COMPUTER MODELLING FOR ELECTRO-PNEUMATIC

ROBOT CONTROL.

V. Toulis (MSc.)

A Thesis submitted for the degree of Doctor of Philosophy
in the Faculty of Engineering, Department of Mechanical
Engineering of the University of Surrey.

December 1980
ABSTRACT.

Initial work on path generation routines for the University of Surrey Manipulator demonstrated the need for an improved control strategy. In view of the difficulties encountered in obtaining such a strategy by analytical means it was decided to investigate the feasibility of generating switching profiles for individual degrees of freedom of the manipulator by a suitable computer simulation. A computer implementation of an exponential constant volume pressure transient formed the basis for a simulation of one degree of freedom of the manipulator that requires a minimum of experimental data for its operation. Using this approach a simulation of the trajectory of a single axis of the manipulator under the control of a simple switch-over sequence was carried out in forward and reverse time. It was argued that such a simulator package is suitable for the generation of switching profiles for the axis and a number of suggestions were made concerning the use of such simulators as the basis for the control of the arm.
AKNOWLEDGEMENTS.

I would like to express my gratitude to my supervisor Dr. P.J. Drazan for his help, guidance and support throughout my research. I would also like to thank Mr. M.F. Jeffery for his help with microprocessor software, Mr. R. Sneddon for his assistance with computing and Mr. D. Bailey for his help in the early experimental work.

V. Toulis.
TABLE OF CONTENTS.

----------------------------------

CHAPTER 1. INTRODUCTION ............ - - - - - - - - - 1

1.1 GENERAL INTRODUCTION ............ - - - - - - - - - 1

1.2 ANALYSES OF MANIPULATOR ARM

KINEMATICS AND DYNAMICS ............. - - - - - - - - - 2

1.2.1 Kinematic analyses of
manipulator arms. .................. - - - - - - - - - 2

1.2.2 Dynamic analyses of
manipulator arms. .................. - - - - - - - - - 5

1.2.3 Methods for reducing the computing
effort involved in the derivation and solution
of the equations of motion of manipulators. .... 7

1.3 THE UNIVERSITY OF SURREY MANIPULATOR. .... 10

1.3.1 Brief description of the manipulator
and its operating system. ............ - - - - - - - - - 10

1.3.2 Difficulties encountered in using
existing methods of kinematic and dynamic
analyses with the University of Surrey
manipulator. ........................ - - - - - - - - - 13

1.3.3 Developments leading to the present
research. .......................... - - - - - - - - - 15

CHAPTER 2. THE USE OF CO-ORDINATE GENERATION

BY INTERPOLATION ROUTINES FOR THE CONTROL

OF THE MANIPULATOR ARM. .......... - - - - - - - - - 19

2.1 INTRODUCTION. ................. - - - - - - - - - - 19

2.2 THE CO-ORDINATE GENERATION ROUTINES. .... 20

2.2.1 Linear co-ordinate generation. .... - - - - - - 20

2.2.2 Programming linear co-ordinate

    generation routines in assembly language. ..... 28
2.2.3 Incorporation of the co-ordinate
generation routine into SUMOS. - - - - - - - - 40
2.3 MANIPULATOR PERFORMANCE WITH THE
SUMOS X1 PACKAGE. - - - - - - - - - - 41
2.4 USE OF DIFFERENT CONTROL ALGORITHMS. - - - 45
2.5 CONTROL OF MANIPULATOR SPEED. - - - - - - 50
2.6 THE USE OF ALTERNATIVE INTERPOLATION
ROUTINES - - - - - - - - - - - - - - - - - - - - - 54
2.7 CONVERSION OF INTERPOLATION ROUTINE
CO-ORDINATES TO INTERNAL MANIPULATOR
CO-ORDINATES. - - - - - - - - - - - - - - - - - - 56
2.8 LIMITATIONS OF THE SUMOS CONTROL
STRATEGY - - - - - - - - - - - - - - - - - - - - - - 60

CHAPTER 3. SYSTEM MODELLING. EXPONENTIAL
REPRESENTATION OF THE PRESSURE TRANSIENT. - - 68
3.1 INTRODUCTION. - - - - - - - - - - - - - - - - 68
3.2 EXPERIMENTAL DERIVATION OF THE
VALVE FLOW CHARACTERISTIC. - - - - - - - - 70
3.2.1 Experimental setup and relevant
considerations. - - - - - - - - - - - - - - - - 70
3.2.2 Derivation of the flow characteristic. - 73
3.2.3 Computer simulation of the
experimental arm. - - - - - - - - - - - - - - - - 75
3.3 PIECE-WISE LINEARISATION OF THE FLOW
CHARACTERISTIC. - - - - - - - - - - - - - - - - 77
3.4 EXPONENTIAL REPRESENTATION OF THE
PRESSURE TRANSIENT. - - - - - - - - - - - - 80
3.5 COMPUTER SIMULATION OF THE EXPONENTIAL
PRESSURE TRANSIENT. - - - - - - - - - - - - 84
3.6 PERFORMANCE OF THE XPPRTR PACKAGE. - - - 89
3.6.1 Determination of the reference
simulation time constant $\tau$.  

89

3.6.2 Factors affecting the value of the
simulation time constant.

91

3.6.3 Evaluation of the pressure transients.

96

CHAPTER 4. SYSTEM MODELLING, COMPUTER
SIMULATION OF THE ARM.

99

4.1 INTRODUCTION.

99

4.2 COMPUTER SIMULATION OF THE ARM FOR
SMALL DISPLACEMENTS.

101

4.2.1 Verification of the first order lag
transfer function.

102

4.2.2 Modification of the simulation time
constant for changing volumes.

103

4.2.3 Evaluation of the small displacement
simulation of the arm.

104

4.3 COMPUTER SIMULATION OF THE ARM FOR
LARGE DISPLACEMENTS.

109

4.3.1 Representation of the valve-motor
combination for large displacements.

109

4.3.2 Computer generation of flow
characteristic data.

112

4.3.3 Performance of the simulator packages
for large displacements.

114

4.4 IMPLICATIONS OF SUCCESSFUL LARGE
DISPLACEMENT SIMULATION.

119

CHAPTER 5, GENERATION OF SWITCHING
PROFILES BY COMPUTER SIMULATION IN
FORWARD AND REVERSE TIME.

121

5.1 INTRODUCTION.

121
5.2 SIMPLE SWITCH-OVER SIMULATION
IN FORWARD TIME
5.2.1 Selection of the mode of switch-over.
5.2.2 Simulation of arm performance after
simple switch-over.
5.2.3 Examination of an alternative
switching sequence.
5.2.4 Limitations of the simple switch-over
simulation for control purposes.
5.3 SIMPLE SWITCH-OVER SIMULATION IN
REVERSE TIME
5.3.1 Reasons for carrying out a simulation
in reverse time.
5.3.2 Method of carrying out a simple
switch-over simulation in reverse time.
5.3.3 Difficulties encountered in carrying
out a simple switch-over simulation in
reverse time.
5.4 EVALUATION OF THE NEED FOR STORAGE OF
SWITCH-OVER OR INITIALISATION VALUES FOR
THE GENERATION OF A SWITCHING PROFILE.
CHAPTER 6. THE USE OF COMPUTER-GENERATED
SWITCHING PROFILES FOR THE CONTROL OF THE
ARM.
6.1 INTRODUCTION.
6.2 A STRATEGY FOR THE DETERMINATION OF THE
SWITCH-OVER POINT WITH THE AID OF THE REVERSE
TIME SIMULATOR.
6.2.1 Basic procedure.
6.2.2 Determination of the switch-over criterion. 

6.2.3 Evaluation of the mode of determination of the switch-over point. 

6.3 TENTATIVE EVALUATION OF CONTROL STRATEGIES THAT UTILISE A SWITCHING PROFILE GENERATED BY THE FORWARD TIME SIMULATOR. 

6.4 SUGGESTIONS FOR AN INTEGRATED CONTROL SCHEME BASED ON THE GENERATION OF SWITCHING PROFILES BY COMPUTER SIMULATION. 

6.5 SUGGESTIONS FOR THE IMPROVEMENT AND EXPANSION OF THE BASIC CONTROL SCHEME. 
6.5.1 Coping with limitations in the accuracy of computer generated switching profiles. 
6.5.2 Expanding the basic control scheme for use with the manipulator. 
6.5.3 Suggestions for the incorporation of path generation routines in the manipulator control package. 

6.6 CONCLUSIONS. 

LIST OF REFERENCES. 

APPENDIX 1. Derivation of equation (10) (Chapter 3). 

APPENDIX 2. Listings of XPPRTR. 

APPENDIX 3. Listings of MANIPR.
APPENDIX 4. Listing of routines for the updating of variables from MANESPR and MANTDPR.

APPENDIX 5. Listing of relevant routines of the forward and reverse time simulators VMSTDPR and REVVMS.

APPENDIX 6. Determination of the dynamic friction of the arm.

APPENDIX 7. Tables of switch-over positions and initialisation pressure for forward and reverse time simulation.

APPENDIX 8. Listing of REVCON.
**NOMENCLATURE**

- **M** [kg] mass of air
- **P** [bar] pressure (gauge, unless otherwise indicated)
- **X** [rad] motor displacement
- **T** [K] temperature
- **V** [m³] volume
- **t** [sec] time
- **R** [N·m] gas constant
- **R** [N·m] gradient of the flow characteristic
- **S** Laplace operator
- **τ** [sec] time constant
- **T** [N·m] torque
- **I** [kg·m²] moment of inertia
- **r** [m] manipulator section length
- **θ_{1,2,3}** [rad] manipulator joint angles
Photograph of the University of Surrey Manipulator
CHAPTER 1.

INTRODUCTION.

1.1 GENERAL INTRODUCTION.

Industrial robots have been used in increasing numbers in a variety of industrial applications in the past few years. Such applications include assembly line manipulation, use of robots in hazardous environments and use of robots in specially configured work stations. (Some descriptions of robot applications can be found in Refs. [1]-[15].) This increased use of industrial robots has been, to a large extent, brought about by the availability of relatively cheap computing capability (which is necessary for the control of a robot) that was brought about by the advent of microprocessors and by the need in modern industry to increase productivity.

The need to increase the capability of industrial robots to carry out complex manipulation tasks has resulted in comprehensive analyses that aim at describing the kinematics and controlling the dynamics of robot arms. The various approaches used in these analyses and attempts at reducing the considerable amount of computing that is necessary for carrying them out are outlined in the first section of this chapter.

In the following section a description of the University of Surrey manipulator arm is given and early attempts aimed at modelling the components of the arm and controlling its trajectory are examined in the context of the work carried out by other researchers. It is pointed out that the kinematic and dynamic analyses carried out by other researchers are largely
inapplicable to the Surrey manipulator because of the
non-linear nature of its drive components (pneumatic motors).
The stage is thus set for the description of research aimed at
specifying trajectories for the manipulator, obtaining an
improved modelling of its components, and deriving from this
modelling a suitable strategy for the control of the arm.

1.2 ANALYSES OF MANIPULATOR ARM KINEMATICS AND DYNAMICS.

1.2.1 Kinematic analyses of manipulator arms.

Early work in this direction dealt with sophisticated
manipulation applications i.e. manipulation in space or
manipulation carried out by prosthetic arms. As a result the
analyses produced were comprehensive and dealt with manipulator
arms that had a relatively large number of degrees of freedom
and considerable motional redundancy (Ref. [16]). One
objective of such analyses is the ability to specify the motion
of the arm in a system of co-ordinates which are convenient and
visible to an operator (i.e. Cartesian co-ordinates). This
objective can be accomplished by co-ordinated rate control, a
strategy that commands several joints of the arm to move
simultaneously at time-varying rates. The mathematical
representation of such a strategy (which is essentially a
conversion of the velocities of the manipulator end point to
velocities of the individual joints of the arm) is given in
Refs. [17], [18], [19].

A more generalised representation of manipulator motion is
given in Ref. [20], where relationships between the position
and orientation of the manipulator endpoint and the internal robot co-ordinates (joint angles) are expressed in group theory notation. This work is extended to cover the position and orientation of the gripper in Ref. [21]. Both the above approaches deal with the problem of singularities which occur in the solution of the equations of motion of the manipulators under consideration. These singularities are due to the high motional redundancy possessed by these arms (this is due to their large number of degrees of freedom) which results in a number of possible configurations of the arm for a given position and orientation of its endpoint in three-dimensional space.

In Ref. [22] it is pointed out that a high motional redundancy is generally possessed by living organisms and speculation as to the need for manipulators to be have an anthropomorphic configuration is examined. This evaluation is carried further in Ref. [23] where an analysis of the movements of the extremities of vertebrae is used to demonstrate that some of the possible movements of these extremities are not used, presumably because of their inability to satisfy natural criteria (i.e., economy of effort). This analysis is extended to manipulators with the aid of the "Interrelated Functions" concept (an Interrelated Function is a way of describing the extent of the role of a given actuator in the total motion of manipulator joints). Using this concept in conjunction with a minimum energy criterion provides the basis for the synthesis of the appropriate configuration of an anthropomorphic arm. A rather different viewpoint to the question of motional redundancy is taken in Ref. [24]. Although
it is acknowledged that highly evolved biological organisms possess a high degree of motional redundancy it is stressed that the biological criteria for the selection of certain configurations are not known and that criteria for industrial manipulators should be evolved independently and not by comparison to various life forms. Criteria relating to a minimum volume of motion, maximum speed of motion and a uniform distribution of the volume of motion over the degrees of freedom (this being a compromise criterion) are discussed. The concept of configuration space is also introduced in this paper, configuration space being an n-dimensional space (where n is the number of degrees of freedom of the arm) in which each point represents a particular configuration of the arm.

The procedure of deriving arm (or motion) configurations in order to comply with certain criteria is reversed in Ref. [25] where an evaluation is carried out of the relationship between the structure of a manipulator and the types of motion it can perform. A similar evaluation linking the number of manipulator joints to the complexity of its kinematic equations is carried out in Ref. [26]. In both cases, generalised methods for the solution of the manipulators' kinematic equations are given.

From the above it can be seen that the question of the solution of the equations of motion of a manipulator (i.e. the conversion of motion in a "visible" co-ordinate system to motion of the individual manipulator joints—or axes—) has been examined in some detail. Comparatively little work has been done, however, on the description of various types of path that are suitable for manipulation operations (with the exception of
straight line paths in various directions). One reason for this is the ability to generate such paths from standard mathematical functions. A more relevant reason is the tendency in current manipulator use to "teach" a working sequence to the manipulator rather than analyse the motions of this sequence and supply them to the manipulator in mathematical (or command) form.

1.2.2 Dynamic analyses of manipulator arms.

The work outlined above deals with an analysis of manipulator arms that is purely kinematic. This analysis must be extended to cover the dynamic properties of arms possessing inertia if it is to be used for the synthesis of control strategies that would enable manipulator arms to perform co-ordinated motion.

The principal obstacle that is encountered in the study of dynamic control of arms that possess a large number of degrees of freedom is the complexity of the equations of motion. A procedure for the computer generation and computer simplification of such equations is described in Reference [27].

In order to overcome this complexity a method which obtains the equations of motion by considering a linearised form of Lagrange’s equations (for small motions about a nominal position and zero nominal velocity) was developed. (Ref. [17]) The manner in which this method is used for the synthesis of a strategy that enables the co-ordinated motion of a three-Joint
Another approach using Lagrange's equations (though not in a linearised form) to derive the equations of a robot's dynamics is described in Refs. [30],[31]. Here, non-linear control (state-space) theory is used to arrive at explicit non-linear control laws which provide an overall system behaviour where all the outputs are completely de-coupled and the dynamics can be chosen arbitrarily. It is claimed that such control laws, in contrast to linear control and de-coupling concepts, do not need re-adjustment for different working conditions and hence provide a considerable saving in the computing effort needed for their derivation and implementation.

In Reference [32] it is argued that a straightforward application of Lagrange's equations to obtain the mathematical models of the dynamics of manipulators (which are considered as open-loop kinematic chains) is not suitable for digital computers because of the number of differentiations involved. As an alternative a kinetostatic approach using a Generalised Inertia Matrix (a matrix which depends at any time upon the arbitrary configuration of the manipulator) leads to closed form solution of the mathematical model without having to actually perform the differentiations required by Lagrange's equations. It is again argued that this approach results in a considerable saving in the amount of computing effort and storage required for the automatic generation of mathematical models of manipulators.
An approach which does not use Lagrange's equations for the derivation of a mathematical model for manipulator dynamics is described in Refs. [33]-[36]. Equations for the driving forces (or torques) exerted on the manipulator joints are derived instead from the kinematic representation of the trajectory of the arm by multiplying the relevant acceleration and inertia terms for each joint (D'Alembert method). As the resulting equations are quite complex various methods are used to reduce the computing effort required for their solution. These methods and similar strategies used by other researchers to the same end are outlined below.

1.2.3 Methods for reducing the computing effort involved in the derivation and solution of the equations of motion of manipulators.

The complexity of the equations of motion of industrial manipulators becomes significant when one considers that the control strategies which are obtained from their solution have to be implemented on-line. (A general account of the problems arising from this is given in Ref. [37].) If solutions for these equations are to be obtained in the short time intervals which one associates with on-line control extremely fast (and therefore expensive) computers have to be used. In order to reduce the computation overhead associated with the on-line control of manipulators we can (a) attempt to simplify their equations of motion and (b) attempt to carry out as much of the computation as possible off-line.

Both approaches are used in Refs. [33]-[36].
Equations of motion are simplified by considering specific sequences of motion of the manipulator endpoint ('synergies') and these synergies are used in turn to calculate (off-line) a series of torques (for the three degrees of freedom that implement the positioning of the manipulator endpoint) that will carry out the synergy in question. Off-line calculations are also used to determine constants for simple feedback-control routines that control the orientation of a gripper attached to the manipulator and the fine positioning of the arm (using force-feedback).

Although the off-line calculation of control signal sequences for a manipulator reduces the amount of computing that has to be carried out in real time for on-line control it increases the amount of computer memory required for the storage of the pre-computed control sequences. The relative advantages and disadvantages of on-line derivation and solution of manipulator equations as opposed to off-line computation and storage of the relevant information in look-up tables are discussed in Ref. [38]. A procedure for organizing look-up tables obtained from the solution of the equations of motion of a manipulator arm using the Configuration Space concept is described in Ref. [39].

Examples of the generation of control sequences by on-line and off-line procedures for specific industrial robots are given in Refs. [40],[41],[42]. In the case of the Cincinnati Milacron robot the kinematic co-ordinate transformations and dynamic calculations necessary for the determination of a control sequence are carried out on-line by using efficient
algorithms based on straight line motion only and taking into account certain mechanical constraints of the robot. The resident Computer Path Control System also includes features that enable an operator to command linear manipulator motion during a TEACH operation and the ability to adjust manipulator motion to cater for changes in speed of a conveyor belt. (Refs. [40], [41]) In the case of the Unimate welding robot a (detachable) Teach Assist Computer is used to carry out co-ordinate conversions on-line (but at reduced speeds) during a TEACH operation in order to enable the robot to move in directions specified by the operator. The resulting internal co-ordinates and control signals are stored in the memory of a smaller computer for use in the real-time operation of the robot. (Ref. [42])

A slightly different solution to the necessity of reducing the time taken by on-line computing is described in Ref. [43]. In this case, in addition to simplified kinematic equations (corresponding to a specific manipulator configuration), dedicated computer hardware (multiplication and division boards and a sine table) is used. The use of an interpolation procedure which relates the density of position co-ordinate interpolation with speed instructions an uses the resulting relationships to simplify the kinematic equations of the robot is of some interest. Further mention of interpolation procedures is made in Reference [44] where a scheme using the same number of interpolation intervals is suggested as a means of obtaining smooth manipulator motion between two points without carrying out a laborious kinematic analysis. The difficulty of commanding prescribed types of path between two points when using such
procedures and in particular the difficulty encountered in programming a manipulator to follow a straight-line path without kinematic analysis are detailed in Ref. [45] and some compensation procedures are suggested. This type of problem is examined in more detail in Reference [46]. It is pointed out that errors will occur in attempting to guide the endpoint of non-orthogonal manipulators (i.e., manipulators of a cylindrical or polar configuration) along a straight line path without a full kinematic analysis. The types and magnitudes of such errors are examined and some strategies for their elimination are proposed.

1.3 THE UNIVERSITY OF SURREY MANIPULATOR.

1.3.1 Brief description of the manipulator and its operating system.

The University of Surrey manipulator is a three degree-of-freedom arm designed as a low cost, high capability system suitable for the manipulation of small objects (up to 5 kg).

In order to achieve low cost, easy maintenance and a high power-to-motor-volume ratio the manipulator axes are driven by pneumatic motors rather than by hydraulic or electrical servo-drives as is generally the case with industrial manipulators. Because there are no suitable analogue valves working at pressures of up to 6 bars available on the market, on-off solenoid valves are used to supply the motors with compressed air. The use of compressed air as the driving
medium and of on-off valves to control its supply introduces non-linearities in the system making its control difficult. This problem is overcome by delegating the control of the arm to a microcomputer that has sufficient processing power to compensate for the effects of such non-linearities. The microcomputer receives feedback position signals from all the manipulator axes, processes these signals to extract position and velocity information, processes this information following a suitable control algorithm and thus sends control signals to the on-off solenoid valves which supply the motors driving the manipulator axes.

Structurally the manipulator has a polar configuration using rotary axes and bearings for simplicity and to reduce overall dimensions. Each arm segment is constructed from two overlapping U-shaped sections in a parallelogram linkage to provide rigidity and fixed orientation platforms at the arm extremities.

The specification of the manipulator hardware and of the microcomputer used for its control has undergone a number of changes in the course of the research associated with the manipulator. The basic specification and the modifications it has undergone are detailed in Refs. [61]-[65]. A block diagram of the complete system (taken from Ref. [65]) is given in Figure 1. A block diagram of the manipulator hardware (taken from Ref. [65]) is given in Figure 2.
Figure 1. Block diagram of the complete system (one degree-of-freedom).

Figure 2. Block diagram of the manipulator hardware (one degree-of-freedom).
1.3.2 Difficulties encountered in using existing methods of kinematic and dynamic analyses with the University of Surrey manipulator.

A number of strategies for the control of manipulator motion were outlined in the previous section. These strategies are generally based on a kinematic analysis of the manipulator in question and of an extension of this analysis to take into account the dynamics of the manipulator. The solutions of the resulting equations of motion provide a basis for the synthesis of a suitable control strategy for the manipulator. Most of the above analyses are based on the assumption that the equations of motion of the manipulator are linear (or suitably linearised), although some workers have modified their control strategies to cater for the presence of non-linearities in the control loop (Refs. [24], [29], [30]). All of the above strategies, however, take for granted the ability of the system components to apply any value of controlling torque or force (within the capabilities of the driving components) that is specified for an axis of the manipulator by the control algorithm. This is a reasonable assumption as the driving components of most industrial manipulators are electrical or hydraulic servo-motors in which, because of their analogue nature, the driving torque can take any value (within a certain range).

This assumption is not valid in the case of the University of Surrey manipulator. Analyses aiming at the identification of this manipulator's system components (Refs. [64], [65], [66], [68]) point out that the use of on-off solenoid
Valves for the control of the motors and the compressibility of the air in the motor chambers introduce non-linearities in the control path. The presence of these non-linearities and, in particular, the digital (on-off) nature of the control interface (solenoid valves) limits the system's ability to apply specific values of driving torque to the manipulator axes. In fact, driving torque values can only be approximately specified. In view of this the analyses outlined in the previous section and, in particular, the work which is related to the derivation of control strategies by solving the equations of motion of the system, is of limited use in the formulation of a control strategy for the University of Surrey manipulator.

In spite of these difficulties an attempt was made to obtain an analytical representation for the valve-motor combination by carrying out a small-displacement stability analysis in which the motor volume can be considered constant. This resulted in the valve-motor combination being represented by a 1st order lead transfer function (Refs. [64],[68]). In view of this a control strategy that implemented a 1st order lead term by switching the motor valves according to a (straight-line) switching profile based on position and velocity thresholds was devised for the control of the manipulator. The performance of this strategy is evaluated in Refs. [64],[68]. Briefly, this strategy did not result in the optimal control of the manipulator transient and, furthermore, was unable to eliminate the onset of limit-cyclings (which is chiefly due to the on-off nature of the control interface) that occurs about the target position necessitating the use of additional thresholds and brakings.
action to control the arm. The resulting control software was designated SUMOS and is described in Refs. [65],[67],[68]. (The inability of this strategy to provide an optimal transient is explained by the large changes in motor volume which occur during the transient. These changes render invalid the assumption on which the control strategy is based; i.e. that the motor volume can be considered constant, thus making this strategy unsuitable for the control of the transient. The presence of limit cycling in the target region, where the displacements are sufficiently small for changes in motor volume to be considered negligible, is rather more difficult to explain. It is argued in Chapter 4, that it is due to the incorrect implementation of the first order lead term which is necessary for the stability of the arm.)

1.3.3 Developments leading to the present research.

It was hoped that an improved manipulator trajectory would be achieved if an interpolation routine that generates additional co-ordinates between pre-taught co-ordinates were incorporated into the SUMOS package. It was also hoped that such a routine would enable us to specify the type of path followed by the manipulator by choosing appropriate interpolation routines. The development and evaluation of such routines is carried out in detail in Chapter 2. It is sufficient to say at this point that although such routines resulted in an improved transient for the manipulator they only achieved this improvement at the expense of accuracy. (This is basically due to the shortcomings of the SUMOS package which are mentioned above.) It was also demonstrated that although the overall
speed of the manipulator can be controlled to some extent by changing the interpolation interval. A proper path specification can only be obtained following a full kinematic analysis (which involves a co-ordinate transformation that is too time consuming for use with on-line path generation routines).

The shortcomings of the SUMOS control algorithm highlighted the need for a better representation of the system than that which is provided by small displacement stability analysis. Such an analysis, carried out for a single degree of freedom arm and using comprehensive compressible flow equations, is described in Ref. [47]. The solution of these equations is only possible for specific conditions, making the representation used to describe the pneumatically powered arm under consideration rather unsuitable for general use.

The difficulties which were encountered in obtaining a control strategy for the manipulator by a conventional analysis led to the decision to carry out the analysis of the system by simulating the control loop of a manipulator axis on a digital computer after identifying the non-linearities, evaluating the various system parameters. The first, rather elementary, identification and simulation of system components (described in Ref. [64]) demonstrated the need for a better identification of the non-linearities in the valve-motor pair. To achieve this end an experimental one degree-of-freedom arm was built (incorporating the same drive components as the actual manipulator) and a data acquisition system was used in conjunction with this arm to monitor the values of the arm variables in the course of various trajectories. The arm was
generally controlled by a microprocessor implementing control algorithms similar to those used in the actual manipulator. (Descriptions of the hardware and software configurations of this system are found in Refs. [66],[68],[70],[71],[72].)

This system was used to monitor the pressure transients that occurred when a constant volume was pressurised or de-pressurised through the valve. The pressure transient data was then processed in a digital computer to obtain an experimental flow characteristic for the valve in use. (This work is described in Refs. [66],[68],[71].) Data from this flow characteristic was used to represent the valve in a simulator package (SIMVALVE) that achieved an accurate simulation of a trajectory of the experimental arm. (Refs. [73],[75].)

The research which is described in this thesis is an extension of this work. The emphasis of the research is placed on replacing experimental data with information generated by computer simulation. A procedure for simulating constant volume pressure transients is described in Chapter 3. A method for using such computer generated transients to obtain flow characteristic information is described in Chapter 4. In this manner a simulation of the performance of the arm can be carried out with minimal recourse to experimental data. In Chapter 5 it is argued that the simulation of the trajectory of the arm under the control of a simple switch-over sequence can provide a switching profile that can be used for the control of the arm. Procedures of carrying out such a simulation in forward and reverse time are described and their limitations are
discussed. Some control schemes for the arm using the above simulations are evaluated in Chapter 6 and suggestions about procedures that will lead to an integrated control scheme for the complete manipulator are made.
CHAPTER 2

THE USE OF CO-ORDINATE GENERATION BY INTERPOLATION ROUTINES
FOR THE CONTROL OF THE MANIPULATOR PATH.

2.1 INTRODUCTION.

The University of Surrey manipulator and its operating system software SUMOS were described briefly in the previous chapter and are described in detail in References [61]-[65].

SUMOS drives the manipulator through a series of pre-taught co-ordinates using a control algorithm that is based on the ON-OFF implementation of a first order lead term (by using a suitable switching line). It has been demonstrated (Ref. [64],[68]) that a control algorithm of this type cannot control the manipulator adequately on its own. It is therefore supplemented by a system of thresholds about each target point.

One of our research objectives has always been to guide the manipulator along an optimal or, at any rate, an improved type of path. It was thought that we would go some way towards accomplishing this objective by being able to specify the type of path to be followed by the manipulator between two pre-taught points. To this end a number of co-ordinate generation routines were developed that generate a series of co-ordinates (initially in two dimensions) between two given points, the co-ordinates being governed by a specific type of relationship in each case.

The routines were tested for correct operation, initially on an ICL 1905F mainframe computer and eventually on the INTEL 8080 microprocessor on which SUMOS was then being run. Finally a
routine generating co-ordinates connected by a parabolic relationship in two dimensions between two given points was chosen for incorporation into SUMOS itself. The resulting version of SUMOS performed as expected i.e., each point, whether pre-taught or generated by the interpolation routine was attained with full accuracy, utilising the full control capability of the SUMOS control algorithm.

As this resulted in excessively fine control because of the system of thresholds employed by SUMOS around each target point it was decided to use a simplified form of control for attaining the co-ordinates generated by the interpolation routine while retaining the full control algorithm for pre-taught points. The resulting version of SUMOS provided an acceptable trade-off between accuracy and smoothness of trajectory between two points. It was also found that by specifying different interpolation increments for the co-ordinate generation routine the overall velocity of the manipulator could be dictated to a certain extent.

The work mentioned above is described in this chapter.

2.2 THE CO-ORDINATE GENERATION ROUTINES.

2.2.1 Linear co-ordinate generation.

The first co-ordinate generation routine that was examined generates between two points co-ordinates (in two dimensions) that are related by the equation:
where $K$ is a constant. The co-ordinates of the points are supplied by the programmer who also specifies the size of the interpolation increment (for the dimension in which the distance between the supplied points is the greatest). The routine then uses this information to calculate the number of interpolation steps between the two points and the size of the second interpolation increment. These calculations are carried out during the initialisation part of the routine.

In the second part of the routine (which is designed to be carried out on-line) the starting point co-ordinates are incremented by the appropriate increment in each dimension and the resulting co-ordinates are output to a suitable peripheral. This procedure is repeated until one of the generated co-ordinates exceeds the supplied finish point co-ordinates. The routine then outputs the finish point co-ordinates to the peripheral device and stops. This routine was designated POINTGEN 1. A flowchart for this routine is listed below:
Supplied starting point co-ordinates XSTART, YSTART.

Supplied finish point co-ordinates XFIN, YFIN.

Supplied increment length L.

Flowchart for POINTGEN 1.
A programme based on this flowchart was run on the ICL 1905F computer and the generated co-ordinates were plotted with the aid of a plotting package (FORTSPLOT) resident in that computer. The resulting plot is shown in Figure 1.

Figure 1. Computer plot of points generated by POINTGEN 1.

(The slight non-linearity apparent in this plot is due to lack of resolution in the plotter. A check on the values of the generated co-ordinates [which were also supplied by the programme and are supplied in a separate table below] confirms the correct operation of the routine.)
Table of co-ordinates generated by POINTGEN 1

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.00000</td>
<td>3.00000</td>
</tr>
<tr>
<td>9.00000</td>
<td>5.16667</td>
</tr>
<tr>
<td>13.00000</td>
<td>7.33333</td>
</tr>
<tr>
<td>17.00000</td>
<td>9.50000</td>
</tr>
<tr>
<td>21.00000</td>
<td>11.66667</td>
</tr>
<tr>
<td>25.00000</td>
<td>13.83333</td>
</tr>
<tr>
<td>29.00000</td>
<td>16.00000</td>
</tr>
<tr>
<td>30.00000</td>
<td>16.00000</td>
</tr>
</tbody>
</table>

From this table it can be seen that the interpolation increments in both dimensions are reasonably large. This is because the differences between the start and finish point co-ordinates in each dimension are of similar magnitude. Let us now consider the case in which the difference between the start and finish co-ordinates in one dimension is considerably larger than that in the second dimension. In this case, POINTGEN 1 would calculate a relatively small interpolation increment for this second dimension. If the resulting co-ordinates were to be supplied to the SUMOS control algorithm as the co-ordinates of successive target points this would result in excessively fine control of one axis of the manipulator because of the small increment size. It was therefore thought desirable to increment co-ordinates in both dimensions with increments of the same size regardless of the relative magnitudes of the differences between the start and finish point co-ordinates. A routine that accomplishes this is POINTGEN 2. POINTGEN 2 operates in the same way as POINTGEN 1 up to the point where
the interpolated co-ordinates are generated. It then examines the generated co-ordinate for the dimension that has the smaller interpolation increment. If the difference between this co-ordinate and the last co-ordinate to be output in this dimension (this is stored in memory) is greater than the specified increment length the co-ordinate is output to a peripheral and stored for further comparisons. Otherwise no co-ordinate is output in this dimension and the co-ordinate retained in memory for reference remains the same. A flowchart for POINTGEN 2 is listed below:

Flowchart for POINTGEN 2.

SD denotes the dimension with the smallest distance between start and finish point co-ordinates.

LD denotes the dimension with the greatest distance between start and finish point co-ordinates.
START

XFIN-XSTART=XLENGTH
YFIN-YSTART=YLENGTH

NO

XLENGTH \geq YLENGTH

YES

N=YLENGTH \% L

SDINC=XLENGTH/N

XSTART=SDSTART

XFIN=SDFIN

YFIN=LDFIN

X=SD

Y=LD

M=0

SD=SDSTART

LD=LDSTART

STORE=SDSTART

Output SD, LD using appropriate dimension assignments.

LD=LD+L

SD=SD+SDINC

NO

SD-STORE \geq L

YES

I/F LD

STORE=SD

Output STORE in place of SD

M=N+1

M \geq N

NO

SD=SDFIN

LD=LDFIN

Output SD, LD using the appropriate dimension assignments.

END
A programme based on this flowchart was run on the ICL 1905F computer and a table of the resulting co-ordinates together with a plot of the coordinates output to the FORTSPLOT package are shown in Figure 2. From these it can be seen that POINTGEN 2 performs as expected generating comparable co-ordinate increment lengths in both dimensions, although the co-ordinates which are actually output by the routine are no longer related by a simple linear relationship.

Table of co-ordinates generated by POINTGEN 2.

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.00000</td>
<td>3.00000</td>
</tr>
<tr>
<td>9.00000</td>
<td>3.00000</td>
</tr>
<tr>
<td>13.00000</td>
<td>7.33333</td>
</tr>
<tr>
<td>17.00000</td>
<td>7.33333</td>
</tr>
<tr>
<td>21.00000</td>
<td>11.66667</td>
</tr>
<tr>
<td>25.00000</td>
<td>11.66667</td>
</tr>
<tr>
<td>29.00000</td>
<td>16.00000</td>
</tr>
<tr>
<td>30.00000</td>
<td>16.00000</td>
</tr>
</tbody>
</table>
Figure 2. Computer plot of co-ordinates generated by
POINTGEN 2.

2.2.2 Programming linear co-ordinate generation routines in
assembly language.

Once it was shown that POINTGEN 1 and POINTGEN 2 performed
satisfactorily the feasibility of their use with SUMOS control algorithm had to be established. Therefore the
routines had to be re-written in assembly language using the
INTEL 8080 instruction set in order to be compatible with the
rest of SUMOS. POINTGEN 1 was dealt with first because of its
relative simplicity and because POINTGEN 2 requires a certain
amount of added flexibility in the assignment of dimensions to
the co-ordinates it outputs. (This can be seen from the POINTGEN
2 flowchart.)

As the INTEL 8080 instruction set is designed for use on a
micro-processor its facilities are relatively limited compared
to those provided by a high level programming language such as
FORTRAN 1900. In view of this the original flowchart of POINTGEN
1 was re-written in greater detail so as to describe operations
capable of being carried out by the 8080 instruction set. A copy
of this revised flowchart is listed below:
INTEL 8080 Flowchart for POINTGEN V1.

START
READ XSTART BUFFER
READ XFIN BUFFER
SUBTRACT XSTART FROM XFIN
PLACE RESULT IN XLENGTH BUFFER
READ YSTART BUFFER
READ YFIN BUFFER
SUBTRACT YSTART FROM YFIN
PLACE RESULTS IN YLENGTH BUFFER
COMPARE CONTENTS OF XLENGTH AND YLENGTH BUFFERS
NO
XLENGTH > YLENGTH?
YES
READ LEN BUFFER
READ YLENGTH BUFFER
YLENGTH%L Integer division
PLACE RESULT IN N BUFFER
READ XLENGTH BUFFER
XLENGTH/N Ordinary division
PLACE RESULT IN XINC BUFFER
LOAD XINC BUFFER WITH LEN
LOAD K BUFFER WITH 0

READ LEN BUFFER
READ XLENGTH BUFFER
XLENGTH%L Integer division
PLACE RESULT IN N BUFFER
READ YLENGTH BUFFER
YLENGTH/N Ordinary division
PLACE RESULT IN YINC BUFFER
LOAD XINC BUFFER WITH LEN
LOAD K BUFFER WITH 0
LOAD X BUFFER WITH XSTART CONTENTS
LOAD Y BUFFER WITH YSTART CONTENTS
OUTPUT X,Y
READ X BUFFER
READ XINC BUFFER
ADD X TO XINC
PLACE RESULT IN X BUFFER
READ Y BUFFER
READ YINC BUFFER
ADD Y TO YINC
PLACE RESULT IN Y BUFFER
OUTPUT X,Y
INCREMENT K BUFFER
COMPARE CONTENTS OF K & N BUFFERS

K ≤ N  NO

YES

LOAD X BUFFER WITH XFIN CONTENTS
LOAD Y BUFFER WITH YFIN CONTENTS
OUTPUT X,Y
END
It can be seen that POINTGEN 1 includes two division operations (one integer division that determines the number of increments and one ordinary division that determines the size of the smaller increment) that are beyond the capabilities of the 8080 instruction set as it stands. Integer division can be performed in fixed point format by utilising a subroutine UDIV that is available in the INTEL programme library. In order to perform ordinary division the numbers in question must be converted to floating point format before using a Floating Point Package subroutine FDIV. (The Floating Point Package - written by Mr. M. F. Jeffery of the Department of Mechanical Engineering - enables the INTEL 8080 microprocessor to carry out numerical operations in the floating point format.) Because of this all interpolation increments must be calculated in floating point format and all co-ordinate incrementation must also be carried out in this format.

A programme incorporating the features mentioned above and based on the revised POINTGEN 1 flowchart was written in the 8080 instruction set. This programme was designated POINTGEN V1 and it makes provision for converting the co-ordinates it generates back into fixed point format and outputting them to output ports 6&7 of the INTEL 8080 microprocessor. The signals output by the microprocessor are converted to analogue voltages by two D/A converters and these voltages are used to drive an X-Y plotter. The programme also generates control signals that dictate plotter pen-up or pen-down action and provides a suitable delay between the outputting of each set of co-ordinates in order to enable the plotter to move to these co-ordinates. In this way the generated co-ordinates can be
examined directly on the X-Y plotter.

A diagram of the hardware configuration used for this is shown in Figure 3.

**Figure 3**: Hardware configuration for plotting POINTGEN V1 results.
The actual plotter output is shown in Figure 4.

Figure 4: Plotter output of POINTGEN V1.

We can see from the co-ordinates output by the plotter that POINTGEN V1 performs as expected i.e. it generates between two points a series of co-ordinates (in two dimensions) that are related by a straight line equation. However, the programme uses floating point addition to carry out co-ordinate incrementation and software that performs addition in the floating point format is generally considerably slower than the corresponding software in the fixed point format. It was
therefore thought that the co-ordinate incrementation carried out by POINTGEN V1 might be too slow to be carried out on line if the co-ordinate generation routine were to be incorporated into SUMOS. If on the other hand the co-ordinate incrementation were to be carried out in the fixed point format (effectively using integer rather than ordinary division for increment calculation) the generated co-ordinates would no longer be related by a straight line equation.

In view of the above considerations it was decided to abandon the straight line relationship between the co-ordinates in favour of a simple relationship based on the equation of a suitable smooth curve. The simplest equations of this type are the equations of the conic sections, i.e., circle, parabola, ellipse and hyperbola. Of these the equation of the parabola with its vertex at the origin i.e.,

\[ Y = 4AX^2 \]

where \( A \) is a constant was considered the most suitable because it is sufficiently simple and because the constant \( 4AX \) can be calculated without ambiguity from the co-ordinates of two points on the parabola if one point is at the vertex of the parabola. (Apart from the above reasons a parabola offers further desirable features in terms of manipulator path specification. These will be discussed later in this chapter.) A flowchart for co-ordinate generation using this equation is listed below:
Flowchart for parabolic co-ordinate generation.

START

4A = (YFIN)**2/(XFIN)

X = XSTART

Y = YSTART

[OUTPUT X, Y]

TEMP = X + LEN

TEMP < XFIN ?

YES

X = TEMP

Y = \sqrt{4A \times X}

[OUTPUT X, Y]

X = XFIN

Y = YFIN

[OUTPUT X, Y]

END
This flowchart includes two multiplications, one ordinary division and one square root extraction operation. Although the execution times for the above types of operation are comparable for fixed and floating point routines it was thought that fixed point routines would result in reasonably uncomplicated software (as the resulting co-ordinates would not have to be re-converted to fixed point format) while the relationship between the co-ordinates would remain essentially parabolic. A plot comparing a true parabola and one generated by fixed point operations using the above flowchart is shown in Figure 5.
It can be seen that the curve produced by using fixed point operations is acceptably smooth and can therefore be used as the basis for a co-ordinate generation routine. In view of this the flowchart for the generation of parabolic co-ordinates was implemented in the fixed point format. It incorporates INTEL library routines for multiplication of unsigned integers (UMUL), single byte precision integer division (UDIV) and fixed point square root extraction (SORTF). It also includes all the facilities found in POINTGEN V1 for the outputting of...
co-ordinates to an X-Y plotter. The resulting package was designated CURVEGEN. A plot of the co-ordinates generated by CURVEGEN is shown in Figure 6.
These can be seen to match the co-ordinates generated by implementing a parabolic equation in fixed point arithmetic. We can therefore say that the parabolic co-ordinate generation routine performs satisfactorily.

2.2.3 Incorporation of the co-ordinate generation routine into SUMOS

The main modification that had to be made to CURVEGEN to enable it to be incorporated into SUMOS consisted of deleting the part of CURVEGEN that output the generated co-ordinates to the INTEL 8080 output ports and making provisions for supplying these co-ordinates as set-points to the SUMOS control algorithm.

In order to accomplish this the off-line part of CURVEGEN was placed in a suitable place in the initialisation section of SUMOS and the on-line part that actually generates the
co-ordinates or accesses them from memory replaced the routine in SUMOS that accesses pre-taught set-point co-ordinates from memory while in the RUN mode. As these modifications render the TEACH and EDIT modes of SUMOS superfluous it was decided to carry them out on a simplified version of SUMOS that did not include these modes. The version chosen was an experimental package that caused a single axis of the manipulator to move between two points (the co-ordinates of which were stored in memory) under the control of the full SUMOS control algorithm (SUMOS XOA). Apart from the modifications that deal with co-ordinate generation the package had to be expanded to cater for feedback (encoder outputs) in three axes and supplied with the Interrupt Service Routines of a full three-axis version of SUMOS (SUMOS BKN SYS80D) that generate the thresholds necessary for the on-line implementation of the control algorithm switching line.

Although the package now had the ability to cope with set point co-ordinates in three dimensions (manipulator axes) the co-ordinates generated by CURVEGEN were in two dimensions only. These were supplied as set-point co-ordinates to the lift and rotation axes of the manipulator. The resulting package was designated SUMOS X1.

2.3 MANIPULATOR PERFORMANCE WITH THE SUMOS X1 PACKAGE.

One of the original motives behind generating co-ordinates that are related by a specific type of mathematical relationship was the desirability of being able to specify the
type of path that the manipulator end-point would follow in three-dimensional space. On examining the geometry of the manipulator, however, it becomes immediately obvious that although the co-ordinates which are supplied as set point co-ordinates to the axes of the manipulator by SUMOS are related by an approximately parabolic equation this relationship does not apply to the co-ordinates of the manipulator end-point in space. The implications of this will be discussed in a later part of this chapter.

As the type of path that is followed by the manipulator end-point in three dimensional space could not be evaluated at this stage the performance of the manipulator with the SUMOS X1 package was evaluated from the point of view of smoothness of the trajectory followed by the manipulator between the start and finish points. It was thought that for this type of evaluation it would be sufficient to monitor the progress of one axis of the manipulator through its working cycle. The LIFT axis was chosen and arrangements were made to output the encoder signal from this axis to a D/A converter as well as to the input port of the INTEL 8080 microprocessor (this provides SUMOS with a digital position signal for the axis). The analogue voltage produced by the D/A converter was then used to drive one axis of an X-Y plotter. The other axis of the plotter moved at constant velocity. In this manner a plot of the LIFT axis position against time was obtained through a working cycle of the manipulator.

Such a plot can be seen in Figure 7.
Figure 7: Motion of the Lift axis under the SUMOS X1 Package.

The working cycle of the manipulator starts at the starting point, the co-ordinates of which are stored in memory, proceeds through the points, the co-ordinates of which are generated by the interpolation routine until it reaches the finish point, whose co-ordinates have also been stored in memory, and, having reached that, returns to the starting point. The start and finish point co-ordinates and the desired interpolation increment length are initially supplied along with other SUMOS operating parameters in software form when the package is loaded into the INTEL 8080 microprocessor. They may
be modified with the aid of a MONITOR programme. (This is an
in-house firmware package resident in the INTEL system.)

In SUMOS X1 all set-point co-ordinates, whether stored in
memory or generated by the interpolation routine, are attained
with full accuracy utilising the full capability of the SUMOS
control algorithm.

The SUMOS control algorithm essentially implements a first
order lead term in the ON-OFF mode. It accomplishes this by
following a switching line that is defined by position and
velocity thresholds. (Ref.[62],[63]) It has however been
demonstrated (Ref. [64]) that such a control algorithm is not
sufficient for the control of the manipulator because of the
highly nonliner nature of the pneumatic drive
components. Because of this SUMOS utilises additional thresholds
and braking action when the axis approaches set-point
coor-ordinates in order to attain them with sufficient accuracy.

Because this complex control action is used for all set
point co-ordinates in the manipulator working cycle the path of
the manipulator through these co-ordinates is not particularly
smooth. This may be readily seen in Figure 7.

We may therefore say that the use of interpolation routines
does not result in a smoother trajectory when used with the
full SUMOS control algorithm. In view of this an alternative
strategy was explored.
2.4 USE OF DIFFERENT CONTROL ALGORITHMS.

As SUMOS X1 did not achieve the desired smooth trajectory between the supplied start and finish points it was decided that in order to obtain smoother progress of the manipulator through the co-ordinates generated by the interpolation routine a simpler control algorithm would be used to drive the manipulator through these co-ordinates even though this might result in some loss of accuracy. The full SUMOS control algorithm was retained for attaining the programmed start and finish points with maximum accuracy. The resulting package was given the designation SUMOS X2.

In order to better evaluate the performance of the manipulator with SUMOS X2 it is useful to compare in detail the full SUMOS control algorithm and the simplified version used in attaining co-ordinates generated by interpolation in SUMOS X2.

The original SUMOS control algorithm is designed to provide acceleration, retardation and coasting commands for the actuator system associated with each degree of freedom of the manipulator. These commands depend on whether the manipulator has exceeded retard (positional) and velocity thresholds in each degree of freedom. The retard thresholds are supplied by the programmer (in software form or via MONITOR) while the velocity thresholds are dynamically calculated and are set proportional to the current value of the absolute error times a modifying term.

Outside the retard threshold, if the velocity threshold is
exceeded coasting is instigated. Inside the retard threshold retardation occurs as a function of velocity if the velocity threshold is exceeded. In all other circumstances acceleration occurs. The control action also provides braking action for each degree of freedom. This is dictated in the region of the target position by the current velocity of the manipulator with respect to a velocity brake threshold.

The control algorithm used for the co-ordinates generated by the interpolation routine in the SUMOS X2 package provides only acceleration and coasting commands for each degree of freedom. No braking action is provided for these co-ordinates.

If the manipulator is outside the retard threshold and does not exceed the velocity threshold acceleration occurs.

If the manipulator is outside the retard threshold but exceeds the velocity threshold coasting is instigated.

If the manipulator is inside the retard threshold for one axis coasting is also instigated. Precautions are taken to prevent the coasting from stopping if the axis overshoots outside the region surrounding the target point which is defined by the retard threshold.

If the manipulator is inside the retard threshold for all axes the control algorithm returns to the interpolation routine which generates the next set of co-ordinates.

The response of the manipulator with the SUMOS X2 package
is shown in Figure 8 where the position of the lift axis is again plotted against time throughout a working cycle of the manipulator.
It can be seen, by comparing Figures 7 and 8, that the motion of the manipulator under the control of the SUMOS X2 package is considerably smoother than that obtained under the SUMOS X1 package which utilises the original control algorithm for attaining all points. However, the points which are generated by the interpolation routine are attained less accurately under the SUMOS X2 package. This is considered to be a reasonable trade-off provided that the start and finish point co-ordinates which are stored in memory are attained with maximum accuracy.

From Figure 8 it may be seen that when the first interpolated point is reached by the lift axis the axis stops for a certain amount of time before it moves to the next co-ordinate. This is due to:
a) The fact that all axes must be inside their respective retard thresholds before the next set of co-ordinates is generated, which means that the control of the three degrees of freedom of the manipulator is no longer fully de-coupled.

b) The approximately parabolic relationship between the lift and rotation co-ordinates generated by the interpolation routine. This results in greater incrementation of the rotation co-ordinates compared to that of the corresponding lift co-ordinates during the initial steps of the interpolation. An alternative interpolation routine might provide a remedy for this problem. Such routines will be discussed later in this chapter. As a temporary remedy different retard thresholds were set for the two degrees of freedom in order to cause the lift axis to coast at a later point in its path. This improves the smoothness of the trajectory of the lift axis as can be seen in Figure 9.
Figure 9. Motion of the Lift axis under the SUMOS X2 package with Adjusted Retard Thresholds.

The smooth trajectory obtained with the SUMOS X2 package demonstrates that interpolation routines can result in a smoother manipulator trajectory when used with a suitable control strategy. The ability of the manipulator to follow smoothly, if with reduced accuracy, co-ordinates spaced at regular intervals between programmed points enhances the original point-to-point control of the manipulator somewhat in the direction of continuous path control.

2.5 CONTROL OF MANIPULATOR SPEED

So far, because the manipulator is controlled by an ON-OFF control strategy it has only been possible to control the
maximum speed of the manipulator by imposing a velocity threshold, exceeding which causes the manipulator to coast.

As the SUMOS X2 package achieves a fairly smooth trajectory through co-ordinates generated by an interpolation routine it was thought that control of the overall speed of the manipulator axes could be achieved to some extent by varying the increment length used in the interpolation routine. Plots of the motion of the lift axis against time for different interpolation increments are shown in Figures 10 and 11.
Figure 10. Motion of the Lift axis under the SUMOS X2 package with small Interpolation increment.
Figure 11: Motion of the Lift axis under the SUMOS X2 package with large Interpolation Increment.

It can be seen from figures 9, 10 and 11 that the over-all gradient of the position vs. time plot, i.e., the overall velocity of the axis, increases with increasing interpolation increment size. While fine velocity control of the manipulator axes cannot be exercised in this way this technique can be used to provide overall speed options when programming the manipulator. Furthermore, the relationship between interpolation increment size and overall speed may indicate ways of specifying the velocities of individual axes if alternative interpolation routines are used.

The implications of using alternative interpolation routines and the possibilities for augmented control strategies
that arise from these are discussed below.

2.6 THE USE OF ALTERNATIVE INTERPOLATION ROUTINES.

As it has been mentioned previously the mathematical relationship between the co-ordinates generated by the interpolation routine does not dictate the type of path that is followed by the manipulator end-point in three-dimensional space. Therefore the choice of an approximately parabolic relationship was adhered to more because it would hopefully result in a smooth trajectory through the interpolated points than because a particular type of path was required.

It was seen (Figure 6) that this parabolic relationship results in a more rapid incrementation of the rotation co-ordinates compared to that of the corresponding lift co-ordinates during the first steps of the interpolation and that this brings about a loss in trajectory smoothness (Figure 8). It was thought that this might be remedied by a routine that generates interpolation increments of the same or, at least, comparable size in both dimensions. If such a routine is not to run into implementation problems because of differences in the number of interpolation increments required in each dimension when the distance between the start and finish points is not the same in both dimensions an interpolation scheme of the type proposed in POINTGEN 2 must be used. (This would have to be implemented in fixed point to enable on-line implementation but this is no longer a problem as an exact straight line relationship between co-ordinates is no longer required.)
We know that POINTGEN 2 does not generate co-ordinates in both dimensions in each interpolation step. If such a routine is used in conjunction with the SUMOS X2 control strategy it might result in excessive coasting of the axis whose co-ordinates have not been incremented, thus bringing about that precise loss of trajectory smoothness that we are trying to avoid.

An alternative scheme that might be considered is that proposed in POINTGEN 1. (Again implemented in fixed point format to enable on-line execution.) This type of interpolation routine produces interpolation increments that are approximately proportional to the distance that has to be covered in each dimension. As the overall manipulator speed roughly depends on the size of interpolation increment used this type of routine might enable us to implement a "Synchronous Motion" scheme (a scheme usually encountered in manipulators with analogue drives) in which the velocity of each axis depends on the distance that has to be travelled by the axis thus enabling all axes to approach the finish point simultaneously. In general this results in smoother manipulator motion. However as the interpolation increments produced by such a routine differ in size for each dimension the implementation of this routine in conjunction with the SUMOS X2 control strategy might again result in a loss of trajectory smoothness thus offsetting any advantage gained by a "Synchronous Motion" scheme.

We can therefore see that the use of alternative interpolation routines is not likely to result in smoother manipulator motion and that SUMOS X2, possibly with suitably
Adjusted retard thresholds represents the maximum possible improvement in the manipulator performance if a smoother trajectory between two points is our sole objective.

2.7 CONVERSION OF INTERPOLATION ROUTINE CO-ORDINATES TO INTERNAL MANIPULATOR CO-ORDINATES.

It has been pointed out that the original intention behind using interpolation routines that generate co-ordinates that are related by specific types of relationship was the desire to be able to program the type of path that the manipulator end-point would follow in three-dimensional space. This would be particularly useful as an aid in the TEACH operation as point-to-point co-ordinates for each type of path would not have to be programmed individually by the operator. It might also enable us to carry out a better analytical definition of the criteria involved in an optimal trajectory (i.e., maximum velocity, minimum trajectory time, maximum path smoothness etc., as described in Ref. [24]).

In order for the manipulator end point to follow a particular path in three-dimensional space the Cartesian co-ordinates defining points in this space (and which can be generated by suitable interpolation routines) have to be converted to internal robot co-ordinates (i.e., joint angles) that take into account the geometry of a particular manipulator. In the case of the Surrey manipulator the conversion is carried out as follows:
From the above diagrams we can see that the following equations apply:

\[ x = r \times (\sin \theta_3) \times (\cos \theta_1 + \cos \theta_2) \]  
\[ y = r \times (\sin \theta_1 + \sin \theta_2) \]  
\[ z = r \times (\cos \theta_3) \times (\cos \theta_1 + \cos \theta_2) \]

Squaring the above equations gives:

\[ x^2 = r^2 \times (\sin \theta_3)^2 \times (\cos \theta_1 + \cos \theta_2)^2 \]  
\[ y^2 = r^2 \times (\sin \theta_1 + \sin \theta_2)^2 \]  
\[ z^2 = r^2 \times (\cos \theta_3)^2 \times (\cos \theta_1 + \cos \theta_2)^2 \]

Adding equations (4) and (6) gives:

\[ (x^2 + z^2)/(r^2) = (\cos \theta_1 + \cos \theta_2)^2 \]

Substituting the above in (6) gives:

\[ z = r \times (\cos \theta_3) \times (x + z)/(r) \]

Hence:
\[ \cos \Theta_3 = \frac{z^2}{(x + z)^2} \] - - - - - - - - - - - - (9)

Therefore:

\[ \Theta_3 = \arccos \sqrt{\frac{z^2}{(x + z)^2}} \] - - - - - - - - - - - - (10)

Equation (7) gives:

\[ \cos \Theta_1 + \cos \Theta_2 = \sqrt{\frac{2}{x + z}} (\frac{2}{r}) \] - - - - - - - - - - - - (11)

Equation (2) gives:

\[ \sin \Theta_1 + \sin \Theta_2 = \left(\frac{y}{r}\right) \] - - - - - - - - - - - - - - - - - - (12)

Equation (11) becomes:

\[ \sqrt{\frac{2}{x + z}} (\frac{2}{r}) = \]

\[ = 2 \cos(\frac{\Theta_1 + \Theta_2}{2}) \cos(\frac{\Theta_1 - \Theta_2}{2}) \] - - - - - - - - - - - - (13)

Equation (12) becomes:

\[ \left(\frac{y}{r}\right) = \]

\[ = 2 \sin(\frac{\Theta_1 + \Theta_2}{2}) \cos(\frac{\Theta_1 - \Theta_2}{2}) \] - - - - - - - - - - - - (14)

Dividing equation (14) by equation (13) gives, after manipulation:

\[ \Theta_1 + \Theta_2 = 2 \arctan \sqrt{\frac{2}{x + z}} \] - - - - - - - - - - - - (15)

Squaring equations (13) and (14) and adding gives, after manipulation:

\[ \Theta_1 - \Theta_2 = 2 \arccos \sqrt{\frac{2}{x + y + z}} \] - - - - - - - - - - - - (16)

Adding equations (15) and (16) gives:
\[
\theta_1 = \arctan \sqrt{\frac{2}{y^2}(x+z)} + \arccos \sqrt{\frac{2}{x+y+z}}(2\pi r)
\]

Subtracting equation (16) from equation (15) gives:

\[
\theta_2 = \arctan \sqrt{\frac{2}{y^2}(x+z)} - \arccos \sqrt{\frac{2}{x+y+z}}(2\pi r)
\]

In order to estimate approximately the time required by a microprocessor to calculate \(\theta_1, \theta_2\) and \(\theta_3\) when \(x, y, z\) and \(r\) are supplied the approximate times required for the execution of various operations are given below (in fixed point format):

- Addition, Subtraction .......... 19 microseconds
- Multiplication (UMUL) .......... 250 "
- Division (UDIV) ................. 350 "
- Square Root Extraction (SQRTF) 600 "
- Trigonometric Functions ....... 950 "

The above times were derived by adding up the instruction execution times for the relevant routines in the INTEL programme library. The trigonometric function estimate was derived from a cosine calculation function.

If we assume that inverse trigonometric functions and ordinary trigonometric functions have similar execution times the following approximate estimates can be made for the calculation of the internal robot co-ordinates \(\theta_1, \theta_2\) and \(\theta_3\) from the Cartesian co-ordinates \(x, y\) and \(z\) and the manipulator arm radius \(r\):
In addition to these times the time taken to generate the cartesian co-ordinates on-line must be taken into account. As the interpolation routines that generate these co-ordinates involve addition and possibly multiplication and square root extraction operations the value to be added should be of the order of 100 to 500 microseconds.

It is clear that with an operating system based on 1 millisecond interrupts (as is the case with SUMOS) the above calculations cannot be carried out on-line unless a faster microprocessor and/or more efficient algorithms are used.

2.8 LIMITATIONS OF THE SUMOS CONTROL STRATEGY.

The original objectives behind the adoption of interpolation routines in SUMOS was a) the ability to specify various types of path that the manipulator end-point would follow in three dimensional space or b) failing that to enable the manipulator to move smoothly between two points by making it follow a series of interpolated co-ordinates. Objective (a) has been frustrated at this stage by the inability of the available microprocessor to carry out the necessary calculations on line. Objective (b) has been accomplished to a certain extent by trading off the accuracy with which set-point co-ordinates...
are achieved for a smoother trajectory. (As is the case when attaining interpolated co-ordinates in SUMOS X2.)

This situation highlights the limitations inherent in the original SUMOS control algorithm. This algorithm essentially implements a first order lead term which is based on a simplified representation of the (highly non-linear) pneumatic drive components of the manipulator. As this basic term does not adequately represent the components in the manipulator control loop its control action is not sufficiently suited to the characteristics of the system so as to cause each axis of the manipulator to attain exactly its set-point position while it decelerates to 0.0 velocity at the precise instant this happens.

Because of this a complicated series of thresholds AND braking action are used to ensure that each manipulator axis eventually attains its set-point position (usually after a number of small overshoots). The control strategy behind these thresholds and braking action is essentially empirical and is not based on any additional insight of the system characteristics. Because of this a number of overshoots around the target point cannot be avoided when the full SUMOS control algorithm is used to drive the manipulator to this point. (If the various thresholds are incorrectly set these overshoots can degenerate into a limit cycle about the target point.)

This is essentially what is responsible for the relative lack of smoothness in the manipulator trajectory when attaining set-point co-ordinates with full accuracy (as is the case with
SUMOS X1 - see Figure 7) and is also the reason why accuracy has to be traded off for trajectory smoothness in SUMOS X2.

The essential limitation of the control algorithm, therefore, lies in its inadequate representation of the system under control. Because of this it was decided at this stage that further work on interpolation routines and path definition would be pointless until a modified control algorithm (based on better modelling of the system) which would enable each manipulator axis to attain its set-point co-ordinates smoothly and accurately was developed.

In addition to the basic shortcomings of the original control algorithm it was suspected that the braking action (which occurs when a zero positional error criterion is satisfied) was not being implemented as promptly as intended, thus permitting further axis overshoots.

A way of checking this was to use the START-UP facility in SUMOS. This facility applies the brake on each axis when all three axes are placed manually at their starting point co-ordinates. The braking action is again dictated by the satisfaction of a zero positional error criterion. It was found that when the error limits for the satisfaction of this criterion are set to the same values used in the RUN mode of SUMOS (they are habitually set to larger values for the START-UP procedure) braking does not occur if the manipulator approaches the starting point co-ordinates with sufficient speed. To determine the maximum axis speed at which braking does occur the lift axis was moved manually with an oscillatory
motion and with both sides of its pneumatic motor being de-pressurised. The oscillatory motion was started at high speed and the speed was gradually reduced with each successive oscillation. When the speed was reduced sufficiently for SUMOS to recognise the zero error condition when it occurred braking was instigated and no further oscillations of the lift axis were possible. A position-time plot was obtained for this test using the facilities available for testing the SUMOS X1 and SUMOS X2 packages. This plot can be seen in Figure 12.

![Figure 12](image)

**Figure 12.** Position-time plot of Lift axis during START-UP test.

The gradient of this plot immediately prior to the point where braking occurred gives an indication of the maximum axis velocity at which the braking action can be relied upon to perform as expected. This velocity was estimated at approx. 1.04 rad/sec.

As this velocity could conceivably occur in normal SUMOS
operation further investigation was warranted. It was first suspected that the duration of the Interrupt Service Routines that supply all the feedback data to the SUMOS control algorithm was sufficiently long for the zero error condition to be missed between successive interrupts. To test this the Interrupt Service Routines were shortened to the point where only the calculations necessary for the verification of the zero error condition for one axis only (Lift) were performed. This did not bring about any significant change in the results of the START-UP test. A similar test was carried out using the SYS80 microprocessor configuration which incorporates the SBC10 board that generates internal interrupts and can therefore use faster software for the Interrupt Service Routines. Again this did not affect the results of the START-UP test significantly. This ruled out the actual duration of the Interrupt Service Routines as the cause of the insufficient promptness of the braking action during the START-UP test.

It was then decided to investigate the timing of the interrupt service routines and the control signal output routines to establish the respective sampling rates in each case. To this end the SUMOS X2 package was modified so as to set and then reset certain bits of an INTEL8080 output port when an Interrupt Service Routine starts and finishes and certain other bits when a control signal generation routine starts and finishes. The resulting package was designated SUMOS X2T. The pulses generated by SUMOS X2T during normal manipulator operation were monitored on an oscilloscope. Polaroid photographs of the oscilloscope traces for various settings of the time scale are shown in Figure 13.
Figure 13. Input and Output Sampling pulses obtained by SUMOS X2T.

A)

Time scale: 2ms/div.

B)

Time scale: 5ms/div.

A)

Time scale: 20ms/div.

A) Interrupt Service Routines Pulses.

B) Control Algorithm Pulses.
An examination of the pulses produced by SUMOS X2T produced the following results:

Duration of parameter update for a single axis by the Interrupt Service Routines (input sampling duration) = 1.8 ms

Interval between parameter updates of the same axis (input sampling interval) = 13.2 ms

Duration of control algorithm (output sampling duration) = 0.4 ms

Interval between successive applications of the control algorithm (output sampling interval) - set by internal clock - = 20 ms

From the above results we can see that the input and output sampling rates employed by SUMOS are neither synchronous nor identical. This situation makes their evaluation using sampling theory impossible. It does however provide a qualitative explanation for the lack of promptness in the braking action as the input and output sampling cycles can be seen to overlap approximately every 60 ms. This overlapping rather than the duration of the Interrupt Service Routines is what causes the control algorithm to miss the zero error condition when the manipulator axis is moving above a certain velocity. The above tests verify that apart from its inherent limitations, which stem from an inadequate representation of the system under control, the SUMOS control algorithm cannot always implement the braking action that is designed to overcome the above
limitations. If the braking action is to be made more reliable the input sampling rates of the system will have to be investigated. However it was decided earlier on that rather than proceed in this direction work would be carried out aimed towards obtaining better modelling of the system. Based on this modelling a new control strategy could then be devised that would do away with the need for braking action.

This work is described in the following chapter.

NOTE: The work dealing with parabolic co-ordinate interpolation routines and the use of various control algorithms with such routines is described in a paper submitted to the 10th International Symposium on Industrial Robots - Milan 1980- (Ref. [69]).
3.1 INTRODUCTION.

The need for better modelling of the manipulator axis was pointed out in the previous Chapter. The component of the manipulator axis which is least amenable to analytical representation is the valve and motor combination because:

a) The valve is described by a nozzle equation for compressible flow:

\[ \frac{dM}{dt} = f(P_d/P_u) \]

relating the rate of flow of mass \( M \) to the ratio of downstream and upstream valve pressures \( P_d \) and \( P_u \) respectively. The function \( f \) depends upon the physical properties of the valve and has not hitherto been determined.

b) The motor is described by the equation (derived from the gas equation):

\[ \frac{dP}{P} = \frac{dM}{M} - \frac{dX}{X} \]

where \( P \) is the pressure in the motor chamber under consideration, \( M \) is the mass of air in the chamber and \( X \) is the motor displacement. Because of the interdependence of the pressures in the motor chambers, the driving torque of the motor and the resulting motor displacement, an independent solution for all the variables in this equation cannot be obtained.
Therefore this equation cannot be used as it stands for the analytical representation of the motor.

Early work aimed at obtaining comprehensive modelling for the valve and motor combination (Ref. [66]) initially resulted in the derivation of a flow characteristic for the valve from data gathered with the aid of a data acquisition system connected to an experimental one degree of freedom arm.

Data from the flow characteristic was then used in conjunction with a simulator programme which includes the motor equation and an accurate representation of the dynamics of the arm and simulates the progress of the arm between specified positions under the control of various types of control algorithm. This produced a simulation which agrees well with results obtained from the experimental arm moving between the same positions. (Ref. [66])

It was thought that, having obtained an experimental flow characteristic and a basis for the simulation of the arm, the next step should be to divorce the simulation of the arm from experimental data and generate the flow characteristic analytically. This approach was made possible by the following facts:

a) The experimental flow characteristic was initially obtained from the pressure transients logged during pressurisation and depressurization of a constant volume through the valve.
b) The pressure transients mentioned above are very similar to those described by exponential rise and exponential decay functions.

It was found that if a sufficiently narrow region of the flow characteristic was examined the relationship between mass flowrate and chamber pressure could be considered to be linear (piece-wise linearisation). Using this linear relationship it became possible to express a constant volume pressure transient as an exponential rise or decay and to implement such a function in a computer. The work leading to this exponential representation is described in this chapter.

3.2 EXPERIMENTAL DERIVATION OF THE VALVE FLOW CHARACTERISTIC.

The experimental derivation of the flow characteristic is described in detail in Refs. ([66],[71],[72]). Some features of this derivation that are relevant to the work described in this chapter are mentioned below:

3.2.1 Experimental setup and relevant considerations.

The experimental setup that is used for the derivation of the flow characteristic is essentially a single degree of freedom version of the three-degree of freedom manipulator. It is controlled by a microprocessor and incorporates pressure sensors in the motor chambers, a position sensor and a data acquisition system, usually resident in another microprocessor. The hardware - and software - configuration of the system has
been modified on a number of occasions (Refs. [66],[70],[71],[72]) but the basic operational features described above have not changed. A schematic of one such configuration (the schematic is obtained from Ref. [66]) is shown in Figure 1.

![Schematic Diagram](image)

**Figure 1.** Schematic of experimental one degree of freedom system.

In terms of component representation the most complex part of this system is that which deals with the compressible flow through the valve supply line to the motor chamber. A typical flow path for one of the system configurations used is shown in Figure 2. (The schematic is obtained from Ref. [66].)
In the above configuration, air from an independently regulated source is allowed to flow to a distribution manifold through a suitably activated solenoid valve and via the supply line to the motor chamber. The flow path (using Burkert solenoid valves and Festo connectors) contains 14 changes in cross-sectional area and 4 abrupt changes of direction. When analysed by classical thermodynamic and fluid mechanic methods, each of these has to be treated separately, together with allowances for local friction losses, in order to calculate accurately the final flow rate. This presents a virtually insoluble set of simultaneous equations involving the instantaneous pressure at each section along the path - only the source pressure and the initial value of the chamber being known.
In order to circumvent this difficulty, it was decided to obtain the flow characteristics empirically. The method adopted takes account of the entire valve, supply line and motor chamber transfer characteristics.

The basic assumption of the method is that the mass flow rate of air passing into or out of the motor chamber is dependent only upon the end pressures of the system. Therefore, as the supply pressure and the atmospheric pressure can be considered constant for filling and venting operations respectively, the mass flow rate is a function of the chamber pressure only.

3.2.2 Derivation of the flow characteristic.

Working on the assumption that we can express the mass flow rate solely as a function of the chamber pressure we must devise a method for extracting this function from experimental data. The procedure for doing so is based on the following equation:

\[
\frac{dM}{dt} = \frac{V}{R(g)*T} \frac{dP}{dt}
\]

(This equation is obtained by differentiating the Gas Equation - Equation (2), listed later in this chapter-)

The first experimental step in this procedure is to obtain a pressure transient for a constant volume of the motor chamber following a step signal to the solenoid valve. From this transient (obtained by the data acquisition system) we then obtain values of \(\frac{dP}{dt}\) and hence, using the above
equation, values of \((\text{dM/dt})\) for various points in time throughout the transient. These are then correlated to pressure values for the same times in the transient and mass flowrate is plotted against pressure to provide an empirical flow characteristic. (A variety of computing techniques have been used to carry out this procedure. These are described in References [66],[71],[72].) Flow characteristics obtained using this procedure for filling and venting operations of the motor chamber are shown in Figures 3 and 4.

![Diagram of flow characteristics](image)

**Figure 3.** Experimental flow characteristic for filling operation (charge).
In order to simulate the behaviour of the experimental arm with a digital computer, the pressure and mass flowrate data which are contained in the above flow characteristic were placed in a look-up table in the computer memory thus eliminating the need for an analytical function relating these manipulator variables. The motor chamber was represented by the equation:

\[
\frac{dP}{P} = \frac{dM}{M} - \frac{dX}{X}
\]
and the dynamics of the arm were described by a series of simple equations relating chamber pressures to driving torque and hence to the motion variables of the arm i.e., acceleration, velocity and displacement. All the above equations were used to update manipulator variables at simulated time (not real time) intervals thus resulting in a simulator package for the arm.

A full description of this type of simulator (using experimental data stored in a look-up table) is given in References [73],[75]. A comparison of the simulated performance of the arm (as generated by the simulator) and the actual behaviour of the arm is made in Refs. [65],[66]. At this point it is sufficient to say that the performance of the simulator package is satisfactory if we overlook the need to store a certain amount of flow characteristic data in the computer memory.

One feature of this simulator (retained in subsequent versions) which is relevant to the work described in the following sections is the relative simplicity of the equations used for the representation of the various components of the manipulator arm. As in most types of purely digital simulation of systems complex equations are avoided and the simplicity of the selected representation is compensated by updating each system variable at very small simulated time intervals thus dealing with relatively small changes of the variable in question. The relevance of this feature will become evident in the following section.
3.3 PIECE-WISE LINEARISATION OF THE FLOW CHARACTERISTIC.

Let us now consider a generalised flow characteristic for the valve. We know from the theoretical examination of gas flow through restrictions (Ref. [48]) that the relationship between mass flow rate and pressure in such cases is complex (even when considering the kind of simplified situation that is amenable to theoretical analysis) but continuous. Therefore any plot of this relationship generally takes the form of a fairly smooth curve. This being the case, we can consider the flow characteristic shown in Figure 3 (experimental pressurisation flow characteristic) as sufficiently representative of the system under examination and as sufficiently general for any peculiarities of this particular system not to affect the general representation.

A flow characteristic of this type is shown in Figure 5.

![Figure 5. Generalised flow characteristic.](image)
Let \( P_c \) be the steady state pressure at which no mass flows through the valve. If we consider a suitably narrow region of the flow characteristic we can argue that the flow characteristic in the region under consideration can be replaced by a straight line. This means that if sufficiently small changes in pressure and mass flow rate are considered the relationship between pressure and mass flow rate can be taken to be linear.

Expressing such a relationship for point \( (o) \) on the flow characteristic gives:

\[
\frac{1}{R(o)} = \frac{1}{(P(o) - P_c(o))} \]

Where \([1/R(o)]\) is the gradient at point \( (o) \), hence:

\[
R(o) = \frac{[P_c(o) - P(o)]}{M(o)}
\]

\( R(o) \) is a term that represents the resistance of the valve. The gradient \([1/R(o)]\) is different for each point of the flow characteristic. If, however, the gradient is calculated for each point the relationship between mass flow rate and pressure can be given by equation (1).

This piece-wise linearisation of the flow characteristic provides us, for the first time, with a relationship between the mass flow rate through the valve and the pressure in the motor chamber which can be used in an analytical representation of the driving components of the arm. One must, however, always bear in mind that this relationship is only valid when applied to small changes in the variables under consideration. As the
Projected use of this relationship is the derivation of the equations necessary for a computer simulation of the system. This limitation is relatively unimportant because, as we mentioned previously, the philosophy behind computer simulation involves the substitution of complex relationships by relatively simple ones in which the variables are updated at small time intervals. Therefore, the changes in the variables between successive updating operations are likely to be sufficiently small for a simplified representation to be valid.
3.4 EXPONENTIAL REPRESENTATION OF THE PRESSURE TRANSIENT.

The linear relationship between mass flowrate and chamber pressure which was described in the previous section relies on the correct evaluation of the flow characteristic gradient \( R(o) \) for its correct implementation. As at this point this gradient could not be derived analytically but had to be evaluated from an experimentally obtained flow characteristic it was decided to re-examine the method of obtaining this experimental flow characteristic with the aim of replacing it with results produced analytically.

The most important feature of this method is that the data actually obtained from the experimental setup are constant volume pressure transients. Every other step of the method is carried out by computer processing of these transients. It would therefore be desirable to be able to generate such transients by computer, thus eliminating the need for experimental data in the derivation of the pressure-mass flowrate relationship, which can be used to simulate the behaviour of the valve.

As a first step the constant volume pressure transients which had been used for the derivation of the experimental flow characteristic were examined. A pair of such transients, obtained from the data acquisition system during the pressurisation and depressurisation of one motor chamber of the arm (while the position of the arm was kept constant thus maintaining a constant motor chamber volume), are shown in Figures 6 and 7.
Figure 6. Experimental constant volume pressure transient for pressurisation.
Figure 7. Experimental constant volume pressure transient for de-pressurisation.

It can be seen that these transients bear a close resemblance to plots of exponential rise and exponential decay functions. Because of this an analysis of the flow through the valve was carried out (starting from first principles of compressible flow) in an attempt to link such functions to the representation of the pressure transient. This analysis is shown below:
Consider the Equation of State for gases:

\[ P*V = M*R(s)*T \]  \hspace{1cm} (2)

where \( R(s) \) is the gas constant for air.

ASSUMING ISOTHERMAL EXPANSION \((dT = 0)\) and differentiating equation \((2)\) we have:

\[
\frac{\partial M}{\partial P} \quad \frac{\partial M}{\partial V} \quad (3)
\]

The partial derivatives w.r.t. \( P \) and \( V \) from equation \((2)\) are:

\[
\frac{\partial M}{\partial P} = V \quad \frac{\partial M}{\partial V} = R(s)*T \]  \hspace{1cm} (4a)

\[
\frac{\partial M}{\partial P} = P \quad \frac{\partial M}{\partial V} = R(s)*T \]  \hspace{1cm} (4b)

Substituting \((4a)\) and \((4b)\) in equation \((3)\) gives:

\[
\frac{1}{R(s)*T} \quad (V*dP + P*dV) \quad (5)
\]

ASSUMING THE MOTOR VOLUME TO BE CONSTANT \((dV = 0)\) equation \((5)\) becomes:

\[
\frac{\partial M}{\partial P} \quad (6)
\]

Differentiating equation \((6)\) w.r.t. time gives:

\[
\frac{dM}{dt} \quad \frac{V}{R(s)*T} \quad \frac{dP}{dt} \quad (7)
\]
(The above is the equation used in the derivation of the experimental flow characteristic.)

Substituting \((dM/dt)\) from (7) into equation (1) gives:

\[
\frac{dP}{dt} = \frac{1}{R(s) \cdot T} \cdot (P_c - P) \quad \cdots (8)
\]

where \(R(p)\) is the generalised notation for the gradient of the flow characteristic.

If we define a time constant \(\tau = \frac{V \cdot R(p)}{[R(s) \cdot T]}\) and re-write equation (8) in s-notation we have:

\[
P = P_c \cdot \frac{1}{1 + \tau \cdot s} \quad \cdots (9)
\]

For a step change of magnitude \(P_c\) from an equilibrium state we have, by re-writing equation (9) in the time domain:

\[
P = P_c \cdot (1 - e^{-t/\tau}) \quad \cdots (10)
\]

The derivation of equation (10) is given in APPENDIX 1.

3.5 COMPUTER SIMULATION OF THE EXPONENTIAL PRESSURE TRANSIENT.

The simulator software for the manipulator axis accomplishes the updating of the manipulator variables (at simulated 1 millisecond intervals) by going through a loop that evaluates and updates these variables at fixed time intervals \(\delta t\) (\(\delta t\) is set to 1 millisecond). Each time the programme goes through the loop the interval \(\delta t\) is added to the value of the simulated elapsed time since the start of the simulation. If the
The exponential pressure transient described by equation (10) is to be simulated by a digital computer. This equation must be modified in order to become compatible with the above procedure.

Consider the exponential pressure transient described by equation (10):

$$ P = P_c \times (1 - e^{-t/\tau}) $$  \hspace{1cm} (10)

Let this function describe an exponential pressure rise between \( P = 0.0 \) and \( P = P_c \). Such an exponential rise is shown below:

For compatibility with the simulator software let us suppose that time is expressed as the sum of discrete and equal time increments \( \delta t \). Therefore, the value of the time variable at any point is given by \( t = n \delta t \). For the sake of simplicity let \( \tau = 1 \). Equation (10) then becomes:

$$ P = P_c \times (1 - e^{-n \delta t}) $$
For \( n=1 \) this becomes:

\[
P(1) = P_c \ast (1 - e^{-\delta t})
\]  \quad (a)

For \( n=2 \) we have:

\[
P(2) = P_c \ast (1 - e^{-2\delta t})
\]  \quad (b)

Now consider the following representation for \( P(2) \):

\[
P(2) = P(1) + (P_c - P(1)) \ast (1 - e^{-\delta t})
\]  \quad (c)

Substituting \( P(1) \) from (a) into (c) we have:

\[
P(2) = P(1) + P_c - P(1) - P_c e^{-\delta t} + P(1) e^{-\delta t}
\]

\[
= P_c - P_c e^{-\delta t} + P_c \ast (1 - e^{-\delta t}) e^{-\delta t}
\]

\[
= P_c - P_c e^{-\delta t} + P_c e^{-\delta t} - P_c e^{-2\delta t}
\]

\[
= P_c \ast (1 - e^{-2\delta t})
\]

From the above we can see that the expressions (b) and (c) are equivalent. The exponential pressure transient can therefore be represented by an expression of the general form:

\[
P(n) = P(n-1) + (P_c - P(n-1)) \ast (1 - e^{-\delta t / \tau})
\]  \quad (11)

Equation (11) can provide the basis for a programme that will generate an exponential pressure transient on a digital computer.

Before proceeding to describe such a programme, however, it is useful to examine whether or not the assumptions which are
made directly or indirectly, in the derivation of the exponential pressure transient representation are in fact valid for such an application. These assumptions are the following:

(i) Direct assumptions:

a) Isothermal expansion.

b) Constant volume expansion.

(ii) Assumptions implicit in using equation (1) to relate mass flowrate to pressure:

c) That the mass flowrate varies linearly with the pressure.

d) That the mass flowrate is a function only of the chamber pressure.

If we consider the manner in which a computer simulation of the pressure transient is carried out (i.e., updating of variables at small time intervals, hence small variable increments) we can immediately see that the criteria for valid piece-wise linearisation of the flow characteristic (c) are satisfied. For the same reason we may consider that the pressure changes in each simulation step are sufficiently small for the expansion to be considered isothermal (a).

The assumption of a constant volume expansion is satisfied directly as we have dealt with constant volume expansion
throughout the above analysis (b),

Finally the assumption that the mass flowrate is a function only of the chamber pressure is justified by the fact that the pressure on one side of the valve is always constant (supply or atmospheric). As the theoretical relationship between mass flowrate and pressure for compressible flow across a valve generally involves the ratio of the pressures on the two sides of the valve (Ref. [48]) it can be expressed as a function of a single pressure if the other pressure is constant. This can be said to apply to the similar (though not identical because of the effect on the flow of the various connectors) relationship expressed by the experimental flow characteristic.

Having ascertained that the analysis leading to an exponential representation of a constant volume pressure transient is compatible with the computer implementation of this representation, a simulator programme that generates and plots such a transient was written. This programme is based on the general simulator package for the manipulator arm but it does not include the motor representation or the routines that simulate the dynamic performance of the arm. Instead it only deals with the pressure build-up in one motor chamber the volume of which is considered to be constant. The pressure transient generation is based directly on equation (11).

The initial pressure in the motor chamber has to be supplied by the programmer together with the value of the time constant \( \tau \). The desired final pressure in the motor chamber also has to be supplied. This, in conjunction with the initial
pressure in the motor chamber determines whether the motor chamber is being pressurised or de-pressurised. The position of the arm is considered to be constant. (In this programme we assume that the time constant $\tau$ remains the same for all values of $R(p)$. The validity of this assumption will be investigated shortly.)

The programme was designated XPPRTR and its performance is discussed in the following section. A listing of XPPRTR is supplied in Appendix 2.

3.6 PERFORMANCE OF THE XPPRTR PACKAGE.

3.6.1 Determination of the reference simulation time constant $\tau$.

The performance of XPPRTR was evaluated as follows: The initial and desired final pressure were set so as to specify pressurisation of the motor chamber. Then a range of values of the time constant $\tau$ were supplied to the programme and the one that resulted in the transient that matched the experimental pressurisation transient (Figure 6) most closely was designated the simulation time constant for this volume. The procedure was then repeated for the de-pressurisation of the motor chamber. The values for the simulation time constants for pressurisation and de-pressurisation of a calibrated 308 cc volume were 0.25 and 1.50 seconds respectively. The transients produced by XPPRTR when using the above time constants are shown in Figures 8 and 9.
Figure 8. Simulated constant volume pressure transient for pressurisation.
Figure 9: Simulated constant volume pressure transient for de-pressurisation.

3.6.2 Factors affecting the value of the simulation time constant.

The time constant $\tau$ was defined as:

$$\tau = \frac{[V \times R(\rho)]}{[R(g) \times T]}$$

The values of the volume $V$, gas constant $R(g)$ and absolute temperature $T$ are considered to be constant throughout the pressure transient. The gradient $R(\rho)$ of the flow characteristic however cannot be said to remain constant throughout the range
of pressures that occur in the motor chamber during pressurisation. (An examination of the de-pressurisation flow characteristic shown in Figure 7 indicates that $R(p)$ can be considered constant during de-pressurisation.) This indicates that the simulation time constant should be varied with pressure during pressurisation. In order to determine the relationship between $R(p)$ and chamber pressure the gradient $R(p)$ of the experimental flow characteristic for pressurisation (Figure 3) was measured for a range of pressures covering the whole of the flow characteristic. A table of these measurements and the resulting values of $R(p)$ is shown below:
<table>
<thead>
<tr>
<th>$F$</th>
<th>$\dot{M}$</th>
<th>$\delta P$</th>
<th>$\dot{M}$</th>
<th>$R(F) = (\dot{M}/\delta P)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>4.30</td>
<td>0.50</td>
<td>0.20</td>
<td>0.40</td>
</tr>
<tr>
<td>0.50</td>
<td>4.10</td>
<td>0.50</td>
<td>0.20</td>
<td>0.40</td>
</tr>
<tr>
<td>1.00</td>
<td>3.90</td>
<td>0.50</td>
<td>0.20</td>
<td>0.40</td>
</tr>
<tr>
<td>1.50</td>
<td>3.70</td>
<td>0.50</td>
<td>0.20</td>
<td>0.40</td>
</tr>
<tr>
<td>2.00</td>
<td>3.50</td>
<td>0.50</td>
<td>0.20</td>
<td>0.40</td>
</tr>
<tr>
<td>2.50</td>
<td>3.30</td>
<td>0.50</td>
<td>0.20</td>
<td>0.40</td>
</tr>
<tr>
<td>3.00</td>
<td>3.10</td>
<td>0.50</td>
<td>0.20</td>
<td>0.40</td>
</tr>
<tr>
<td>3.50</td>
<td>2.70</td>
<td>0.50</td>
<td>0.40</td>
<td>0.80</td>
</tr>
<tr>
<td>4.00</td>
<td>2.30</td>
<td>0.50</td>
<td>0.40</td>
<td>0.80</td>
</tr>
<tr>
<td>4.50</td>
<td>1.80</td>
<td>0.50</td>
<td>0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>4.60</td>
<td>1.70</td>
<td>0.10</td>
<td>0.10</td>
<td>1.00</td>
</tr>
<tr>
<td>4.70</td>
<td>1.55</td>
<td>0.10</td>
<td>0.15</td>
<td>1.50</td>
</tr>
<tr>
<td>4.80</td>
<td>1.40</td>
<td>0.10</td>
<td>0.15</td>
<td>1.50</td>
</tr>
<tr>
<td>4.90</td>
<td>1.25</td>
<td>0.10</td>
<td>0.15</td>
<td>1.50</td>
</tr>
<tr>
<td>5.00</td>
<td>1.10</td>
<td>0.10</td>
<td>0.15</td>
<td>1.50</td>
</tr>
<tr>
<td>5.10</td>
<td>0.80</td>
<td>0.10</td>
<td>0.30</td>
<td>3.00</td>
</tr>
<tr>
<td>5.20</td>
<td>0.45</td>
<td>0.10</td>
<td>0.35</td>
<td>3.50</td>
</tr>
<tr>
<td>5.30</td>
<td>0.00</td>
<td>0.10</td>
<td>0.45</td>
<td>4.50</td>
</tr>
</tbody>
</table>

A plot of the values of $R(F)$ against pressure is shown in Figure 10.
Figure 10. Plot of $R(p)$ against chamber pressure for pressurisation.

From this plot it can be seen that $R(p)$ has approximately constant (but different) values (0.4 and 1.1) through the pressure ranges of 0 – 3 bar and 3 – 5 bar respectively and that it varies linearly with pressure for pressures above 5 bar, the equation for this variation being:

$$R(p) = 8.33p - 40.23$$
As the time constant $\tau$ is inversely proportional to $R(p)$ the variation in $R(p)$ can be compensated in XPPRTR by incorporating a routine which:

(a) Divides $\tau$ by 0.4 if the chamber pressure is between 0.0 and 3.0 bar.

(b) Divides $\tau$ by 1.1 if the chamber pressure is between 3.0 and 5.0 bar.

(c) Divides $\tau$ by $(0.33P - 40.23)$ if the chamber pressure exceeds 5.0 bar.

This routine is pointed out in the (comprehensive) XPPRTR listing in Appendix 2.

The pressure transient generated by XPPRTR while incorporating this routine is shown in Figure 11!
3.6.3 Evaluation of the pressure transients.

A comparison of the experimental pressure transients shown in Figures 6 and 7 with the corresponding ones generated by XPRTR and shown in Figures 8 and 9 demonstrates that once the appropriate simulation time constant had been selected XPRTR could simulate the experimental pressure transients quite closely.

The pressure transient generated by XPRTR when correcting the time constant $\tau$ for changes in $R(p)$ (Figure 11) does not
produce a significantly better simulation of the experimental transient (although some improvement is evident). Further improvement could be brought about by a better representation of the variation of $R(p)$ with pressure but the overall improvement in the simulator performance when the time constant is corrected for this variation is not sufficient to warrant the added complexity in the representation of the valve that is brought about by the inclusion of this correction. In view of this it was thought sufficient to consider the value of $\tau$ for both pressurisation and de-pressurisation as constant throughout the pressure range when using XPRTR to simulate a constant volume pressure transient in the manipulator motor chamber.

From the above it becomes evident that the only experimental data needed for the correct simulation of a constant volume pressure transient are the values of the simulation time constants. Once these have been determined from experimental results it becomes possible to produce these transients entirely by computer. The generation of flow characteristic information by computer thus becomes theoretically feasible eliminating the need for look-up table storage of flow characteristic data. The applications of this in the full manipulator axis simulator will be discussed in the following chapter.

One further point that must be made at this stage is that the successful simulation of a constant volume pressure transient by the computer implementation of an exponential function provides us with added insight in the analytical
representation of the valve. This will again be discussed in the following chapter.
4.1 INTRODUCTION.

In the previous chapter the representation of a constant volume pressure transient by an exponential function and the feasibility of simulating such a transient by computer were demonstrated. The significance of this is twofold:

(a) It has been argued (Ref. [64]) that because of the exponential form of the constant volume pressure transient the transfer function of the valve-motor combination when attempting an analysis for stability in small (i.e. when motor displacement is sufficiently small for the motor volume to be considered constant) is of the form:

\[ \frac{T_d}{T} = \frac{1}{1 + Ts}\]

Where \(T_d\) is the driving torque resulting from the pressure \(P\) in the motor chamber while \(T\) is the torque produced by the motor at the supply pressure.

The representation of the valve-motor combination by this transfer function has been the main analytical justification of the use of a first order lead term in the manipulator control algorithm (SUMOS package.) It has been demonstrated (Ref. [64]) that this type of control algorithm is not adequate for the control of the manipulator arm, even for the region about the target position, where the displacement of the arm is sufficiently small for the stability in small analysis to be
applicable. This is because the control strategy used (SUMOS package) does not implement the lead term necessary for adequate system stability correctly. The experience gained in simulating the exponential constant volume pressure transient highlights the shortcomings of the original strategy and suggests an alternative control strategy for stability in the target region.

(b) Even if sufficiently good control is obtained for the target region by an improved implementation of a first order lead term this does not provide a solution for the inadequacies of the original control strategy which are due to our failure to take into account the representation of the motor in the control strategy. It was mentioned in the previous chapter that an exact solution to the motor equation cannot be obtained because of the interdependence of the terms of this equation. Therefore a switching profile that takes into account the performance of the motor can only be produced by a computer simulation of the arm. So far one limitation of this approach has been the reliance on experimentally derived flow characteristic data. The ability to generate a constant volume pressure transient enables us to produce such data by computer thus removing this limitation and rendering feasible the computer generation of suitable switching profiles for the arm.

The suggested control strategy for improved performance around the target region and the computer simulation of the arm without experimental data are described in this chapter.
It was shown in the previous chapter that the constant volume pressure transients in the motor chamber are approximately similar to those produced by exponential rise and exponential decay functions. As the driving torque produced by the motor can be said to be proportional to the pressure in the motor chamber (if friction is ignored) the transfer function of the valve-motor combination for small displacements (where the effect of displacement of the motor on the chamber pressure can be considered negligible and the motor volume can be considered constant for the purpose of analysis) is of the form:

\[
\frac{T_d}{T} = \frac{1}{1 + Ts}
\]

As was mentioned in Ref. [64] this is the main justification for using a first order lead term for the control of the manipulator arm. The main limitation of such a control term is that it does not represent the characteristics of the arm for larger displacements where the rate of change of the motor volume is such that it cannot conceivably be considered constant. The above transfer function however can adequately represent the valve-motor combination when the motor displacement is sufficiently small and can therefore be used as the basis for a control strategy for the arm about the target region to effect a smooth and accurate approach to the target point, minimise or hopelessly eliminate overshoots and eliminate the need for braking action in the vicinity of the target point.
4.2.1 Verification of the 1st order led transfer function.

In view of the above considerations it was decided to check the validity of representing the valve-motor combination by a 1st order led T,F. for small displacements. The most convenient way of accomplishing this is by computer simulation. The existing manipulator simulator (described in Refs. [73],[75]) was modified for this purpose. The valve and motor representation (which in this simulator relies heavily on experimentally derived flow characteristic data) was eliminated completely. It was replaced by a routine that generates exponential pressure transients for the two sides of the motor, similar to that used in XPRTR. The reasoning behind this was that since the first order led transfer function of the valve-motor combination is derived from the exponential constant volume pressure transients, replacing the valve-motor representation by a routine that generates such transients would simulate this transfer function correctly.

The equations that simulate the dynamics of the arm were retained and the original first order lead control algorithm, then resident in the simulator was also retained. (This was done because the control software for the experimental arm—the SAMOS package, described in Ref. [70]—incorporated this control algorithm and the computer simulation could then be readily compared with experimental data obtained from the data acquisition system.)
4.2.2 Modification of the simulation time constant for changing volumes.

It was thought that although the volume of the motor can be considered constant for the purpose of system representation when considering small displacements (thus enabling the representation of the valve-motor combination by a 1st order lag T.F.), a more accurate simulation of this transfer function would be accomplished if the small changes in motor volume that do occur were taken into account in the simulation.

The basis for the calculation of the pressure increment in each successive simulation step is equation (11) in Chapter 3. The parameters needed for the correct implementation of this equation are the value of the current pressure in the motor chamber, the value of the "desired" steady state pressure and the value of the simulation time constant.

We know from the analysis carried out in Chapter 3 that the value of the simulation time constant \( \tau \) is proportional to the volume of the motor chamber under consideration. A routine that multiplies the reference simulation time constant (derived from experimental results with the aid of XPRTR) with a volume-dependent modifying term was added to the basic pressure transient simulation routine. This routine is based on the equation:

\[
\tau = \tau(\text{ref}) \times \left[ \frac{V(\text{current})}{V(\text{ref})} \right]
\]

where \( \tau(\text{ref}) \) is the reference time constant and \( V(\text{ref}) \) the
volume at which it was obtained.

The addition of this routine means that, rather than use a single exponential constant volume pressure transient the simulator selects the pressure transient that corresponds to the current motor chamber volume in each simulation step and uses it to calculate the pressure increment for this step.

It was thought that this piece-wise simulation of the pressure transient would minimise, to some extent, the inaccuracy inherent in the assumption that the volume remains constant for small displacements of the arm and hence in the representation of the valve-motor combination by a 1st order lag term which is based on this assumption.

The resulting simulator package was designated MANIPR. A listing of MANIPR is given in APPENDIX 3.

4.2.3 Evaluation of the small displacement simulation of the arm.

The performance of the MANIPR package was evaluated by comparing the trajectory of the experimental arm between two points under the control of the SAMOS package and the simulation of this trajectory as produced by MANIPR.

The actual trajectory of the arm, plotted in the phase plane from data acquired by the data acquisition system is shown in Figure 1.
The corresponding trajectory as produced by MANIFR is shown in Figure 2.

Figure 1. Phase-plane plot of experimental arm trajectory.
Figure 2. Phase plane plot of arm trajectory as simulated by MANIPR.
A comparison of the two plots shows that the simulation carried out by MANIPR is not accurate for the greatest part of the trajectory of the arm. This is because the valve-motor representation used in MANIPR is not compatible with the large arm displacement encountered in the early part of the trajectory. The simulation by MANIPR of the small amplitude oscillations of the arm about the target point is visibly more successful even though the state of the arm at the beginnings of these oscillations has not been accurately simulated because of the initial discrepancy between the actual trajectory and its simulation by MANIPR.

This indicates that although the simulation of the arm trajectory produced by MANIPR is not entirely successful the valve-motor representation (1st order lag) used in MANIPR is correct, in principle, for small displacements of the arm.

The manner of implementing this representation in the computer, however, raises the following points:

(a) The exponential pressure transient is simulated not in one but in two motor chambers (as the simulated motor configuration has two sides).

(b) The simulation time constant does not remain constant but is proportional to the chamber volume for each chamber.

(a) indicates that an improved valve-motor representation would not use one 1st order lag term but two different terms in parallel, each term having a separate time constant. (T1s and
(b) indicates that the time constant used in each case is not in fact a constant but is a function of the relevant chamber volume i.e. of the position of the arm.

The transfer function of the valve-motor combination, therefore, could be more appropriately given as:

\[
\frac{T_d}{T} = \frac{1}{1 + f(X)S} \cdot \frac{1}{1 + f(X)S} \cdot \frac{1}{1 + f(X)S}
\]

This can be used as the basis of a first order lead control algorithm in which the value of the time constant and hence of the coefficient of the velocity term is not a constant but a function of the position of the arm. This should provide, if correctly implemented, a suitable strategy for the control of the arm in the region of the target point.
4.3 COMPUTER SIMULATION OF THE ARM FOR LARGE DISPLACEMENTS.

4.3.1 Representation of the valve-motor combination for large displacements.

The inability of the MANIPR simulator to reproduce the complete trajectory of the arm accurately highlights the inadequacy of the constant volume assumption (which forms the basis of the valve-motor representation implemented by MANIPR) for larger arm displacements.

In order to accurately simulate the arm for these larger displacements the valve-motor representation must include the effect of a varying motor chamber volume.

A general representation for the variable volume expansion of gases is given by the Equation of State for gases:

\[ P \cdot V = N \cdot R \cdot (g) \cdot T \]

After partial differentiation and manipulation this gives:

\[ \left( \frac{dP}{P} \right) = \left( \frac{dM}{M} - \frac{dV}{V} \right) \]

If the motor chamber has a constant cross-section motor volume is proportional to the motor displacement \( X \). Hence:

\[ \left( \frac{dP}{P} \right) = \left( \frac{dM}{M} - \frac{dX}{X} \right) \]
in Chapter 3) shows that the mass flowrate through the valve can be expressed as a function of the pressure in the motor chamber.

\[ M = f(P) \] (3i)

Equation (3) indicates that the pressure in the motor chamber is dependent on the mass flowrate into the chamber and on the rate of displacement of the arm.

The rate of displacement of the arm depends on the driving torque exerted by the motor which depends in turn on the pressure in the motor chamber (if friction is ignored for the sake of simplicity):

\[ \frac{T(\text{ref})}{\text{ref}} \times \frac{P}{P(\text{ref})} \times \frac{1}{I} \] (3ii)

Where \( T(\text{ref}) \) is the torque produced by the motor at a reference pressure value \( P(\text{ref}) \) - the supply pressure - and \( I \) is the moment of inertia of the arm.

The motor of the experimental arm actually has a two chamber configuration, therefore the driving torque is a function of the difference between the chamber pressures on the two sides of the motor \( P_1 \) and \( P_2 \). Hence:

\[ \frac{T(\text{ref})}{\text{ref}} \times \frac{(P_1 - P_2)}{P(\text{ref})} \times \frac{1}{I} \] (3iii)

This situation results when the arm is represented in block diagram form (Chapter 1, Figure 2) in a number of internal
feedback loops in the valve-motor segment of the diagram.

Equations (3), (3i) and (3ii) are simultaneous non-linear differential equations. A set of equations of this type is not generally amenable to an explicit solution which could be used to represent the valve-motor combination (and form the basis for a control strategy).

The above equations however are perfectly adequate for the simulation of the arm by a digital computer in which all the variables are updated in each simulation step and used to calculate the values of the variables for the following step. A number of simulator packages that utilise various versions of equations (1), (2) and (3) for the representation of the motor have already been developed (Refs. [73],[75]). These can successfully simulate the complete trajectory of the arm. Their major disadvantage, however, is that they have to rely on look-up tables of experimentally derived flow characteristic data which (a) take up a considerable amount of computer memory and (b) become useless when a different valve is used necessitating further experiments for the derivation and storage of new flow characteristic data.

It was thought that as the derivation of the experimental flow characteristic invariably involves data from a constant volumetric pressure transient and as such transients can now be generated by computer it would be possible to eliminate or, at least, alleviate our dependence on experimental data for the derivation of the flow characteristic information which is used for the simulation of the valve.
4.3.2 Computer generation of flow characteristic data.

The method of obtaining the flow characteristic of the valve from an experimental constant volume pressure transient was described in Chapter 3.

Essentially it entails the following steps:

(a) Extracting the value of $(dP/dt)$ for various values of the pressure from the transient.

(b) Multiplying this by the constant $[V]/[R(s)*T]$ to obtain the value of $(dM/dt)$.

(c) Associating this value with the corresponding value of the pressure.

Let us now consider the equation which forms the basis for the simulation of the constant volume pressure transient (equation (11), Chapter 3):

$$ P(n) = P(n-1) + (P_c - P(n-1)) \times (1 - e^{-\delta t/\tau}) $$

This gives:

$$ P(n) - P(n-1) = (P_c - P(n-1)) \times (1 - e^{-\delta t/\tau}) $$

$[P(n) - P(n-1)]$ is the simulated constant volume pressure increment occurring in the simulated time interval $\delta t$. It can therefore be said to simulate $(dP/dt)$ for pressure values between $P = P(n-1)$ and $P = P(n)$.
In view of this equation (5) could be used as the basis for a routine that calculates the flow characteristic of the valve without using experimental pressure transient data other than for the determination of the reference simulation time constant \( \tau \). This would, however, necessitate the storage of the generated data in lookup-table form thus occupying a certain amount of computer memory. In view of this it was decided to devise a routine that would generate the necessary flow characteristic data in the course of the simulation itself.

In the actual simulation the motor chamber volume does not remain constant. In the previous section, however, it was shown that equation (4) can be correctly implemented for varying volumes by modifying the reference simulation time constant \( \tau \) with the aid of a volume-dependent multiplier. (This is, in effect, a piece-wise selection of parts of different constant volume pressure transients corresponding to the actual volume and pressure values in each simulation step.)

Using the above reasoning we can argue that equation (5) can be correctly implemented for varying volumes in the same way.

If the value of \( (dP/dt) \) obtained from equation (5) in each simulation step is multiplied by \( [V]/[R(k)] \times \tau \) where \( V \) is the value of the current volume a value of \( (dM/dt) \) can be obtained. Using this approach we are generating a value of \( (dM/dt) \) by effectively taking a constant volume pressure transient corresponding to the current volume and extracting the gradient \( (dP/dt) \) corresponding to the current pressure. We
then multiply this gradient with the term $\frac{\mu}{\mu(x)T}$ which can be considered a constant for each individual simulation step.

What this amounts to is the implementation of the flow characteristic derivation procedure in a piece-wise fashion. As the implementation of various expressions in a piece-wise manner is quite compatible with computer simulation the procedure described above can be used in a simulator package for the manipulator axis to generate the flow characteristic data representing the valve in each simulation step.

4.3.3 Performance of simulator packages for large displacements.

Once the simulation of the valve has been accomplished (as described in the previous section) the simulation of the motor can be carried out by the computer implementation of equation (1) or equation (3).

A simulator package based on MANIPR and incorporating the above valve representation in conjunction with a motor simulation based on the Equation of State for gases (equation (1)) was designated MANESPR.

A similar simulator package incorporating a motor simulation based on a Total Differential notation (equation (3)) was designated MANTDPR.

Both packages produced identical simulations of the
manipulator trajectory. Listings of the relevant sections of these packages are given in APPENDIX 4.

A plot of the trajectory of the arm (in the phase plane) as simulated by MANTDPFR is shown in Figure 3. A plot of the position transient of the same trajectory is shown in Figure 4. A plot of the corresponding pressure transients for the two sides of the motor and of the resultant driving pressure is shown in Figure 5.
Figure 3. Phase plane plot of arm trajectory simulated by MANTDPF.
Figure 4. Position-time plot of arm trajectory simulated by MANTDPR.
Figure 5. Pressure transients during arm trajectory as simulated by MANTDPRI.

PRESSURE
[BAR]

TIME-S

P1
P2
P1-P2
Comparing the actual trajectory of the experimental arm
with the simulated trajectory generated by MANTDPR in the phase
plane (Figures 1 and 3) shows that the simulation performed by
MANTDPR is accurate for the whole of the trajectory. This
validates the valve and motor simulation used in MANTDPR and
thus generates a basis for the simulation of the manipulator
with minimal reliance on, and no storage of experimental data
(experimental data are only used in the evaluation of the
reference simulation time constant).

4.4 IMPLICATIONS OF SUCCESSFUL LARGE DISPLACEMENT
SIMULATION.

An examination of Figures 3, 4 and 5 shows that the first
order lead control strategy used (both in the experimental arm
and in the simulator) to control the arm cannot do so ade-
ately and results in a number of oscillations about the
target point. The limitations of this type of control term which
uses an oversimplified linear switching profile have been
discussed in previous chapters. Furthermore, the inability to
solve the equations representing the valve-motor combination
and hence derive a suitable switching profile analytically was
discussed earlier in this chapter.

The ability to simulate the manipulator arm by computer
without the need for storage of flow characteristic data opens
up the possibility of the generation of suitable switching
profiles for the manipulator by computer thus side-stepping the
problem posed by our inability to derive suitable switching
profiles analytically. The generation of switching profiles by
computer is discussed in the following chapter.
5.1 INTRODUCTION.

The limitations of using a linear switching profile for the control of the manipulator arm were discussed in previous chapters. In general, when a system contains a non-linearity that is represented as a switching line in the phase plane, this results in changes of sign in the output of the system occurring at the points at which the switching line intersects the phase plane plot of the rest of the system. In practical terms this means that at those points of intersection the non-linearity which is represented by the switching line brings about a series of switch-overs in the system which are invariably seen as oscillations in the time domain. Clearly, the way to minimise these oscillations is to reduce the number of points of intersection of the switching line and the phase plane plot of the rest of the system.

As in our case the switching line does not represent a system non-linearity but is generated by the control algorithm, the points of intersection mentioned above could be kept to a minimum if the switching profile were to match the performance of the system after a single switch-over operation that takes place after the first intersection of the phase plane plot of the performance of the system and the switching profile generated by the control algorithm.

As was explained in previous chapters the performance of the arm cannot be represented analytically. Therefore a
mathematical expression for such a switching profile cannot be obtained. In Chapter 4, however, it was demonstrated that the performance of the arm can be simulated quite successfully on a digital computer. It was reasoned therefore that the required switching profile could be obtained by simulating a single switch-over operation with the aid of a suitable simulator programme and storing the resultant phase-plane data in a look-up table in lieu of an analytical switching function.

A slightly different approach of using a computer simulation of the arm for the generation of a suitable switching profile was also attempted. This consisted of simulating the performance of the arm in reverse time, i.e., starting at (or rather near) the steady state at the target point and proceeding backwards in the time domain. The reason for using this approach was that it was thought at one point that it would bring about a control strategy that would be easier to implement on-line than one involving the use of switching profile data generated by forward simulation.

It was found that both approaches were successful in generating the required switching profiles. It was also found, however, that both approaches present considerable limitations as far as their generalised use in a control strategy is concerned.

The generation of switching profiles using the above methods and the problems which were encountered in obtaining them are discussed in this chapter.
5.2 SIMPLE SWITCH-OVER SIMULATION IN FORWARD TIME.

5.2.1 Selection of the mode of switch-over.

In general the intersection of the phase plane plot of a system with a switchings line (which is due to the existence of particular types of non-linearity in the system) brings about a reversal of the sign of the output of the system component associated with the switchings line.

In the case of the manipulator arm the components associated with switchings operations are the valves controlling the supply of air to the motor chambers on the two sides of the arm. These can be set to "HIGH", connecting the relevant motor chamber to the air supply (which is maintained at constant pressure), or "LOW", venting the motor chamber to atmosphere.

The presence of two separate components that can be subjected to switchings would cause considerable problems if the switchings profile were to be derived analytically. However, as the procedure used to obtain the switchings profile merely involves the simulation of the response of the arm to a single switch-over operation and the storage of the resultant phase-plane data it is sufficient, in this case, to be consistent about the mode of switch-over used to obtain the switchings profile and the mode of switchings to be used in the implementation of this profile.

At the start of the trajectory of the arm both valves are set to "HIGH" and hence both motor chambers are connected with
the compressed air supply. Then, in order to accelerate the arm
the appropriate motor chamber is vented to atmosphere while the
other chamber is retained at supply pressure.

For the sake of a consistent notation, let us assume that in
this case the velocity and acceleration of the arm are positive
and let us designate the motor chamber which is kept at supply
pressure Side 1 of the motor and the motor chamber which is
vented to atmosphere Side 2. Therefore, in order to accelerate
the arm, the switching state of the valves through which sides 1
and 2 of the motor are supplied is 1-0.

It can be seen that once the arm is in motion it can be
decelerated by either of the following switching signals:

a) 1-1, and b) 0-1.

It can be argued that switching signal (b) can result in a
greater rate of deceleration of the arm than signal (a) as the
complete reversal of the valve connections for sides 1 and two
of the motor results in greater pressure differences between
the two motor chambers and hence a higher decelerating
torque. The trouble with using this switching mode is that after
the arm has decelerated to zero velocity a considerable (and
increasing) decelerating torque is still exerted on the arm, as
in the steady state Side 1 of the motor tends to atmospheric
pressure while Side 2 of the motor tends to the supply
pressure. This will eventually cause a reversal of the
direction of motion of the arm and this reversal can only be
avoided by the application of braking action. On the other
hand, using switching signal (a) results in the eventual stop of the arm without a reversal of its direction of motion as the pressures in the two motor chambers tend to the same value (supply) in the steady state and hence the decelerating torque tends to zero.

As one of the reasons for seeking an improved control algorithm was the wish to eliminate the need for braking action (since this cannot be implemented with absolute reliability as was demonstrated in Chapter 2) switching signal (a) was chosen for the derivation of the switching profile.

5.2.2 Simulation of arm performance after simple switch-over.

Before a suitable simulation of the behaviour of the arm under the control of a simple switch-over command was carried out data was obtained from the data acquisition system that could be used to verify the validity of the simulation. The experimental arm was subjected to a sequence of 1-1,1-0,1-1 signals by means of its controlling processor and the data acquisition system was used to log and plot the trajectories (phase-plane) and pressure transient data. The switch-over was carried out at an arbitrary point in time. The resultant phase-plane trajectory is shown in Figure 1. The pressure transients for sides 1 and 2 of the motor are shown in Figure 2.
Figure 1. Phase plane trajectory for simple 1-1, 1-0, 1-1 switch-over from the experimental arm.
Figure 2. Pressure transients for sides 1 and 2 of the motor for simple 1-1,1-0,1-1 switch over from the experimental arm.
A simulation of this switch-over was carried out as follows:

The control algorithm was removed from the MANTDPR simulator and replaced with a routine that simply implemented the switch-over sequence which was carried out in the experimental test. The 1-0 to 1-1 switch-over was implemented when the arm reached a certain position. This position could be specified by the operator. The resulting package was designated VMSTDPR.

A listing of the relevant parts of VMSTDPR is given in APPENDIX 5.

The accuracy of the simulation carried out by VMSTDPR was checked as follows:

The starting position of the arm for the simulator was set to the same value as that used in the data acquisition test. The switch-over position of the simulator was then varied until the same finish point was obtained from the simulator as that which was reached by the arm in the aforementioned test. A phase plane plot of this trajectory was then produced by VMSTDPR together with the corresponding position transient and pressure transients for sides 1 and 2 of the motor. These are shown in Figures 3, 4 and 5 respectively.
Figure 3. Phase plane trajectory for simulated 1-1,1-0,1-1
switch-over produced by VMSTDPR.
Figure 4. Position-time plot for simulated 1-1,1-0,1-1 switch-over produced by VMSTDPR.
Figure 5. Pressure transients of sides 1 and 2 of the motor for simulated 1-1,1-0,1-1 switch-over trajectory produced by VMSTIFPR.
A comparison between the experimental and simulated phase plane trajectories and pressure transients shows that UMSTDPR carries out an accurate simulation of the trajectory of the arm under the control of a simple switch-over sequence when the switch-over point is known. Such a simulation can therefore be used for the derivation of a switching profile which will provide the basis for a suitable control algorithm for the arm.

An examination of the pressure transients (simulated or experimental) occurring in sides 1 and 2 of the motor while the arm is controlled by the simple switch-over sequence described above reveals the following features:

(i) Although the pressure on side 1 of the motor is nominally supposed to remain the same as the supply pressure there is in fact a pressure drop below supply pressure and a subsequent pressure rise to supply pressure as the arm accelerates and decelerates respectively.

(ii) On side 2 of the motor the pressure is nominally supposed to rise to the value of the supply pressure after switch-over. It does in fact exceed this pressure and drops to supply pressure as the arm decelerates.

Both (i) and (ii) are due to the fact that the mass flowrate across the valves connecting sides 1 and 2 of the motor to supply or atmosphere (this mass flowrate is governed by the pressure in each motor chamber) cannot sufficiently counteract the effect of the change in volume of the motor chambers (increase for side 1, decrease for side 2) which is due
to the motion of the arm. The resulting expansion in side 1 and compression in side 2 of the motor accounts for (i) and (ii).

Because of these features of the pressure transients the arm begins to decelerate at an earlier point in time (after switch-over) and the decelerating torque (which depends on the pressure difference between the two sides of the motor) is greater than would have been the case if the above expansion and compression effects did not take place.

5.2.3 Examination of an alternative switching sequence.

It was thought, in view of the actual performance of the arm under switch-over mode (a), that a comparison of this performance and that obtained with switch-over mode (b) (which, because of the reversal of both valve settings, results in maximum deceleration) would be of interest.

In order to accomplish this the control sequence in VMSTDPR was modified so as to implement a 1-0 to 0-1 switch-over when the switch-over point was reached. Braking action was simulated by setting acceleration and velocity to zero when the target position (which is specified by the programmer) was attained. In order for the pressures in both simulations to have the same steady state values a further switch-over to 1-1 was implemented when the arm reached the designated target position and the arm stopped by braking. Start, switch-over and target positions were given the same values as those which were set (or, in the case of the target position, attained) in VMSTDPR.
The resulting simulator package was designated VMSTDPRB. A listing of the relevant parts of this package is given in APPENDIX 5. The phase plane trajectory, position transient and pressure transients produced by this simulator are shown in Figures 6, 7 and 8 respectively.
Figure 6. Simulated phase plane trajectory for 1-0, 0-1 switch-over as produced by VMSTDRPB.
Figure 7. Simulated position-time plot for 1-0,0-1 switch-over as produced by VMSTDPRB.
Figure 8. Pressure transients of sides 1 and 2 of the motor for simulated 1-0, 0-1 switch-over trajectory produced by VMSTDPRB.
A comparison of the phase plane trajectories and position-time plots produced by the simulation of the two switching sequences (Figures 3 and 6 and Figures 4 and 7 respectively) shows that although the reversal of both valve settings on switch-over (switch-over mode (b)) does result in a greater rate of deceleration of the arm the sain in deceleration is not as great as was initially supposed. This is due to the increase in decelerating torque which occurs because of expansion and compression effects when switch-over mode (a) is implemented. The above comparison provides further Justification for the simulation of the switch-over mode (a) in order to obtain switching profile data as the sain in deceleration torque which is brought about by switch-over mode (b) is not sufficient to warrant the added complexity of the braking action necessary with this mode.

5.2.4 Limitations of the simple switch-over simulation for control purposes.

It was shown, earlier in this chapter, that the VMSTDPR simulator can simulate accurately the trajectory of the arm in the phase plane between a starting point and a target point under the control of a simple switch-over sequence if the switch-over point is known. Clearly, simulated phase plane trajectory data occurring after the switch-over can be used for the derivation of a switching profile. The main drawback in this approach is the need to know the switch-over point before such a profile can be generated. This would necessitate a considerable number of runs (off-line) of the VMSTDPR simulator for different start and switch-over points and storage of
switch-over point co-ordinates corresponding to pairs of start and target point co-ordinates in a look-up table.

A set of phase plane trajectories and pressure transients produced by VMSTDPR and corresponding to various start and target positions (the switch-over positions had to be determined by trial and error) are shown in Figures 9, 10 and 11 respectively.
Figure 9. Set of simulated phase plane plots produced by VMSTDFR.
Note! The starting position used in the plots in Figures 10 and 11 is 0.05*STR where STR signifies the displacement corresponding to the swept volume of the motor.

Figure 10. Set of simulated pressure transients for side 1 of the motor produced by VMSTDPR. (Same starting position, different switch-over and target positions.)
Figure 11. Set of simulated pressure transients for side 2 of the motor produced by VMSTDBR. (Same starting position; different switch-over and target positions.)
This results in a considerable computing time and storage overhead and constitutes the major limitation of using this approach for the purpose of obtaining switching profile data for use in the control of the arm.

Another limitation of this approach becomes apparent when the simulated pressure transients shown in Figure 5 are compared with the corresponding experimental ones shown in Figure 2. In the experimental transients a reversal of the sign of the pressure difference between the two sides of the motor (and hence of the driving torque) occurs when the pressures are approaching the steady state. This sign reversal is not simulated in the pressure transients produced by VMSTDPR. The reason for this is that the dynamic friction representation used in VMSTDPR is over-simplified and does not represent the variation of dynamic friction with velocity accurately at low arm velocities. (An analysis of dynamic friction measurements and the derivation of the current representation of dynamic friction is given in APPENDIX 6.) Essentially, the dynamic friction is assumed to be constant throughout the velocity range. Although this representation is reasonably accurate for the greatest part of the velocity range it results in friction terms of excessive magnitude at low velocities. (As is described in APPENDIX 6, lack of sufficiently accurate data precludes a more accurate representation of the dynamic friction at low velocities.) As a result of this the changes in the sign of the driving torque and the resultant oscillations around the target position which actually take place in the motion of the experimental arm are not simulated accurately by VMSTDPR.
An improved friction representation was attempted by incorporating a routine in VMSTDPR which set the dynamic friction term to zero when the arm velocity became smaller than 0.03 rad/sec. The resultant simulated position transient and an expanded plot of the pressure transients when the pressures approach the steady state are shown in Figures 12 and 13.
Figure 12. Simulated position-time plot (final part) produced by VMSTDPR with improved dynamic friction representation.
Figure 13. Simulated pressure transients (final part) of sides 1 and 2 of the motor produced by VMSTDPR with improved dynamic friction representation.
From Figures 12 and 13 it can be seen that although some oscillations about the target point are now present in the simulated trajectory they are more exaggerated than those actually occurring in the experimental arm. We can therefore conclude that although the friction representation used in this simulation constitutes a step in the right direction it is some distance from being an accurate representation of the dynamic friction-velocity relationship which actually occurs in the experimental arm. Hence any simulation of the arm which uses this representation and any switching profile derived from such a simulation is likely to be inadequate for the control of the arm about the target point.

However, the basis for a strategy which is suitable for the control of the arm precisely in this region of its trajectory has been already suggested in Chapter 4. Therefore, the limitation posed upon the suggested method for obtaining a switching profile by the inadequate representation of the dynamic friction can be easily overcome.
5.3 SIMPLE SWITCH-OVER SIMULATION IN REVERSE TIME.

5.3.1 Reasons for carrying out a simulation in reverse time.

It was seen that the major limitation present in using a computer simulation of a simple switch-over to derive an appropriate switching profile for the arm was the need for the switch-over point to be known before a successful simulation could be carried out.

It was thought that this limitation could be overcome if the simulation was started at the end of the trajectory and carried out in reverse time, thus proceeding "towards" the switch-over point. In this manner the actual trajectory of the arm as it accelerates would be monitored as would the one simulated in reverse time (starting at the finish point). The intersection of the two trajectories would indicate that the switch-over point had been reached (as the actual arm trajectory represents the behaviour of the arm before switch-over, while the simulated trajectory represents when simulating in reverse time - the behaviour of the arm after switch-over). In this manner the switch-over point could be derived by the control software rather than accessed from memory. The switching profile generated by the reverse time simulation could then be used as the basis for feedback control of the arm after switch-over. (This approach is correct in principle but presents considerable difficulties in its actual implementation. These will be discussed in the following chapter.)
Another reason for carrying out a simulation in reverse time was the assumption that as the arm settles to a steady state at the end of its trajectory the final values of the variables used in describing the trajectory (especially the pressures) could be specified with relative ease thus making the specification of the initial conditions for the simulation in reverse time a relatively simple matter. This assumption, however, implies an over-simplified representation of the arm which would lead to ambiguities in the simulation of the arm trajectory. The problems posed by such ambiguities are discussed later in this chapter.

5.3.2 Method of carrying out a simple switch-over simulation in reverse time.

The procedure for carrying out a simulation of the arm in forward time essentially consists of updating the values of the variables that describe the behaviour of the arm (according to a set of simple expressions) at discreet, but small, time intervals. In order to carry out a simulation of the arm in REVERSE time this procedure is retained but its direction is reversed. In order to accomplish this reversal the following modifications had to be incorporated in the simple switch-over simulator VMSTDPR:

The sign of the time interval between simulation steps $\Delta t$ was changed in all operations involving $\Delta t$. (These operations carry out the calculation of the velocity and position increments for each simulated time interval.)
The piece-wise simulation of the flow characteristic data (mass increment) was not modified as it was reasoned that the magnitude of the constant volume pressure increment (on which the flow characteristic data generation is based) does not change if the constant volume pressure transient in question is examined in forward or reverse time. The sign of the calculated mass increment, however, does change when examined in reverse time and this was catered for by carrying out a sign change of the mass increment for each simulation step DELM (dM/dt) in all operations involving DELM. (These operations carry out the updating of the value of the mass in each motor chamber and hence calculate the pressure increment.) All other parts of the programme were retained the same as in VMSTDPR.

The resultant package was designated REVVMS. A listing of REVVMS is given in APPENDIX 5.

5.3.3 Difficulties encountered in carrying out a simple switch-over simulation in reverse time.

It was stated previously that one of the reasons for which simulation in reverse time was attempted was the supposed ease with which the initial conditions for such a simulation could be specified.

The initial conditions which must be supplied for any simulation of the arm (whether in forward or reverse time) are the values of the chamber pressures in sides 1 and 2 of the motor and the initial values of acceleration, velocity and position of the arm. In the case of a simulation in reverse time
it was initially considered adequate to set as initial conditions the steady state values of the above variables which occur at the end of the trajectory of the arm i.e after the arm has stopped. These values are zero in the case of acceleration, velocity and position and the initial pressure values are equal to the supply pressure.

One problem that arises from initialising a reverse time computer simulation in this manner is the inability to carry out the piece-wise generation of flow characteristic data. The generation of this data requires the calculation in each simulation step of a constant volume pressure increment (dP/dt). This calculation is carried out by implementing Equation (5) in Chapter 4. In forward time simulation this equation calculates the pressure increment by utilising the difference between the current value of the motor chamber pressure and a relevant steady-state value that this pressure should eventually attain. (In the case of the simulation of a 1-0 to 1-1 switch-over this value is equal to the supply pressure.)

In reverse time simulation this "eventual" steady-state value is the value of the pressure set at the beginning of the simulation. Although this value is given no values of the "current" pressure corresponding to other points in time are known. If the calculation of (dP/dt) is to be carried out using the method described above these values have to be assumed.

It was thought that the simulator could be suitably initialised by setting the values of the motor chamber
Pressures at values slightly different from that of the supply pressure. It was quickly realised that the selection of these initialisation values was not as simple a matter as was originally thought. There were two reasons for this:

(a) REVVMS was designed to carry out the same simulation as VMSTDPR, only in reverse. It was shown previously that VMSTDPR (because of limitations in the representation of dynamic friction at low velocities) does not simulate the behaviour of the arm accurately at the end of its trajectory. Because of this, if REVVMS were to carry out a correct simulation of the trajectory of the arm the initialisation values for the simulation should be selected from a part of the trajectory for which the representation of the dynamic friction is adequate i.e. a point in the trajectory which occurs before the final small overshoots around the target point have taken place.

Such a point is the point at which the arm decelerates to zero velocity for the first time. At this point, as was determined by obtaining a data dump from a simulation in forward time carried out by VMSTDPR, the acceleration of the arm is sufficiently small to be taken as zero and the pressures can be set equal to those obtained from the forward time simulation. (The forward time simulation described is that illustrated in Figures 3, 4 and 5 but with the improved dynamic friction representation illustrated in Figures 12 and 13.)

(b) The difficulty outlined in (a) is not particularly significant as it could be eliminated if a better friction representation were available. The main shortcomings of the
procedure described above initialises the simulator with a single set of values. Clearly a simulation in reverse time using these initialisation values will result in a SINGLE trajectory. As we can see from Figure 10, however, a number of trajectories, each corresponding to different starting and switch-over points, converge at each target point. Clearly, different values of the initialisation pressures (representing the different pressure variations that correspond to different trajectories) will have to be set if REVVMS is to simulate those trajectories accurately in reverse time. (These values can be obtained as described in (a) above by running simulations in forward time for the appropriate start and switch-over positions.)

The above considerations indicate that switching profile generation in reverse time displays limitations of the same kind as those encountered in forward time, i.e., a considerable computing overhead for the generation and storage of initialisation values (necessary for the specification of particular trajectories) and an inadequate switching profile in the region around the target point.

REVVMS was run using the initialisation values mentioned in (a) above. The reverse time phase plane trajectory, position transient and pressure transients are shown in Figures 14, 15, and 16 respectively.
Figure 14. Simulated phase-plane trajectory produced by REVUMS.
Figure 15. Position-time plot simulated in reverse time by REVUNS.
Figure 16. Pressure transients for sides 1 and 2 of the motor simulated in reverse time by REVU MS.
It can be seen that the plots produced by reverse time
simulation agree with the corresponding ones produced by
forward time simulation up to a point but seriously diverge
from them after this point. This happens because REVUMS
continues to carry out the simulation in reverse time past the
time at which switch-over occurs. As the valve settings and
all operational parameters change at switch-over a reverse time
simulation based on post-switch-over conditions is no longer
valid, hence the divergence. Specifically, it was shown by
examining a data dump of the values produced by REVUMS, if the
reverse time simulation is allowed to continue for a
sufficiently long time sufficient mass flow takes place out of
the motor chambers for the mass value in the chambers to become
negative, making nonsense of further simulation.

As the switch-over point is not known in a reverse time
simulation the only way of minimising the invalidity of this
simulation is to stop it when the values which it generates for
any of the manipulator variables become obviously
unrealistic. This can be accomplished by stopping the simulator
when the value of the mass in any one of the motor chambers
becomes zero.

An improved version of REVUMS incorporates a routine that
does this. This routine is listed in APPENDIX 5. The phase plane
trajectory, position transient and pressure transients produced
by this package are shown in Figures 17, 18 and 19 respectively.

It can be seen from these figures that the divergence
between forward and reverse time simulations has been
considerably reduced. This divergence cannot be eliminated altogether as the point at which reverse time simulation ceases to be realistic (i.e., the switch-over point) is not known. However, the divergence now present is sufficiently small for the switch-over point to be determined with the aid of reverse time simulation data. The procedure for doing this and its evaluation will be given in the next chapter.
Figure 17. Phase plane trajectory produced by REVUMS with improved termination.
Figure 18. Position-time plot of trajectory simulated in reverse time by REVUMS with improved termination.
Figure 19. Pressure transients for sides 1 and 2 of the motor simulated in reverse time by REVUMS with improved termination.
5.4 EVALUATION OF THE NEED FOR STORAGE OF SWITCH-OVER OR INITIALISATION VALUES FOR THE GENERATION OF A SWITCHING PROFILE.

It was pointed out that the major limitation in using forward or reverse time simulation to obtain switching profiles for the arm was the need to store values of the switch-over point or of the initialisation pressures for all (or at least a considerable number of) combinations of start and target positions.

It was decided to investigate the feasibility of obtaining a mathematical relationship between the switch-over point (or the initialisation pressures) for each trajectory of the arm and the combination of start and target points for this trajectory, so as to eliminate this limitation. To this end the forward simulator VMSTDPR was used to simulate trajectories corresponding to various start and switch-over positions and the values of start, switch-over and finish positions as well as the values of the pressures in the two sides of the motor when the arm first decelerated to zero velocity were recorded for all these trajectories. (Tables of these values are shown in APPENDIX 7.) Plots of switch-over position against target position and of pressure difference between the pressure in each chamber (at zero arm velocity) and the supply pressure against target position were made for each starting position. These are shown in Figures 20-21 and 22 respectively.
Figure 20. Plot of switch-over positions vs. target positions.
Figure 21. Plot of $P_{\text{Supply}} - P_{\text{Vel}=0}$ vs. target position for side 1 of the motor.

The plot shows three lines labeled 'a', 'b', and 'c', with corresponding equations:

- Line 'a': $y = 0.5(x - 0.25)$
- Line 'b': $y = 0.52(x - 0.25)$
- Line 'c': $y = 0.58(x - 0.38)$

The x-axis represents the target position (x STR), and the y-axis represents the difference in pressure ($P_{\text{Supply}} - P_{\text{Vel}=0}$) in bars.
Figure 22. Plot of $P(\text{Vel}=0) - P(\text{Supply})$ vs. target position for side 2 of the motor.

\begin{align*}
y_a &= 0.122 - \frac{(0.71-x)^2}{1.46} \\
y_b &= 0.154 - \frac{(0.73-x)^2}{0.96} \\
y_c &= 0.121 - \frac{(0.86-x)^2}{1.10}
\end{align*}
From these plots it can be seen that for each starting position the relationship between the switch-over position (or the motor chamber pressure at zero velocity) and the target position is regular and can be represented by linear or parabolic equations (as can be seen in Figures 21 and 22), with very small errors. This situation can be explained qualitatively if we examine Figures 11 and 12 in which pressure transients for the two sides of the motor for trajectories with the same starting position but different target positions are shown. It can be seen that the transients are essentially similar in shape which is to be expected as they portray the same essential sequence of events happening over increasing space and time scales. In view of this any values associated with particular points in this sequence of events (i.e., the switch-over point or the point at which the arm decelerates to zero for the first time) are bound to vary regularly as the trajectory of the arm becomes longer.

So far no relationship has been derived between the coefficients of the equations representing the plots shown in Figures 20, 21 and 22 and the starting position for each plot. Such a relationship would enable the switch-over point (or the pressures when the arm first decelerates to zero velocity) to be expressed in terms of both start and target positions.

It can be argued however, in view of the above qualitative analysis, that such a relationship could be expressed by a fairly simple equation if a sufficient data base (i.e., a large number of simulations) were available for its derivation. The implications of this are evaluated in the following chapter.
CHAPTER 6.

THE USE OF COMPUTER-GENERATED SWITCHING PROFILES FOR THE
CONTROL OF THE ARM: AN EVALUATION OF PROPOSED STRATEGIES.

6.1 INTRODUCTION.

The generation of switching profiles for the control of the arm by simulation of the trajectory of the arm in forward or reverse time was described in the previous chapter. The original intention was to store the switching profile data generated by this simulation (wholly or selectively) and to use it for the control of the arm after switch-over. It was then argued that the switching profile obtained by the simulation of the trajectory of the arm in reverse time could be used for the determination of the switch-over point as well as for the control of the arm after switch-over. A mode of implementing such a strategy is described in this chapter and its limitations are evaluated. An alternative strategy in which the switch-over point is assumed to be known and in which parts of the switching profile are generated more or less on-line by selective use of a forward time simulator is also tentatively evaluated. Finally, a procedure is described for the derivation of the mathematical relationship which relates the initialisation data which is necessary for the correct operation of the simulator packages with the combinations of start and target points for each simulation. This procedure is evaluated from the point of view of computing effort requirement and the possibility of producing a comprehensive self-teaching package that requires a minimum of experimental data for the control of the arm is briefly examined. Further suggestions are made for the inclusion in such a package of a strategy suitable for the control of the arm in the target
region (where the computer-generated switching profiles are of limited use) and for the use of the package in conjunction with co-ordinate generation routines.

6.2 A STRATEGY FOR THE DETERMINATION OF THE SWITCH-OVER POINT WITH THE AID OF THE REVERSE TIME SIMULATOR.

6.2.1 Basic Procedure.

The basic procedure for the determination of the switch-over point with the aid of the reverse time simulator essentially consists as was described in the previous chapter of letting the arm and the reverse time simulator run concurrently and switching the valves over at the point when the two trajectories intersect (or come in the close vicinity of each other, depending on the accuracy of the simulation). The reasoning behind this procedure is that, as the actual trajectory of the arm reflects the arm's pre-switch-over behaviour and the reverse time simulation simulates the arm's post-switch-over behaviour, the point of intersection of the two trajectories will be the point of transition from one condition to the other i.e. the switch-over point.

One problem that becomes immediately apparent when the implementation of this strategy is considered is the difficulty encountered in running the reverse time simulation concurrently with the trajectory of the arm. Even if the real-time scale of the reverse time simulation can be specified the time scale of the arm's trajectory is rather more difficult to determine as it depends upon the overall manipulator velocity and the length
of the manipulator trajectory which in turn are interdependent.

Another problem associated with this strategy is the ambiguity of the concept of the intersection of the actual trajectory of the arm and that generated by its simulation in reverse time.

An intersection of the phase-plane trajectories is ambiguous because of the parametric nature of time in this representation. An intersection of the velocity transients is also ambiguous as it may occur more than once and not necessarily at the switch-over point.

An intersection of the position transients would be suitable if the termination of the reverse time simulation at the switch-over point could be implemented. As this is not the case, however, (see Chapter 5) the position transients may intersect at a position in which the reverse time simulation is no longer valid.

The above considerations highlight the ambiguity inherent in the mode of selection of the switch-over criterion which forms the basis of this strategy. If this ambiguity is to be removed the switch-over criterion must be examined in more detail.

6.2.2 Determination of the switch-over criterion.

If we examine the simple switch-over trajectories produced by simulation in forward time by VMSTDPR, which are shown in
Figure 10 (Chapter 5) we can see that they are not exactly symmetrical about the middle of the trajectory. This is mainly due to the effect of dynamic friction (which opposes the motor torque during acceleration but assists it during deceleration). An examination of the switch-over points corresponding to these trajectories shows that these occur, in general, before the middle of the trajectory. This can be explained (a) by the lack of symmetry of the trajectory and (b) by the fact that deceleration does not begin immediately after the valves are switched over but only after the pressure in side 2 of the motor has built up to the point where it exceeds the pressure in side 1 of the motor.

In view of this there can be no certainty that the arm and the reverse time simulator will attain the switch-over position simultaneously even if they are made to operate in identical time scales. The fact that the reverse time simulator extends, erroneously, beyond the switch-over point complicates the issue even further. An examination of phase plane trajectories produced by forward and reverse time simulations shows that they may produce identical velocity values simultaneously more than once. There is, however, no guarantee that these coincidences of velocity will first occur at the switch-over point. The above analysis indicates that the use of independent criteria of position or velocity coincidence is not sufficient for the determination of the switch-over point. It was decided, therefore, that a combined criterion involving both position and velocity would have to be used.

The first problem that had to be overcome in applying such
A criterion was the inability of the arm and the reverse time simulator to attain the switch-over position simultaneously (even when operating on identical time scales). The only way to accomplish this was by abandoning the concept of running the reverse time simulator simultaneously with the arm, storing the phase plane data which was generated by a complete run of the reverse time simulator in an array in the computer memory and using a search routine to access the necessary data for comparison with position and velocity data fed back from the arm during its trajectory. Using this procedure the switch-over point could be determined as follows:

For any position of the arm the phase plane data generated by the reverse time simulator would be checked. If the position attained by the arm was included in this data the velocity (stored in memory) corresponding to this position would be compared with the actual velocity of the arm. If the arm velocity exceeded the simulated velocity switch-over would be implemented. This switch-over criterion is justified by arguing that if the actual velocity of the arm exceeds that produced by the reverse time simulator for the same position the kinetic energy of the arm will be higher than that which the arm is assumed to possess at this position by the reverse time simulation. Therefore the kinetic energy loss which occurs in the decelerating path which is portrayed by the reverse time simulator is not sufficient to stop the arm at the designated target position. If the switch-over were to be carried out at the point at which the velocity of the arm just exceeds the corresponding one produced by reverse time simulation the kinetic energy excess of the arm would be sufficiently small.
for the arm to stop on the target position.

A simulation of this procedure was carried out by using the reverse time simulator to produce and store a phase plane trajectory in an array in memory and using the forward simulator (in which the switch-over point was not supplied), in conjunction with the above procedure for the determination of the switch-over point, to simulate the actual performance of the arm. Provisions were made for the display of both phase plane trajectories on the same plot. The resulting package was designated REVCON. A listing of REVCON is given in APPENDIX 8. A plot of the phase plane trajectory of the arm produced by REVCON (which includes the phase plane trajectory produced by the reverse time simulation) is shown in Figure 1.
Figure 1. Phase plane trajectories produced by REVCON.
An examination of the simulated trajectory of the arm which is shown in Figure 1 indicates that the valves are switched over too early. The reason for this can be understood if we re-consider the work carried out with the aim of providing a correct termination of the reverse time simulator which was described in the previous chapter. It was argued that if the reverse time simulation is continued beyond the switch-over point it will cease to be valid. It was also pointed out that as the switch-over point is not known this situation cannot be completely remedied and that we cannot avoid some divergence of the path generated by the reverse time simulator from the actual manipulator path at positions corresponding to parts of the trajectory that occur before switch-over. It can be seen, therefore, that the inability to terminate the reverse time simulator correctly affects the performance of a combined position-velocity criterion as well as that of the position coincidence criterion discussed previously.

This divergence was illustrated by by-passing the switch-over point determination routine in REVCON and supplying the appropriate switch-over point (for the target point in use) externally to the forward time simulator. The resulting phase plane trajectories produced by equivalent simulations in forward and reverse time are shown in Figure 2.
Figure 2. Equivalent phase plane trajectories produced by forward and reverse time simulation.
From Figure 2 it can be seen that the divergence between the trajectories produced by forward and reverse time simulation becomes significant only for parts of the trajectory occurring before switch-over, that is, as the switch-over point is approached (in forward time) the trajectories converge. This convergence was used in REVCONE as a criterion for the determination of the switch-over point which is not affected by the incorrect termination of the reverse time simulator. This was done as follows: The search routine that compares the position of the arm to the phase plane data (produced by reverse time simulation) stored in memory was retained. When the arm position matched one stored in memory the actual and stored velocities were again compared but this time switch-over was only implemented when the difference between the two velocities was smaller than a specified percentage (0.03) of the simulated velocity of the arm.

The relevant parts of this version of REVCONE are listed in APPENDIX 8. The simulated phase plane trajectories (in forward and reverse time) produced by REVCONE using the improved switch-over criterion are shown in Figure 3.
Figure 3. Simulated phase plane trajectories produced by REVCON using a convergence criterion for switch-over.

VELOCITY-POSITION

VEL-RD/S

0

.79

1.57

0

-3

POSITION--RD

GVEL = .25
6.2.3 Evaluation of the mode of determination of the switch-over point.

We can see from Figure 3 that the phase plane trajectory produced by the forward time simulator matches that produced by the reverse time simulator quite closely after switch-over. This indicates that the criterion which was used to determine the switch-over point (i.e., the assumed convergence, at and after switch-over, of the trajectory of the arm and its simulation in reverse time) is valid in principle. The above procedure, however, incorporates a number of features that render it less than ideally suited to the on-line control of a manipulator arm. These features are the following:

(a) The need to initialise the reverse time simulator with the correct pressure values that correspond to the combination of start and target positions under consideration.

(b) The need to store the phase plane data generated by the reverse time simulator in order to be able to determine the switch-over point.

(c) The need to carry out a repeated search through this data in order to determine the switch-over point.

In the previous chapter it was argued that an equation connecting the initialisation pressures for the reverse time simulator with the start and target positions of the trajectory being simulated could be derived. Therefore point (a) becomes relatively unimportant if one is prepared to carry out the
relatively large amount of computation necessary for the derivation of such an equation, as this computation can be carried out off-line.

Point (b) is also relatively unimportant as the phase plane data which is generated by the reverse time simulator has to be stored in any case to provide the switching profile that will be used for the control of the arm after switch-over. The memory requirement for this data can be reduced by storing in memory only the phase plane data taken at relatively large time intervals in the course of the reverse time simulation as this data should be sufficient for the generation of an adequate switching profile. In this case the accuracy of the method for determining the switch-over point is retained by storing in full detail the phase plane data generated at the end of the reverse time simulation as earlier data (simulating the arm after switch-over) is not necessary for this purpose.

Points (a) and (b) pose a considerable computing and some storage requirement but cannot be said to have a directly adverse effect as far as the ability of this method to control the manipulator arm on-line is concerned. However, the search through the data generated by the reverse time simulator (in order to determine the switch-over point) has to be carried out on-line. This search increases the amount of on-line computation necessary for the control of the arm considerably and constitutes the major disadvantage of this method as far as on-line control of the arm is concerned.
6.3 Tentative Evaluation of Control Strategies That Utilise
A Switching Profile Generated by the Forward Time Simulator.

The search routine described above, which is necessary for the determination of the switch-over point when using phase plane data obtained from a simulation in reverse time and which has to be carried out on-line can be avoided if a switching profile based on phase plane data generated by a simulation in forward time is used for the control of the arm.

The only data needed for a successful simulation of the trajectory of the arm in forward time is the position of the switch-over point corresponding to the selected combination of start and target points. In this case too, an equation relating the position of the switch-over point and the start and target positions of any trajectory can be derived if sufficient off-line computation is carried out. (The procedure is outlined in Chapter 5.)

Therefore, a strategy utilising a forward time simulation for the generation of the switching profile poses a comparable (off-line) computation requirement to the strategy described in the previous section (which uses reverse time simulation) and eliminates, as the switch-over point is known, the need for an on-line search routine which constitutes the main drawback of the strategy which uses the reverse time simulator.

The elimination of the on-line search routine renders a strategy which uses a simulation in forward time for the generation of phase plane (switching profile) data preferable
to one which uses a simulation in reverse time for the same purpose (and to determine the switch-over point).

We have so far considered strategies in which the switching profile data (obtained by forward or reverse time simulation) which will eventually be used for the control of the arm is generated off-line and stored in memory. It has already been suggested (in the evaluation of the strategy using the reverse time simulator) that a saving in memory space may be achieved by saving switching profile data that are generated by the simulator at relatively large time intervals. This saving can be obtained regardless of the type of simulator which is used to generate the data.

In the case of the strategy (described in the previous section) which utilises the reverse time simulator, the off-line generation of the switching profile data and their subsequent storage in memory cannot be avoided as the data is needed for the determination of the switch-over point.

In the case of a strategy that uses data obtained from a forward time simulation it might be possible to arrange for the simulation to proceed at a comparable time scale as that of the actual trajectory of the arm anticipating slightly the progress of the arm. After switch-over, the simulator could be used to generate a switching profile that slightly precedes the path of the arm. This profile could then be used for the control of the arm. In this case simulator data which refers to a part of the trajectory which has been already traversed by the arm need no longer be stored in memory. Such a strategy, therefore, could use
a limited amount of computer memory repeatedly for the storage of switching profile data which would be generated more or less on-line.

The savings in computer memory that is brought about by the virtual elimination of the need to store switching profile data may be offset by:

(a) The problems that such a strategy poses as regards the co-ordination of the progress of the manipulator and the generation of switching profile data by a forward time simulator.

(b) The fact that the on-line simulation of the arm may be more time-consuming than the accessing of pre-computed switching profile data from memory.

Because of the above considerations, the choice between on-line or off-line generation of the switching profile will depend on the relative merits of the two approaches. The savings in computer memory which is a feature of on-line switching profile generation will have to be evaluated in the light of the memory requirement for the storage of pre-computed switching profile data. Similarly, the on-line computation time which is required for the generation of parts of a switching profile will have to be compared to the time required for the retrieval of the pre-computed switching profile from memory.

The above comparisons will have to take into consideration the properties (memory capacity and instruction execution
times) of the computer or microprocessor system actually used for the control of the arm.

(If the off-line generation of the switching profile is selected a strategy based on the use of forward time simulation is preferable to one using a reverse time simulation for the reasons described previously.)

6.4 SUGGESTIONS FOR AN INTEGRATED CONTROL SCHEME BASED ON THE GENERATION OF SWITCHING PROFILES BY COMPUTER SIMULATION.

A number of suggested strategies for the control of a one degree-of-freedom arm by using switching profiles which are generated by a simulation of the trajectory of the arm were described to a greater or lesser extent in the preceding sections of this chapter. These strategies pre-suppose the capability of carrying out the simulation of a trajectory of the arm (under the control of a simple switch-over sequence) between any start and target points. A scheme for the acquisition of this capability leading to the correct implementation of the above strategies is listed below:

The first test in such a scheme would involve the determination of the values of the time constants of the valves on both sides of the motor when a constant volume is pressurised or vented through the valves. This test would require the acquisition of two constant volume pressure transients for each of the valves (these can be easily obtained from the experimental setup, if it is fitted with the valves in question, with the aid of the data acquisition system). The
required time constants could then be obtained by repeated comparison with simulated pressure transients according to the procedure described in Chapter 3. (This is the only part of the scheme that requires experimental data.)

Once the values of these time constants have been obtained, the generation of flow characteristic data becomes possible, rendering feasible the simulation of the arm's performance by computer. It is now possible to simulate the trajectory of the arm under the control of a simple switch-over sequence (using a simulator based on VMSTDP). By carrying out a number of such simulations, data relating the switch-over points (or the chamber pressures when the arm decelerates to zero velocity for the first time) to combinations of start and target points can be obtained. This data can then be subjected to curve fitting routines so as to obtain a mathematical relationship between start and target point combinations and switch-over position or pressure (at zero velocity) values.

One suggested procedure for this would involve plotting, say, switch-over points against target points for the same starting point and repeating the procedure for other starting points. Once simple curves had been fitted to these plots the coefficients of the curves could be plotted against the values of the starting point. This part of the scheme would have to be executed on a mainframe computer with access to curve fitting routines and would provide the capability of simulation of a simple switch-over trajectory for any start and target positions (in forward or reverse time).
This capability having been obtained any of the strategies described earlier in this chapter could be used for the control of the arm. These strategies could probably be carried out on a fast micro-processor with sufficient memory capacity and/or sufficiently fast on-line computing capability. (It is in view of the properties of the system in use that a decision will have to be made regarding the choice of a strategy that generates a switching profile on-line or pre-computes the switching profile data off-line and merely accesses them for the on-line control of the arm.)

6.5 SUGGESTIONS FOR THE IMPROVEMENT AND EXPANSION OF THE BASIC CONTROL SCHEME.

6.5.1 Coping with limitations in the accuracy of computer-generated switching profiles.

The control scheme described in the previous section should guide the arm along a smooth trajectory between a start and a target point by implementing a single switch-over of the valves. According to the analysis made in chapter 5 regarding the use of simulation-generated switching profile data this single switch-over should be sufficient if carried out at the right position to guide the manipulator to the target point. If, however, the representation of the arm which is implemented by the simulator is not completely accurate, the trajectory of the arm will deviate from that dictated by the switching profile data. In this case the control algorithm will implement further switch-overs to guide the arm back on the trajectory described by the switching profile. Because this
strategy was conceived with the aim of providing a minimum number of switch-overs, the number of switch-over operations that are actually implemented by the control algorithm is likely to be small, as long as the switching profile generated by the simulator package represents the performance of the arm with reasonable accuracy.

It was pointed out in Chapter 5, however, that the programmes that simulate the trajectory of the arm in forward and reverse time in order to provide switching profile data both display a similar inability to simulate the arm's performance accurately around the target point (because of their inadequate representation of dynamic friction at low velocities). It is therefore reasonable to assume that the portion of the switching profile which is generated by simulating the arm's trajectory around the target region cannot adequately control the arm in this region.

The basis for a strategy for the control of the arm in this region was described in Chapter 4. The analysis leading to this strategy assumed small displacements about the target point and, by implication, small rates of displacement (as the analysis is essentially based on a constant volume representation of the motor). It is reasonable to argue that under the control of the simulator-generated switching profile the arm will approach the region of the target position travelling at a relatively small velocity. The conditions are therefore satisfied for the application of a feedback control strategy based on a constant volume representation of the motor in the target region.
The improved control scheme would, therefore, use the computer-generated switching profile to guide the arm along the greatest part of its trajectory but would resort to the above feedback control strategy (based on the constant volume motor representation) for the control of the arm as it approaches the target position.

6.5.2 Expanding the basic control scheme for use with the manipulator.

The control scheme considered so far is suitable for the control of an arm with a single degree of freedom. If it is to be applied to a manipulator with more than one degree of freedom it must be suitably duplicated to provide switching profile data and to implement appropriate control algorithms for all the degrees of freedom of the manipulator. In order to accomplish this it is sufficient to repeat the procedure described in the previous section for all the manipulator axes.

This will necessitate additional data acquisition from the experimental arm if different valves are used to supply the motors driving the axes, in order to derive the basic simulation time constants for each valve-motor configuration. Once these basic time constants have been obtained the basis for the simulation of simple switch-over trajectories for each degree of freedom will have been provided. In carrying out these simulations for the purpose of obtaining switching profile data care must be taken to supply the simulator with suitable values for the properties of each degree of freedom (i.e., maximum motor torque, valve delay times, motor volumes etc.). Particular
attention must be paid to the values of the moments of inertia supplied to the simulator package as the moment of inertia about each manipulator joint is not fixed but depends on the position of all the arm segments that rotate directly or indirectly about that joint.

The difficulty posed by such changing moment of inertia values may be overcome if simplifying assumptions about the interrelation of these values are made, enabling their simplified representation in the simulations. Should such simplifying assumptions result in an inaccurate simulation of the manipulator trajectory an alternative approach would have to be attempted. The simulations of all the degrees of freedom of the manipulator would have to be carried out in parallel and the moment of inertia value about each manipulator joint would have to be calculated from the position of all the relevant arm segments in each simulation step. Clearly, the use of this approach for the generation of switching profiles for any manipulator trajectory would require considerable computing effort.

6.5.3 Suggestions for the incorporation of path generation routines in the manipulator control package.

The control scheme discussed up to this point is designed to control each axis of the manipulator while this axis executes a trajectory between two specified (start and target) co-ordinates. As routines that generate a series of interpolated points between such start and finish co-ordinates in more than one dimension have already been described (Chapter 2) it is
useful to comment on the possibility of the use of such
routines in conjunction with the above control scheme.

The investigation of path interpolation and path generation
routines was abandoned at a fairly early stage for the
following reasons:

(a) The improvement on the trajectory of the arm brought
about by the use of interpolation routines had to be obtained
at the expense of accuracy as the control algorithm which was
in use at the time was not sufficiently suited to the system
characteristics to implement a smooth approach to specified
target co-ordinates with any degree of accuracy.

(b) The generation of interpolated points according to
specified mathematical relationships could not be extended to
three-dimensional space because of the on-line computing effort
that had to be expended for the conversion of the Cartesian
co-ordinates in question to internal robot co-ordinates (Joint
angles) that would enable the manipulator end-point to move
along a path (in three dimensional space) specified by these
relationships.

Point (a) is no longer valid as a control strategy that
will implement a smooth and accurate approach to a specified
target point now exists. Point (b) becomes relatively
unimportant when the control scheme as a whole is considered as
most of the operations related to the generation of suitable
switching profiles (and possibly the generation of the
switching profile itself) are carried out off-line. In view of
This it could be argued that the co-ordinate conversion procedure could be carried out by off-line computation. It would therefore become possible, if co-ordinate generation and conversion routines were incorporated in the control package, to specify the path that the manipulator end-point would follow in three-dimensional space. Once the start and target point co-ordinates and the path to be followed between them by the manipulator end-point had been specified in three-dimensional space interpolated co-ordinates and the switching profiles necessary for their smooth attainment by the manipulator axes could be calculated by the attendant path interpolation and simulation software. (Such a calculation would be more suitably carried out off-line.) The resultant switching profiles could then be used for the control of the arm along the specified trajectory.

It should be pointed out at this stage that as the manipulator would not have to stop when attaining interpolated co-ordinates the simulator package would have to be modified so that the simulation of the trajectory between successive interpolated co-ordinates would be initialised with the correct (non-steady-state) values of the variables involved in the simulation (as obtained at the end of the trajectory between the previous interpolated co-ordinates).

Once path generation capability has been combined with a suitable control strategy for the manipulator a number of possibilities present themselves. These include speed regulation by means of specifying the increment between interpolated co-ordinates, synchronous motion of the manipulator axes etc. At
this stage (which is at present highly speculative) criteria would have to be determined for the selection of an optimal type of path for the manipulation tasks that are to be undertaken by the manipulator.
6.6 CONCLUSIONS.

An examination of the work described in this Thesis leads to the following conclusions:

(i) It is possible to specify the type of path to be followed by the manipulator end-point between two points in three-dimensional space but the coordinate conversion which is needed for the specification of the individual joint angles that must be supplied to the manipulator control package to actually follow this path is time-consuming and not really suitable for on-line computation (although such an evaluation is relative, depending on the speed of the processor used for this task and the efficiency of the software involved).

(ii) The use of interpolation routines (which form the basis of such path specification) was demonstrated in principle but was shown to be of limited use when implemented in conjunction with the control algorithm then in use (1st order lead) as this algorithm was not based on an adequate representation of the system.

(iii) A representation of the pressure transient in a constant volume supplied by the valve can be obtained analytically, after certain simplifying assumptions have been made, by expressing such a transient as an exponential rise or decay function. This representation enables the simulation of constant volume pressure transients by digital computer. This simulation enables, in turn, the generation of flow characteristic data for the valve by the computer with a
minimal recourse to experimental data. The procedure adopted in this simulation provides additional insight in the mode of implementation of the original (first order lead) control algorithm and suggests an improved mode of implementation which is suitable for the control of the arm (one degree of freedom) when small displacements are being considered.

(iv) An analytical representation, suitable for control purposes, of the valve-motor combination is difficult because of the interdependence of the variables involved in the representation. Our ability, however, to generate flow characteristic data by computer makes the simulation of the valve-motor combination by computer possible by the piece-wise generation of the parts of the flow characteristic that correspond to the conditions prevailing in each simulation step. Once the simulation of the performance of the valve motor combination has been accomplished the simulation of a single degree of freedom of the manipulator becomes a relatively easy matter.

(v) A simulated trajectory of a single degree of freedom of the manipulator (under the control of a simple switch-over sequence) provides a switching profile suitable for the control of the arm. It is possible to carry out such a simulation in forward or reverse time, but both types of simulation become inaccurate at the region surrounding the target point. As a strategy for the control of the axis in this region has already been suggested this is not a great disadvantage.

(vi) The use of data generated by computer simulation for
the control of a single degree of freedom was demonstrated in principle. This approach provides the basis for a comprehensive control package for the manipulator using simulation-generated switching profiles for all degrees of freedom and thus by-passes the need to represent the manipulator analytically for control purposes. Such a comprehensive control package is likely to require considerable off-line computational effort. In view of this the incorporation of path generation routines and routines that carry out Cartesian-to-robot co-ordinate conversion would not constitute an excessive computing overhead and could be incorporated into the overall control package, thus considerably enhancing the capability of the robot to carry out a variety of manipulation tasks.
LIST OF REFERENCES.

1. FOUR MILLION HOURS OF ROBOT FIELD EXPERIENCE.
   J.F. Engelberger, 4th International Symposium on Industrial
   Robots (ISIR), Tokyo 1974, p. 133.

2. SELECTED APPLICATIONS OF VFW-FOKKER FLEXIBLE HANDLING
   SYSTEMS.

3. MHU INDUSTRIAL ROBOTS IN PRODUCTION SYSTEMS.

4. 5 YEARS OF INDUSTRIAL ROBOT EXPERIENCE AT VW.
   F. Weissberger, 8th ISIR, Stuttgart 1978, p. 78.

5. A SYSTEMS APPROACH TO ROBOT USE IN DIE CASTING.

6. INDUSTRIAL ROBOT FOR HEAT TREATMENT AND APPLICATIONS.

7. APPLICATION OF AN INDUSTRIAL ROBOT FOR PALLETTISING OF
   BOXES IN A COMPLICATED PATTERN.

8. CONTINUOUS PATH CONTROLLED ROBOT IN ARC-WELDING
   APPLICATIONS.
9. MICROMANIPULATION OF SMALL CIRCUITS.

10. APPLICATION OF AN INDUSTRIAL ROBOT TO A COLD FORGING PROCESS.

11. MODELL OF A "FLEXIBLE MANUFACTURING CELL" FOR ROTATIONAL PARTS.

12. GRINDING ROBOT ON A CROOKED SPATIAL SURFACE.
K. Okamoto et al., 7th ISIR, Tokyo 1977, p. 615.

13. MATERIAL HANDLING DEVICE FOR IRREGULARLY SHAPED HEAVY WORKS.

14. APPLICATION OF THE SIEMENS INDUSTRIAL ROBOT IN MACHINE FEEDING.

15. ROBOTS IN INDUSTRY.

16. ANALYSIS OF MECHANISMS AND ROBOT MANIPULATORS.
17. THE MATHEMATICS OF COORDINATED CONTROL OF PROSTHETIC ARMS AND MANIPULATORS.

18. RESOLVED MOTION RATE CONTROL FOR MANIPULATORS AND HUMAN PROSTHESES.

19. A KINEMATIC NOTATION FOR LOWER PAIR MECHANISMS BASED ON MATRICES.

20. THE DESCRIPTION AND CONTROL OF THE STATE OF A ROBOT.

21. FORCE FEEDBACK CONTROL.

22. SHOULD AN INDUSTRIAL ROBOT BE MORE ANTHROPOMORPHIC SYSTEM IN THE FUTURE THAN IT IS NOW.

23. "INTERRELATED FUNCTIONS" CONCEPT APPLIED IN MANIPULATORS.
24. A COMPUTER GUIDANCE OF MANIPULATORS.

25. STRUCTURAL AND KINEMATICAL ANALYSIS OF ROBOTS AND MANIPULATORS.

26. A KINEMATICAL CLASSIFICATION OF ROBOT MANIPULATORS.

27. PROGRESS REPORT ON DIGITAL SIMULATION.

28. APPLIED OPTIMAL CONTROL.

29. USE OF RESOLVED RATES TO GENERATE TORQUE HISTORIES FOR ARM CONTROL.

30. PATH CONTROL FOR A REDUNDANT TYPE OF INDUSTRIAL ROBOT.

31. CONTROL OF INDUSTRIAL ROBOTS BY MEANS OF MICROPROCESSORS.
32. MATHEMATICAL MODEL OF AN INDUSTRIAL MANIPULATOR: A CLOSED FORM SOLUTION USING LAGRANGE'S METHOD.

33. DYNAMIC CONTROL OF ANTHROPOMORPHIC MANIPULATORS.
M. Vukobratovic; D. Stokic; D. Hristic, 4th ISIR, Tokyo 1974, p. 229.

34. CONTROL OF ANTHROPOMORPHIC MANIPULATORS IN INDUSTRIAL APPLICATION.

35. DYNAMIC CONTROL OF MANIPULATORS AND SOME PROJECTS OF INDUSTRIAL APPLICATION.
M. Vukobratovic; D. Hristic, 7th ISIR, Tokyo 1977, p. 131.

36. INDUSTRIAL ANTHROPOMORPHIC MANIPULATORS FOR HIGH POSITIONING ACCURACY REQUIREMENTS.
D. Hristic; M. Vukobratovic; O. Dzigurski, 8th ISIR, Stuttgart 1978, p. 478.

37. REAL TIME PROBLEMS IN COMPUTER CONTROL OF ROBOTS.
P. Coiffet; J. Dumas; P. Molinier, 7th ISIR, Tokyo 1977, p. 145.

38. ANALYTICAL EQUATIONS vs. TABLE-LOOKUP: A UNIFYING CONCEPT.
39. MANIPULATOR CONTROL USING THE CONFIGURATION SPACE METHOD.

40. COMPUTED PATH CONTROL FOR AN INDUSTRIAL ROBOT.

41. MOVING LINE APPLICATIONS WITH A COMPUTER CONTROLLED ROBOT.

42. ROBOT PATH CONTROL BY OFF-LINE COMPUTER.
W. Perzley, 8th ISIR, Stuttgart 1978, p. 338.

43. COMPUTER CONTROLLED INDUSTRIAL ROBOT FOR PAINTING.
S. Ando et al., 8th ISIR, Stuttgart 1978, p. 689.

44. FOUNDRIES ROBOTS AND PRODUCTIVITY.

45. DEVELOPMENT OF BASIC CONTROL PROGRAMS FOR INDUSTRIAL ROBOTS WITH ARTIFICIAL INTELLIGENCE.

46. CONTROL OF WORKING ORGAN MOVEMENT SPEED IN THE MANIPULATOR NON-ORTHOGONAL SYSTEM OF CO-ORDINATES.
47. APPLICATION OF PNEUMATIC INDUSTRIAL ROBOTS FOR ASSEMBLY.

48. THE ANALYSIS AND DESIGN OF PNEUMATIC SYSTEMS.

49. PNEUMATIC AND HYDRAULIC CONTROL SYSTEMS.

50. COMPUTER SIMULATION OF CONTINUOUS SYSTEMS.

51. SIMULATION OF MECHANICAL SYSTEMS. AN INTRODUCTION.

52. SIMULATION OF CONTROL SYSTEMS.
I. Troch (Editor), North-Holland, 1978.

53. PROCEEDINGS OF THE 1973 SUMMER COMPUTER SIMULATION CONFERENCE.

54. AUTOMATIC CONTROL ENGINEERING.

55. INTRODUCTION TO CONTROL THEORY INCLUDING OPTIMAL CONTROL.
D.N. Burshes, A. Graham, Ellis Horwood, 1980.
56. MODERN CONTROL ENGINEERING.

57. CONTROL AND DYNAMIC SYSTEMS.

58. ANALYSIS AND DESIGN OF NON-LINEAR FEEDBACK CONTROL SYSTEMS.

59. NON-LINEAR AUTOMATIC CONTROL.

60. ANALYSIS OF NON-LINEAR CONTROL SYSTEMS.

References dealing with the University of Surrey Manipulator.

61. AN OPTO-PNEUMATIC MANIPULATING ARM.

62. MICROPROCESSOR CONTROL AND PNEUMATIC DRIVE OF A MANIPULATING ARM.

63. RECENT DEVELOPMENTS IN THE DESIGN OF A NOVEL ELECTRO-PNEUMATIC ROBOT.
P.J. Drazan, M.F. Jeffery, J.M. Zarek, 2nd CISM-IFToMM
64. CONTROLLABILITY OF AN ELECTRO-PNEUMATIC MANIPULATOR.

65. SOME ASPECTS OF AN ELECTRO-PNEUMATIC INDUSTRIAL MANIPULATOR.

66. SIMULATION OF AN ELECTRO-PNEUMATIC MANIPULATOR SYSTEM.

67. CONTROL OF ROBOT DYNAMICS BY MICROCOMPUTERS.

68. CONTROL OF ROBOT DYNAMICS.
P. J. Drazan, Conference of Swiss Association for Automation, Zurich, March 1980.

69. THE USE OF INTERPOLATION ROUTINES FOR PATH GENERATION AND CONTROL OF AN ELECTRO-PNEUMATIC INDUSTRIAL MANIPULATOR.
P. J. Drazan, V. Toulis, 10th ISIR, Milan 1980, p. 313.
The following References are Internal Departmental Reports (written by R. Sneddon) concerning SRC research on:

Investigation and Development of Control Strategies for an Electro-Pneumatic manipulator.

70. Work report on SAMOS, (Report No. 20)

71. Work report on VALVE CHARACTERISTICS (Report No. 21).

72. COLLECTION AND ANALYSIS OF DATA FROM THE SAMOS RIG, (Report No. 22).

73. Programme report on SIMVALVE (Report No. 27).

74. Programme report on SAMTRAN (Report No. 28).

75. Subroutine report on VALVE (Report No. 31).

76. Subroutine report on VELEST (Report No. 30).
Consider the expression for the motor chamber pressure given by equation (9):

\[ P = \frac{P_c}{1 + \tau S} \quad (9) \]

For a step input of magnitude \( P_c \) this becomes:

\[ P = \frac{P_c}{1 + \tau S} \cdot \frac{1}{S} \]

Expanding the above into partial fractions gives:

\[ \frac{1}{1 + \tau S} = \frac{A}{1 + \tau S} + \frac{B}{S} \]

Therefore we have:

\[ 1 = A + B \tau \]

Equating equal power coefficients in the above equation gives:

\[ A + B \tau = 0 \]
\[ B = 1 \text{ Therefore } A = -\tau \]

Therefore we have:

\[ P = \frac{P_c - \tau}{S} \quad + \quad \frac{P_c}{1 + \tau S} \quad + \quad \frac{1}{(1/\tau) + S} \]

Taking the inverse Laplace transform of the above in order to obtain an expression in the time domain we have:
\[ P(t) = (-P_c) \times e^{-t/\tau} + P_c \]

Hence:

\[ P(t) = P_c \times (1 - e^{-t/\tau}) \]
APPENDIX 2.

Listing of the constant volume pressure transient simulator XPPRTR.
DIMENSION X(1000),PR(1000),PR2(1000),V2(1000),Z2(1000),X2(1000)
*IPOSN(20),MX(15),MT(15),MP(15),ML2(25),MY(15),ML(25),MGVEL(4)
COMMON DELT,DELM

THESE STATEMENTS HOLD THE GRAPH TITLES (2 PER ARRAY ELEMENT)

DATA MX(1)/2HPO/,MX(2)/2HSI/,MX(3)/2HTI/,MX(4)/2HON/, 
*FX(5)/2H-/!,MX(6)/2HRE/,MY(1)/2HVE/,MY(2)/2HL-/!,MY(3)/2HRD/, 
*MY(4)/2H/S/,ML(1)/2HVE/,ML(2)/2HLO/,ML(3)/2HCI/,ML(4)/2HTY/, 
*ML(5)/2H-P/,ML(6)/2HSO/,ML(7)/2HTI/,ML(8)/2H10/,ML(9)/2HN/, 
*MGVEL(1)/2H6V/,MGVEL(2)/2HCL/,MGVEL(3)/2H=/, 
*MT(1)/2HTI/,MT(2)/2HME/,MT(3)/2H-S/, 
*ML2(1)/2HP0/,ML2(2)/2HSCI/,ML2(3)/2HTI/,ML2(4)/2HON/, 
*ML2(5)/2H-T/,ML2(6)/2HIN/,ML2(7)/2HE/, 
*MP(1)/2HPR/,MP(2)/2HES/,MP(3)/2HSU/,MP(4)/2HRE/, 
*DATERM/=0.0/

PI = 3.14159
E = 2.7182818
NOP=999

N1 = 1
N2 = 0
N3 = 0
N4 = 0

; NO. OF CYCLES IN SIMULATION

BEN=1.23
; AIR DENSITY

SUP=1.0

STR=1.57
; STROKE OF ACTUATING CYLINDER

DELT=0.001
; TIME BETWEEN SIMULATION CYCLES

TIMSIM=5.0
; TOTAL SIMULATION TIME

P0=5.0E5
PA=1.0E5
P1=P0
P2=PA
P1V=P1
P2V=P2
P1D=P1V
P2D=P2V
FR1(1)=P1
PR2(1)=P2

DELX=0.0
; INITIAL PRESSURE SETTINGS

VDEL = 0.054
; VALVE DELAY

L = IFIX(VDEL/DELT)
; NO. OF CYCLES IN VALVE DELAY SIMULATION
**Format 27H** MANTIPULATOR SIMULATION/27H

**-----------------------------**
** THE INITIAL CONDITIONS ARE: */*

**-----------------------------**

**C**

```
WRITE(1,10) A, DEN, GUP, P0, PA, STR, XS, FR0, FR, HMOT, DELT, DZN
* GVEL, GACC
```

```
10 FORMAT (3H1 A, F6.4/5H, DEN, F5.3/5H, GUP, F5.3/4H, P0 = E10.2
*/4H, PA, F10.2/5H, STR = F5.2/4H, XS = F5.2/75H, FR0, F5.2/4H
*/4H FR = F5.2/6H, NAS = F4.2/
*/9H TIME INTERVAL USED, DELT = F6.3/6H, DZN = F7.4/8H, GVEL = F5.2/8H
*/ GACC = F5.2//**
```

**C**

```
WRITE (1,6)
```

```
6 FORMAT (1X, 5H TIME, 3X, 4H ACC, 3X, 5H VEL2, 4X, 3H X2, 5X, 3H P1, 5X
*/3H P2, 4X, 4H N1, 4X, 4H P2N, //**
```

**C**

```
ISUMPR = 0
XCURR = XS
```

**NOw PErFORM THE SIMULATION AND STORE THE RESULTS IN ARRAYS**

**C**

```
LOOPCT = N0P + 5
CO 500 N = 1, LOOPCT
```

```
N2 = N2 + 1
N3 = N3 + 1
N4 = N4 + 1
```

**CYCLE COUNTERS INCREMINEED, TEST THEM LATER**

**C**

```
CALCULATE NEW PRESSURE FOR NEXT CYCLE
```

```
P1 = P1 + DELP1
```

**C**

```
SET LIMITS ON POSSIBLE VALUES OF P1
```

```
IF (P1. GT. P0) P1 = P0
IF (P1. LT. PA) P1 = PA
```

**C**

```
TIME = TIME + DELT
IF (N2.H. NE. DELT) G0TC 150
```

```
M = N/5
X(M+1) = XCURR
V2(M+1) = VEL2
X2(M+1) = X(M+1)
PR1(M) = P1
PR2(M) = P2
N2 = 0
```

**PERFORMANCE FIGURES SAVED**

```
CONTINUE
```

**C**

```
GOTO 451
```

```
P1NDIS = P1N/100000.
P2NDIS = P2N/100000.
P10DIS = P1O/100000.
```

128) 130. FORMAT(F6.3,3(F9.4,1X),2(F9.0,1X),2(F3.0,1X),2(I3,1X))
129) C
130) WRITE(1,135)TIMEX1,TIMEX2,P10DIS,P20DIS,TAU1,TAU2,DELP1,DELP2
131) 135 FORMAT(2F7.3,2X,2F3.0,2X,2F8.4,2X,2F9.0)
132) 431 CONTINUE
133) C
134) 202 PIN=PA;
135) C ; SIDE 1 PRESSURISED
136) C
137) C RESPONSE DELAY OF THE VALVE PAIR
138) C
139) 113 IF(PIN.EQ.P1D) GO TO 119
140) IF(PIN.EQ.P1V) GO TO 116
141) J=L
142) PIN=P1V
143) 110 J=J-1
144) IF(J.GE.0) GO TO 119
145) P1D=P1V
146) C NOW DETERMINE THE PRESSURE CHANGES
147) C
148) C ASSIGN TIME CONSTANT
149) 11" "  TAU1=1.5
150) C
151) GOTO 300
152) C
153) ( CORRECT FOR FLOW CHARACTERISTIC (PRESSURISATION ONLY)
154) C
155) C
156) IF(PIN.EQ.PA) GO TO 300
157) C VERIFY PRESSURISATION
158) C
159) IF(P1.GE.5.0E5) GO TO 100
160) C
161) C
162) IF(P1.GE.3.0E5) GO TO 200
163) TAU1 = (TAU1/0.4)
164) GO TO 300
165) 200 TAU1 = (TAU1/1.1)
166) GO TO 300
167) 100 TAU1 = (TAU1/(8.33*(P1/1.0E5)-40.23))
168) C EVALUATE PRESSURE CHANGE
169) 300 DELP1=(P10-P1)*(1-E**(-(DELT)/TAU1))
170) C
171) 499 VEL1=VEL2
172) C
173) C
174) 500 CONTINUE;
175) C
176) GOTO 9
177) C
178) C NOW PLOT THE RESULTS HELD IN THE ARRAYS
179) C
180) C
181) C NOW DRAW PRESSURE TRANSIENT
182) C
183) C
184) C SET UP AXIS VALUES
185) C
CALL GOC

NOW DRAW AND LABEL AXES

CALL GRAPH(1,MT,MP,ML2,3,4,7,2,XZERO,XMAX,5,1,YZERO,YMAX,6,1)

CALL MOVTO2(55.,55.)

TIME = 0.0

RENORMALISE THE Y AXIS RANGE (X10E5)

YZERO = 0.0

YMAX = YMAX*1E5

XRANG = XPIXL/(XMAX-XZERO)

YRANG = YPIXL/(YMAX-YZERO)

XOFFSET = 255-XPIXL

YOFFSET = 255-YPIXL

DO 440 K=1,NOP

XSCAL = ((TIME-(XZERO))*XRANG)+XOFFSET

YSCAL = ((PR1(K)-(YZERO))*YRANG)+YOFFSET

CALL LINTO2(XSCAL,YSCAL)

TIME = TIME+(DELT*5.0)

440 CONTINUE

STOP

CALL MOVTO2(55.,130.)

TIME = 0.0

NOW DRAW P2

DO 445 K=1,NOP+4

XSCAL = ((TIME-(XZERO))*XRANG)+XOFFSET

YSCAL = ((PR2(K)-(YZERO))*YRANG)+YOFFSET

CALL LINTO2(XSCAL,YSCAL)

TIME = TIME + (DELT*4.0)

445 CONTINUE

TIME = 0.0

CALL MOVTO2(55.,130.)

PLOT P1-P2 ON THE SAME AXES

DO 450 K=1,NOP+4

XSCAL = ((TIME-(XZERO))*XRANG)+XOFFSET

YSCAL = ((PR1(K)-PR2(K))-(YZERO))*YRANG)+YOFFSET

CALL LINTO2(XSCAL,YSCAL)

TIME = TIME + (DELT*4.0)

450 CONTINUE
APPENDIX 3.

----------

Listings of the full arm simulator MANIPR. (This package does not implement the full valve-motor representation for large displacements.)
DIMENSION X(1000), PR1(1000), PR2(1000), V2(1000), Z2(1000), X2(1000)

*IP0SN(20), MX(15), NT(15), MP(15), ML2(25), MY(15), ML(25), MGVEL(4)

COMMON DELT, DELM

THESE STATEMENTS HOLD THE GRAPH TITLES (2 PER ARRAY ELEMENT)

DATA MX(1)/2HPO/, MX(2)/2HSI/, MX(3)/2HTI/, MX(4)/2HON/,
  MX(5)/2HRD/, MX(6)/2HVE/, MX(7)/2HL0/, MX(8)/2HCC/, MX(9)/2HHT/,
  MX(10)/2HCR/, MX(11)/2HVE/, MX(12)/2HL0/, MX(13)/2HCC/, MX(14)/2HHT/,
  MX(15)/2HCR/

DATA MP(1)/2HPR/, MP(2)/2HES/, MP(3)/2HSU/, MP(4)/2HRE/

VOL1 = 0.000076
VOL2 = 0.000076
VOLC = 0.000264
VOLCAL = 0.000300

CALIBRATED DEAD VOLUMES AND SWEEP VOLUMES FOR MOTOR

DEBUG = 0.0
PI = 3.14159265359
E = 2.7182818

N = 1
N2 = 0
N3 = 0
N4 = 0

; CONTROL. VELOCITY AND ARRAY UPDATE CYCLE COUNTERS

STR = PI/2

; STROKE OF ACTUATING CYLINDER

XS = STR*11/16
IXSCOD = IFIX(XS*256/STR)

FR = 6.00

; DYNAMIC FRICTION

FR0 = 8.70

; STATIC FRICTION

TMAX = 104.24

; MAXIMUM TORQUE

PTIA = 0.94

; INERTIA

GVEL = 0.25

; CONTROL PARAMETERS (SEE ALGORITHM)

DELT = .001

TIMSIM = 5.0

; TOTAL SIMULATION TIME
C ; INITIAL PRESSURE SETTINGS
75) C x(1) = .25* STR
76) C ; INITIAL STARTING POSITION
77) C DELX = 0.0
78) C X2 = X(1)
79) C VLLOF1 = 0.054
80) C VLLOF2 = 0.054
81) C VLLON1 = 0.009
82) C VLLOM2 = 0.009
83) C ICYOF1 = IFIX(VLLOF1/DELT)
84) C ICYOF2 = IFIX(VLLOF2/DELT)
85) C ICYON1 = IFIX(VLLON1/DELT)
86) C ICYON2 = IFIX(VLLOM2/DELT)
87) C ; NO. OF CYCLES IN VALVE DELAY SIMULATION DURING DISCHARGE
88) C ; NO. OF CYCLES IN VALVE DELAY SIMULATION DURING CHARGE
89) C NPOP = 16
90) C ; NO. OF SAMPLES TAKEN FOR VELOCITY ESTIMATION
91) C NFACT = (NPOP**2)
92) C ; VELOCITY SAMPLER DIVISOR FACTOR
93) C VEL1 = 0.0
94) C V2 = 0.0
95) C TIME = 0.0
96) C
97) C WRITE(1,2)
98) C FORMAT /// 27H MANIPULATOR SIMULATION/27H
99) C ***** /// 28H THE INITIAL CONDITIONS ARE: ///
100) C WRITE(1,10) A, DEN, GUP, P0, PA, STR, XS, FR0, FR, H0, MOT, DELL, DZN
101) C + GVEL, GA CC
102) C
103) 10 FORMAT(3H =, F6,4/5H DEN =, F5,3/5H GUP =, F5,3/4H P0 =, E 10,2
104) + /4H PA =, F10,2/5H STR =, F5,2/4H XS =, F5,2/5H FR0 =, F5,2/
105) + /4H FR =, F5,2/6H MASS =, F4,2/
106) + /25H TIME INTERVAL USED + DELT =, F6,5/6H DZN =, F7,5/6H GVEL =, F5,2/8H
107) + GACC =, F5,2///
108) C WRITE(1,6)
109) C FORMAT(1X,5H TIME, 3X, 4H ACC, 5X, 5H VEL2, 4X, 3H X2, 5X, 3H P1, 5X,
110) + 3H P2, 4X, 4H PIN, 4X, 4H P2N, ///
111) C ISUPR = 0
112) C XCURR = X(1)
113) C
114) C NOW PERFORM THE SIMULATION AND STORE THE RESULTS IN ARRAYS
115) C
116) C LOOPCT = NPOP * 5
117) C GO 500 N = 1, LOOPCT
118) C
N4 = N4 + 1
CALCULATE NEW PRESSURE FOR NEXT CYCLE
P1 = P1 + DELP1
SET LIMITS ON POSSIBLE VALUES OF P1
IF (P1 > P0 .AND. P1 < P0) P1 = P0
IF (P1 < P0 .AND. P1 > P0) P1 = P0

SIMILARLY PROCESS SIDE 2'S PRESSURE CHANGE FOR NEW P2
P2 = P2 + DELP2
IF (P2 > P0 .AND. P2 < P0) P2 = P0
IF (P2 < P0 .AND. P2 > P0) P2 = P0
MARKER = 1
IF (DEBUG .EQ. 0) WRITE (1,9) MARKER

MECHANICAL HARDWARE
IF (V1 .LE. 0.0) GOTO 104 ; IS ARM STATIONARY?
FRIC = FRIC
IF (P1 > P2) FRIC = -FRIC ; FRICTION OPPOSES MOVEMENT OR FORCE
IF (ABS ((P1 - P2) * MAX / 7.145) .GT. abs (FRIC)) GOTO 106 ; MOVEMENT?

ACC = 0.
GOTO 118

FRIC = (V1 / (1 + (ABS (V1)))) * FRIC * (-1)
FRICIONAL FORCE ALWAYS OPPOSES DIRECTION OF MOVEMENT

ACC = (MAX * (P1 - P2) / 7.145 + FRIC) / RTIA
FORCES (PRESSURE & FRICTION) ADDED AS VECTORS

VEL2 = (ACC * DELT) * VEL1 ; NEW VELOCITY
GOTO 780
IF (VEL2 > 2.0) VEL2 = 2.0
IF (VEL2 < -2.0) VEL2 = -2.0
SET LIMIT ON NEW VELOCITY
DELX = (VEL2 + VEL1) * DELT / 2.
IF (XCURR > 0.1 .AND. VEL < 0.0) DELX = 0.0
IF (XCURR < 0.1 .AND. VEL > 0.0) DELX = 0.0 ; TRAVERSE LIMIT OF MOTOR
XCURR = XCURR + DELX ; NEW POSITION
IXCODE = IFIX(XCURR * 256 / STR)
IPART(N4) = IXCODE ; GET INTEGER VALUE OF XCURR IN RANGE 1-256
TIME = TIME + DELT ; 1MS SAMPLE OF POSITION (1-256 RESOLUTION)
IF (N2 .NE. 5) GOTO 150 ; IF ON A 5 MS INTERVAL, UPDATE THE ARRAYS
c = 1.5
X(N+1) = XCURR
V2(M+1) = VEL2
V2(M+1) = XCM+1
PR1(M) = P1
PR2(M) = P2
N2 = 0 ; PERFORMANCE FIGURES SAVED

150 CONTINUE

159 CONTINUE

999 if (DEBUG.EQ.0,0) GOTO 431
999 P1NDIS = P1/100000.
999 P2NDIS = P2/100000.
999 P1DIS = P1/100000.
999 P2DIS = P2/100000.

130 WRITE((1,130) TIME,ACC,VEL2,XCURR,P1,P2,P1NDIS,P2NDIS,N2,N3
130 FORMAT(F6.3,F9.1,1X,2(F9.1,1X),2(F3.0,1X),2(I3,1X))
130 CONTINUE
130 WRITE((1,135)P1DIS,P2DIS,TAU1,TAU2,DELP1,DELP2
130 FORMAT(2X,2F3.0,2X,2F8.4,2X,2F9.3)
1431 CONTINUE

140 CONDITIONAL BRANCHING LOGIC ALGORITHM

150 if (N4.NE.NPOP) GOTO 113
150;

160 ISUMPS = 0
160 GO TO 260
160
160 ISUMPS = ISUMPS + IPOS(N)
160 
160 SUMDF = ISUMPS-ISUMPR ; SUM OF SEVERAL PAST POSITIONS CALCULATED
160 
160 SUMDF = ISUMPS - ISUMPR ; DIFFERENCE OF NEW AND PREVIOUS SUM.
160 
160 SMPVEL = SUMDF/(DELT*FACT) ; AVERAGE VELOCITY DURING PREVIOUS
160 SAMPLING PERIOD
160
160 N4 = 0
160
160 NOW SET THE VALVES ACCORDING TO THE AVAILABLE DATA
160
170 N4 = 0
170 DXVEL = GVEL*SMPVEL
170 XCOR = IXCODE+DXVEL
170 ER = IXSCOD - XCOR
170 IF (IXSCOD.NE.IXCODE) GOTO 109 ; IS ARM IN DEAD ZONE?
170 P1N=P0
170 P2N=P0 ; BOTH CYLINDERS PRESSURISED
170
180 GO TO 113
180
180 IF (ER.GT.0) GOTO 202
180
180 P1N=PA
180 P2N=PA ; SIDE 2 PRESSURISED
2 = SIDE 1 PRESSURISED

MARKER = 3
IF(DEBUG*NE.0.0) WRITE(1,19)MARKER

RESPONSE DELAY OF THE VALVE PAIR

113 IF(PIN.EQ.PI0) GO TO 131
114 IF(PIN.EQ.PI0) GO TO 110
115 J=ICYOF1
116 IF(PIN.EQ.P0) J=ICYO1
117 P1V=PIN
118 C=J-1
119 IF(J.GE.0) GO TO 131
120 IF(P0=PI0) GO TO 111
121 IF(P2N.EQ.P20) GO TO 121
122 K=ICYOF2
123 IF(P2N.EQ.P0) K=ICYO2
124 P2V=P2N
125 K=K-1
126 IF(K.GE.0) GO TO 111
127 P2N=P2V
128 MARKER = 4
129 IF(DEBUG*NE.0.0) WRITE(1,19)MARKER

NOW DETERMINE THE PRESSURE CHANGES IN EACH SIDE OF THE MOTOR

ASSUMING A CONSTANT VOLUME PRESSURE TRANSIENT REPRESENTATION WITH
UPDATED TIME CONSTANT VALUES TO CATER FOR THE CHANGE IN VOLUME.

EVALUATE TIME CONSTANT FOR SIDE 1 OF MOTOR

111 IF(PID.EQ.PA) GO TO 112

CHECK IF PRESSURIZING

112 IF PRESSURISING ASSIGN TIME CONSTANT

TAU1 = 0.20

CORRECT FOR FLOW CHARACTERISTIC GRADIENT VARIATION.

113 IF (P1.GE.5.0E5) GO TO 100
114 IF (P1.GE.3.0E5) GO TO 200
115 TAU1 = (TAU1/0.4)
116 GO TO 300
117 200 TAU1 = (TAU1/1.1)
118 GO TO 300
119 100 TAU1 = (TAU1/(8.33*(P1/1.0E5)-40.23))
120 CORRECT FOR VOLUME VARIATION

301 TAU1=TAU1*(VOL1+(XCURR/STR)*VOLC)/(VOLCAL)
302 GO TO 114
303 IF DE-PRESSURIZING SET DE-PRESSURIZATION TIME CONSTANT

304 TAU1 = 0.5
305 IF (P1.GE.5.2E5) GO TO 115
306 TAU1 = 15.0
307 IF (P1.GE.5.0E5) GO TO 115
310 TAU1=TAU1*(VOL01+(XCURR/STR)*VOLC)/(VOLCAL)
314) CFL1=-(P10-P1)*(1-E**(-(DELT)/TAU1))
315) MARKER = 5
316) IF (DEBUG NE. 0.0) WRITE (1, 19) MARKER
317) C
318) C      REPEAT ABOVE PROCEDURE FOR OTHER SIDE OF MOTOR.
319) C
320) 222 IF (P20.EQ.PA) GO TO 223
321)  TAU2 = 0.28
322) IF (P2.EQ.3.0E5) GO TO 331
323) TAU2 = (TAU2/0.4)
324) GO TO 333
325) 332 TAU2 = TAU2/1.1)
326) GO TO 333.
327) 331 TAU2 = (TAU2/8.33*(P2/1.0E5)-40.23))
328) 333 TAU2 = TAU2*(VOLU2*/(STR-XCURR)/STR)*VOLC)/(VOLCAL)
329) GO TO 224
330) 223 TAU2 = 0.5
331) IF (P2.EQ.5.2E5) GO TO 225
332) TAU2 = 15.0
333) 225 TAU2 = TAU2*(VOLU2*/(STR-XCURR)/STR)*VOLC)/(VOLCAL)
334) 224 DELP2=(P20-P2)*(.1-E**(-(DELT)/TAU2))
335) MARKER = 6
336) IF (DEBUG NE. 0.0) WRITE (1, 19) MARKER
337) C
338) C
339) 495 VEL1=VEL2
340) C
341) C
342) 590 CONTINUE
343) C
344) C
345) C
346) C
347) C
348) C
349) C
350) C
351) C
352) XZERO = 1.0
353) XMAX = 1.3
354) YZERO = -2.0
355) YMAX = 2.0
356) XPIXL = 250
357) YPIXL = 250
358) C
359) C
360) C
361) C
362) C
363) CALL GOC
364) CALL GRAPH(2, MX, MY, ML, 6, 4, 9, 2, XZERO, XMAX, 5, 2, YZERO, YMAX, 4, 1)
365) C
366) CALL MOVTO2(130., 130.)
367) C
368) C
369) YRANG = YPIXL/(YMAX-YZERO)
370) XRANG = XPIXL/(XMAX-XZERO)
371) XOFFSET = 255-XPIXL
372) YOFFSET = 255-YPIXL
DO 400 K=1,NOP

YSCALE = ((V2(K)-(YZERO))\*YRANG)+YOFFST
IF (YSCALE.LT.0.) GOTO 400
IF (YSCALE.GT.255.) GOTO 400
XSCALE = ((X(K)-XZERO)\*XRANG)+XOFFST
IF (XSCALE.LT.0.) GOTO 400
IF (XSCALE.GT.255.) GOTO 400
CALL LINT02(XSCALE,YSCALE)

CONTINUE
CALL MVTO2(170.,200.)
CALL CHAARR(MGVEL,3,2)
CALL MVBY2(0.*0.)
CALL CHAFIX(GVEL,5,2)
XSCALE = (XS - XZERO)*XRANG+XOFFST
CALL MVTO2(XSCALE,150.)
CALL LINRY2(0.,-10.)

TARGET POINT NOW MARKED IN THE GRAPH

CALL GOC

SET UP SECOND PICTURE AXES

XZERO = 0.0
XMAX = TIMSIM
YZERO = 0.0
YMAX = STR

XPIXL = 200
YPIXL = 200
XOFFST = 255-XPIXL
YOFFST = 255-YPIXL

CALL GRAPH(1,MT,MS,ML2,3,6,7,2,XZERO,XMAX,5,14,YZERO,YMAX,5,2)
CALL MVTO2(55.,55.)
TIME = 0.0

XRANG = XPIXL/(XMAX-XZERO)
YRANG = YPIXL/(YMAX-YZERO)

DO 420 K=1,NOP,4

YSCALE = ((TIME-(XZERO))*XRANG)+YOFFST
YSCALE = ((X2(K)-(YZERO))*YRANG)+YOFFST
CALL LINT02(XSCALE,YSCALE)
TIME = TIME+(DELT*4.*5)

CONTINUE
YSCALE = ((XS-(YZERO))*YRANG)+(255-YPIXL)
CALL MVTO2(55.,YSCALE)
CALL LINT02(255.,YSCALE)

DRAW IN TARGET LINE
SET UP AXIS VALUES

XZERO = 0.0
XMAX = TIMSIM
YZERO = -10
YMAX = 10
XPIXL = 200
YPIXL = 250

CALL GGC

NOW DRAW AND LABEL AXES

CALL GRAPH(3,MT,MP,ML2,3,4,7,2,XZERO,XMAX,5,1,YZERO,YMAX,8,1)
CALL MOVTO2(55.,130.)
TIME = 0.0

RENORMALISE THE Y AXIS RANGE (X10E5)

YZERO = -1E6
YMAX = 1E6

XRANG = XPIXL/(XMAX-XZERO)
YRANG = YPIXL/(YMAX-YZERO)
XOFFSET = 255-XPIXL
YOFFSET = 255-YPIXL

DO 443 K=1,NOP+4
XSCAL = ((TIME-(XZERO))*XRANG)*XOFFSET
YSCAL = ((PR1(K)-(YZERO))*YRANG)*YOFFSET
CALL LINTO2(XSCAL,YSCAL)
TIME = TIME + (DELT*4.*5)

443 440 CONTINUE

445 440 CONTINUE

449 440 CONTINUE

CALL MOVTO2(55.,130.)
TIME = 0.0

NOW DRAW P2

DO 445 K=1,NOP+4
XSCAL = ((TIME-(XZERO))*XRANG)*XOFFSET
YSCAL = ((PR2(K)-(YZERO))*YRANG)*YOFFSET
CALL LINTO2(XSCAL,YSCAL)
TIME = TIME + (DELT*4.*5)

447 445 CONTINUE

450 445 CONTINUE

TIME = 0.0
CALL MOVTO2(55.,130.)

PLOT P1-P2 ON THE SAME AXES

455 450 K=1,NOP+4
APPENDIX 4.

Listings of the routines for the updating of the manipulator variables from the Equation of State representation (MANESPR simulator) and the total differential representation (MANTDPN simulator). MANESPR and MANTDPN implement the full valve-motor representation for large displacements.
NOW PERFORM THE SIMULATION AND STORE THE RESULTS IN ARRAYS

LOOPCT=NOP+5
DO 500 N=1,LOOPCT
A2 = N2+1
N3 = N3+1
N4 = N4+1
;CYCLE COUNTERS INCREMENTED; TEST THEM LATER
CALCULATE NEW PRESSURE FOR NEXT CYCLE
(DELP1 AND DELP2 ARE CALCULATED FROM A CONSTANT VOLUME PRESSURE TRANSIENT REPRESENTATION WITH UPDATED TIME CONSTANTS TO CATER FOR THE CHANGE IN MOTOR VOLUME AS IN "MANIPR".)

DELV1 = VOLC*(DELV/STR)
VOL1 = VOLD1+VOLC*(XCURR/STR)
DELM1 = (VOL1+DELP1)/RT
FMAS1 = FMAS1 + DELM1
P1 = (FMAS1/VOL1)*RT

SET LIMITS ON POSSIBLE VALUES OF P1
IF(P1.LT.PA) P1=PA

SIMILARLY PROCESS SIDE 2'S PRESSURE CHANGE FOR NEW P2

DELV2 = VOLC*(DELV/STR)
VOL2 = VOLD2 + (STR-XCURR)/STR)*VOLC
DELM2 = (VOL2+DELP2)/RT
FMAS2 = FMAS2 + DELM2
P2 = (FMAS2/VOL2)*RT
IF(P2.LT.PA) P2=PA
MARKER = 1
IF(DEBUG.NE.0.0) WRITE(1,19) MARKER

MECHANICAL HARDWARE

IF(VEL1.NE.0.) GOTO 104 ; IS ARM STATIONARY?
FRIC=FR0
IF(P1.GT.P2) FRIC=-(FRIC) ; FRICTION OPPOSES MOVEMENT OR FORCE
IF(Abs((P1-P2)*THAX/7.14E3)*GT.ABS(FRIC)) GOTO 106 ; MOVEMENT?
FRIC = (VEL1/(ABS(VEL1)))*FR(-1)

FRICIONAL FORCE ALLOWS ONE DIRECTION OF MOVEMENT

ACC=(TMAX*(P1-P2)/7.14E5+FRIC)/KIA

FORCES (PRESSURE & FRICTION) ADDED AS VECTORS

VEL2=(ACC+DELT)*VEL1 ; NEW VELOCITY

GO TO 700

IF(VEL2.GT.2.0) VEL2 = 2.0
IF(VEL2.LT.-2.0) VEL2 = -2.0

SET LIMIT ON NEW VELOCITY

DELX=(VEL2*VEL1)*DELT/2.

IF(XCURR.GT.STR-0.1) AND VEL1.GT.0.0 DELX = 0.0
IF(XCURR.LT.STR-0.1) AND VEL1.LT.0.0 DELX = 0.0

TRAVERSE LIMIT OF MOTOR

XCURR = XCURR + DELX ; NEW POSITION

IXCODE = IFIX(XCURR*256/STR) ; GET INTEGER VALUE OF XCURR IN RANGE 1-256

IPOSN(N4) = IXCODE ; 1MS SAMPLE OF POSITION (1-256 RESOLUTION)

TIME=TIME+DELT

IF(N2.NE.5) GO TO 150 ; IF ON A 5 MS INTERVAL, UPDATE THE ARRAYS

MARKER = 2
IF(DEBUG.NE.0.0) WRITE(1,19) MARKER

X = N/5

XM+1 = XCURR

XV+1 = VEL2

XM+2 = XM+1

P1(H) = P1

P2(H) = P2

A2 = 0 ; PERFORMANCE FIGURES SAVED

CONTINUE

IF(DEBUG.EQ.0.0) GO TO 431

P1DIS = P1H/100000.

P2DIS = P2H/100000.

P1DIS = P1H/100000.

P2DIS = P2H/100000.

DISPLAY VALUES OF VALVE POSITIONS

WRITE(1,130) TIME,ACC,VEL2,XCURR,P1,P2,P1DIS,P2DIS,N2,N3

130 FORMT(F6.5,3(F9.4,1X),2(F9.4,1X),2(F3.0,1X),2(F13.1X))

WRITE(1,135) P1DIS,P2DIS,TAU1,TAU2,DELP1,DELP2

135 FORMT(2X,2F3.0,2X,2F3.0,1X,2F9.4)

CONTINUE

CONDITIONAL BRANCHING LOGIC ALGORITHM

IF(N4.NE.NPOP) GO TO 113 ; VELOCITY ESTIMATE EVERY 16 MS.

DO 260 I = 1,NPOP

ISUMPS = ISUMPS + IPOSN(I)

CONTINUE

SUM OF SEVERAL PAST POSITIONS CALCULATED
MANTODP

NOW PERFORM THE SIMULATION AND STORE THE RESULTS IN ARRAYS

   LOOPC=NOPT
   DO 500 N=1,LOOPC
   N2 = N2+1
   N3 = N3+1
   N4 = N4+1
;CYCLE COUNTERS INCREMENTED, TEST THEM LATER
   CALCULATE NEW PRESSURE FOR NEXT CYCLE

   (DELP1 AND DELP2 ARE FIRST CALCULATED FROM A CONSTANT VOLUME PRESSURE
   TRANSIENT REPRESENTATION THAT USES MODIFIED TIME CONSTANTS TO CATER
   FOR THE CHANGES IN VOLUME AS IN "MANTPR" BEFORE THE ACTUAL PRESSURE
   INCREMENT CALCULATION THAT USES THE TOTAL DIFFERENTIAL EXPRESSION.)

   DELV1 = VOLC*(DELX/STR)
   VOL1 = VOLDI+VOLC*(XCURR/STR)
   DELM1 = (VOL1+DELP1)/RT
   AMAS1 = (2.*FMAS1+DELM1)/2.*
   FMAS1 = FMAS1 + DELM1
   DELP1 = P1*((DELM1/AMAS1)-(DELV1/VOL1))
   P1 = P1+DELP1

   SET LIMITS ON POSSIBLE VALUES OF P1
   IF(P1.LT.PA) P1=PA

   SIMILARLY PROCESS SIDE 2'S PRESSURE CHANGE FOR NEW P2

   DELV2 = VOLC*(DELX/STR)
   VOL2 = VOLD2 + ((STR*XCURR)/STR)*VOLC
   DELM2 = (VOL2+DELP2)/RT
   AMAS2 = (2.*FMAS2+DELM2)/2.*
   FMAS2 = FMAS2 + DELM2
   DELP2 = P2*((DELM2/AMAS2)-(-DELV2/VOL2))
   P2 = P2+DELP2
   IF(P2.LT.PA) P2=PA
   MARKER = 1
   IF(DEBUG.NE.0) WRITE(1,19)MARKER

   MECHANICAL HARDWARE

   IF(VEL1.NE.0) GOTO 104
MA...D

187) IF(P1.GT.P2) FRIC= -(FRIC) ; FRICION OPPOSES MOVEMENT OR FORCE
189) IF(ABS((P1-P2)*TMAX/7.14E5).GT.ABS(FRIC)) GOTO 106 ; MOVEMENT?
190) C
191) C
192) 103 ACC=0.
193) GOTO 118
194) 104 FRIC=(VEL1/(ABS(VEL1)))*FR*(-1)
195) C
196) 105 FRICIONAL FORCE ALWAYS OPPOSES DIRECTION OF MOVEMENT
197) C
198) 106 ACC=(TMAX*(P1-P2)/7.14E5*FRIC)/RTIA
199) C
200) FORCES (PRESSURE & FRICTION) ADDED AS VECTORS
201) C
202) 118 VEL2=(ACC*DELT)+VEL1 ; NEW VELOCITY
203) C
204) 600 IF(VEL2.GT.2.0) VEL2 = 2.0
205) IF(VEL2.LT.-2.0) VEL2 = -2.0
206) C
207) 700 DELX=(VEL2+VEL1)*DELT/2.
208) IF(XCURR.GT.(STR-0.1).*AND.*VEL1.(T-0.0) DELX = 0.0
209) IF(XCURR.LT.0.1).*AND.*VEL1.(T-0.0) DELX=0.0 ; TRAVERSE LIMIT OF MOTOR
210) C
211) XCURR = XCURR + DELX ; NEW POSITION
212) C
213) 800 ICODE = IFIX(XCURR*256/STR) ; GET INTEGER VALUE OF XCURR IN RANGE 1-256
214) C
215) IPOSN(N4) = ICODE ; 1MS SAMPLE OF POSITION (1-256 RESOLUTION)
216) C
217) TIME=TIME+DELT ; IF ON A 5 MS INTERVAL, UPDATE THE ARRAYS
218) C
219) 150 MARKER = 2
220) IF(DEBUG.NE.0.0)WRITE(1,19)MARKER
221) M = N/5
222) X(M+1) = XCURR
223) V2(M+1) = VEL2
224) X2(M+1) = X(M+1)
225) PR1(M) = P1
226) PR2(M) = P2
227) N2 = 0 ; PERFORMANCE FIGURES SAVED
228) C
229) CONTINUE
230) 150
231) C
232) 235) IF(DEBUG.EQ.0.0)GOTO 431
233) P1NDIS = P1N/100000.
234) P2NDIS = P2N/100000.
235) P1ODIS = P1O/100000.
236) P2ODIS = P2O/100000. ; DISPLAY VALUES OF VALVE POSITIONS
237) C
238) WRITE(1,130) TIME,ACC,VEL2,XCURR,P1,P2,P1NDIS,P2NDIS,N2,N3
239) 130 FORMAT(F6.3,3(F9.4,1X),2(F9.0,1X),2(F3.0,1X),2(I3,1X))
240) C
241) WRITE(1,135)P1ODIS,P2ODIS,TAU1,TAU2,DELPI,DELP2
242) 135 FORMAT(2X,2F3.0,2X,2F8.4,2X,2F9.0)
243) C
244) 431 CONTINUE
245) C
246) C
247) CONDITIONAL BRANCHING LOGIC ALGORITHM
APPENDIX 5.

(a) Listing of relevant routines of a simple switch-over (1-0 to 1-1) forward time simulator VMSTDPR.

(b) Listing of relevant routines of a simple switch-over (1-0 to 0-1) forward time simulator (simulating braking action) VMSTDPRB.

(c) Listing of relevant routines of simple switch-over (1-0 to 1-1) reverse time simulator REVUMS.(Original version.)

(d) Listing of routines that carry out an improved termination of REVUMS.
127) 10 FORMAT(3H A=F6.4/5H DEN=F5.3/5H GUP=F5.3/4H P0=F10.2
128) */4H PA=F10.2/5H STR=F5.2/4H XS=F5.2/5H FR0=F5.2/
129) */5H TIME INTERVAL USED*DELT=F6.3/6H DZN =F7.4/8H GVEL =F5.2/8H
130) + GACC =F5.2///)
131) C
132) C
133) WRITE(1,6)
134) 6 FORMAT(1X,5H TIME, 3X,4H ACC, 3X,5H VEL2, 4X,3H X2, 5X,3H P1, 5X
135) / 3H P2, 4X, 4H P1M, 4X, 4H P2N, ///)
136) C
137) C
138) ISUMPR = 0
139) XCURR = X(1)
140) VOL1 = VOL01 + VOLC*(XCURR/STR)
141) VOL2 = VOL02 + VOLC*((STR-XCURR)/STR)
142) FMA1 = (P1*VOL1)/RT
143) FMA2 = (P2*VOL2)/RT
144) C
145) C
146) C
147) C
148) C
149) C
150) C
151) LOOPCT=NOP+5
152) C0 500 N=1, LOOPCT
153) C
154) N2 = N2+1
155) N3 = N3+1
156) N4 = N4+1
157) C
158) C
159) C
160) C
161) C
162) DELV1 = VOLC*(DELV/STR)
163) VOL1 = VOL1+DELV1
164) DELM1 = (VOL1*DELP1)/RT
165) AMAS1 = (2.*FMA1+DELM1)/2
166) FMA1 = FMA1+DELM1
167) DELP1 = P1*((DELM1/AMAS1)-(DELV1/VOL1))
168) P1 = P1+DELP1
169) C
170) C
171) C
172) C
173) C
174) C
175) C
176) C
177) C
178) C
179) C
180) C
181) C
182) C
183) C
184) C
185) C
186) C

NOW PERFORM THE SIMULATION AND STORE THE RESULTS IN ARRAYS

CALCULATE NEW PRESSURE FOR NEXT CYCLE

DELV2 = VOLC*(DELV/STR)

DELV2 = VOL2-DELV2

DELV2 = (VOL2*DELP2)/RT

AMAS2 = (2.*FMA2+DELM2)/2

FMA2 = FMA2+DELM2

DELP2 = P2*((DELM2/AMAS2)-(DELV2/VOL2))

P2 = P2+DELP2

IF(P2.LT.PA) P2=PA

MARKER = 1

IF(DEBUG.EQ.2.0) WRITE(1,19) MARKER
169) C
191) C
192) C
FRIC=FRO
194) IF(PI.GT.P2) FRIC=-(FRIC)
195) C
196) IF(ABS((P1-P2)*TMAX/7.14E5).GT.ABS(FRIC)) GOTO 106
197) C
198) C
199) 103 ACC=0.
200) GOTO 118
201) 104 IF(VEL1.LT.0.03) FR=0.0
202) IF(VEL1.LE.0.03) FR=6.0
203) C
204) FRICTIONAL FORCE ALWAYS OPPOSES DIRECTION OF MOVEMENT
205) C
206) 105 ACC=(TMAX*(P1-P2)/7.14E5+FRIC)/PTIA
207) C
208) C
209) C
210) 118 VEL2=(ACC*DELT)+VEL1
211) C
212) GOTO 730
213) 666 IF(VEL2.GT.2.0) VEL2 = 2.0
214) IF(VEL2.LT.-2.0) VEL2 = -2.0
215) C
216) 700 DELX=(VEL2+VEL1)*DELT/2.
217) IF(XCURR.GT.(STR-0.1) .AND. VEL1.GT.0.0) DELX = 0.0
218) IF(XCURR.LT.0.1 .AND. VEL1.LT.0.0) DELX=0.0
219) C
220) XCURR = XCURR + DELX
221) C
222) TIME=TIME+DELT
223) IF (N2.NE.5) GOTO 150
224) C
225) MARKER = 2
226) IF(DEBUG.EQ.2.0) WRITE(1,19)MARKER
227) 327 M = N/5
228) X(M+1) = XCURR
229) Y(M+1) = VEL2
230) X2(M+1) = X(M+1)
231) PR1(M) = P1
232) PR2(M) = P2
233) N2 = 0
234) C
235) 150 CONTINUE
236) C
237) C
238) IF(DEBUG.EQ.0.0) GOTO 431
239) 999 P1NDIS = P1N/100000.
240) P2DIS = P1N/100000.
241) P1DDIS = P1D/100000.
242) P2DDIS = P2D/100000.
243) C
244) WRITE(1,130) TIME,ACC,VEL2,XCURR,P1,P2,P1NDIS,P2NDIS,N2,N3
245) 130 FORMAT(F6.3,3(F9.4,1X),2(F9.4,1X),2(F3.0,1X),2(13,1X))
246) C
247) WRITE(1,135)P1DDIS,P2DDIS,TAU1,TAU2,DELP1,DELP2
248) 135 FORMAT(2X,2F3.0,2X,2F8.4,2X,2F9.0)
CONTINUE

SWITCH - OVER ROUTINE
XSO DENOTES THE SWITCH-OVER POSITION AND IS SUPPLIED EARLIER IN THE
PROGRAMME.

IF (XCURR.GE.XSO) GO TO 202
P1N=P0
P2N=PA
SIDE 1 PRESSURISED
GO TO 113
202 P1N = P0
P2N = P9
BOTH SIDES PRESSURISED.

ALL OTHER PARTS OF THE PROGRAMME ARE AS IN "MANTDPR".

MARKER = 3
IF (DEBUG.EQ.2.0) WRITE(1,19) MARKER

RESPONSE DELAY OF THE VALVE PAIR

113 IF (P1N.EQ.P1U) GO TO 131
114 IF (P1N.EQ.P1V) GO TO 110
J=ICYOF1
P1V=P1N
119 J=J-1
120 IF (J.GE.0) GO TO 131
P10=P1V
121 IF (P2N.EQ.P20) GO TO 111
122 IF (P2N.EQ.P2V) GO TO 121
K=ICYDF2
P2V=P2N
126 K=K-1
127 IF (K.GE.0) GO TO 111
128 P2D=P2V
MARKER = 4
IF (DEBUG.EQ.2.0) WRITE(1,19) MARKER

NOW DETERMINE THE PRESSURE CHANGES IN EACH SIDE OF THE MOTOR

111 EVALUATE TIME CONSTANT FOR SIDE 1 OF MOTOR
112 IF (P1D.EQ.PA) GO TO 112
CHECK IF PRESSURIZING

IF PRESSURISING ASSIGN TIME CONSTANT

120 TAU1 = 0.20
GO TO 300

CORRECT FOR FLOW CHARACTERISTIC

117 IF (P1.GE.5.0E5) GO TO 100
118 IF (P1.GE.3.0E5) GO TO 200
304 TAU1 = (TAU1/0.4)
GO TO 300
TAU1 = (TAU1/(8.33*(P1/1.0E5)-0.23))

TAU = TAU1*(VOLD1*(XCURR/STR)*VOLC)/(VOLCAL)

50 TO 114

IF DE-PRESSURIZING SET DE-PRESSURIZATION TIME CONSTANT

TAU1 = 0.50

TAU1 = TAU1*(VOLD1*(XCURR/STR)*VOLC)/(VOLCAL)

EVALUATE PRESSURE CHANGE.

DELP1 = (P1-P1)**(1.0E**(-(DELT)/TAU1))

MARKER = 5

IF (DEBUG.EQ.2.0) WRITE(1,19)MARKER

REPEAT ABOVE PROCEDURE FOR OTHER SIDE OF MOTOR.

IF (P2D.EQ.PA) GO TO 223

TAU2 = 0.20

GO TO 333

IF (P2.6E-5.0E5) GO TO 331

IF (P2.6E-3.0E5) GO TO 332

TAU2 = (TAU2/0.4)

GO TO 333

TAU2 = (TAU2/1.1)

GO TO 333

TAU2 = (TAU2/(8.33*(P2/1.0E5)-0.23))

TAU2 = TAU2*(VOLD2*(STR-XCURR)/STR)*VOLC)/(VOLCAL)

GO TO 224

TAU2 = 0.50

TAU2 = TAU2*(VOLD2*(STR-XCURR)/STR)*VOLC)/(VOLCAL)

DELP2 = (P2D-P2)**(1.0E**(-(DELT)/TAU2))

MARKER = 6

IF (DEBUG.EQ.2.0) WRITE(1,19)MARKER

VEL1 = VEL2

CONTINUE

NOW PLOT THE RESULTS HELD IN THE ARRAYS

SET UP THE GRAPH VALUES AND AXIS RANGES

XZERO = 0.0

XMAX = STR

YZERO = -3.0

YMAX = 3.0

XPIXL = 250

YPIXL = 250

; [X/Y]PIXL = NO. OF PIC. ELEMENTS PER AXIS
*4VEL,CACC
**3*1.34/5H_DEN=*F5.3/5H GUP=,*F5.3/4H P0=,*E10.2
*+4H PA=,*F10.2/5H STR=,*F5.2/4H XS=,*F5.2/5H FR0=,*F5.2/7
*25H TIME INTERVAL USED+DELT=,*F6.3/6H DZN =,*F7.4/8H GVEL =,*F5.2/8H
*GACC =,*F5.2//**

WRITE(1,6)
**3*1.3+5H TIME, 3X,4H ACC,3X,5H VEL,2X,4X,3H X2,5X,3H P1,5X,
*3H P2,4X,4H PIN,4X,4H P2N,**

ISUMPR = 0
XCURR = X(1)
VOL1 = VOLD1 + VOLC*(XCURR/STR)
VOL2 = VOLD2 + VOLC*((STR-XCURR)/STR)
FAM1 = (P1*VOL1)/RT
FAM2 = (P2*VOL2)/RT

NOW PERFORM THE SIMULATION AND STORE THE RESULTS IN ARRAYS.

LOOPCT=N0P*5
DO 500 N=1,LOOPCT
N2 = N2+1
N3 = N3+1
N4 = N4+1
;CYCLE COUNTERS INCREMENTED, TEST THEM LATER
CALCULATE NEW PRESSURE FOR NEXT CYCLE:

DELV1 = VOLC*(DELX/STR)
VOL1 = VOL1+DELV1
DELM1 = (VOL1*DELP1)/RT
AMAS1 = (2.*FAM1+DELM1)/2.
FAM1 = FAM1 + DELM1
DELP1 = P1*-(DELM1/AMAS1)-(DELV1/VOL1))
P1 = P1 + DELP1

SET LIMITS ON POSSIBLE VALUES OF P1
IF(P1.LT.PA) P1=PA

SIMILARLY PROCESS SIDE 2'S PRESSURE CHANGE FOR NEW P2

DELV2 = VOLC*(DELX/STR)
CHANGE IN VOLUME ON SIDE 2 IS OF OPPOSITE SIGN TO MOTION.

VOL2 = VOL2-DELV2
DELM2 = (VOL2*DELP2)/RT
AMAS2 = (2.*FAM2+DELM2)/2
FAM2 = FAM2 + DELM2
DELP2 = P2*-(DELM2/AMAS2)-(-DELV2/VOL2))
P2 = P2 + DELP2
IF(P2.LT.PA) P2=PA
MARKER = 1
IF(DEBUG.NE.0.0) WRITE(1,19)MARKER
IF(VEL1 .NE. 0.) GOTO 104 ; IS ARM STATIONARY?
FRIC=FRO
IF(P1.GT.P2) FRIC=-(FRIC) ; FRICTION OPPOSES MOVEMENT OR FORCE
IF(ABS((P1-P2)*TMAX/7.14E5).*GT.ABS(FRIC)) GOTO 106 ; MOVEMENT?

103 ACC=0.
GOTO 118
104 FRIC=(VEL1/(ABS(VEL1)))*FR*(-1)
FRICIONAL FORCE ALWAYS OPPOSES DIRECTION OF MOVEMENT

106 ACC=(TMAX*(P1-P2)/7.14E5+FRIC)/RTIA
FORCES (PRESSURE & FRICTION) ADDED AS VECTORS

118 VEL2=(ACC*DELT)+VEL1 ; NEW VELOCITY
IF(XCURR.GT.XS) GO TO 900

120 GOTO 789
IF(VEL2.GT.2.0) VEL2 = 2.0
IF(VEL2.LT.-2.0) VEL2 = -2.0
SET LIMIT ON NEW VELOCITY

125 DELX=(VEL2+VEL1)*DELT/2.
IF(XCURR.GT.(STR-0.1).AND.VEL1.GT.0.0) DELX = 0.0
IF(XCURR.LT.0.1.00.0.1) DELX=0.0
; TRAVERSE LIMIT OF MOTOR

130 XCURR = XCURR + DELX ; NEW POSITION
GO TO 910
DELX=0.0
VEL2=1.0

150 XCURR=XCURR+DELX
NEW XCURRENT

160 TIME=TIME+DELT ; IF ON A 5 MS INTERVAL UPDATE THE ARRAYS
IF (M2.NE.5) GOTO 150
MARKER = 2
IF (DEBUG.NE.0.0) WRITE(1,19) MARKER
M = N/9
X(M+1) = XCURR
V2(M+1) = VEL2
X2(M+1) = X(M+1)
PR1(M) = P1
PR2(M) = P2
N2 = 0 ; PERFORMANCE FIGURES SAVED
CONTINUE

180 IF (DEBUG.EQ.0.0) GOTO 431
P1INDIS = P1N/100000.
P2INDIS = P2N/100000.
P1DIS = P1D/100000.
P2DDIS = P20/100000.
; DISPLAY VALUES OF VALVE POSITIONS
WRITE(1,130) TIME ACC VEL2 XCURR P1 P2 P
190 CONTINUE

200 PRINT*,
END

210 STOP
WRITE(1,135)P1D1S,P2D1S,TAU1,TAU2,DELP1,DELP2
WRITE(1,140)DELM1,DELM2,FMAS1,FMAS2,DELV1,DELV2,VOL1,VOL2
FORMAT(2X,2F3.0,2X,2F8.4,2X,2F9.0)
FORMAT(2X,8(F11.10,2X))
CONTINUE

SWITCH-OVER ROUTINE (ALTERNATIVE MODE).
IF(TARCHK.NE.0.0)GO TO 203
IF(XCURR.GE.XSO)GO TO 202
P1N=PA
P2N=PA
SIDE 1 PRESSURISED
GO TO 113
202 P1N=PA
P2K=PG
SIDE 2 PRESSURISED.
GO TO 113
203 P1N=PA
P2N=PA
BOTH SIDES PRESSURISED
MARKER = 3
IF(DBG,NE.0.0)WRITE(1,19)MARKER
RESPONSE DELAY OF THE VALVE PAIR
113 IF(P1N.EQ.P1D)GO TO 131
IF(P1N.EQ.P1V)GO TO 110
J=ICYOF1
IF(P1N.EQ.P0)J=ICYON1
P1V=P1N
110 J=J-1
IF(J.GE.0)GO TO 131
P1D=P1V
131 IF(P2N.EQ.P2D)GO TO 111
IF(P2N.EQ.P2V)GO TO 121
K=ICYOF2
IF(P2N.EQ.P0)K=ICYON2
P2V=P2N
121 K=K-1
IF(K.GE.0)GO TO 111
P2D=P2V
MARKER = 4
IF(DBG,NE.0.0)WRITE(1,19)MARKER
NOW DETERMINE THE PRESSURE CHANGES IN EACH SIDE OF THE MOTOR
11 EVALUATE TIME CONSTANT FOR SIDE 1 OF MOTOR
IF(P1D.EQ.PA)GO TO 112
CHECK IF PRESSURIZING
IF PRESSURISING ASSIGN TIME CONSTANT
TAU1 = 0.20
GO TO 300
CORRECT FOR FLOW CHARACTERISTIC
IF(P1.GE.5.0E5)GO TO 100
AL1 = (TAU1 0.4)  
GO TO 300  
TAU1 = (TAU1/1.1)  
GO TO 300  
TAU1 = (TAU1/(8.33*(P2/1.0E5)-40.23))  
C  
CORRECT FOR VOLUME VARIATION  
300  
TAU1=TAU1*(VOLD1+(XCURR/STR)*VOLC)/(VOLCAL)  
GO TO 114  
C  
IF DE-PRESSURIZING SET DE-PRESSURIZATION TIME CONSTANT  
112  
TAU1 = 0.50  
115  
TAU1=TAU1*(VOLD1+(XCURR/STR)*VOLC)/(VOLCAL)  
C  
EVALUATE PRESSURE CHANGE.  
114  
DELP1=(P1D-P1)*(1-E**(-(DELT)/TAU1))  
MARKER = 5  
IF (DEBUG .NE. 0.0) WRITE(1,19)MARKER  
C  
REPEAT ABOVE PROCEDURE FOR OTHER SIDE OF MOTOR.  
222  
IF (P2D.EQ.PA) GO TO 223  
TAU2 = 0.20  
GO TO 333  
IF (P2.EQ.5.0E5) GO TO 331  
IF (P2.EQ.3.0E5) GO TO 332  
TAU2 = (TAU2/0.9)  
GO TO 333  
332  
TAU2 = (TAU2/1.1)  
GO TO 333  
331  
TAU2 = (TAU2/(8.33*(P2/1.0E5)-40.23))  
333  
TAU2=TAU2*(VOLD2+(STR-XCURR)/STR*VOLC)/(VOLCAL)  
GO TO 224  
223  
TAU2 = 0.50  
225  
TAU2=TAU2*(VCLD2+(STR-XCURR)/STR*VOLC)/(VOLCAL)  
224  
DELP2=(P2D-P2)*(1-E**(-(DELT)/TAU2))  
MARKER = 6  
IF (DEBUG .NE. 0.0) WRITE(1,19)MARKER  
C  
VEL1=VEL2  
C  
CONTINUE  
C  
NOW PLOT THE RESULTS HELD IN THE ARRAYS  
C  
SET UP THE GRAPH VALUES AND AXIS RANGES  
C  
XYZERO = 0.0  
XMAX = STR  
YZERO = -3.0  
YMAX = 3.0  
XPIXL = 250  
YPIXL = 250
DELT=0.01 ; TIME BETWEEN SIMULATION CYCLES
TMSIM=5.0 ; TOTAL SIMULATION TIME

RT = 283.02*293

P0=5.00E5
PA=1.0E5
F1=465000
P2=535000
F10=P0
P20=P0
PR1(1)=P1
PR2(1)=P2

: INITIAL PRESSURE SETTINGS

X(1)=X3 ; INITIAL STARTING POSITION

VEL1=0.0
V2(1)=0.
TIME=0.

WRITE(1,2)

2 FORMAT(/8H MANIPULATOR SIMULATION/27H

***28H THE INITIAL CONDITIONS ARE://

WRITE(1,10) A,DEN,GUP,P0,PA,STR,XS,F10,FR,HMO1,DELT,D2N

+3VEL,GACC

10 FORMAT(3H A=,F6.4/5H DEN=,F5.3/5H GUP=,F5.3/4H P0=,F10.2

+4H PA=,F10.2/5H STR=,F5.2/4H XS=,F5.2/5H FR0=,F5.2/

+4H TIME INTERVAL USED=,F6.3/6H D2N=,F7.4/8H GVEL =,F5.2/8H

+6GACC =,F5.2/7//

WRITE(1,6)

6 FORMAT(1X,5H TIME, 3X,4H ACC,3X,5H VEL2,4X,3H X2,5X,3H P1,5X,

+3X P2,4X,4H PIN,4X,4H P2N,///)

ISUMPR = 0
XCURR = X(1)
VOL1 = VOLD1 + VOLC*(XCURR/STR)
VOL2 = VOLD2 + VOLC*((STR-XCURR)/STR)
FMA1 = (P1*VOL1)/RT
FMA2 = (P2*VOL2)/RT

RSFTN = 0.0

NOW INITIALISE PRESSURES ACCORDING TO VOLUME
F2=(P0-P1)*(VOL2/VOL1)*P0

SIMULATOR STOP TEST NUMBER

NOW PERFORM THE SIMULATION AND STORE THE RESULTS IN ARRAYS

(Underlined statements denote operations that are altered for reverse time simulation.)
DO 500 N=1,LOOPCT

N2 = N2+1
N3 = N3+1
N4 = N4+1

; CYCLE COUNTERS INCREMENTED, TEST THEM LATER

; CHECK FOR SIMULATOR STOP

IF(RSFIN.NE.0.0) GO TO 900

; IF POSITIVE BY-PASS ALL CALCULATIONS

CALCULATE NEW PRESSURE FOR NEXT CYCLE

DFLV1 = VOLC*(DELX/STR)
VOL1 = VOL1+DFLV1
TAU1 = 0.20
TAU1 = TAU1*(VOL1/VOLCAL)
CELP1 = (P1D-P1)*((1-E**(-(DELX)/TAU1)))
DELM1 = (VOL1*DELP1)/RT
AMAS1 = (2.*FMA1-DELM1)/2.
FMA1 = FMA1 - DELM1
IF(FMA1.LT.0.0) GO TO 800

CHECK FOR UNREALISTIC SIMULATION. (Improved REVUMS)

DFLP1 = P1*((-DELM1/AMAS1)-(DFLV1/VOL1))
P1 = P1 + DELP1

SET LIMITS ON POSSIBLE VALUES OF P1

IF(P1.LT.PA) P1=PA
IF(P1.GE.6.0E5) P1=6.0E5

SIMILARLY PROCESS SIDE 2'S PRESSURE CHANGE FOR NEW P2

DFLV2 = VOLC*(DELX/STR)

VOL2 = VOL2-DFLV2
TAU2 = 0.20
TAU2 = TAU2*(VOL2/VOLCAL)
DFLP2 = (P2D-P2)*((1-E**(-(DELX)/TAU2)))
DELM2 = (VOL2*DELP2)/RT
AMAS2 = (2.*FMA2-DELM2)/2.
FMA2 = FMA2 - DELM2
IF(FMA2.LT.0.0) GO TO 800

CHECK FOR UNREALISTIC SIMULATION

DFLP2 =P2*((DELM2/AMAS2)-(DFLV2/VOL2))
P2 =P2 + UCLP2
IF(P2.GE.6.0E5) P2=6.0E5
IF(P2.LT.PA) P2=PA
MARKER = 1
IF(DEBUG.EQ.2.0) WRITE(1,19)MARKER

MECHANICAL HARDWARE
189) \( 17.14 \times 10^5 / \text{RTIA} \times \text{LT} \times 1.70 \) GO TO 801
190) \( \text{FRIC} = 0.0 \)
191) IF(VEL1.GT.0.03) FRIC=(VEL1/(ABS(VEL1))) \ast (\text{FR}) \ast (-1) \) Improved friction representation.
192) C
193) C
194) C
195) C FORCES (PRESSURE & FRICTION) ADDED AS VECTORS
196) C
197) C
198) C VEL2=(ACC*(-DELT))+VEL1 : NEW VELOCITY
199) 790 DELX=(VEL2+VEL1)*(-DELT)/2.
200) IF(XCURR.GT.(STR-0.1)) DELX=0.0
201) IF(XCURR.LT.0.1) DELX=0.0 ; TRAVERSE LIMIT OF MOTOR
202) C XCURR = XCURR + DELX ; NEW POSITION
203) GO TO 900
204) C IF SIMULATOR IS NO LONGER REALISTIC MAINTAIN EXISTING VALUES.
205) C (ORIGINAL VERSION OF REVVMES DOES NOT CONTAIN THIS CHECK.)
206) C
207) C 899 PSFIN = 1.0
208) DELP1 = 0.0
209) P1 = P1 + DELP1
210) DELP2 = 0.0
211) P2 = P2 + DELP2
212) ACC = 0.0
213) VEL2 = (ACC*(-DELT))+VEL1
214) DELX = 0.0
215) XCURR = XCURR + DELX \nu \nu \nu \nu \nu
216) GO TO 790
217) 800 VEL2=0.0
218) ACC=0.0
219) DELX=0.0
220) TIME=TIME+DELT
221) IF(N2.NE.5) GO TO 150 ; IF ON A 5 MS INTERVAL, UPDATE THE ARRAYS
222) C MARKER = 2
223) IF(DEBUG.EQ.0.0) WRITE(1,19)MARKER
224) K = N/5
225) X(M+1) = XCURR
226) V2(M+1) = VEL2
227) X2(M+1) = X(M+1)
228) PR1(M) = P1
229) PR2(M) = P2
230) K2 = 0 ; PERFORMANCE FIGURES SAVED
231) C
232) C
233) 150 IF(DEBUG.EQ.0.0) GO TO 431
234) 999 P1NDIS = P1IN/100.000.
235) P2NDIS = P2IN/100.000.
236) P1NDIS = P1IN/100.000.
237) P2NDIS = P2IN/100.000.
238) C ; DISPLAY VALUES OF VALVE POSITIONS
239) WRITE(1,130) \( \text{TIME}, \text{ACC}, \text{VEL2}, \text{XCURR}, \text{P1}, \text{P2}, \text{P1NDIS}, \text{P2NDIS}, N2, N3 \)
240) 130 FORMAT(F6.3,3(F9.4,1X),2(F9.0,1X),2(F3.0,1X),2(13,1X))
241) C
WRITE(1,140)DELM1,DELM2,FMAS1,FMAS2,DELV1,DELV2,VOL1,VOL2
FORMAT(2X,8(F11.10,2X))
CONTINUE

! REMARKER = 3
IF(DEBUG.EQ.2.0)WRITE(1,19)MARKER
MARKER = 6
IF(DEBUG.EQ.2.0)WRITE(1,19)MARKER
VOL1=VOL2
CONTINUE

NOW PLOT THE RESULTS HELD IN THE ARRAYS
SET UP THE GRAPH VALUES AND AXIS RANGES

XZERO = 0.0
XMAX = STR
YZERO = -3.0
YMAX = 3.0
XPIXL = 250
YPIXL = 250

; [X/Y]PIXL = NO. OF PIC. ELEMENTS PER AXIS

NOW DRAW THE AXES
CALL CE906N
CALL SCALE(0.7)
CALL GRAPH(2,MY,ML,6,4,9,2,XZERO,XMAX,5,2,YZERO,YMAX,4,1)
CALL MOVT02(136.,139.)

YRANG = YPIXL/(YMAX-YZERO)
XRANG = XPIXL/(XMAX-XZERO)
XOFFSET = 255-XPIXL
YOFFSET = 255-YPIXL

PLOTTING CONSTANTS NOW EVALUATED

DO 400 K=1,400

YSCAL = ((V2(K)-(YZERO))*YRANG)+YOFFSET
IF (YSCAL.LT.0.) GOTO 400
IF (YSCAL.GT.255.) GOTO 400
XSCAL = ((X(K)-XZERO)*XRANG)+XOFFSET
IF (XSCAL.LT.0.) GOTO 400
IF (XSCAL.GT.255.) GOTO 400

CALL LINT02(XSCAL,YSCAL)
CONTINUE

400 CONTINUE
Determination of the dynamic friction of the arm.

As no adequate representation of the dynamic friction in the motor existed, it was decided to use the data acquisition system (Refs. [66], [72]) in order to derive a relationship between the dynamic friction and the velocity of the arm. The following procedure was used for this:

The trajectory of the experimental arm (under the control of a simple switch-over sequence) was monitored by the data acquisition system and a position transient and transients of the pressures $P_1$ and $P_2$ on the two sides of the motor were obtained. The position transient was differentiated twice with respect to time to obtain velocity and acceleration transients. Pressure transient data was manipulated in order to extract a transient of the driving torque of the motor from the pressure difference ($P_1 - P_2$).

The velocity and acceleration transients are shown in Figures A6:1 and A6:2 respectively. The driving torque transient is shown in Figure A6:3.

The acceleration transient data was then divided by the moment of inertia of the arm in order to obtain a transient of the effective driving torque on the arm. These values were subtracted from the corresponding driving torques in order to obtain a transient of the frictional torque. The software that carries out the above operations is described in Refs.
Friction values from this transient were plotted against the corresponding values of velocity and pressure difference (P1-P2). A plot of friction vs. velocity is shown in Figure A6:4. A plot of friction vs. pressure difference (P1-P2) is shown in Figure A6:5.

An examination of Figure A6:4 shows that the friction remains essentially constant throughout the velocity range. No friction values are plotted in the low velocity region. This is due to the lack of reliable acceleration values in this region. The inability to calculate acceleration values accurately at low rates of displacement is a basic shortcoming of the software that is used for this operation (ref. [76]).

Because of this, the original dynamic friction representation (which assumes the dynamic friction to be constant for all values of velocity) cannot be improved. The value of the constant friction torque was estimated (from Figure A6:4) at 6.00 Nm. This value was supplied to the simulator packages.

Because it was originally thought that the seals which are present in the motor vane contribute significantly to the dynamic friction, the plot of friction vs. pressure difference across the vane (P1-P2) was examined in order to establish whether an unambiguous relationship between dynamic friction and pressure differential exists. No such relationship could be derived from Figure A6:5. It was therefore decided not to include a representation of the dynamic friction as a function of pressure differential across the motor in the simulator.
Figure A6:1. Velocity transient of the experimental arm under the control of a simple switch-over sequence.
Figure A6:2. Acceleration transient of the experimental arm under the control of a simple switch-over sequence.
Figure A6.3. Pressure difference transient of the experimental arm under the control of a simple switch-over sequence.
Figure A6.4. Plot of velocity vs. dynamic friction torque for the experimental arm.
Figure A6:5. Plot of pressure difference torque vs. dynamic friction torque for the experimental arm.
This Appendix contains tables of switch-over positions corresponding to various start and target positions. It also lists the pressures on both sides of the motor (P1 and P2) and their difference from the supply pressure (P0), taken at the point in time when the arm first decelerates to zero velocity, for the same values of start and target positions. Start, target and switch-over positions are expressed as fractions of STR where STR is the angular displacement corresponding to the swept volume of the motor.

Starting point at 0.05*STR

<table>
<thead>
<tr>
<th>Switch-over *STR</th>
<th>Target *STR</th>
<th>P1 P0-P1</th>
<th>P2 P2-P0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.065</td>
<td>0.19</td>
<td>498563</td>
<td>500861</td>
</tr>
<tr>
<td>0.080</td>
<td>0.26</td>
<td>494361</td>
<td>504336</td>
</tr>
<tr>
<td>0.100</td>
<td>0.33</td>
<td>490764</td>
<td>507922</td>
</tr>
<tr>
<td>0.125</td>
<td>0.39</td>
<td>487169</td>
<td>511264</td>
</tr>
<tr>
<td>0.150</td>
<td>0.45</td>
<td>484172</td>
<td>513583</td>
</tr>
<tr>
<td>0.175</td>
<td>0.51</td>
<td>481620</td>
<td>515209</td>
</tr>
<tr>
<td>0.200</td>
<td>0.56</td>
<td>479435</td>
<td>516323</td>
</tr>
<tr>
<td>0.250</td>
<td>0.64</td>
<td>474963</td>
<td>517847</td>
</tr>
<tr>
<td>0.300</td>
<td>0.67</td>
<td>471486</td>
<td>517978</td>
</tr>
<tr>
<td>0.400</td>
<td>0.86</td>
<td>465393</td>
<td>515715</td>
</tr>
</tbody>
</table>
### Starting Point at 0.25*STR

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.256</td>
<td>0.33</td>
<td>499047</td>
<td>50050</td>
<td></td>
</tr>
<tr>
<td>0.270</td>
<td>0.41</td>
<td>491374</td>
<td>504705</td>
<td></td>
</tr>
<tr>
<td>0.280</td>
<td>0.45</td>
<td>488885</td>
<td>507100</td>
<td></td>
</tr>
<tr>
<td>0.310</td>
<td>0.56</td>
<td>483475</td>
<td>511569</td>
<td></td>
</tr>
<tr>
<td>0.350</td>
<td>0.64</td>
<td>478405</td>
<td>514366</td>
<td></td>
</tr>
<tr>
<td>0.390</td>
<td>0.73</td>
<td>474339</td>
<td>515401</td>
<td></td>
</tr>
<tr>
<td>0.430</td>
<td>0.80</td>
<td>470973</td>
<td>515226</td>
<td></td>
</tr>
<tr>
<td>0.470</td>
<td>0.86</td>
<td>468128</td>
<td>514166</td>
<td></td>
</tr>
</tbody>
</table>

### Starting Point at 0.45*STR

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.461</td>
<td>0.56</td>
<td>490900</td>
<td>503920</td>
<td></td>
</tr>
<tr>
<td>0.470</td>
<td>0.61</td>
<td>487443</td>
<td>506816</td>
<td></td>
</tr>
<tr>
<td>0.480</td>
<td>0.64</td>
<td>484837</td>
<td>508760</td>
<td></td>
</tr>
<tr>
<td>0.505</td>
<td>0.73</td>
<td>479887</td>
<td>511441</td>
<td></td>
</tr>
<tr>
<td>0.530</td>
<td>0.80</td>
<td>476474</td>
<td>512170</td>
<td></td>
</tr>
<tr>
<td>0.560</td>
<td>0.86</td>
<td>472965</td>
<td>512078</td>
<td></td>
</tr>
</tbody>
</table>
(a) Listings of REVCON incorporating the original position-velocity switch-over criterion.

(b) Listings of REVCON file incorporating the modified (convergence) switch-over criterion.
103 ACC = 0.
GOTO 118

104 FRIC = (VEL1/(ABS(VEL1))) * FRIC * (-1)
   FRICITIONAL FORCE ALWAYS OPPOSES DIRECTION OF MOVEMENT

106 ACC = (TMAX * (P1 - P2) / 7.14E5 + FRIC) / RTIA
   FORCES (PRESSURE & FRICTION) ADDED AS VECTORS

118 VEL2 = (ACC * DELT) + VEL1
   ; NEW VELOCITY
   DELX = (VEL2 * VEL1) * DELT / 2.
   IF (XCURR * GT (STR - 0.1) AND. VEL1 * GT 0.0) DELX = 0.0
   IF (XCURR * LT 0.01 AND. VEL1 * LT 0.0) DELX = 0.0
   ; TRAVERSE LIMIT OF MOTOR

   XCURR = XCURR + DELX
   ; NEW POSITION
   TIME = TIME + DELT
   IF (N2 * NE 5) GOTO 150
   ; IF ON A 5 MS INTERVAL, UPDATE THE ARRAYS

   MARKER = 2
   IF (DEBUG * NE 0.0) WRITE(1, 19) MARKER
   M = N / 5
   X(M+1) = XCURR
   V2(M+1) = VEL2
   X2(M+1) = X(M+1)
   P1(M) = P1
   P2(M) = P2
   N2 = 0
   ; PERFORMANCE FIGURES SAVED

150 CONTINUE

999 IF (DEBUG * EQ 0.0) GOTO 431
   PINDIS = PIN / 100000.
   P2NDIS = P2N / 100000.
   P1DIS = P1D / 1000000.
   P2DDIS = P2D / 1000000.
   ; DISPLAY VALUES OF VALVE POSITIONS
   WRITE(1, 130) TIME, ACC, VEL2, XCURR, P1, P2, PINDIS, P2NDIS, N2, N3
   FORMAT(FE. 3, 3(F9.4, 1X), 2(F9.0, 1X), 2(F13.1X))

130 WRITE(1, 135) PINDIS, P2NDIS, TAU1, TAU2, DELP1, DELP2
   FORMAT(2X, 2(F9.0, 1X), 2X, 2(F9.0, 1X))

135 WRITE(1, 140) DELM1, DELM2, FMAS1, FMAS2, DELV1, DELV2, VOL1, VOL2
   FORMAT(2X, 8(F11.10, 2X))

140 CONTINUE

431 GO TO 699

DICTATED SWITCH-OVER

IF (XCURR GE XSO) GO TO 202
   PIN = P0
   P2N = P0

202 SIDE 1 PRESSURISED
   GO TO 113
   PIN = P0
   P2N = P0

113 BOTH SIDES PRESSURISED.
BEGIN

IF (SIND $\neq 0.0$) GO TO 711
IF (XCURR $\leq$ RX(IRVARC)) GO TO 720
IRVARC = IRVARC - 1
IF (XCURR $\leq$ RX(IRVARC)) GO TO 710
GO TO 790

P1N = P0
P2N = PA
SIND = 1.0
GO TO 113

BOTH SIDES PRESSURISED

P1N = P0
P2N = PA
SIDE 1 PRESSURISED

RESPONSE DELAY OF THE VALVE PAIR

113 IF (P1N $\equiv$ P1D) GO TO 131
IF (P1N $\equiv$ P1V) GO TO 110
J = ICYOF1
IF (P1N $\equiv$ P0) J = ICYON1
P1V = P1N

110 J = J - 1
IF (J $\geq$ 0.0) GO TO 131
P1D = P1V

131 IF (P2N $\equiv$ P2D) GO TO 111
IF (P2N $\equiv$ P2V) GO TO 121
K = ICYOF2
IF (P2N $\equiv$ P0) K = ICYON2
P2V = P2N

121 K = K - 1
IF (K $\geq$ 0.0) GO TO 111
P2D = P2V
MARKER = 4
IF (DEBUG $\neq 0.0$) WRITE (1, 19) MARKER

NOW DETERMINE THE PRESSURE CHANGES IN EACH SIDE OF THE MOTOR

EVALUATE TIME CONSTANT FOR SIDE 1 OF MOTOR

11 IF (P1D $\equiv$ P1A) GO TO 112

CHECK IF PRESSURIZING

IF PRESSURISING ASSIGN TIME CONSTANT

TAU1 = 0.20
GO TO 300

CORRECT FOR FLOW CHARACTERISTIC

IF (P1 $\geq$ 5.0E5) GO TO 100
IF (P1 $\geq$ 3.0E5) GO TO 200
TAU1 = (TAU1 / 0.4)
GO TO 300

00 TAU1 = (TAU1 / 1.1)
GO TO 300

00 TAU1 = (TAU1 / 0.33 * (P1 / 1.0E5 - 40.23))

CORRECT FOR VOLUME VARIATION

END
THIS IS AN IMPROVED VERSION OF "REVCON" THAT UTILISES THE REVERSE TIME SIMULATOR MORE EFFECTIVELY.

"REVCON" IS A PACKAGE THAT CARRIES OUT A SIMULATION OF THE MANIPULATOR ARM BASED ON THE "VMSTOD" PACKAGE. INSTEAD OF USING A PREDETERMINED SWITCH-OVER POINT DATA FROM A REVERSE TIME SIMULATION BASED ON "OBTEST" IS STORED IN AN ARRAY. WHEN THE ARM'S CURRENT POSITION MATCHES ONE STORED IN THE ARRAY AND ITS VELOCITY IS WITHIN 3% OF THAT STORED IN THE DATA FOR THIS POSITION SWITCH-OVER OCCURS.

IN THIS WAY THE DECREASING CONVERGENCE BETWEEN THE FORWARD AND REVERSE TIME SIMULATORS WHICH OCCUR BECAUSE OF SWITCH-OVER IS FULLY UTILISED.

THESE STATEMENTS HOLD THE GRAPH TITLES (2 PER ARRAY ELEMENT)

DATA MX(1)/2HPO/, MX(2)/2HSI/, MX(3)/2HTI/, MX(4)/2HON/,  
* MX(5)/2H-/-/, MX(6)/2HRO/, MY(1)/2HVE/, MY(2)/2HL-/, MY(3)/2HRD/,  
* MY(4)/2H-/-/, ML(1)/2HVE/, ML(2)/2HL0/, ML(3)/2HCI/, ML(4)/2HTY/,  
* ML(5)/2H-/-/, MGVEL(1)/2HGV/, MGVEL(2)/2HEL/, MGVEL(3)/2HE=/,  
* MT(1)/2HTI/, MT(2)/2HME/, MT(3)/2H-S/,  
* ML2(1)/2HPO/, ML2(2)/2HSI/, ML2(3)/2HTI/, ML2(4)/2HON/,  
* ML3(1)/2H-/-/, ML3(2)/2HIM/, ML3(3)/2HCE/,  
* MP(1)/2HPR/, MP(2)/2HEC/, MP(3)/2HSU/, MP(4)/2HRC/,  
* DTERM=0.0  

FORMAT(8HGET HERE, 14)  

VOLD1 = 0.000076  
VOLD2 = 0.000076  
VOLC = 0.000264  
VOLCAL = 0.000300  

CALIBRATED DEAD VOLUMES AND SWEEP VOLUMES FOR MOTOR  
DEBUG = 0.0  
PI = 3.14159265359  
E = 2.7182818  
NOP = 999  
; NO. OF CYCLES IN SIMULATION  
STR = PI/2  
; STROKE OF ACTUATING CYLINDER  
XS = STR*865/1000  
; TARGET POSITION FOR MECHANISM  
FR = 6.0  
; DYNAMIC FRICTION  
FRO = 8.70  
; STATIC FRICTION  
TMAX = 104.24  
; MAXIMUM TORQUE  
RTIA = 4.8  
; INERTIA
DELT=0.001  ; TIME BETWEEN SIMULATION CYCLES
TMSIM=5.0  ; TOTAL SIMULATION TIME
RT = 283.02-293.

N=1
N2=0
N3=0
N4=0

REVERSE TIME ARRAY COUNTERS.
PN0=5.0E5
PA=1.0E5
P1=464800
P2=516500
PID=PN0
P2D=PD0
RX(1)=XS
DELY=0.0
VEL1=0.0
RV2(1)=0.0
TIME=0.0

REVERSE TIME ARRAY GENERATION INITIALISED

XCURR=RX(1)
VOL1=VOL01+VOLC*(XCURR/STR)
VOL2=VOL02+VOLC*((STR-XCURR)/STR)
FMAS1=(P1*VOL1)/RT
FMAS2=(P2*VOL2)/RT

"FINAL POINT" PARAMETER SETTING
RSFIN=0.0
REVERSE TIME SIMULATOR STOP TEST NUMBER

NOW PERFORM REVERSE TIME SIMULATION AND STORE THE RESULTS IN ARRAYS

LOOPCT=NOP*5
DO 850 N=1,LOOPCT
N2=N2+1
N3=N3+1

REVERSE CYCLE COUNTERS INCREMENTED; TEST THEM LATER

IF (RSFIN .NE. 0.0) GO TO 900

IF CHECK POSITIVE BY PASS ALL CALCULATIONS

CALCULATE PRESSURE

DELV1=VOLC*(DELY/STR)
VOL1=VOL1+DELV1
TAU1=0.20
TAU1=TAU1*(VOL1/VOLCAL)
DELP1=(PI-P1)*(1-E**(-(DELT)/TAU1))
DELM1=(VOL1*DELP1)/RT
IF (FMS1.LT.0.0) GO TO 800
DELN1=P1*(-DELM1/AMAS1)-(DELV1/VOL1)
P1=P1+DELN1

C SET LIMITS ON POSSIBLE VALUES OF P1
IF (P1.LT.PA) P1=PA
IF (P1.GE.6.0E5) P1=6.0E5

C SIMILARLY EVALUATE PRESSURES ON SIDE 2 OF THE MOTOR
DELV2=VOLC*(DELX/STR)

C CHANGE IN VOLUME ON SIDE 2 IS OF OPPOSITE SIGN TO MOTION
VOL2=VOL2-DELV2
TAU2=0.20
TAU2=TAU2*(VOL2/VOLCAL)
DELN2=(P20-P2)*(1-E**(-DELT)/TAU2)
AMAS2=(2.*FMS2-DELM2)/2
FMS2=FMS2-DELM2
IF (FMS2.LT.0.0) GO TO 800
DELV2=P2*((-DELM2/AMAS2)-(DELV2/VOL2))
P2=P2+DELV2
IF (P2.GE.6.0E5) P2=6.0E5
IF (P2.LT.PA) P2=PA

MECHANICAL HARDWARE

FRIC=0.0
IF (VEL1.NE.0.0) FRIC=(VEL1/(ABS(VEL1)))*FR*(-1)

ACC=(TMAX*(P1-P2)/7.14E5+FRIC)/RTIA
VEL2=(ACC+(-DELT))*VEL1
DELN=(VEL2+VEL1)*(-DELT)/2
IF (XCURR.GT.(STR-0.1)) DELX=0.0
IF (XCURR.LT.-0.1) DELX=0.0

C TRAVERSE LIMIT OF MOTOR
XCURR=XCURR+DELX
GO TO 900

IF REVERSE SIMULATOR STOP CHECK IS POSITIVE
MAINTAIN ALL PARAMETER VALUES

RSFIN=1.0
P1=P1+DELN1
DELN2=0.0
P2=P2+DELN2
ACC=0.0
VEL2=(ACC+(-DELT))*VEL1
DEL1=0.0
XCURR=XCURR+DELX
TIME=TIME+DELT
IF (N2.NE.5) GO TO 750

M=N/5
RX(M+1)=XCURR
RV2(M+1)=VEL2
N2=0

VEL1=VEL2
REVERSE TIME POSITION AND VELOCITY ARRAYS GENERATED
INITIALISE FORWARD SIMULATION

CONTROL, VELOCITY AND ARRAY UPDATE CYCLE COUNTERS

REVERSE ARRAY COUNTER

ARRAY SEARCH INDICATOR

INITIAL PRESSURE SETTINGS

INITIAL STARTING POSITION

SWITCH-OVER POSITION.

DELX = 0.0

X2 (1) = X (1)

VC = 0.05

VDC = 0.05

VDD = 0.05

ICV = IFIX (VDC / DELT)

ICY = IFIX (VDC / DELT)

ICYON = IFIX (VDD / DELT)

ICYO2 = IFIX (VDD / DELT)

NPOPN = 16

NFACT = (NPOPN + 2)

VELOC = (NFACT / 2)

VELOC = (NFACT / 2)

TIME = 0.0

WRITE (1, 2)

2 FORMAT (///27H MANIPULATOR SIMULATION/27H

******* //28H THE INITIAL CONDITIONS ARE://)

WRITE (1, 10) A, DEN, GUP, P0, PA, STR, XS, FR0, FR, HM, DELT, DZN

*GVEL, GACC

10 FORMAT (///3H A, F6.4/5H DEN, F5.3/5H GUP, F5.3/4H P0, E10.2

+4H PA, F10.2/5H STR, F5.2/4H XS, F5.2/5H FR0, F5.2/

+4H FR, F5.2/6H MASS, F4.2/)
WRITE(1, 6)
6  FORMAT(1X, 5H TIME, 3X, 4H ACC, 3X, 5H VEL2, 4X, 3H X2, 5X, 3H P1, 5X, *3H P2, 4X, 4H P1N, 4X, 4H P2N, //)

ISUMPR = 0
XCURR = X(1)
VOL1 = VOLD1 + VOLC*(XCURR/STR)
VOL2 = VOLD2 + VOLC*((STR-XCURR)/STR)
FMAS1 = (P1*VOL1)/RT
FMAS2 = (P2*VOL2)/RT

NOW PERFORM THE SIMULATION AND STORE THE RESULTS IN ARRAYS

LOOPCT=MOP*5
DO 500 N=1, LOOPCT

N2 = N2+1
N3 = N3+1
N4 = N4+1

CYCLE COUNTERS INCREMENTED, TEST THEM LATER
CALCULATE NEW PRESSURE FOR NEXT CYCLE

DELV1 = VOLC*(DELX/STR)
VOL1 = VOL1+DELV1
DELN1 = (VOL1*DELPP1)/RT
AMAS1 = (2.*FMAS1+DELN1)/2
FMAS1 = FMAS1 + DELN1
DELPP1 = P1*((DELN1/AMAS1)-(DELV1/VOL1))
P1 = P1 + DELPP1

SET LIMITS ON POSSIBLE VALUES OF P1
IF(P1.LT.PA) P1=PA

SIMILARLY PROCESS SIDE 2'S PRESSURE CHANGE FOR NEW P2

DELV2 = VOLC*(DELX/STR)
VOL2 = VOL2-DELV2
DELN2 = (VOL2*DELPP2)/RT
AMAS2 = (2.*FMAS2+DELN2)/2
FMAS2 = FMAS2 + DELN2
DELPP2 = P2*((DELN2/AMAS2)+DELV2/VOL2))
P2 = P2 + DELPP2
IF(P2.LT.PA) P2=PA
PARKER = 1
IF(DEBUG.NE.0.0) WRITE(1,19)MARKER

MECHANICAL HARDWARE

IF(VEL1.NE.0.0) GOTO 104
IF (P1.GT.P2) FRIC = -(FRIC)
IF (ABS((P1-P2)*TMAX/7.14E5) GOTO 106; MOVEMENT?

GOTO 118

FRIC = (VEL1/(ABS(VEL1)))*FRIC

FFHICTIONAL FORCE ALWAYS OPPOSES DIRECTION OF MOVEMENT

ACC = TMAX*(P1-P2)/7.14E5+FRIC)/RTIA

FORCES (PRESSURE & FRICTION) ADDED AS VECTORS

VFL2 = (ACC*DELTA)*VEL1

; NEW VELOCITY

DELD = (VEL2+VEL1)*DELTA/2.

IF (XCURR.GT.(STR-0.1).AND.VEL1.GT.0.0) DELX = 0.0

IF (XCURR.LT.0.1.1. AND.VEL1.LT.0.0) DELX = 0.6

; TRAVERSE LIMIT OF MOTOR

XCURR = XCURR + DELX

; NEW POSITION

TIME = TIME + DELT

IF (N2.GT.0.0) GO TO 150

; IF ON A 5 MS INTERVAL UPDATE THE ARRAYS

MARKER = 2

IF (DEBUG.GT.0.0) WRITE(1,19) MARKER

M = N/5

X(N+1) = XCURR

V(N+1) = VEL2

X(N+1) = X(N+1)

PR1(M) = P1

PR2(M) = P2

K2 = 0

; PERFORMANCE FIGURES SAVED

CONTINUE

CONTINUE

IF (DEBUG.EQ.0.0) GOTO 431

P1DIS = P1N/1000000.

P2DIS = P2N/1000000.

P1DDIS = P1D/1000000.

P2DDIS = P2D/1000000.

; DISPLAY VALUES OF VALVE POSITIONS

WRITE (1,130) TIME, Acc, VEL2, XCURR, P1, P2, P1DIS, P2DIS, N2, N3

FORMAT(F6.3,3(F9.4,1X),2(F9.4,1X),2(F9.4,1X)),2(F13.1X))

WRITE (1,135) P1DDIS, P2DDIS, TAU1, TAU2, DELP1, DELP2

FORMAT(2X,2F3.0,2X,2F3,0,2X,2F3,0),2(F9.4,1X)

WRITE (1,140) DELM1, DELM2, FMAS1, FMAS2, DELV1, DELV2, VOL1, VOL2

FORMAT(2X,4(F11.10),2X)

CONTINUE

CONTINUE

GO TO 599

; FACILITY FOR DICTATED SWITCH-OVER

IF (XCURR.EQ.X0) GO TO 202

P1N = P0

P2N = PA

SIDE 1 PRESSURISED
C  P2N = P0  BOTH SIDES PRESSURISED.
GO TO 113

C  REVERSE TIME ARRAY SEARCH AND SWITCH-OVER ROUTINE

C  IF(SIND NE 0.0) GO TO 711
C  IF(XCURR LT RX(IRVARC)) GO TO 720
C  IRVARC=IRVARC-1
C  IF(XCURR LE RX(IRVARC)) GO TO 710
C  GO TO 700
C  IF(ABS(VEL2-RV2(IRVARC)) GT (0.03*VEL2)) GO TO 720
C  P1N=P0
C  P2N=P0
C  SIND=1.0
C  GO TO 113

C  BOTH SIDES PRESSURISED

C  P1N=P0
P2N=PA

C  RESPONSE DELAY OF THE VALVE PAIR

C  IF(P1N.EQ.P1D) GO TO 131
C  IF(P1N.EQ.P1V) GO TO 110
C  J=ICYOF1
C  IF(P1N.EQ.P0) J=ICYON1
C  P1V=P1N
C  J=J-1
C  IF(J.GE.0) GO TO 131
C  P1D=P1V
C  IF(P2N.EQ.P2D) GO TO 111
C  IF(P2N.EQ.P2V) GO TO 121
C  K=ICYOF2
C  IF(P2N.EQ.P0) K=ICYON2
C  P2V=P2N
K=K-1
C  IF(K.GE.0) GO TO 111
C  P2D=P2V
C  MARKER = 4
C  IF(DEBUG NE 0.0) WRITE(1,19)MARKER
C  NOW DETERMINE THE PRESSURE CHANGES IN EACH SIDE OF THE MOTOR

C  EVALUATE TIME CONSTANT FOR SIDE 1 OF MOTOR
C  IF(P1D.EQ.PA) GO TO 112

C  CHECK IF PRESSURIZING

C  IF PRESSURISING ASSIGN TIME CONSTANT
C  TAU1 = 0.20
C  GO TO 300

C  CORRECT FOR FLOW CHARACTERISTIC

C  IF (P1 .GE .5 .0E5) GO TO 100
C  IF(P1 .GE .3 .0E5) GO TO 200
C  TAU1 = (TAU1/0.4)
C  GO TO 300
GO TO 300  
TAU1 = (TAU1/(8.33*(P1/1.0E5)-40.23))  

CORRECT FOR VOLUME VARIATION  

TAU1=TAU1*(VOL1+((XCURR/STR)*VOL)/(VOLCAL))  
GO TO 114  

IF DE-PRESSURIZING SET DE-PRESSURIZATION TIME CONSTANT  
TAU1 = 0.50  

TAU1=TAU1*(VOL1+((XCURR/STR)*VOL)/(VOLCAL))  

EVALUATE PRESSURE CHANGE.  

DPLP1=(P10-P1)*(1-E**((-DELT)/TAU1))  
MARKER = 5  
IF (DEBUG NE.0.0) WRITE(1,19)MARKER  

REPEAT ABOVE PROCEDURE FOR OTHER SIDE OF MOTOR.  

IF(P20.EQ.PA) GO TO 223  
TAU2 = 0.20  
GO TO 333  
IF (P2.EQ.5.0E5) GO TO 331  
IF (P2.EQ.3.0E5) GO TO 332  
TAU2 = (TAU2/0.4)  
GO TO 333  
TAU2 = (TAU2/8.33*(P2/1.0E5)-40.23))  
TAU2=TAU2*((VOL2+((STR-XCURR)/STR)*VOL)/(VOLCAL))  
GO TO 324  
TAU2 = 0.50  
TAU2=TAU2*((VOL2+((STR-XCURR)/STR)*VOL)/(VOLCAL))  
DEL2=(P20-P2)*((1-E**((-DELT)/TAU2))  
MARKER = 6  
IF (DEBUG NE.0.0) WRITE(1,19)MARKER  

VEL1=VEL2  

CONTINUE  

NOW PLOT THE RESULTS HELD IN THE ARRAYS  
SET UP THE GRAPH VALUES AND AXIS RANGES  

XZERO = 0.0  
XMAX = STR  
YZERO = -3.0  
YMAX = 3.0  
XPIXL = 250  
YPIXL = 250  

; [X/Y]PIXL = NO. OF PIC. ELEMENTS PER AXIS
CALL CC906N
CALL SCALE(0.7)
CALL GRAPH(Z, XMAX, YMAX, 6, 4, 9, 2, XZERO, YZERO, YMAX, 4, 1)
CALL MOVTO2(130, 130)

YRANG = YPIXL / (YMAX - YZERO)
XRANG = XPIXL / (XMAX - XZERO)
XOFFSET = 255 - XPIXL
YOFFSET = 255 - YPIXL

PLOTTING CONSTANTS NOW EVALUATED

DO 400 K = 1, 400

YSCAL = ((V2(K) - (YZERO)) * YRANG) + YOFFSET
IF (YSCAL LT 0.) GOTO 400
IF (YSCAL GT 255.) GOTO 400
XSCAL = ((X(K) - XZERO) * XRANG) + XOFFSET
IF (XSCAL LT 0.) GOTO 400
IF (XSCAL GT 255.) GOTO 400

CALL LINTO2(XSCAL, YSCAL)

CONTINUE:

FORWARD SIMULATION PLOTTED

PLOT REFERENCE REVERSE TIME SIMULATION

XSCAL = (R(K) - XZERO) * XRANG + XOFFSET
YSCAL = (RV2(K) - YZERO) * YRANG + YOFFSET
CALL MOVTO2(XSCAL, YSCAL)

DO 410 K = 1, 400

YSCAL = (RV2(K) - (YZERO)) * YRANG + YOFFSET
IF (YSCAL LT 0.) GOTO 410
IF (YSCAL GT 255.) GOTO 410
XSCAL = (R(K) - XZERO) * XRANG + XOFFSET
IF (XSCAL LT 0.) GOTO 410
IF (XSCAL GT 255.) GOTO 410

CALL LINTO2(XSCAL, YSCAL)

CONTINUE:

CALL MOVTO2(170, 200)
CALL CHAARR(MGVEL, 3, 2)
CALL MOVXY216.0
CALL CHAPX6(GVEL, 5, 2)
XSCAL = (XS - XZERO) * XRANG + XOFFSET
CALL MOVTO2(XSCAL, 150)
CALL LINBY2(0.0, -10)

TARGET POINT NOW MARKED IN THE GRAPH

CALL PICCLE

SET UP SECOND PICTURE AXES

XZERO = 0.0
XMAX = TIMSIM/2
YZERO = 0.0
YMAX = STR
XPIXL = 200
YPIXL = 200
XOFFSET = 255-XPIXL
YOFFSET = 255-YPIXL
CALL GRAPH(1,MT,ML,4,7,2,XZERO,XMAX,5,1,YZERO,YMAX,5,2)
TIME = 0.0
XRANG = XPIXL/(XMAX-XZERO)
YRANG = YPIXL/(YMAX-YZERO)
DO 420 K=1,400,2
XSCAL = ((TIME-(XZERO))*XRANG)+XOFFSET
YSCAL = ((X2(K)-(YZERO))*YRANG)+YOFFSET
CALL LINTO2(XSCAL,YSCAL)
TIME = TIME+(DELT*2.*5)
CONTINUE
YSCAL = ((X2-(YZERO))*YRANG)+(255-YPIXL)
CALL LINTO2(255,YSCAL) ; DRAW IN TARGET LINE
CALL PICCLE // CLEAR THE SCREEN
NOW THE THIRD SET OF GRAPHS, P1, P2 AND P1-P2
SET UP AXIS VALUES
XZERO = 0.0
XMAX = 2.0
YZERO = 0.0
YMAX = 6.0
XPIXL = 200
YPIXL = 250
NOW DRAW AND LABEL AXES
CALL GRAPH(1,MP,ML,4,7,2,XZERO,XMAX,5,1,YZERO,YMAX,8,1)
CALL MOVT02(55,130)
TIME = 0.0
RENORMALISE THE Y AXIS RANGE (X10E5)
YZERO = 0.0E6
YMAX = 0.6E6
XRANG = XPIXL/(XMAX-XZERO)
YRANG = YPIXL/(YMAX-YZERO)
XOFFSET = 255-XPIXL
YOFFSET = 255-YPIXL
DO 440 K=1,400,2
XSCAL = ((TIME-(XZERO))*XRANG)+XOFFSET
CALL LINT02(XSCAL,YSCAL)
623) TIME = TIME+(DELT*2.*5)
624) C
625) 440 CONTINUE
626) C
627) CALL MOVT02(55.,130.)
628) TIME = 0.0
629) C
630) C
NOW DRAW P1
631) C
632) DO 445 K=1,400.2
633) C
634) XSCAL = ((TIME-(XZERO))*XRANG)*XOFFST
635) YSCAL = ((PRI(K)-(YZERO))*YRANG)*YOFFST
636) CALL LINT02(XSCAL,YSCAL)
637) TIME = TIME + (DELT*2.*5)
638) C
639) 445 CONTINUE
640) C
641) TIME = 0.0
642) CALL MOVT02(55.,130.)
643) C
644) C
PLOT P1-P2 ON THE SAME AXES
645) GO TO 450
646) C
647) DO 450 K=1,400.2
648) C
649) XSCAL = ((TIME-(XZERO))*XRANG)*XOFFST
650) YSCAL = ((PRI(K)-PR2(K)-(YZERO))*YRANG)*YOFFST
651) CALL LINT02(XSCAL,YSCAL)
652) TIME = TIME + (DELT*2.*5)
653) C
654) 450 CONTINUE
655) C
656) C
657) CALL DEVCN
658) STOP
659) END
660) C
661)