return", Circ. Res., **8**, 1059-1075.

Beneken, J. E. W. (1964), "Electronic analog computer of the human blood circulation", in Pulsatile Blood Flow, in E. O. Attinger, E. O. (Ed.), McGraw-Hill, 423-432.

Beneken, J. E. W. (1965), "A mathematical approach to cardiovascular function", Institute of Medical Physics TNO, Utrecht, Holland. Report No. 2-4-5/6.

Beneken, J. E. W., and De Wit, B. (1967), "A physical approach to haemodynamic aspects of the human cardiovascular system", in E. B. Reeve and A. C. Guyton (Eds.), Physical Bases of Circulatory Transport: Regulation and Exchange, W. B. Saunders, Philadelphia.

Beneken, J. E. W., and Rideout, V. C. (1968), "The use of multiple models in cardiovascular system studies: transport and perturbation methods", IEEE Trans. Biomed. Eng., **BME-15**, 281-289.

Boyers, D. G., Cuthbertson, J. G., Luetscher, J. A. (1972), "Simulation of the human cardiovascular system: a model with normal responses to changes of posture, blood loss, transfusion, and autonomic blockade", Simulation, **18B**, 197-205.

Carson, E. R. , Cobelli, C., and Finkelstein, L. (1983), The Mathematical Modelling of Metabolic and Endocrine Systems, J. Wiley & Sons, Chichester.

Date, C. J. (1981), An Introduction to Database Systems. Addison-Wesley, Reading, Mass, 3rd Ed.

Date, C. J. (1986), An Introduction to Database Systems. Academic Press, New York, 4th Ed.

Dick, D. E., and Kendrick, J. E., Matson, G. L., Rideout, V. C. (1968), "Measurement of non-linearity in the arterial system of the dog by a new method"", Circ. Res. **22**. 101-111

Green, J. H. (1972), An introduction to human physiology. Oxford University Press, London, 3rd Ed.

Green, J. F. (1977), Mechanical Concepts in Cardiovascular and Pulmonary Physiology. H. Kimpton, London.

Grodins, F. S. (1959), "Interactive cardiovascular physiology: a mathematical synthesis of cardiac and blood vessel haemodynamics", Quart. Rev. Biol., **34**, 93-116.

Guyton, A. C. (1955), "Determination of cardiac output by equating venous return curves with cardiac response curves" Physiol. Rev., **35**, 123-129.

Guyton, A. C., Coleman, T. G., Granger, H. J. (1972), "Circulation: overall regulation". Ann. Rev. Physiol., **34**, 13-46.

Guyton, A. C. (1979), "On the value of large models of biological systems: A critical commentary on Yates's paper...". Cybernetics & Information Sci., **2**, 71-72.

Guyton, A. C. (1980), Arterial Pressure and Hypertension, W. B. Saunders, Philadelphia.

Harreveld, A. Van, Feigen, G. A., Lerman, L. S. (1949), "Haemodynamics of aortic occlusion", Am. J. Physiol., **157**, 168-176.

Harvey, W. (1628), Excertatio de Motu Cordis et Sanguinis in Animalibus, (translated by K. J. Franklin, Movement of the Heart and Blood in Animals, Blackwell Scientific Publications, Oxford, 1957).

Horowitz, E. (1984), Fundamentals of Programming Languages. Springer-Verlay, London, 2nd Ed.

Huikeshoven, F. J., Hope, I. D., Power, G. G., Gilbert, R. D., Longo, L. D. (1985), "A mathematical model of the fetal circulation". Am. J. Physiol., **249**, R192-R202.

Hyndman, B. W. (1970), A Digital Simulation of the Human Cardiovascular System and its use in the Study of Sinus Arrhythmia, Ph.D. Thesis, Imperial College, London.

Jones R. T. and Kantrowiz A. R. (1965), "Analysis of certain heamodynamic effects of an artificial ventricle", Engineering in Medicine and Biology, Proceedings of the 18th Annual Conference, 79.

Katona, P. G., Barnett, G. O., Jackson, W. D. (1967), "Computer simulation of the blood pressure control of heart period", in P. Kezdi (Ed.), Baroreceptors and Hypertension, Pergamon, Oxford, 191-199.

Leaning, M. S. (1980), The Validity and Validation of Mathematical Models, Ph.D. Thesis, City University, London.

Leaning, M. S. (1986), "A database-oriented architecture for large-scale simulation models". Technical Report No. 86/3, Medical School Computer Unit, Royal Free Hospital, London.

Leaning, M. S., Pullen, H. E., Carson, E. R., Finkelstein, L. (1983a), "Modelling a complex biological system: the human cardiovascular system. I. Methodology and model description", Trans. Inst. Meas. Contr., **5**, 71-86.

Leaning, M. S., Pullen, H. F., Carson, E. R., Al-Dahan, M., Rajkumar, N., Finkelstein, L. (1983b), "Modelling a complex biological system: the human cardiovascular system. II. Model validation", Trans. Inst. Meas. Contr., **5**, 87-98.

Leaning, M. S., Nicolosi E. (1986), "MODEL-Software for knowledge-based modelling of compartmental systems". Biomed. Meas. Infor. Contr., **1**, 171-181.

Leaning M. S. and Fardipour P. (1987), "A pulsatile cardiovascular simulation program (CVS1)". Technical report 87/4, Royal Free Hospital, London.

Newman F. H. and Searle V. H. L. (1957), <u>The General Properties of Matter</u>. E. Arnold Ltd, London.

Noordergraaf, A., Verdouw, P. D., and Boom, H. B. K. (1963), "The use of an analog computer in a circulation model". Progress in Cardiovascular Diseases, **5**, 419-439.

Martin P. J., Levy M. N., Zieske H. (1969), "Bilateral carotid sinus control of ventricular performance in the dog". Circ. Res., **24**, 321-335.

Mccann, R. (1982), <u>Introduction to Ordinary Differential Equations</u>. The Camelot Press Ltd., Southampton.

McDonald, D. A. (1974), <u>Blood flow in arteries</u>. The Camelot Press Ltd., Southampton.

Piene, H., Smiseth, O. A., Refsum, H., Tyberg, J. V. (1983), "Apparent depression of right ventricular function after selective reduction of left ventricular inotropy: an interpretation of experimental data utilising a computer-based circulatory model". Med. & Biol. Eng. & Comput., **21**, 548-556.

Pullen, H. E. (1976), Studies in the Modelling and Simulation of the Human Cardiovascular System with Application to the Effects of Drugs, Ph.D. Thesis, City University, London.

Rushmer, R. F. (1976), Cardiovascular Dynamics, W. B. Saunders, Philadelphia, 4th Ed.

Scher A. M., and Young A. C. (1963), "Servoanalysis of carotid sinus reflex effects on peripheral resistance". Circ. Res., **12**, P152-162.

Snyder, M. F., and Rideout, V. C. (1969), "Computer simulation studies of the venous circulation", IEEE Trans. Biomed. Eng., **BME-16**, 325-334.

Solomon, E. P., and Davis, P. W. (1983), Human Anatomy and Physiology, CBS College printing, New York.

Spriet, J. A. and Vansteenkiste, G. (1982), Computer Aided Modelling and Simulation, Academic Press, London.

Uttamsingh, R. J., Leaning, M. S., Bushman, J. A., Carson, E. R., Finkelstein L. (1985), "Mathematical model of the human renal system", Med. and Biol. Eng. and Comput., **23**, 525-535.

Wexelblat R. L. (1981), History of Programming Languages. Academic Press, New York.

Yamamoto, W. S. (1985), "Converting mathematical models of physiological systems to relational database schemes for analysis and comparison". IEEE Trans. Biomed. Eng., **BME-32**, 273-276.

Yates, F. E. (1979), "Physical biology: A basis for modelling living systems". Cybernetics & Information Sci., **2**, 57-70.

APPENDIX 1

DICTIONARY LISTING

The listing of dictionary files `entity`, `attribute` and `ent_att` which were discussed in Chapter 5 are given below.

entname	**entgroup**	**definition**	**occurance**
entity	dictionary	definitions of all entities in pulse	200
attribut	dictionary	definitions of all attribut in pulse	200
ent_att	dictionary	definitions of all relations in pulse	200
comp_d	circulation	definitions of all compartments in pulse	30
f_comp_d	circulation	definitions of all flows in pulse	60
artery_p	circulation	contained all arterial parameters	10
vein_p	circulation	contained all venous parameters	10
atrium_p	circulation	contained all atrial parameters	4
ventri_p	circulation	contained all ventricular parameter	4
f_art_p	circulation	contained all flows from arteries	20
f_vein_p	circulation	contained all flows from veins	20
f_atri_p	circulation	contained all flows from atria	4
f_vent_p	circulation	contained all flows from ventricles	4
baro_d	neural control	definitions of all baroreceptors	4
cns_in_d	neural control	definitions of all CNS input	4
heart_d	neural control	definition of heart rate control	1
periph_d	neural control	definition of peripheral resistance	1
contrc_d	neural control	definition of myocardial contractility	1
venous_d	neural control	definition of venous tone	1
baro_p	neural control	contained all baroreceptors parameter	4
cns_in_p	neural control	contained all CNS inputs parameter	16
heart_p	neural control	contained heart rate parameter	1
periph_p	neural control	contained peripheral resistance parameter	1
contrc_p	neural control	contained myocardial contractility param.	1
venous_p	rneural control	contained venous tone parameter	1
drug_d	pharmacokinetics	definition of all drugs	5
drug_e_d	pharmacodynamics	definition of all drug effects	5
drug_p	pharmacokinetics	contained all drug parameters	5
drug_e_p	pharmacodynamics	contained all drug effect parameters	5
drug_inp	pharmacokinetics	contained drug inputs	5

Figure A1.1 **Entity (`entity`) file listing**

attname	**type**	**definition**
entname	character	name of the entity in pulse
entgroup	character	group which the entity belongs to
definition	character	definition of all entities,attributes
occurrence	numeric	number of occurrences of the entity
attname	character	name of attribute in pulse
type	character	type of the attribut in pulse
key	logical	if the attribute is a key to entity
model_code	numeric	the code for definig the models
comp_type	character	the type for the compartment (artery)
region	character	the region for the compartment (thorax)
name	character	the name for the compartment (leg veins)
anat_det	character	any anatomical details
reference	character	any litrature reference
from	character	the code for from compartment (e.g. A1)
to	character	the code for to compartment (e.g. A2)
link	character	the code for linking compartment (A1A2)
descriptn	character	the description of the link compartment
i	numeric	index for arteries,veins,atrium,ventricle
ci	numeric	compliance
ki	numeric	viscoelastisity constant in arteries
pregion	numeric	regional pressure (-4 to +4)
cond_code	character	the code defining the condition of model
comp_code	character	the code defining the compartment (A1)
cio	numeric	intial value for compliance in veins
vui	numeric	unstressed volume
alphai	numeric	comliance constant in veins
vio	numeric	initial volume
neural	character	effect of neural control in flows,veins
ais	numeric	systolic elastance for heart chambers
aid	numeric	diastolic elastance for heart chambers
lambda1	numeric	heart period constant in heart chambers
lambda2	numeric	heart period constant in heart chambers
lambda3	numeric	heart period constant in heart chambers
j	numeric	index for flows in circulatory compartments
Rji	numeric	resistance to flows between compartments
lenji	numeric	length of tubes in flows from compartments
anglji	numeric	effect of gravitational effect with angle
Lji	numeric	inertance of flows between compartments
Fji	numeric	initial flow values
pcc	numeric	critical pressure in lungs
betai	numeric	collapsible veins constant
valve	character	if valve exist in veins compartment
Aji	numeric	area of vessel in ventricular compartments
baro_code	character	a code for baroreceptors
x1	numeric	state variable value
x2	numeric	state variable value
slow_time	numeric	time to pass through the slow filter
fast_time	numeric	time to pass through the fast filter
gain1	numeric	first filter gain in baroreceptors
gain2	numeric	second filter gain in baroreceptors
thrsh_pres	numeric	threshold pressure in baroreceptors
cns_in_cod	character	the CNS input code
control	character	type of control .(e.g. heart rate)
contributn	numeric	the contribution of the control parameter
bhrt	numeric	heart rate CNS input
x3	numeric	state variable in heart rate

```
x5          numeric    state variable in heart rate
x6          numeric    state variable in heart rate
T1          numeric    upper time limit in heart rate
T2          numeric    lower time linit in heart rate
thmax       numeric    maximum value for heart period
thmin       numeric    minimum value for heart period
bprt        numeric    peripheral resistance CNS input
prmax       numeric    maximum value for peripheral resistance
prmin       numeric    minimum value for peripheral resistance
bmct        numeric    myocardial contractility CNS input
t           numeric    time constant through filter
mcmin       numeric    minumum value for myocardial contractility
mcmax       numeric    maximum value for myocardial contractility
bvtt        numeric    venous tone CNS input
vtmin       numeric    minimum value for venous tone
vtmax       numeric    maximum value for vanous tone
cgain       numeric    gain for compliance
vugain      numeric    gain for unstressed volume
```

Figure A1.2 **Attribute (`attribut`) file listing**

entname	attname	key
entity	entname	.T.
entity	entgroup	.F.
entity	definition	.F.
entity	occurrence	.F.
attribut	attname	.T.
attribut	type	.F.
attribut	definition	.F.
ent_att	entname	.T.
ent_att	attname	.T.
ent_att	key	.F.
comp_d	model_code	.T.
comp_d	comp_code	.T.
comp_d	comp_type	.F.
comp_d	region	.F.
comp_d	name	.F.
comp_d	anat_det	.F.
comp_d	reference	.F.
f_comp_d	model_code	.T.
f_comp_d	from	.F.
f_comp_d	to	.F.
f_comp_d	link	.T.
f_comp_d	descriptn	.F.
baro_d	model_code	.T.
baro_d	baro_code	.T.
baro_d	comp_code	.T.
baro_d	descriptn	.F.
cns_in_d	model_code	.T.
cns_in_d	baro_code	.T.
cns_in_d	cns_in_cod	.T.
cns_in_d	descriptn	.F.
heart_d	model_code	.T.
heart_d	descriptn	.F.
periph_d	model_code	.T.
periph_d	descriptn	.F.
contrc_d	model_code	.T.
contrc_d	descriptn	.F.
venous_d	model_code	.T.
venous_d	descriptn	.F.
artery_p	model_code	.T.
artery_p	cond_code	.T.
artery_p	comp_code	.T.
artery_p	i	.F.
artery_p	ci	.F.
artery_p	vui	.F.
artery_p	ki	.F.
artery_p	vio	.F.
artery_p	pregion	.F.
vein_p	model_code	.T.
vein_p	cond_code	.T.
vein_p	comp_code	.T.
vein_p	i	.F.
vein_p	cio	.F.
vein_p	vui	.F.
vein_p	alphai	.F.
vein_p	vio	.F.

```
vein_p      pregion    .F.
vein_p      neural     .F.
ventri_p    model_code .T.
ventri_p    cond_code  .T.
ventri_p    comp_code  .T.
ventri_p    i          .F.
ventri_p    vui        .F.
ventri_p    ais        .F.
ventri_p    aid        .F.
ventri_p    lambda1    .F.
ventri_p    lambda2    .F.
ventri_p    lambda3    .F.
ventri_p    vio        .F.
ventri_p    pregion    .F.
atrium_p    model_code .T.
atrium_p    cond_code  .T.
atrium_p    comp_code  .T.
atrium_p    i          .F.
atrium_p    vui        .F.
atrium_p    ais        .F.
atrium_p    aid        .F.
atrium_p    lambda1    .F.
atrium_p    lambda2    .F.
atrium_p    vio        .F.
atrium_p    pregion    .F.
f_art_p     model_code .T.
f_art_p     cond_code  .T.
f_art_p     link       .T.
f_art_p     j          .F.
f_art_p     i          .F.
f_art_p     Rji        .F.
f_art_p     lenji      .F.
f_art_p     anglji     .F.
f_art_p     Lji        .F.
f_art_p     Fji        .F.
f_art_p     pcc        .F.
f_art_p     neural     .F.
f_vein_p    model_code .T.
f_vein_p    cond_code  .T.
f_vein_p    link       .T.
f_vein_p    j          .F.
f_vein_p    i          .F.
f_vein_p    Rji        .F.
f_vein_p    lenji      .F.
f_vein_p    anglji     .F.
f_vein_p    vui        .F.
f_vein_p    betai      .F.
f_vein_p    Fji        .F.
f_vein_p    valve      .F.
f_vent_p    model_code .T.
f_vent_p    cond_code  .T.
f_vent_p    link       .T.
f_vent_p    j          .F.
f_vent_p    i          .F.
f_vent_p    Rji        .F.
f_vent_p    Fji        .F.
f_vent_p    Lji        .F.
f_vent_p    Aji        .F.
```

```
f_atri_p   model_code .T.
f_atri_p   cond_code  .T.
f_atri_p   link       .T.
f_atri_p   j          .F.
f_atri_p   i          .F.
f_atri_p   Rji        .F.
f_atri_p   Fji        .F.
baro_p     model_code .T.
baro_p     cond_code  .T.
baro_p     baro_code  .T.
baro_p     x1         .F.
baro_p     x2         .F.
baro_p     slow_time  .F.
baro_p     fast_time  .F.
baro_p     gain1      .F.
baro_p     gain2      .F.
baro_p     thrsh_pres .F.
cns_in_p   model_code .T.
cns_in_p   cond_code  .T.
cns_in_p   cns_in_cod .T.
cns_in_p   control    .F.
cns_in_p   contributn .F.
heart_p    model_code .T.
heart_p    cond_code  .T.
heart_p    bhrt       .F.
heart_p    x3         .F.
heart_p    x5         .F.
heart_p    x6         .F.
heart_p    T1         .F.
heart_p    T2         .F.
heart_p    fast_time  .F.
heart_p    slow_time  .F.
heart_p    thmax      .F.
heart_p    thmin      .F.
periph_p   model_code .T.
periph_p   cond_code  .T.
periph_p   bprt       .F.
periph_p   x2         .F.
periph_p   x3         .F.
periph_p   fast_time  .F.
periph_p   slow_time  .F.
periph_p   prmax      .F.
periph_p   prmax      .F.
periph_p   gain1      .F.
contrc_p   model_code .T.
contrc_p   cond_code  .T.
contrc_p   bmct       .F.
contrc_p   x2         .F.
contrc_p   t          .F.
contrc_p   mcmin      .F.
contrc_p   mcmax      .F.
venous_p   model_code .T.
venous_p   cond_code  .T.
venous_p   bvtt       .F.
venous_p   x2         .F.
venous_p   t          .F.
venous_p   vtmin      .F.
venous_p   vtmax      .F.
venous_p   cgain      .F.
venous_p   vugain     .F.
```

Figure A1.3 **Entity/attribute (`ent_att`) file listing**

APPENDIX 2

LISTING OF EQUATIONS IN THE SIMULATION ENGINE

The listing of circulatory equations file `result1.equ` which was discussed in Chapter 5 is given below.

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$
$                         P U L S E   V e r s i o n   1
$
$                         -----------------------------
$
$
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$
$                Full equations of Circulatory Fluid Mechanics
$
$                ---------------------------------------------
$
$
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


///////////////////////////////////////////////////////////////////
/           N.B. see DICTIONARY menu for all attribute definitions
/
///////////////////////////////////////////////////////////////////


*******************************************************************
                              1. Units
*******************************************************************

Length         cm
Volume         ml
Pressure       mmHg
Time           sec
Compliance     ml/mmHg
Elastance      mmHg/ml
Flow           ml/sec
Inertance      mmHg/ml/SQUARE(sec)
```

```
**********************************************************************
                            2.  Pressures
**********************************************************************


                            2.1 Arteries
                            ------------

For all arteries i;

P[i] = ((V[i]-VU[i])/C[i]) + ((K[i]/C[i])dV[i]/dt)


######################### END OF ARTERIES #########################


                            2.2 Veins
                            ---------

For all veins i;

P[i] = (V[i]-sigmaVU[i]) (sigmaC[i])

   sigmaVU[i] = VU[i]/CnsVVu                       if neural is true
   sigmaVU[i] = VU[i]                              if neural is false

   C[i] = (initial compliance Cio)                 if V[i] >
sigmaVU[i]
   C[i] = (initial compliance Cio)(alfa[i])        if V[i] <=
sigmaVU[i]

   sigmaC[i] = CnsVC/C[i]                          if neural is true
   sigmaC[i] = 1/C[i]                              if neural is false


######################### END OF VEINS ############################


                            2.3 Ventricles
                            --------------

For all ventricles i;

P[i] = a[i](t) (V[i] - VU[i])

    a[i](t) = X(t) (sigmaas[i] - ad[i]) + ad[i]

    sigmaas[i] = (CnsM)(as[i])                          if neural is
true
    sigmaas[i] = as[i]                                  if neural is
false

     TAV = TAS - lambda1
     TVS = lambda2 + (lambda3)(TH)
    X(t) = 0                    if tcardiac <= TAV or tcardiac >
TAV+TVS
```

```
    X(t) = sin(π(tcardiac-TAV)/TVS)  otherwise


###################### END OF VENTRICLES ########################


                         2.4 Atria
                         ---------

For all atrial i;

P[i] = a[i](t) (V[i]- VU[i])

  a[i](t) = X(t) (sigmaas[i] - ad[i]) + ad[i]

  sigmaas[i] = (CnsM)(as[i])                          if neural is
true
  sigmaas[i] = as[i]                                  if neural is
false

   TAS = lambda1 + (lambda2)(TH)
  X(t) = 0                          if   tcardiac >  TAS
  X(t) = sin(π(tcardiac/TAS))       if   tcardiac <= TAS


######################### END OF ATRIA ##########################


******************************************************************
                         3.  Flows
******************************************************************

                         3.1 From Arteries
                         -----------------

For all arterial flows from i to j;

sigmaR[j,i] = CnsP(R[j,i])    if neural is true
sigmaR[j,i] = R[j,i]          if neural is false

if **** Pulmonary flow ****

  F[j,i] = (P[i] - P[j])/sigmaR[j,i]      if P[j] > Pcc
  F[j,i] = (P[i] - Pcc)/sigmaR[j,i]       otherwise

if **** Systemic flow ****

  if L[j,i] > 0

  dF[j,i]/dt = (P[i]+Preg[i]-P[j]-Preg[j]-G[j,i]-
(sigmaR[j,i])(F[j,i]))/L[j,i]

  where
```

```
   G[j,i] = (n,g)(len[j,i])(sin(angl[j,i]))     where n,g=0.7807 with
n=1
   , = density of blood = , g = gravitational constant = 9.81


  if L[j,i] = 0

  F[j,i] = (P[i]+Preg[i]-P[j]-Preg[j]-G[j,i])/sigmaR[j,i]
```

######################## END OF FROM ARTERIES ######################

3.2 From Veins

```
For all venous flows from i to j

temp1 = ((P[i]+Preg[i]-P[j]-Preg[j]-G[j,i])(SQUARE(V[i]))/RV2

   where RV2 = (R[j,i])(SQUARE(VU[i]))

if   temp1 >  0
        F[j,i] = temp1

if   temp1 <= 0 and valve = true

        F[j,i] = 0

if   temp1 <= 0 and valve = false

        F[j,i] = (beta[i])(temp1)
```

######################### END OF FROM VEINS ########################

3.3 From Ventricles

```
For all ventricular flows from i to j

if Lji > 0

dF[j,i]/dt = (P[i]-P[j]-(R[j,i])(F[j,i])-F2A2)/L[j,i]

where F2A2 = (,/2) SQUARE((F[j,i])/(A[j,i]))

if Lji = 0

F[j,i] = (P[i]-P[j])/R[j,i]
```

####################### END OF FROM VENTRICLES #####################

```
                       3.4 From Atria
                       --------------

For all atrial flows from i to j

F[j,i] = (P[i]-P[j])/R[j,i]


######################## END OF FROM ATRIA ########################
```

The listing of neural control equations file `result2.equ` which was discussed in Chapter 5 is given below.

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$
$                        P U L S E   V e r s i o n   1
$
$                        ------------------------------
$
$
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$
$                 Full equations of Neural Control
$
$                 --------------------------------
$
$
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


////////////////////////////////////////////////////////////////////
/           N.B. see DICTIONARY menu for all attribute definitions
/
////////////////////////////////////////////////////////////////////


********************************************************************
*                            4.  Neural control
*
********************************************************************
```

5.1 baroreceptors

```
for all baroreceptors i,

temp3 = a1*((x1 - PT[i])+ a2*x2)

B[i]  = temp3   if  temp3 > 0
      = 0       if  temp3 <= 0

dx1 = (P[i] - x1)/t1
dP[i]/dt = 0 if dP[i]/dt <= 0
dx2 = (dP[i] - x2)/t2

dB[i]/dt = a1*(dx1 + a2*dx2)  if  temp3>0
dB[i]/dt = 0                     otherwise


####################### END OF BARORECEPTORS #######################
```

5.2 heart rate

```
x7 = b2*(x3 + x6)

Heart period = THmax if x7 >= THmax
Heart period = x7    if THmin < x7 <= THmax
Heart period = THmin if x7 <= THmin

for all baroreceptors i,

cns input for heart rate (BHR) = Σ cnsconst1[i]*B[i]

for all baroreceptors i,

rate of change of cns input for heart rate (dBHR/dt) = Σ
cnsconst1[i]*dB[i]/dt

 if BHR <= BHRT (heart rate constant input)
x1       =  0.0
dx1      =  0.0
x4       =  BHR

 if BHR > BHRT (heart rate constant input)
x1       = b1*(BHR - BHRT)
dx1      = b1*DBHR
x4       = BHRT

 if   dx1 > 0          x2 = t1
 if   dx1 <= 0         x2 = t2

dx3 = (x1 - x3)/x2
dx5 = (x4 - x5)/t3
dx6 = (x5 - x6)/t4
```

```
######################### END OF HEART RATE #########################
```

5.3 peripheral resistance

```
x4 = b3*x3 + (1 - b3)*x2

peripheral resistance effect (CnsP) (effect on resistances in
arteries) = x4

for all baroreceptors i,

cns input for peripheral resistance (BPR) = Σ cnsconst2[i]*B[i]

x1 = BPRTmin if BPR >  BPRT (peripheral resistance constant input)
x1 = BPRTmax if BPR <= BPRT (peripheral resistance constant input)


dx2 = (x1 - x2)/t1
dx3 = (x1 - x3)/t2


################## END OF PERIPHERAL RESISTANCE ###################
```

5.4 myocardial contractility

```
myocardial contractility effect (CnsM) (effect on heart chambers) =
x2

for all baroreceptors i,

cns input for myocardial contractility (BMC) = Σ cnsconst3[i]*B[i]

x1 = BMCTmin if BMC >  BMCT (myocardial contractility constant input)
x1 = BMCTmax if BMC <= BMCT (myocardial contractility constant input)


dx2 = (x1 - x2)/t1


################# END OF MYOCARDIAL CONTRACTILITY #################
```

5.5 venous tone

```
x3 = 1 + Cgain  * (x2 - 1)
x4 = 1 + Vugain * (x2 - 1)

venous tone input effect (CnsVC) (effect on compliance in the veins)
= x3
venous tone effect (CnsVVu) (effect on unstressed volume in the
veins) = x4

for all baroreceptors i,
cns input for venous tone (BVT) = Σ cnsconst3[i]*B[i]

x1 = BVTTmin     if    BVT >  BVTT (venous tone constant input)
x1 = BVTTmax     if    BVT <= BVTT (venous tone constant input)


dx2 = (x1 - x2)/t1

######################### END OF VENOUS TONE ########################
```

APPENDIX 3

LISTING OF THE SIMULATION ENGINE

The listing of Pascal program `harvey.pas` which was discussed in Chapter 5 is given below.

*** Program : HARVEY.PAS**
*** Author : Parvin Fardipour**
*** Date : April 25, 1988**
*** Notice : Copyright 1989**

*** Pascal program to simulate cardiovascula models.**

```
program Harvey;

uses
  Dos, Crt, Printer, Graph;
```

{ Pulsatile network of ventricles, atria, arteries & veins }

```
type
```

{data types for graphics. PointType is declared in the Graph Unit}

```
  RealPointType = record
                     X,Y: real;
                  end;
  PlotArray     = array[1..200] of PointType;
  RealPlotArray = array[1..200] of RealPointType;
  TimeStoreType = array[1..200] of real;
```

{data types for model}

```
  StateSpace1 = array[1..40] of real;
  StateSpace2 = array[1..35,1..35] of real;
  StateSpace3 = array[1..65] of real;
  StateSpace4 = array[1..10] of real;
  StateSpace5 = array[1..4] of real;

  ArtPar    = record
              i : integer;
              Ci,Vui,Ki,Vi0,KCi,Pregion : real;
              sensedA : string[4];
              end;

  VeinPar   = record
```

```
            i : integer;
            Ci0,Vui,alfai,Vi0,Pregion : real;
            Ci,alfaCi0,sigmaVui,sigmaCi : real;      {computed}
            neural : string[4];
            end;

  VentPar    = record
            i : integer;
            Vui,ais,aid,TAV,TVS,lambda1,lambda2,lambda3,Vi0,Pregion
: real;
            aisd,ait,x,TAVS,piTVS : real;  {computed}
            end;

  AtrPar     = record
            i : integer;
            Vui,ais,aid,TAS,lambda1,lambda2,Vi0,Pregion : real;
            aisd,ait,x,piTAS : real;    {computed}
            end;

  FromArtPar  = record
              j,i : integer;
              Rji,lenji,anglji,Lji,Fji,Pcc,invLji : real;
              neural : string[4];
              Gji,sigmaRji : real;     {computed}
              end;

  FromVeinPar = record
              j,i : integer;
              Rji,lenji,anglji,Vui,betai,Fji : real;
              valve : string[4];
              sigmaVui,Gji,invRV2ji : real;        {computed}
              end;

  FromVentPar = record
              j,i : integer;
              Rji,Lji,Aji : real;
              rhoAji,Fji : real;         {computed}
              left : string[4];
              end;

  FromAtrPar  = record
              j,i : integer;
              Rji,Fji : real;
              end;

  HeartCircPar = record
               nart,nvein,nvent,natr,ncomp : integer;
               nfromart,nfromvein,nfromvent,nfromatr : integer;
               Preg,DX,D2X : StateSpace1;
               Art  : array[1..10] of ArtPar;
               Vein : array[1..10] of VeinPar;
               Vent : array[1..2] of VentPar;
               Atr  : array[1..2]  of AtrPar;
               FromArt  : array[1..20] of FromArtPar;
               FromVein : array[1..20] of FromVeinPar;
               FromVent : array[1..4]  of FromVentPar;
               FromAtr  : array[1..4]  of FromAtrPar;
               end;
```

```
BarPar         =  record
                  i : integer;                  {index of sensed artery}
                  x1,x2,dx1,dx2   : real;
                  dpres,t1,t2,a1,a2,PTi : real;
                  end;

CnsInputPar    =  record
                  i : integer;                  {index of sensed artery}
                  ai : real;
                  end;

HeartRatePar   =  record
                  CnsInp : array[1..4] of CnsInputPar;
                  b1,b2,t1,t2,t3,t4,THmin,THmax,BHRT : real;
                  x1,x2,x3,x4,x5,x6,x7,dx1,dx3,dx5,dx6,BHR,DBHR :
real;
                  end;

PerResPar      =  record
                  CnsInp : array[1..4] of CnsInputPar;
                  b1,b2,b3,t1,t2,BPRT : real;
                  x1,x2,x3,x4,dx2,dx3,BPR : real;
                  end;

MyoConPar      =  record
                  CnsInp : array[1..4] of CnsInputPar;
                  b1,b2,t1,BMCT : real;
                  x1,x2,dx2,BMC : real;
                  end;

VenTonPar      =  record
                  CnsInp : array[1..4] of CnsInputPar;
                  a : array[1..4] of real;
                  b1,b2,b3,b4,t1,BVTT : real;
                  x1,x2,x3,x4,dx2,BVT : real;
                  end;


CnsControlPar =   record
                  nbar : integer;
                  Bar  : array[1..4] of BarPar;
                  Periph,Myocar,Venous : string[4];
                  cnsconst1,cnsconst2,cnsconst3,cnsconst4 :
array[1..4] of real;
                  HeartRate : HeartRatePar;
                  PerRes    : PerResPar;
                  MyoCon    : MyoConPar;
                  VenTon    : VenTonPar;
                  end;
```

{data types for simulation}

```
  MeasCompDef =   record
                  icomp : integer;
                  plotmin,plotmax : real;
                  end;
  WindDef = record
            x1,y1,x2,y2: integer;
            xscale,yscale: real;
            end;

  MeasFcompDef = record
                 ifrom,ito : integer;
                 plotmin,plotmax : real;
                 end;

  MeasOtherDef = record
                 plotmin,plotmax : real;
                 end;


  SimulGraphPar = record
                  integstep,simulationtime : real;
                  ndraw : integer;
                  flag,namevar : array[1..6] of string[6];
                  title : array[1..6] of string[38];
                  name : array[1..6] of string[8];
                  w: array[1..5] of WindDef;
                  MeasComp  : array[1..6] of MeasCompDef;
                  MeasFcomp : array[1..6] of MeasFcompDef;
                  MeasOther : array[1..6] of MeasOtherDef;
                  end;

  ModelPar = record
             HeartCirc     : HeartCircPar;
             CnsControl    : CnsControlPar;
{            Drugs         : DrugsPar;}
             SimulGraph    : SimulGraphPar;
             end;
```

{global variables used in the program}

```
var

  Model : ModelPar;
  integpres,MP,difMP,P,V,DV,ETSR : StateSpace1;
  integF,F,DF : stateSpace2;
  TH,CnsH,CnsP,CnsM,CnsVC,CnsVVu,tcardiac : real;
  CnsHea,CnsPer,CnsMyo,CnsVen : Real;
  B,DB,EL : StateSpace4;
  SV,CO : StateSpace5;
  BangBang : string[4];
  NewState : StateSpace3;
  kstate,row1,row2,row3,col1,col2,row4 : integer;


{***********************************************************}

procedure FormatField(x: real;var w,d: integer);
```

{Determines the minimum field width, w, and decimal width, d, for a given real number, x. The difficult bit is finding d. The procedure is called by DrawXYAxes procedure written below.}

```
var
  s: string;

begin
  d:=1; {just to try}
  str(trunc(x),s);
  w:=length(s)+1+d;
end;

{**********************************************************}

procedure DrawXYAxes(XMin,XMax,XStep,YMin,YMax,YStep: real;
                     XTitle,YTitle,MainTitle: string);
```

{Draws and labels X & Y axes in current viewport. Axes run from XMin and YMin to XMax and YMax with labels at intervals of XStep and XStep. On exit the viewport settings are re-set to include the plotting area. Labels are printed in decimal format. A 2 line header is also allowed. All sizing adjustments for text are based on the current SetTextStyle. XTitle and YTitle are the titles for the x and y axes. MainTitle runs across the top in the 2nd line of the header.

There are 2 coordinate systems used:
1. pixel coordinates, in which all plotting is done;
2. graphics coordinates, in which the data are supplied.

The procedure is called by MakeScreen procedure written below.}

```
var
  Yptop,Ypbot,Xplt,Xprt: integer;     {axis offsets in pixels}
  Xpmax,Ypmax: integer;               {viewport size in pixels}
  Xpstep,Ypstep: integer;             {axis intervals in pixels}
  Xp,Yp: integer;                     {general position in pixels}
  X,Y: real;                          {general position in graphics}
  labl: string;
  ViewPort: ViewPortType;
  wx,wy,dx,dy: integer;               {width and decimal formatting
                                       for x & y labels}

begin
  {calculate formatting parameters}
  FormatField(XMax+XStep,wx,dx);
  FormatField(Ymax+YStep,wy,dy);

  {calculate pixel offsets for axes}
  Yptop:=2*TextHeight('1');          {allow 2 text lines}
  Ypbot:=2*TextHeight('1')+5;        {          "          + tic}
  str(YMax:wy:dy,labl);
  Xplt:=TextWidth(labl)+5;           {allow for YMax + tic}
  str(YMax:wy:dy,labl);
  Xprt:=TextWidth(labl) div 2;       {allow for 1/2 XMax width}

  GetViewSettings(ViewPort);
  with ViewPort do
  begin
  Xpmax:=x2-x1;
  Ypmax:=y2-y1;
end;

{Draw X axis}
Yp:=Ypmax-Ypbot;
Line(Xplt,Yp,Xpmax-Xprt,Yp);

{Tick & label X axis}
SetTextJustify(CenterText,TopText);
SetTextStyle(DefaultFont,HorizDir,1);
Xpstep:=round((Xpmax-Xprt-Xplt)*XStep/(XMax-XMin));
Xp:=Xplt;
X:=XMin;
repeat
  MoveTo(Xp,Yp);
  LineRel(0,4);
  str(X:wx:dx,labl);
  OutText(labl);
  X:=X+XStep;
  Xp:=Xp+Xpstep;
until X>XMax;

{Title X axis}
MoveTo(Xplt+(Xpmax-Xplt-Xprt) div 2,Ypmax-TextHeight('1'));
OutText(XTitle);

{Draw Y axis}
Xp:=Xplt;
Line(Xp,Yptop,Xp,Ypmax-Ypbot);

{Tick & label Y axis}
SetTextJustify(RightText,CenterText);
SetTextStyle(DefaultFont,HorizDir,1);
Ypstep:=round((Ypmax-Yptop-Ypbot)*YStep/(YMax-YMin));
Yp:=Ypmax-Ypbot;
Y:=YMin;
repeat
  MoveTo(Xp,Yp);
  LineRel(-4,0);
  str(Y:wy:dy,labl);
  OutText(labl);
  Y:=Y+YStep;
  Yp:=Yp-Ypstep;
until Y>YMax;

{Title Y axis}
SetTextStyle(DefaultFont,VertDir,1);   {Set to vertical text}
MoveTo(0,Yptop+(Ypmax-Yptop-Ypbot) div 2);
SetTextJustify(LeftText,CenterText);
OutText(YTitle);
SetTextStyle(DefaultFont,HorizDir,1);{Reset to original settings}

{Main Title}
MoveTo(Xpmax div 2,TextHeight('1'));
SetTextJustify(CenterText,CenterText);
OutText(MainTitle);
```

```
  {Reset Viewport to plotting area}
  with ViewPort do
    SetViewPort(x1+Xplt,y1+Yptop,x2-Xprt,y2-Ypbot,ClipOn);

end;  {DrawXYAxes}

{*****************************************************************}

procedure PlotXYGraph(NumPoints: word; XY: RealPlotArray;
                      XMin,XMax,Ymin,YMax: real; Symbol: integer);
```

{Plots NumPoints of x,y coordinate pairs supplied in XY. The data are plotted in the current viewport, which is detected automatically by the procedure. The scaling for the x and y axes are XMin -> XMax and YMin -> YMax. These must match the values supplied in the DrawXYAxes call, if the data are to be plotted correctly on axes.

RealPlotArray and PlotArray are arrays of real and integer coordinate pairs. These should have global type declarations as follows:

```
      type
         RealPointType = record
                    X,Y,: real;
                  end;
          PlotArray:      array[1..200] of PointType;
            RealPlotArray: array[1..200] of RealPointType;
```

PointType is declared in the Graph Unit as:

```
      type
         PointType = record
                  X,Y,: integer;
               end;
```

The data are plotted according to the value of Symbol:

```
     Symbol=0   =>  line plot
     Symbol=1   =>  plotted as o
     Symbol=2   =>  plotted as +
     Symbol=3   =>  plotted as x
     Symbol=4   =>  plotted as *
```

end of intro}

```
var
  Xpmax,Ypmax: integer;              {Viewport size in pixels}
  XDiff, YDiff: real;                {Axis lengths in graphics coords}
  Viewport: ViewPortType;
  XYp: PlotArray;                    {PlotArray in pixels}
  ipoint: integer;

begin
  {Convert XY to pixel coordinates}
  GetViewSettings(ViewPort);
  with ViewPort do
  begin
    Xpmax:=x2-x1;
    Ypmax:=y2-y1;
  end;
  XDiff:=XMax-XMin;
  YDiff:=YMax-YMin;
  for ipoint:=1 to NumPoints do
  begin
    XYp[ipoint].X:=Xpmax*round(1000*(XY[ipoint].X-Xmin)/Xdiff) div
1000;
    XYp[ipoint].Y:=Ypmax-Ypmax*round(1000*(XY[ipoint].Y-Ymin)/Ydiff)
div 1000;
  end;
  if Symbol=0 then
    DrawPoly(NumPoints,XYp)
  else
  if Symbol<=4 then
  begin
    SetTextJustify(CenterText,CenterText);
    for ipoint:=1 to NumPoints do
    begin
      MoveTo(XYp[ipoint].X,XYp[ipoint].Y);
      case Symbol of
        1: OutText('o');
        2: OutText('+');
        3: OutText('x');
        4: OutText('*');
      end;  {case}
    end;
  end;

end;


{***********************************************************}

procedure StartGraphic;
```

{Initialise graph drivers. The procedure is called by Final procedure written below.}

```
var
  GraphDriver, GraphMode, ErrorCode: integer;  {for Graph Unit}
  ViewPort: ViewPortType;

begin
  GraphDriver:=Detect;  {set flag; do detection}
  InitGraph(GraphDriver,GraphMode,'');
  ErrorCode:=GraphResult;
  if ErrorCode<>grOk then     {error ?}
  begin
    Writeln('Graphics error: ',GraphErrorMsg(ErrorCode));
    Writeln('Program aborted...');
    Halt(1);
  end;
end;
```

```
{**************************************************************}

procedure ReadNewState;
```

{The state variables are saved into a text file called statvar.res. These variables are read into an array called NewState. The name is appropriate since these are an intial state variables. The procedure is called by Main body of the program.}

```
var
  modelfile9,resultfile9 : text;
  idraw,nstate : integer;

begin
  assign(modelfile9,'\pulse\statvar.res');
  reset(modelfile9);
  assign(resultfile9,'\pulse\result6.res');
  rewrite(resultfile9);
    readln(modelfile9,nstate);
    For idraw:=1 to nstate do
    begin
      readln(modelfile9,NewState[idraw]);
      writeln(resultfile9,NewState[idraw]:9:4);
    end;
  close(modelfile9);
  close(resultfile9);
end;  {of procedure ReadNewState}

{**************************************************************}

procedure OutputOptions;
```

{reads the simulation file for graphical & numerical representation. These desired variables with their limits are saved by 'PULSE' into a text file called model3.def. The procedure is called by Initial procedure written below.}

```
var
  modelfile1 : text;
  resultfile1 : text;
  idraw : integer;

begin
  assign(modelfile1,'\pulse\model3.def');
  reset(modelfile1);
  assign(resultfile1,'\pulse\result4.res');
  rewrite(resultfile1);
  with Model,SimulGraph do
  begin
    readln(modelfile1,integstep);
    writeln(resultfile1,'integstep=',integstep);
    readln(modelfile1,simulationtime);
    writeln(resultfile1,'simulationtime=',simulationtime);
    readln(modelfile1,ndraw);
    writeln(resultfile1,'ndraw=',ndraw);
    for idraw:=1 to ndraw do
    begin
      readln(modelfile1,flag[idraw]);
      readln(modelfile1,namevar[idraw]);
      readln(modelfile1,name[idraw],title[idraw]);
      write(resultfile1,'flag[',idraw,']=',flag[idraw]);
      write(resultfile1,'  namevar[',idraw,']=',namevar[idraw]);
      writeln(resultfile1);
      write(resultfile1,'name[',idraw,']=',name[idraw]);
      write(resultfile1,'  title[',idraw,']=',title[idraw]);
      writeln(resultfile1);
    end;

writeln(resultfile1,'*************************************************
****');
      writeln(resultfile1);
    For idraw:=1 to ndraw do
    begin
      if flag[idraw]='compar' then
      begin
        with MeasComp[idraw] do
        begin
          readln(modelfile1,icomp,plotmin,plotmax);
          writeln(resultfile1,'idraw = ',idraw,'  icomp=',icomp,'
plotmin=',plotmin,'  plotmax=',plotmax);
        end;
      end
      else if flag[idraw]='fcompa' then
      begin
        with MeasFcomp[idraw] do
        begin
          readln(modelfile1,ito,ifrom,plotmin,plotmax);
          writeln(resultfile1,'idraw = ',idraw,'  ito=',ito,'
ifrom=',ifrom,'  plotmin=',plotmin,'  plotmax=',plotmax);
        end;
      end
      else if flag[idraw]='others' then
      begin
        with MeasOther[idraw] do
        begin
          readln(modelfile1,plotmin,plotmax);
          writeln(resultfile1,'idraw = ',idraw,'  plotmin=',plotmin,'
plotmax=',plotmax);
        end;
      end;
    end;
  end;
  close(modelfile1);
  close(resultfile1);
end;  {of procedure OutputOptions}
```

```
{*****************************************************************}

procedure InitialNeural;
```

{reads the output file written by 'PULSE' for the neural control. That is if neural control is selected by the user. The initial values for the neural control is saved into a file called result2.res. The procedure is called by Initial procedure written below.}

```
var
  modelfile2,resultfile2 : text;
  ibar : integer;

begin

  assign(modelfile2,'\pulse\model2.def');
  reset(modelfile2);
  assign(resultfile2,'\pulse\result2.res');
  rewrite(resultfile2);
  with Model,CnsControl do
  begin
    readln(modelfile2,nbar);
    for ibar:=1 to nbar do
    begin
      with Bar[ibar] do
      begin
        readln(modelfile2,i,t1,t2,a1,a2,PTi);
        x1:=NewState[kstate];
        kstate:=kstate+1;
        x2:=NewState[kstate];
        kstate:=kstate+1;
        writeln(resultfile2);
        writeln(resultfile2,'**** Baroreceptors ****');
        writeln(resultfile2,'Neural Control is ',BangBang);
        writeln(resultfile2,'sensed artery ',i);
        writeln(resultfile2,'State variable 1 ',x1);
        writeln(resultfile2,'State variable 2 ',x2);
        writeln(resultfile2,'Time constant 1 ',t1);
        writeln(resultfile2,'Time constant 2 ',t2);
        writeln(resultfile2,'Weighting factor 1',a1);
        writeln(resultfile2,'Weighting factor 2',a2);
        writeln(resultfile2,'Thereshold pressure',PTi);
      end;
    end;

    with HeartRate do
    begin
      for ibar:=1 to nbar do
      begin
        with CnsInp[ibar] do
        begin
          readln(modelfile2,i,ai);
          cnsconst1[i]:=ai;
        end;
      end;
      readln(modelfile2,BHRT,t1,t2,t3,t4,THmax,THmin,b1,b2);
      x1:=0.0;
      x3:=NewState[kstate];
    kstate:=kstate+1;
    x5:=NewState[kstate];
    kstate:=kstate+1;
    x6:=NewState[kstate];
    kstate:=kstate+1;
    writeln(resultfile2);
    writeln(resultfile2,'**** Heart Rate ****');
    for ibar:=1 to nbar do
    begin
      with CnsInp[ibar] do
      begin
        writeln(resultfile2,'sensed artery  ',i);
        writeln(resultfile2,' a[',i,'] ',cnsconst1[i]);
      end;
    end;
    writeln(resultfile2,'State variable 3 ',x3);
    writeln(resultfile2,'State variable 5 ',x5);
    writeln(resultfile2,'State variable 6 ',x6);
    writeln(resultfile2,'Time constant 1 ',t1);
    writeln(resultfile2,'Time constant 2 ',t2);
    writeln(resultfile2,'Time constant 3 ',t3);
    writeln(resultfile2,'Time constant 4 ',t4);
    writeln(resultfile2,'Maximum TH',THmax);
    writeln(resultfile2,'Minimum TH',THmin);
    writeln(resultfile2,'Weighting factor 1',b1);
    writeln(resultfile2,'Weighting factor 2',b2);
  end;  {with heart rate}

  readln(modelfile2,Periph);
  if Periph='true' then
  begin
   with PerRes do
    begin
      for ibar:=1 to nbar do
      begin
        with CnsInp[ibar] do
        begin
          readln(modelfile2,i,ai);
          cnsconst2[i]:=ai;
        end;
      end;
      readln(modelfile2,BPRT,t1,t2,b1,b2,b3);
      x2:=NewState[kstate];
      kstate:=kstate+1;
      x3:=NewState[kstate];
      kstate:=kstate+1;
      writeln(resultfile2,'**** Peripheral Resistance ****');
      writeln(resultfile2,'Peripheral Resistance is ',Periph);
      writeln(resultfile2,'Peripheral Baroreceptor',BPRT);
      writeln(resultfile2,'State variable 2 ',x2);
      writeln(resultfile2,'State variable 3 ',x3);
      writeln(resultfile2,'Time constant 1 ',t1);
      writeln(resultfile2,'Time constant 2 ',t2);
      writeln(resultfile2,'Weighting factor 1',b1);
      writeln(resultfile2,'Weighting factor 2',b2);
      writeln(resultfile2,'Weighting factor 3',b3);
    end;
  end;
```

```
    readln(modelfile2,Myocar);
    if Myocar='true' then
    begin
      with MyoCon do
      begin
        for ibar:=1 to nbar do
        begin
          with CnsInp[ibar] do
          begin
            readln(modelfile2,i,ai);
            cnsconst3[i]:=ai;
          end;
        end;
        readln(modelfile2,BMCT,t1,b1,b2);
        x2:=NewState[kstate];
        kstate:=kstate+1;
        writeln(resultfile2);
        writeln(resultfile2,'**** Myocardial Contractility ****');
        writeln(resultfile2,'Myocardial Contractility is ',Myocar);
        writeln(resultfile2,'Myocardial Baroreceptor',BMCT);
        writeln(resultfile2,'State variable 2 ',x2);
        writeln(resultfile2,'Time constant 1 ',t1);
        writeln(resultfile2,'Weighting factor 1',b1);
        writeln(resultfile2,'Weighting factor 2',b2);
      end;
    end;
    readln(modelfile2,Venous);
    if Venous='true' then
    begin
      with VenTon do
      begin
        for ibar:=1 to nbar do
        begin
          with CnsInp[ibar] do
          begin
            readln(modelfile2,i,ai);
            cnsconst4[i]:=ai;
          end;
        end;
        readln(modelfile2,BVTT,t1,b1,b2,b3,b4);
        x2:=NewState[kstate];
        kstate:=kstate+1;
        writeln(resultfile2);
        writeln(resultfile2,'**** Venous Tone ****');
        writeln(resultfile2,'Venous Tone is ',Venous);
        writeln(resultfile2,'Venous Baroreceptor',BVTT);
        writeln(resultfile2,'Time constant 1 ',t1);
        writeln(resultfile2,'Weighting factor 1',b1);
        writeln(resultfile2,'Weighting factor 2',b2);
        writeln(resultfile2,'Weighting factor 3',b3);
        writeln(resultfile2,'Weighting factor 4',b4);
      end;
    end;
  end; {of with Model,CnsControl do}
close(modelfile2);
close(resultfile2);
end;  {of procedure InitialNeural}
```

```
{****************************************************************}

procedure HeartControl;
```

{initialises the neural control variables in the circulatory model, the results are saved into a file called result8.res. The procedure is called by Initial procedure written below.}

```
var
  ibar : integer;

  resultfile8:text;

begin
  assign(resultfile8,'\pulse\result8.res');
  rewrite(resultfile8);

    with Model,CnsControl do
    {Neural Control}
    begin
      if Periph='true' then
      begin
        with PerRes do
        begin
          x4:=b3*x3+(1-b3)*x2;
          CnsP:=x4;
        end; {with PerRes}
      end {if Periph='true'}
      else
        CnsP:=1.0;

      if Myocar='true' then
      begin
        with MyoCon do
        begin
          CnsM:=x2;
        end; {with MyoCon}
      end {if Myocardial='true'}
      else
        CnsM:=1.0;

      if Venous='true' then
      begin
        with VenTon do
        begin
          x3:=1+b3*(x2-1);
          x4:=1+b4*(x2-1);
          CnsVC:=x3;
          CnsVVu:=x4;
        end; {of VenTon}
      end {if Venous='true'}
      else
      begin
        CnsVC:=1.0;
        CnsVVu:=1.0;
      end;
  writeln(resultfile8,'periph = ',periph);
  writeln(resultfile8,'myocar = ',myocar);
  writeln(resultfile8,'venous = ',venous);
  writeln(resultfile8,'CnsP = ',CnsP);
  writeln(resultfile8,'CnsM = ',CnsM);
  writeln(resultfile8,'CnsVC  = ',CnsVC);
  writeln(resultfile8,'CnsVVu = ',CnsVVu);
  close(resultfile8);
    end; {with Model,CnsControl}

end; {of HeartControl}

{*****************************************************************}

procedure InitialCirculation;
```

{reads the output file for circulatatory fluid dynamics and saves the initial values for the circulation into a file called result1.res. The procedure is called by Initial procedure written below.}

```
var

  modelfile3,resultfile3 : text;
  iart,ivein,ivent,iatr,icomp,idraw : integer;
  ifrart,ifrvein,ifrvent,ifratr : integer;

begin

  assign(modelfile3,'\pulse\model1.def');
  reset(modelfile3);
  assign(resultfile3,'\pulse\result1.res');
  rewrite(resultfile3);

  kstate:=1;
  with Model,HeartCirc do
  begin
    readln(modelfile3,nart);
    for iart:=1 to nart do
    begin
      with Art[iart] do
      begin
        readln(modelfile3,i,Ci,Vui,Ki,Pregion);
        Vi0:=NewState[kstate];
        kstate:=kstate+1;
        V[i]:=Vi0;
        Preg[i]:=Pregion;
        writeln(resultfile3);
        writeln(resultfile3,'Compartment ',i,' *** Artery ***');
        writeln(resultfile3,'Compliance ',Ci);
        writeln(resultfile3,'Unstressed volume ',Vui);
        writeln(resultfile3,'Viscoelasticity ',Ki);
        writeln(resultfile3,'Preg[',i,'] ',Preg[i]);
        writeln(resultfile3,'Initial volume ',Vi0);
      end;
    end; {of arteries}

    readln(modelfile3,nvein);
    for ivein:=1 to nvein do
    begin
      with Vein[ivein] do
      begin
        readln(modelfile3,i,Ci0,Vui,alfai,Pregion);
        readln(modelfile3,neural);
        Vi0:=NewState[kstate];
        kstate:=kstate+1;
        V[i]:=Vi0;
        alfaCi0:=alfai*Ci0;
        Preg[i]:=Pregion;
        writeln(resultfile3);
        writeln(resultfile3,'Compartment ',i,' *** Vein ***');
        writeln(resultfile3,'Compliance ',Ci0);
        writeln(resultfile3,'Unstressed volume ',Vui);
        writeln(resultfile3,'Alfai ',alfai);
        writeln(resultfile3,'Preg[',i,'] ',Preg[i]);
        writeln(resultfile3,'Initial volume ',Vi0);
        writeln(resultfile3,'Neural effect',neural);
      end;
    end; {of veins}

    readln(modelfile3,nvent);
    for ivent:=1 to nvent do
    begin
      with Vent[ivent] do
      begin
  readln(modelfile3,i,Vui,ais,aid,lambda1,lambda2,lambda3,Pregion);
        Vi0:=NewState[kstate];
        kstate:=kstate+1;
        V[i]:=Vi0;
        Preg[i]:=Pregion;
          writeln(resultfile3);
        writeln(resultfile3,'Compartment ',i,' *** Ventricle ***');
        writeln(resultfile3,'Systolic elastance ',ais);
        writeln(resultfile3,'Diastolic elastance ',aid);
        writeln(resultfile3,'Unstressed volume ',Vui);
        writeln(resultfile3,'CO[',i,'] = ',CO[i]);
        writeln(resultfile3,'Preg[',i,'] ',Preg[i]);
        writeln(resultfile3,'lambda1',lambda1);
        writeln(resultfile3,'lambda2',lambda2);
        writeln(resultfile3,'lambda3',lambda3);
        writeln(resultfile3,'Initial volume ',Vi0);
      end;
    end; {of ventricles}

    readln(modelfile3,natr);
    for iatr:=1 to natr do
    begin
      with Atr[iatr] do
      begin
        readln(modelfile3,i,Vui,ais,aid,lambda1,lambda2,Pregion);
        Vi0:=NewState[kstate];
        kstate:=kstate+1;
        V[i]:=Vi0;
        Preg[i]:=Pregion;
        writeln(resultfile3);
        writeln(resultfile3,'Compartment ',i,' *** Atria ***');
        writeln(resultfile3,'Systolic elastance ',ais);
        writeln(resultfile3,'Diastolic elastance ',aid);
        writeln(resultfile3,'Unstressed volume ',Vui);
        writeln(resultfile3,'Preg[',i,'] ',Preg[i]);
        writeln(resultfile3,'lambda1',lambda1);
        writeln(resultfile3,'lambda2',lambda2);
        writeln(resultfile3,'Initial volume ',Vi0);
      end;
    end; {of atria}

    ncomp:=nart+nvein+nvent+natr;

{initialise the end of cardiac cycle variables}
    for icomp:=1 to ncomp do
    begin
      writeln(resultfile3,'Preg[',icomp,'] = ',Preg[icomp]);
      DX[icomp]:=0.0;
      D2X[icomp]:=0.0;
      P[icomp]:=0.0;
      integpres[icomp]:=0.0;
      MP[icomp]:=0.0;
      ETSR[icomp]:=0.0;
      DV[icomp]:=0.0;
    end;

    readln(modelfile3,nfromart);
    for ifrart:=1 to nfromart do
    begin
      with FromArt[ifrart] do
      begin
        readln(modelfile3,j,i,Rji,lenji,anglji,Lji,Pcc);
        readln(modelfile3,neural);
          Gji:=0.7807*lenji*sin(anglji*0.01745);
{        0.7807 = n.rho.g  with n=1}

        Fji:=NewState[kstate];
        kstate:=kstate+1;
        F[j,i]:=Fji;
        DF[j,i]:=0.0;
        if Lji>0 then
        begin
          DX[i]:=DX[i]-F[j,i];
          DX[j]:=DX[j]+F[j,i];
          invLji:=1/Lji;
        end;
        writeln(resultfile3);
        writeln(resultfile3,'**** Flow from Artery ****');
        writeln(resultfile3,'From ',i,' to ',j);
        writeln(resultfile3,'Resistance ',Rji);
        writeln(resultfile3,'Length ',lenji);
        writeln(resultfile3,'Angle ',anglji);
        writeln(resultfile3,'Inertance ',Lji);
        writeln(resultfile3,'Neural effect',neural);
        writeln(resultfile3,'Calculated hydrostatic pressure ',Gji);
      end;
    end; {of flow from arteries}

    readln(modelfile3,nfromvein);
    for ifrvein:=1 to nfromvein do
    begin
```

```
      with FromVein[ifrvein] do
      begin
        readln(modelfile3,j,i,Rji,lenji,anglji,Vui,betai);
        readln(modelfile3,valve);
        Gji:=0.7807*lenji*sin(anglji*0.01745);
{        0.7807 = n.rho.g  with n=1}
        invRV2ji:=1/(Rji*SQR(Vui));

        Fji:=NewState[kstate];
        kstate:=kstate+1;
        F[j,i]:=Fji;
        DF[j,i]:=0.0;
        writeln(resultfile3);
        writeln(resultfile3,'**** Flow from Vein ****');
        writeln(resultfile3,'From ',i,' to ',j);
        writeln(resultfile3,'Resistance ',Rji);
        writeln(resultfile3,'Valve fraction ',betai);
        writeln(resultfile3,'invRV2[',j,',',i,'] = ',invRV2ji);
        if valve='true' then
          writeln(resultfile3,'Valve exists ');
      end;
    end; {of flow from veins}

    readln(modelfile3,nfromvent);
    for ifrvent:=1 to nfromvent do
    begin
      with FromVent[ifrvent] do
      begin
        readln(modelfile3,j,i,Rji,Lji,Aji);
        readln(modelfile3,left);

        Fji:=NewState[kstate];
        kstate:=kstate+1;
        F[j,i]:=Fji;
        DF[j,i]:=0.0;
        DX[i]:=DX[i]-F[j,i];
        DX[j]:=DX[j]+F[j,i];
        if Aji>0 then
          rhoAji:=1.06/(2.0*Aji*Aji*1332);
        integF[j,i]:=0;
        CO[i]:=0.0;
        SV[i]:=0.0;
        writeln(resultfile3);
        writeln(resultfile3,'**** Flow from Ventricle ****');
        writeln(resultfile3,'From ',i,' to ',j);
        writeln(resultfile3,'left = ',left);
        writeln(resultfile3,'Resistance ',Rji);
        writeln(resultfile3,'Initial flow ',F[j,i]);
        writeln(resultfile3,'Inertance ',Lji);
        writeln(resultfile3,'Area ',Aji);
        writeln(resultfile3,'Calculated rhoAji ',rhoAji);
      end;
    end; {of flow from ventricles}

    readln(modelfile3,nfromatr);
    for ifratr:=1 to nfromatr do
    begin
      with FromAtr[ifratr] do
      begin
        readln(modelfile3,j,i,Rji);

        Fji:=NewState[kstate];
        kstate:=kstate+1;
        F[j,i]:=Fji;
        DF[j,i]:=0.0;
        writeln(resultfile3);
        writeln(resultfile3,'**** Flow from Atrium ****');
        writeln(resultfile3,'From ',i,' to ',j);
        writeln(resultfile3,'Resistance ',Rji);
      end;
    end; {of flow from atria}

  end; {of with Model,HeartCirc do}

  {check if neural control needed}
  readln(modelfile3,BangBang);
  with Model,HeartCirc do
  begin
    for iart:=1 to nart do
    begin
      with Art[iart] do
      begin
        readln(modelfile3,sensedA);
        writeln(resultfile3,'sensedA[',iart,'] = ',sensedA);
      end;
    end;
  end;
  close(modelfile3);
  close(resultfile3);
end;  {of procedure InitialCirculation}

{**********************************************************}

procedure Initial;
```

{Calls the above procedures in the appropriate order. The procedure is called by the main body of the program.}

```
begin

  InitialCirculation;
  OutputOptions;
  if BangBang='true' then
  begin
    InitialNeural;;
    HeartControl;
  end
  else
  begin

   {initialise the neural control variable}
    CnsP:=1.0;
    CnsM:=1.0;
    CnsVC:=1.0;
    CnsVVu:=1.0;
  end;
  TH:=NewState[kstate];

end;  {of procedure Initial}

{****************************************************************}

procedure UpdateHeartTimes;
```

{updates heart time, TAS (atrial systolic time),etc. The procedure is called by ModelEquations procedure written below.}

```
var
  ivent,iatr : integer;
  TASnew : real;

begin
  with Model,HeartCirc do
  begin

    for iatr:=1 to natr do
    begin
      with Atr[iatr] do
      begin
        TAS:=lambda1+lambda2*TH;
        TASnew:=TAS;
        piTAS:=pi/TAS;
        aisd:=CnsM*ais-aid;
      end;
    end;

    for ivent:=1 to nvent do
    begin
      with Vent[ivent] do
      begin
        TAV:=TASnew-lambda1;
        TVS:=lambda2+lambda3*TH;
        TAVS:=TAV+TVS;
        piTVS:=pi/TVS;
        aisd:=CnsM*ais-aid;
      end;
    end;

  end; {of with Model,HeartCirc}

end; {of procedure UpdateHeartTimes}

{****************************************************************}

function pos(xx:real):real;
```

{ function used to return the positive value where ever is needed. The function is called by ModelEquations procedure written below.}

```
begin
  if xx>0 then pos:=xx else pos:=0.0;
end;
```

```
{*******************************************************************}

procedure ModelEquations;
```

{drives the equations for the circulatory and neural control model. The procedure is called by AdjustStep procedure written below.}

```
var

  iart,ivein,ivent,iatr,icomp : integer;
  ifrart,ifrvein,ifrvent,ifratr : integer;
  ibar : integer;
  temp : array[1..10] of string[5];
  temp1,temp2,temp3,temp4,temp5,temp6,SLOPE : real;
  Pconst1,Pconst2,Pconst3,difpres,Pchang,ConstCi : StateSpace4;

begin

{drives the circulatory equations}

  with Model,HeartCirc do
  begin
    for iart:=1 to nart do
    begin
      Pconst1[iart]:=0;
      Pconst2[iart]:=0;
      Pconst3[iart]:=0;
      difpres[iart]:=0;
    end;
{pressures}
    for iatr:=1 to natr do
    begin
      with Atr[iatr] do
      begin
        if tcardiac>TAS then
          x:=0.0
        else
          x:=sin(tcardiac*piTAS);
        ait:=x*aisd+aid;
        EL[i]:=ait;
        P[i]:=ait*(V[i]-Vui);
        difMP[i]:=P[i];
      end;
    end; {of atria}

    for ivent:=1 to nvent do
    begin
      with Vent[ivent] do
      begin
        if (tcardiac<=TAV) or (tcardiac>TAVS) then
          x:=0.0
        else
          x:=sin((tcardiac-TAV)*piTVS);
        ait:=x*aisd+aid;
        EL[i]:=ait;
        P[i]:=ait*(V[i]-Vui);
        difMP[i]:=P[i];
        end;
    end; {of ventricles}

    for ivein:=1 to nvein do
    begin
      with Vein[ivein] do
      begin
        if neural='true' then sigmaVui:=Vui/CnsVVu else
sigmaVui:=Vui;
        if V[i]>sigmaVui then Ci:=Ci0 else Ci:=alfaCi0;
        if neural='true' then sigmaCi:=CnsVC/Ci else sigmaCi:=1/Ci;
        P[i]:=(V[i]-sigmaVui)*(sigmaCi);
        difMP[i]:=P[i];
        ConstCi[i]:=sigmaCi;
      end;
    end; {of veins}

{for the calculation of pressure in arteries the following constant
are needed}
     for ifrart:=1 to nfromart do
     begin
       with FromArt[ifrart] do
       begin
         if neural='true' then sigmaRji:=CnsP*Rji else sigmaRji:=Rji;
         if Lji=0 then
         begin
           Pconst1[i]:=P[j]/sigmaRji;
           Pconst2[i]:=1/sigmaRji;
         end;
       end;
    end; {of fromArt for initialization}

    for iart:=1 to nart do
    begin
      with Art[iart] do
      begin
        P[i]:=((V[i]-Vui)+Ki*(DX[i]+Pconst1[i]))/(Ci+Ki*Pconst2[i]);
        difMP[i]:=P[i];
      end;
    end; {of arteries}
    {Calculate mean pressure}

{zero derivative}
    for icomp:=1 to ncomp do
    begin
      DV[icomp]:=0.0;
      DX[icomp]:=0.0;
      D2X[icomp]:=0.0;
    end;

{flows}
    for ifratr:=1 to nfromatr do
    begin
      with FromAtr[ifratr] do
      begin
        temp2:=(P[i]-P[j])/Rji;
        if temp2>0 then F[j,i]:=temp2 else F[j,i]:=0;
        DV[i]:=DV[i]-F[j,i];
        DV[j]:=DV[j]+F[j,i];
    end;
  end; {of flow from atria}

  for ifrvent:=1 to nfromvent do
  begin
    with FromVent[ifrvent] do
    begin
      if Aji>0 then
        DF[j,i]:=(P[i]-P[j]-Rji*F[j,i]-F[j,i]*F[j,i]*rhoAji)/Lji
      else
        DF[j,i]:=(P[i]-P[j]-Rji*F[j,i])/Lji;
      if F[j,i]<0 then F[j,i]:=0;
      DV[i]:=DV[i]-F[j,i];
      DV[j]:=DV[j]+F[j,i];
      if DV[i]<0 then SLOPE:=-200*V[i];
      if DV[i]<SLOPE then DV[i]:=SLOPE;
      if DV[j]<0 then SLOPE:=-200*V[j];
      if DV[j]<SLOPE then DV[j]:=SLOPE;
      DX[i]:=DX[i]-F[j,i];
      DX[j]:=DX[j]+F[j,i];
    end;
  end; {of flow from ventricles}


  for ifrvein:= 1 to nfromvein do
  begin
    with FromVein[ifrvein] do
    begin
      temp6:=SQR(V[i]);
      temp5:=temp6*invRV2ji;
      temp1:=temp5*(P[i]+Preg[i]-P[j]-Preg[j]-Gji);
      if temp1>0 then
        F[j,i]:=temp1
      else
      begin
        if valve='true' then F[j,i]:=0.0 else F[j,i]:=betai*temp1;
      end;
      DV[i]:=DV[i]-F[j,i];
      DV[j]:=DV[j]+F[j,i];
    end;
  end; {of flow from veins}

  for ifrart:=1 to nfromart do
  begin
    with FromArt[ifrart] do
    begin
      if neural='true' then sigmaRji:=CnsP*Rji else sigmaRji:=Rji;
      if Pcc<>0.0 then
      begin
        if P[j]>Pcc then
          F[j,i]:=(P[i]-P[j])/sigmaRji
        else
        F[j,i]:=(P[i]-Pcc)/sigmaRji;
      end
      else {Pcc=0.0}
      begin
        if Lji>0 then
        begin
```

```
            temp4:=(P[i]+Preg[i]-P[j]-Preg[j]-Gji-sigmaRji*F[j,i]);
            DF[j,i]:=temp4*invLji;
            DX[i]:=DX[i]-F[j,i];
            DX[j]:=DX[j]+F[j,i];
            D2X[i]:=D2X[i]-DF[j,i];
            D2X[j]:=D2X[j]+DF[j,i];
          end
          else    {Lji=0}
            F[j,i]:=(P[i]+Preg[i]-P[j]-Preg[j]-Gji)/sigmaRji;
        end;
        DV[i]:=DV[i]-F[j,i];
        DV[j]:=DV[j]+F[j,i];
        if DV[i]<0 then SLOPE:=-200*V[i];
        if DV[i]<SLOPE then DV[i]:=SLOPE;
        if DV[j]<0 then SLOPE:=-200*V[j];
        if DV[j]<SLOPE then DV[j]:=SLOPE;
      end;
    end; {of fromArt}

{for the calculation of dpressure (rate of change of pressure in
arteries following constant are needed}

    for ivein:=1 to nvein do
    begin
      with Vein[ivein] do
      begin
        if neural='true' then sigmaVui:=Vui/CnsVVu else
sigmaVui:=Vui;
        if V[i]>sigmaVui then Ci:=Ci0 else Ci:=alfaCi0;
        if neural='true' then sigmaCi:=CnsVC/Ci else sigmaCi:=1/Ci;
        ConstCi[i]:=sigmaCi;
        Pchang[i]:=ConstCi[i]*DV[i];
      end;
    end; {of veins}

     for ifrart:=1 to nfromart do
     begin
       with FromArt[ifrart] do
       begin
         if neural='true' then sigmaRji:=CnsP*Rji else sigmaRji:=Rji;
         if Lji=0 then
         begin
           Pconst3[i]:=Pchang[j]/sigmaRji;
           Pconst2[i]:=1/sigmaRji;
         end;
       end;
    end; {of fromArt for initialization}

    for iart:=1 to nart do
    begin
      with Art[iart] do
      begin
        if sensedA ='true' then

difpres[i]:=(DV[i]+Ki*(D2X[i]+Pconst3[i]))/(Ci+Ki*Pconst2[i])
        else
          difpres[i]:=0;
      end;
    end; {of arteries}

  end; {of Model,HeartCirc}


{drive the neural control equations}

  if BangBang='true' then
  begin
    with Model,CnsControl do
    {Neural Control}
    begin
      for ibar:=1 to nbar do
      begin
        with Bar[ibar] do
        begin
          temp3:=a1*((x1-PTi)+a2*x2);
          B[i]:=pos(temp3);
          dx1:=(P[i]-x1)/t1;
          dpres:=pos(difpres[i]);
          dx2:=(dpres-x2)/t2;
          if temp3>0 then DB[i]:=a1*(dx1+a2*dx2) else DB[i]:=0.0;
        end;
      end;{of Baroreceptors}

      with HeartRate do
      begin
        x7:=b2*(x3+x6);
        if x7>=THmax then
          TH:=THmax
        else
          if  (x7>THmin) then TH:=x7 else TH:=THmin;
        BHR:=0.0;
        DBHR:=0.0;
        for ibar:=1 to nbar do
        begin
          with CnsInp[ibar] do
          begin
            BHR:=cnsconst1[i]*B[i]+BHR;
            CnsHea:=BHR;
            DBHR:=cnsconst1[i]*DB[i]+DBHR;
          end;
        end;
        if BHR<=BHRT then
        begin
          x1:=0.0;
          dx1:=0.0;
          x4:=BHR;
        end
        else
        begin
          x1:=b1*(BHR-BHRT);
          dx1:=b1*DBHR;
          x4:=BHRT;
        end;
        if (dx1>0) then x2:=t1 else x2:=t2;
        dx3:=(x1-x3)/x2;
        dx5:=(x4-x5)/t3;
        dx6:=(x5-x6)/t4;
      end; {with HeartRate}

      if Periph='true' then
      begin
        with PerRes do
        begin
          x4:=b3*x3+(1-b3)*x2;
          CnsP:=x4;
          BPR:=0.0;
          for ibar:=1 to nbar do
          begin
            with CnsInp[ibar] do
            begin
              BPR:=cnsconst2[i]*B[i]+BPR;
              CnsPer:=BPR;
            end;
          end;
          if BPR>BPRT then x1:=b1 else x1:=b2;
          dx2:=(x1-x2)/t1;
          dx3:=(x1-x3)/t2;
        end; {with PerRes}
      end; {if Peripheral='true'}

      if Myocar='true' then
      begin
        with MyoCon do
        begin
          CnsM:=x2;
          BMC:=0.0;
          for ibar:=1 to nbar do
          begin
            with CnsInp[ibar] do
            begin
              BMC:=cnsconst3[i]*B[i]+BMC;
              CnsMyo:=BMC;
            end;
          end;
          if BMC>BMCT then x1:=b1 else x1:=b2;
            dx2:=(x1-x2)/t1;
        end; {with MyoCon}
      end; {if Myocardial='true'}

      if Venous='true' then
      begin
        with VenTon do
        begin
          x3:=1+b3*(x2-1);
          x4:=1+b4*(x2-1);
          CnsVC:=x3;
          CnsVVu:=x4;
          BVT:=0.0;
          for ibar:=1 to nbar do
          begin
            with CnsInp[ibar] do
            begin
              BVT:=cnsconst4[i]*B[i]+BVT;
              CnsVen:=BVT;
```

```
          end;
        end;
        if BVT>BVTT then x1:=b1 else x1:=b2;
          dx2:=(x1-x2)/t1;
      end; {of VenTon}
    end; {if Venous='true'}
  end; {with Model,CnsControl}
 end; {of if BangBang='true'}

end; {of procedure ModelEquations}
```

{}**

```
procedure Count(var totalno:integer);
```

{to count the number of state variables to copy for next time the user runs the simulation. This is when the user is finished with the simulation and maybe would like to run it again with these state variables values rather than the initialise state variables. This procedure is called by the ValuesSaved procedure written below.}

```
var

  ibar: integer;
  count1,count2 : integer;

begin

{Volume & flow count}
  with Model,HeartCirc do

count1:=nart+nvein+natr+nvent+nfromart+nfromvein+nfromatr+nfromvent;

    { for TH state variable}

  count2:=1;
  {if neural control exist}

  if BangBang='true' then
  begin

  {Cns Control count}
    with Model,CnsControl do
    begin
      for ibar:=1 to nbar do
      begin
        with Bar[ibar] do
          count2:=count2+2;
      end;

      with HeartRate do
        count2:=count2+3;

      if Periph='true' then
      begin
        with PerRes do
          count2:=count2+2;
      end;
      if Myocar='true' then
      begin
        with MyoCon do
          count2:=count2+1;
      end;

      if Venous='true' then
      begin
        with VenTon do
          count2:=count2+1;
      end;
    end; {of Cns Control count}

  end; {if BangBang='true'}

totalno:=count1+count2;
end; {of Count}

{***********************************************************}

procedure Integ(nv: integer;var treal,tstep: real;var Z:StateSpace3);
```

{integrates the aquired variables using Eulers method. The new state variables are saved into an array called Z which is used next time the simulation was run. This array is called by procedure ValuesSaved written below. The procedure is called by AdjustStep procedure written below.}

```
var

  i,j,ifrart,ifrvent,ifratr,ifrvein,ibar,istep: integer;
  dtcardiac: real;

begin
  dtcardiac:=1.0;

{Volume update}
  for i:=1 to nv do
  begin
    V[i]:=V[i]+tstep*DV[i];
    integpres[i]:=integpres[i]+tstep*difMP[i];
    Z[i]:=V[i];
  end;
istep:=nv+1;

  {Flow update}
  with Model,HeartCirc do
  begin
    for ifrart:=1 to nfromart do
    begin
      with FromArt[ifrart] do
      begin
        if Lji>0 then
        begin
          F[j,i]:=F[j,i]+tstep*DF[j,i];
          Z[istep]:=F[j,i];
          istep:=istep+1;
```

```
         end
         else
         begin
           Z[istep]:=F[j,i];
           istep:=istep+1;
         end;
       end;
     end;
     for ifrvein:=1 to nfromvein do
     begin
       with FromVein[ifrvein] do
       begin
         Z[istep]:=F[j,i];
         istep:=istep+1;
       end;
     end;
     for ifrvent:=1 to nfromvent do
     begin
       with FromVent[ifrvent] do
       begin
         F[j,i]:=F[j,i]+tstep*DF[j,i];
         if F[j,i]<0 then F[j,i]:=0;
         integF[j,i]:=integF[j,i]+tstep*F[j,i];
         Z[istep]:=F[j,i];
         istep:=istep+1;
       end;
     end;
     for ifratr:=1 to nfromatr do
     begin
       with FromAtr[ifratr] do
       begin
         Z[istep]:=F[j,i];
         istep:=istep+1;
       end;
     end;
   end; {of flow update}

   {if neural control exist}
   if BangBang='true' then

   begin

     {Cns Control update}

     with Model,CnsControl do
     begin
       for ibar:=1 to nbar do
       begin
         with Bar[ibar] do
         begin
           x1:=x1+tstep*dx1;
           Z[istep]:=x1;
           istep:=istep+1;
           x2:=x2+tstep*dx2;
           Z[istep]:=x2;
           istep:=istep+1;
         end;
       end;
```

```
      with HeartRate do
      begin
        x3:=x3+tstep*dx3;
        Z[istep]:=x3;
        istep:=istep+1;
        x5:=x5+tstep*dx5;
        Z[istep]:=x5;
        istep:=istep+1;
        x6:=x6+tstep*dx6;
        Z[istep]:=x6;
        istep:=istep+1;
      end;

      if Periph='true' then
      begin
        with PerRes do
        begin
          x2:=x2+tstep*dx2;
          Z[istep]:=x2;
          istep:=istep+1;
          x3:=x3+tstep*dx3;
          Z[istep]:=x3;
          istep:=istep+1;
        end;
      end;

      if Myocar='true' then
      begin
        with MyoCon do
        begin
          x2:=x2+tstep*dx2;
          Z[istep]:=x2;
          istep:=istep+1;
        end;
      end;

      if Venous='true' then
      begin
        with VenTon do
        begin
          x2:=x2+tstep*dx2;
          Z[istep]:=x2;
          istep:=istep+1;
        end;
      end;
    end; {of Cns Control update}

  end; {if BangBang='true'}

  {Time update}
  tcardiac:=tcardiac+tstep*dtcardiac;

    Z[istep]:=TH;

end; {of procedure Integ}
```

```
{***********************************************************}

procedure AdjustStep(var treal: real;var Z:StateSpace3);
```

{to calles the two procedures mentioned above; ModelEquations and Integ and then increase the time by the integration step length value.}

```
var
  nv,i,j,ifrart,ifrvent,ibar: integer;

begin
  nv:=Model.HeartCirc.ncomp;
  with Model,SimulGraph do
  begin
    Integ(nv,treal,integstep,Z);
    ModelEquations;
    treal:=treal+integstep;
  end;
end; {of AdjustStep}

{***********************************************************}

procedure MakeScreen;
```

{draws the windows, draws the axes and displays the appropriate units for the selected variables. The procedure is calles by final procedure written below.}

```
var

  idraw,ind: integer;
  xtitle,ytitle,maintitle: string[38];
  ViewPort: ViewPortType;

begin
    with Model,SimulGraph do
  begin
  {set window parameters}
    ind:=10;
    w[1].x1:=ind;
    w[1].y1:=ind;
    w[1].x2:=(GetMaxX div 2)-ind;
    w[1].y2:=(GetMaxY div 2)-ind;
    w[2].x1:=w[1].x2+2*ind;
    w[2].y1:=w[1].y1;
    w[2].x2:=GetMaxX-ind;
    w[2].y2:=w[1].y2;
    w[3].x1:=w[1].x1;
    w[3].y1:=w[1].y2+ind;
    w[3].x2:=w[1].x2;
    w[3].y2:=GetMaxY-ind;
    w[4].x1:=w[2].x1;
    w[4].y1:=w[3].y1;
    w[4].x2:=w[2].x2;
    w[4].y2:=w[3].y2;
    w[5].x1:=w[1].x1;
```

```
w[5].y1:=w[3].y2+ind;
w[5].x2:=w[2].x2;
w[5].y2:=GetMaxY-ind;
xtitle:='Time (sec)';
for idraw:=1 to 4 do
begin
  with w[idraw] do
    SetViewPort(x1,y1,x2,y2,ClipOff);
  if namevar[idraw]='pressu' then
  begin
     ytitle:='mmHg';
     Maintitle:=title[idraw];
  end;
  if namevar[idraw]='volume' then
  begin
     ytitle:='ml';
     Maintitle:=title[idraw];
  end;
  if namevar[idraw]='barore' then
  begin
     ytitle:='AU';
     Maintitle:=title[idraw];
  end;
  if namevar[idraw]='ratbar' then
  begin
    ytitle:='AU';
    maintitle:=title[idraw];
  end;
  if namevar[idraw]='elasta' then
  begin
    ytitle:='mmHg/m2/ml';
    maintitle:=title[idraw];
  end;
  if namevar[idraw]='mepres' then
  begin
    ytitle:='mmHg';
    maintitle:=title[idraw];
  end;
  if namevar[idraw]='etsres' then
  begin
    ytitle:='(mmHg sec)/ml';
    maintitle:=title[idraw];
  end;
  if namevar[idraw]='flowfr' then
  begin
    ytitle:='ml/sec';
    maintitle:=title[idraw];
  end;
  if namevar[idraw]='cnshea' then
  begin
    ytitle:='/sec';
    maintitle:=title[idraw];
  end;
  if namevar[idraw]='cnsper' then
  begin
    ytitle:='Res';
    maintitle:=title[idraw];
  end;
```

```
      if namevar[idraw]='cnsmyo' then
      begin
        ytitle:='Myo';
        maintitle:=title[idraw];
      end;
      if namevar[idraw]='cnsvco' then
      begin
        ytitle:='VC';
        maintitle:=title[idraw];
      end;
      if namevar[idraw]='cnsvvu' then
      begin
        ytitle:='ml';
        maintitle:=title[idraw];
      end;
      if namevar[idraw]='inputh' then
      begin
         ytitle:='AU';
         Maintitle:=title[idraw];
      end;
      if namevar[idraw]='inputp' then
      begin
         ytitle:='AU';
         Maintitle:=title[idraw];
      end;
      if namevar[idraw]='inputm' then
      begin
         ytitle:='AU';
         Maintitle:=title[idraw];
      end;
      if namevar[idraw]='inputv' then
      begin
         ytitle:='AU';
         Maintitle:=title[idraw];
      end;
      if namevar[idraw]='svolum' then
      begin
        ytitle:='ml';
        maintitle:=title[idraw];
      end;
      if namevar[idraw]='coutpu' then
      begin
        ytitle:='ml/sec';
        maintitle:=title[idraw];
      end;


      if flag[idraw]='compar' then
         with MeasComp[idraw] do

DrawXYAxes(0,simulationtime,simulationtime/2,plotmin,plotmax,
                      (plotmax-plotmin)/2,Xtitle,Ytitle,Maintitle);
      if flag[idraw]='fcompa' then
         with MeasFComp[idraw] do

DrawXYAxes(0,simulationtime,simulationtime/2,plotmin,plotmax,
                      (plotmax-plotmin)/2,Xtitle,Ytitle,Maintitle);
      if flag[idraw]='others' then
```

```
         with MeasOther[idraw] do

DrawXYAxes(0,simulationtime,simulationtime/2,plotmin,plotmax,
                     (plotmax-plotmin)/2,Xtitle,Ytitle,Maintitle);
      {reset window}
      GetViewSettings(ViewPort);
      with ViewPort do
      begin
        w[idraw].x1:=x1;
        w[idraw].y1:=y1;
        w[idraw].x2:=x2;
        w[idraw].y2:=y2;
      end;

      if flag[idraw]='compar' then
         with MeasComp[idraw] do
         begin
           w[idraw].xscale:=(w[idraw].x2-w[idraw].x1)/simulationtime;
           w[idraw].yscale:=(w[idraw].y2-w[idraw].y1)/(plotmax-
plotmin);
         end;
      if flag[idraw]='fcompa' then
         with MeasFComp[idraw] do
         begin
           w[idraw].xscale:=(w[idraw].x2-w[idraw].x1)/simulationtime;
           w[idraw].yscale:=(w[idraw].y2-w[idraw].y1)/(plotmax-
plotmin);
         end;
      if flag[idraw]='others' then
         with MeasOther[idraw] do
         begin
           w[idraw].xscale:=(w[idraw].x2-w[idraw].x1)/simulationtime;
           w[idraw].yscale:=(w[idraw].y2-w[idraw].y1)/(plotmax-
plotmin);
         end;
    end; {of idraw loop}
  end;

end; {of MakeScreen}

{**************************************************************}

procedure SimulateNumFile1(var result3:text);
```

{first procedure for numerical simulation. Writes the definition of each abbreviation used on the top of appropriate column heading in the numerical simulation.}

```
var

  idraw: integer;

begin
  with Model,SimulGraph do
  begin
     { write the information above }
     For idraw:=1 to ndraw do
     begin
```

```
       writeln(result3,'  ',name[idraw],'  ',title[idraw]);
       writeln(result3);
      end;
     writeln(result3);
     writeln(result3,'        ***** During Cardiac Cycle *****
');
     writeln(result3);

    {write the first line}
     write(result3,'Time   ');
    For idraw:=1 to ndraw do
      write(result3,'     ',name[idraw]);
  end; {with Model,SimulGraph}

end; { of SimulateNumfile1 }

{******************************************************************}

procedure SimulateNumFile2(var result3:text);
```

{second procedure for numerical simulation. Fills in the appropriate column with the simulated value.}

```
var

  idraw: integer;

begin
  with Model,SimulGraph do
  begin
    For idraw:=1 to ndraw do
    begin
      if flag[idraw]='compar' then
      begin
        with MeasComp[idraw] do
        begin
          if namevar[idraw]='pressu' then
             write(result3,' ',P[icomp]:9:4,'  ');
          if namevar[idraw]='volume' then
             write(result3,' ',V[icomp]:9:4,'  ');
          if namevar[idraw]='barore' then
             write(result3,' ',B[icomp]:9:4,'  ');
          if namevar[idraw]='ratbar' then
             write(result3,' ',DB[icomp]:9:4,'  ');
          if namevar[idraw]='elasta' then
             write(result3,' ',EL[icomp]:9:4,'  ');
          if namevar[idraw]='mepres' then
             write(result3,' ',MP[icomp]:9:4,'  ');
          if namevar[idraw]='etsres' then
             write(result3,' ',ETSR[icomp]:9:4,'  ');
          if namevar[idraw]='svolum' then
             write(result3,' ',SV[icomp]:9:4,'  ');
          if namevar[idraw]='coutpu' then
             write(result3,' ',CO[icomp]:9:4,'  ');
        end;
      end;
      if flag[idraw]='fcompa' then
      begin
```

```
          with MeasFcomp[idraw] do
          begin
             if namevar[idraw]='flowfr' then
                write(result3,' ',F[ito,ifrom]:9:4,'  ');
          end;
        end;
        if flag[idraw]='others' then
        begin
          with MeasOther[idraw] do
          begin
             if namevar[idraw]='cnshea' then
                write(result3,' ',CnsH:9:4,'  ');
             if namevar[idraw]='cnsper' then
                write(result3,' ',CnsP:9:4,'  ');
             if namevar[idraw]='cnsmyo' then
                write(result3,' ',CnsM:9:4,'  ');
             if namevar[idraw]='cnsvco' then
                write(result3,' ',CnsVC:9:4,'  ');
             if namevar[idraw]='cnsvvu' then
                write(result3,' ',CnsVVu:9:4,'  ');
             if namevar[idraw]='inputh' then
                write(result3,' ',CnsHea:9:4,'  ');
             if namevar[idraw]='inputp' then
                write(result3,' ',CnsPer:9:4,'  ');
             if namevar[idraw]='inputm' then
                write(result3,' ',CnsMyo:9:4,'  ');
             if namevar[idraw]='inputv' then
                write(result3,' ',CnsVen:9:4,'  ');
          end;
        end;
      end;
  end;  {with Model,SimulGraph}

end; { of SimulateNumFile2 }

{***********************************************************}

procedure SimulGraphFile(var treal : real);
```

{procedure for graphical simulation. Draws the variables in the appropriate window with the title and the range already defined for the simulation.}

```
var

  idraw,xplot,yplot: integer;
  a: realplotarray;

begin
  with Model,SimulGraph do
  begin
    for idraw:=1 to 4 do
    begin
      with w[idraw] do
      begin
        SetViewPort(x1,y1,x2,y2,ClipOn);
        xplot:=round(treal*xscale);
      end;
      if flag[idraw]='compar' then
      begin
        with MeasComp[idraw] do
        begin
          if namevar[idraw]='pressu' then
          begin
             yplot:=round((plotmax-
P[MeasComp[idraw].icomp])*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='volume' then
          begin
             yplot:=round((plotmax-
V[MeasComp[idraw].icomp])*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='barore' then
          begin
             yplot:=round((plotmax-
B[MeasComp[idraw].icomp])*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='elasta' then
          begin
             yplot:=round((plotmax-
EL[MeasComp[idraw].icomp])*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='mepres' then
          begin
             yplot:=round((plotmax-
MP[MeasComp[idraw].icomp])*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='etsres' then
          begin
             yplot:=round((plotmax-
ETSR[MeasComp[idraw].icomp])*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='svolum' then
          begin
             yplot:=round((plotmax-
SV[MeasComp[idraw].icomp])*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='coutpu' then
          begin
             yplot:=round((plotmax-
CO[MeasComp[idraw].icomp])*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
        end;
      end;
      if flag[idraw]='fcompa' then
      begin
        with MeasFcomp[idraw] do
        begin
```

```
          if namevar[idraw]='flowfr' then
          begin
             yplot:=round((plotmax-
F[MeasFcomp[idraw].ito,MeasFcomp[idraw].ifrom])*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
        end;
      end;

      if flag[idraw]='others' then
      begin
        with MeasOther[idraw] do
        begin
          if namevar[idraw]='cnshea' then
          begin
             yplot:=round((plotmax-CnsH)*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='cnsper' then
          begin
             yplot:=round((plotmax-CnsP)*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='cnsmyo' then
          begin
             yplot:=round((plotmax-CnsM)*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='cnsvco' then
          begin
             yplot:=round((plotmax-CnsVC)*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='cnsvvu' then
          begin
             yplot:=round((plotmax-CnsVVu)*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='inputh' then
          begin
             yplot:=round((plotmax-CnsHea)*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='inputp' then
          begin
             yplot:=round((plotmax-CnsPer)*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='inputm' then
          begin
             yplot:=round((plotmax-CnsMyo)*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
          if namevar[idraw]='inputv' then
          begin
             yplot:=round((plotmax-CnsVen)*w[idraw].yscale);
             PutPixel(xplot,yplot,White);
          end;
        end;
      end;
    end;
    with w[5] do
      SetViewPort(x1,y1,x2,y2,ClipOn);

  end;
end;  {of SimuFile}

{****************************************************************}

procedure ValuesSaved(var Z:stateSpace3);
```

{saves the state variables for the next simulation. This procedure calls the above procedure count to count the state variables and then uses the values in the array Z and writes the values to a file called result5.res. When the simulation is run again by the user, the 'PULSE' package asks the user that the 'simulation variables have already been saved, do you want to start the simulation using these values ?' if the answer is yes then the 'PULSE' package overwrites the text file `statvar.res` with `result5.res`.}

```
var
  i,totalno: integer;
  treal:real;
  resultfile5: text;

begin
  assign(resultfile5,'\pulse\result5.res');
  rewrite(resultfile5);
  Count(totalno);
  writeln(resultfile5,totalno);
  for i:=1 to totalno do
    writeln(resultfile5,Z[i]:9:4);
  close(resultfile5);
end; {of ValuesSaved}

{****************************************************************}

procedure cycle1;
```

{calculates the end of cycle variables such as stroke volume, cardiac output, etc. for the first cycle.}

```
var
  cout:real;
  icomp,ifrvent:integer;

begin
     with Model,HeartCirc do
     begin
       for ifrvent:=1 to nfromvent do
       begin
         with FromVent[ifrvent] do
         begin
            SV[i]:=integF[j,i];
            CO[i]:=SV[i]/TH;
            IF left='true' then cout:=CO[i] else cout:=0;
```

```
          end;
        end;
        for icomp:=1 to ncomp do
        begin
           MP[icomp]:=integpres[icomp]/TH;
           IF cout<>0 then ETSR[icomp]:=MP[icomp]/cout else
ETSR[icomp]:=0.0;
        end;
      end;
end;

{*******************************************************************
}

procedure cycle2;
```

{calculates the end of cycle variables such as stroke volume, cardiac output, etc. for the consecutive cycles not including the first cycle.}

```
var
  cout:real;
  icomp,ifrvent:integer;

begin
     with Model,HeartCirc do
     begin
       for ifrvent:=1 to nfromvent do
       begin
         with FromVent[ifrvent] do
         begin
            SV[i]:=integF[j,i];
            CO[i]:=SV[i]/TH;
            integF[j,i]:=0;
            IF left='true' then cout:=CO[i] else cout:=0;
         end;
       end;
       for icomp:=1 to ncomp do
       begin
        MP[icomp]:=integpres[icomp]/TH;
        integpres[icomp]:=0.0;
        IF cout<>0 then ETSR[icomp]:=MP[icomp]/cout else
ETSR[icomp]:=0.0;
       end;
     end;
end;

{******************************************************************}

procedure SimulateGraphModel;
```

{simulates and draw graphical & numerical representation. Assigns a result3.res file for the numerical simulations and calls the above appropriate procedures. The procedure sets the cardiac time to zero and reads the simulation time. Then initialises the heart time and model equations by calling UpdateHeartTimes and ModelEquations. Then starts the simulation until the simulation time is reached. The numerical simulation is written every 0.05

seconds, and the end of cycle is reached when the cardiac time is equal to heart period, then the cardiac time is set to zero again.}

```
var

  treal,THlast,tres,tcheck,t2,outloop: real;
  result3: text;
  Z:StateSpace3;

begin
  assign(result3,'\pulse\result3.res');
  rewrite(result3);
  tcheck:=0.05;
  tres:=0.0;
  with Model,SimulGraph do
  begin
    tcardiac:=0;
    treal:=0;
    outloop:=0;
    UpdateHeartTimes;
    ModelEquations;
      repeat
      THlast:=TH;
      if treal=0 then t2:=0
      else
      begin
        t2:=simulationtime;
        cycle2;
      end;
      CnsH:=60/TH;
      SimulateNumFile1(result3);
      writeln(result3);
      repeat
        if (tcardiac>=tres) then
        begin
          write(result3,tcardiac:7:4,'    ');
          tres:=tres+tcheck;
          SimulateNumFile2(result3);
          writeln(result3);
        end;
        if t2=0 then cycle1;
        AdjustStep(treal,Z);
        SimulGraphFile(treal);
     until (tcardiac>=THlast) or (KeyPressed);
     if tcardiac>=THlast then
     begin
       writeln(result3);
       writeln(result3,'        *****  End Cardiac Cycle  *****
');
     end;
     UpdateHeartTimes;
     tcardiac:=0;
     tcheck:=0.05;
     tres:=0.0;
     if outloop<0.8 then outloop:=outloop+0.2
     else outloop:=0
         until (treal+outloop>=simulationtime) or (KeyPressed);
  end;
  if KeyPressed then
  begin
    writeln(result3);
    writeln(result3,'         *****     INTERRUPTED        *****
');
  end;
    close(result3);
    ValuesSaved(Z);

end; {of SimulateGraphModel}

{*************************************************************}

procedure Final;
```

{main procedure. Calls the startsgraphic procedure followed by the main above procedures. Once the simulation is over it waits for the users input (any key could be pressed). This is to ensure that if the user is not attending the terminal the screen will not be cleared once the simulation is over. Finally the procedure stops the graphics once the input is entered.}

```
begin
  StartGraphic;
  MakeScreen;
  SimulateGraphModel;
  readln;
  CloseGraph;
end; {of Final}

{*************************************************************}

begin
```

{main body of the program}

```
  ReadNewState;
  Initial;
  Final;

end.
```

APPENDIX 4

LIST OF DATABASE FILES FOR THE 19-SEGMENT MODEL

The list of database files for the 19-segment model mentioned in Chapter 5 are given below.

comp_ code	index	comp_ type	region	name	anat_ det
A1	1	artery	thorax	ascending aorta	first part of gt. artery
A2	2	artery	thorax	aortic arch	Arch of gt. artery
A3	3	artery	thorax	thoracic aorta	Descending part of gt. artery
A4	4	artery	abdomen	intestinal arts.	Intestinal arteries lumped
A5	5	artery	abdomen	abdominal arteries	Abdominal arteries lumped
A6	6	artery	lower limb	leg arteries	Leg arteries lumped
A7	7	artery	upper limb	head/arms arteries	Upper limb arteries lumped
A8	8	artery	lung	pulmonary arteries	Lungs arteries lumped
V1	9	vein	upper limb	head/arms veins	Upper limb veins lumped
V2	10	vein	upper limb	superior vena cava	Upper part of great vein
V3	11	vein	abdomen	intestinal veins	Intestinal veins lumped
V4	12	vein	abdomen	abdominal veins	Abdominal veins lumped
V5	13	vein	lower limb	leg veins	Leg veins lumped
V6	14	vein	lower limb	inferior vena cava	Lower part of great vein
V7	15	vein	lung	pulmonary veins	Lungs veins lumped
AT1	18	atrium	heart	right atrium	Top right chamber of heart
AT2	19	atrium	heart	left atrium	Top left chamber of heart
VE1	16	ventricle	heart	right ventricle	Bottom right chamber of heart
VE2	17	ventricle	heart	left ventricle	Bottom left chamber of heart

Figure A4.1 **compartment definition file (comp_d) listing**

from	to	link	descriptn
A1	A2	A1A2	arterial connection
A1	AT1	A1AT1	heart connection
A2	A7	A2A7	arterial connection
A2	A3	A2A3	arterial connection
A3	AT1	A3AT1	heart connection
A3	A4	A3A4	arterial connection
A3	A5	A3A5	arterial connection
A4	V3	A4V3	arteriovenous connection
A5	A6	A5A6	arterial connection
A5	V4	A5V4	arteriovenous connection
A6	V5	A6V5	arteriovenous connection
A7	V1	A7V1	arteriovenous connection
A8	V7	A8V7	arteriovenous connection
V1	V2	V1V2	venous connection
V2	AT1	V2AT1	venous connection
V3	V6	V3V6	venous connection
V4	V6	V4V6	venous connection
V5	V4	V5V4	venous connection
V6	AT1	V6AT1	heart connection
AT1	VE1	AT1VE1	heart connection
VE1	A8	VE1A8	heart connection
AT2	VE2	AT2VE2	heart connection
V7	AT2	V7AT2	heart connection
VE2	A1	VE2A1	hear connection

Figure A4.2 **compartment flow definition file (`f_comp_d`) listing**

comp_code	i	ci	vui	ki	vio	pregion
A1	1	0.28	53.0	0.04	79.8614	-4.0
A2	2	0.29	61.0	0.04	88.8197	-4.0
A3	3	0.29	59.0	0.04	86.7118	-4.0
A4	4	0.06	17.0	0.04	22.2541	4.0
A5	5	0.21	58.0	0.04	76.1043	4.0
A6	6	0.12	63.0	0.04	73.6318	0.0
A7	7	0.33	114.0	0.04	144.6275	0.0
A8	8	4.30	50.0	0.00	109.1511	-4.0

Figure A4.3 **aterial parameter file (`artery_p`) listing**

comp_code	i	cio	vui	alphai	vio	pregion	neural
V1	9	9.4	552.0	20	521.5427	0.0	true
V2	10	8.3	488.0	20	532.9667	-4.0	fals
V3	11	10.6	607.0	20	568.3799	4.0	true
V4	12	5.1	305.0	20	276.1907	4.0	true
V5	13	4.8	257.0	20	257.8565	0.0	true
V6	14	8.3	488.0	20	533.0754	-4.0	fals
V7	15	8.4	460.0	20	525.6252	-4.0	fals

Figure A4.4 **venous parameter file (`vein_p`) listing**

```
comp_code i     vui      ais    aid    lambda1  lambda2  vio        pregion

AT1        18    30.0    0.15  0.050   0.10     0.09     130.1922 -4.0
AT2        19    30.0    0.28  0.120   0.10     0.09      93.1334 -4.0
```

Figure A4.5 **atrial parameter file (`atrium_p`) listing**

```
comp_code i  vui        ais    aid      lambda1  lambda2  lambda3  vio      pregion

VE1        16  0.0      0.30  0.046    0.04     0.16     0.20    106.9795 -4.0
VE2        17  0.0      1.50  0.067    0.04     0.16     0.20    111.7554 -4.0
```

Figure A4.6 **ventricular parameter file (`ventri_p`) listing**

```
link       j  i     rji       lenji  anglji lji        fji      pcc neural

A1A2       2  1  0.000031  0.0     0.0     0.00043   6.5532   0.0 fals
A1AT1     18  1 12.000000  0.0     0.0     0.00000   0.0000   0.0 fals
A2A7       7  2  0.047000 19.5    90.0     0.01400   2.5506   0.0 fals
A2A3       3  2  0.000090 10.0   -90.0     0.00380  18.4178   0.0 fals
A3AT1     18  3 12.000000 10.0    90.0     0.00000   0.0000   0.0 true
A3A4       4  3  0.001400  8.0   -90.0     0.00270  32.0507   0.0 fals
A3A5       5  3  0.012000 16.0   -90.0     0.01400  -6.5786   0.0 fals
A4V3      11  4  2.300000  0.0     0.0     0.00000   0.0000   0.0 true
A5A6       6  5  0.180000 48.0   -90.0     0.03100   0.0093   0.0 fals
A5V4      12  5 57.000000  0.0     0.0     0.00000   0.0000   0.0 true
A6V5      13  6 15.000000  0.0     0.0     0.00000   0.0000   0.0 true
A7V1       9  7  6.000000  0.0     0.0     0.00000   0.0000   0.0 fals
A8V7      15  8  0.110000  0.0     0.0     0.00000   0.0000   7.0 fals
```

Figure A4.7 **arterial flow parameter file (`f_art_p`) listing**

```
link       j  i     rji     lenji  anglji  vui       betai    fji     valve

V1V2      10  9    0.226    18.0   -90.0  552.0     0.667    0.0000 fals
V2AT1     18 10    0.060     1.5   -90.0  488.0     0.100    0.0000 fals
V3V6      14 11    0.166     8.0    90.0  607.0     1.000    0.0000 fals
V4V6      14 12    0.595    16.0    90.0  305.0     1.000    0.0000 fals
V5V4      12 13    0.300    48.0    90.0  257.0     0.000    0.0000 true
V6AT1     18 14    0.015    10.0    90.0  488.0     0.100    0.0000 fals
V7AT2     19 15    0.007     0.0     0.0  460.0     0.100    0.0000 fals
```

Figure A4.8 **venous flow parameter file (`f_vein_p`) listing**

link	j	i	rji	fji
AT1VE1	16	18	0.003	0.0000
AT2VE2	17	19	0.003	0.0000

Figure A4.9 **atrial flow parameter file (`f_atri_p`) listing**

link	j	i	rji	fji	lji	aji
VE1A8	8	16	0.003	0.0000	0.00018	1.539
VE2A1	1	17	0.003	0.0000	0.00022	1.539

Figure A4.10 **ventricular flow parameter file (`f_vent_p`) listing**

baro_code	comp_code	descriptn
B1	A2	baroreceptor of aortic arch.
B2	A7	baroreceptor of upper arms.

Figure A4.11 **baroreceptor definition (`baro_d`) listing**

baro_ code	i	x1	x2	slow_ time	fast_ time	gain1	gain2	thrsh_ pres
B1	2	102.1393	2.3577	0.8	0.1	1.0	1.0	40.0
B2	7	97.7083	2.8019	0.8	0.1	1.0	1.0	40.0

Figure A4.12 **baroreceptor parameter (`baro_p`) listing**

baro_code	cns_in_cod	descriptn
B1	C1	cns input of aortic arch
B2	C2	cns input of upper arms

Figure A4.13 **cns input definition (`cns_in_d`) listing**

cns_in_cod	control	i	contributn
C1	heart	2	0.3
C2	heart	7	0.7
C1	periph	2	0.3
C2	periph	7	0.7
C1	myocar	2	0.3
C2	myocar	7	0.7
C1	venous	2	0.3
C2	venous	7	0.7

Figure A4.14 **cns input parameter (cns_in_p) listing**

descriptn
heart rate control.

Figure A4.15 **controller definition (e.g. heart_d) listing**

bhrt	x3	x5	x6	t1	t2	fast_ time	slow_ time	thmax	thmin	gain1	gain2
80.0	52.3483	72.3248	72.1437	1.5	4.5	1.0	2.0	2.0	0.3	1.0	0.006

Figure A4.16 **heart rate controller parameter (heart_p) listing**

bprt	x2	x3	fast_time	slow_time	prmax	prmin	gain1
80.0	1.004934	1.003662	4.0	20.0	1.4	0. 6	0.75

Figure A4.17 **peripheral resustance controller parameter (periph_p) listing**

bmct	x2	t	mcmax	mcmin
80.0	1.003888	10.0	1.4	0.6

Figure A4.18 **myocardial contractility controller parameter (contrc_p) listing**

bvvt	x2	t	vtmax	vtmin	cgain	vugain
80.0	1.154200	14.0	1.6	0.7	1.0	1.0

Figure A4.19 **venous tone controller parameter (periph_p) listing**

APPENDIX 5

LIST OF DATABASE FILES FOR THE DRUG EXTENSION

The list of database files for the drug extension discussed in Chapter 8 are given below.

drug_ code	start_ time	stop_ time	comp_ code	input_ type	dose	drug_ units
METHOX	0.0	0.0	UV	injection	70	microgrm/ml
ISOPREN	0.0	0.0	UV	injection	70	microgrm/ml
NORADREN	0.0	0.0	UV	injection	70	microgrm/ml
METHOX	0.0	0.0	CV	injection	70	microgrm/ml
ISOPREN	0.0	0.0	CV	injection	70	microgrm/ml
NORADREN	0.0	0.0	CV	injection	70	microgrm/ml

Figure A5.1 **Drug input (`drug_input`) for the 19-segment model**

drug_code	name	descriptn
methox	methoxamine	methoxamine is a sympathomimetic amine related to adrenaline and its main action is one of vasoconstriction by direct action on alpha receptors in the arteriolar smooth muscle (Godman and Gilman , 1970).
isopren	isoprenaline	isoprenaline is an isopropyl derivative of noradrenaline which is particular active on the beta receptors so that its primary local actions are inhibition of smooth muscle and increase in the force of contraction and heart rate (Green, 1972).
noradren	noradrenaline	noradrenaline is the chemical transmitter of the sympathetic nervous system and is also a hormone released by the adrenal medulla. Its effect is to constrict the arteries by direct action and also to stimulate the heart (Green 1972).

Figure A5.2 **Drug definition (`drug_d`) for the 19-segment model**

drug_ e_code	drug_e code	from	to	mod_par	dir_of_eff	descriptn
m_avr	methox	uv	svc	arterio_venous resistance	.T.	Methoxine is injected into head and arms veins. The total systemic resistance rises indicating that the peripheral vasculature is constricted.
m_mc	methox	uv	svc	myocardial contractility	.T.	The mean systolic and diastolic pressures rise, there is reflex slowing of the heart and the cardiac output and stoke volume fall.
i_avr	isopren	uv	svc	arterio_venous resistance	.F.	estimated total systemic resistance (ETSR) falls indicating peripheral vasodilation.
i_hr	isopren	uv	svc	heart rate	.F.	the heart rate rises, indicating tachcardia.
i_mc	isopren	uv	svc	myocardial contractility	.T.	positive inotropic action of isoprenaline despite the decreasing blood pressure.
n_hr	noradren	uv	svc	heart rate	.F.	hear rate rises with the injection, indicating tachcardia.
n_avr	noradren	uv	svc	arterio-venous resistance	.T.	Estimated total systemic resistance (ETSR) increases indocating vasoconstriction.
n_mc	noradren	uv	svc	myocardial contractility	.T.	The systolic and diastolic pressures rise, and cardiac output falls which causes positive inotropy.

Figure A5.3 **Drug effect definition (drug_e_d) for the 19-segment model**

drug_code	comp_code	koi
METHOX	UV	0.03333
METHOX	CV	0.03333
ISOPREN	UV	0.03333
ISOPREN	CV	0.03333
NORADREN	UV	0.03333
NORADREN	CV	0.03333

Figure A5.4 **Drug parameter (`drug_p`) for the 19-segment model**

drug_e_code	sensitive
M_AVR	400.0
M_MC	50.0
I_AVR	400.0
I_HR	50.0
I_MC	50.0
N_HR	50.0
N_AVR	400.0
N_MC	50.0

Figure A5.5 **Drug effect parameter (`drug_e_p`) for the 19-segment model**