SIMULATION AND 3D ANIMATION OF A SIX-DEGREE-OF-FREEDOM MOTION BASE SYSTEM FOR FLIGHT SIMULATORS

By

GISELE GUIMARÃES

Engenheiro Eletricista

Universidade Federal de Goiás

Goiânia, Goiás

1984

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1988

Thesis
1988
G963A
cop. 2

Oklahoma State Univ. Library

SIMULATION AND 3D ANIMATION OF
A SIX-DEGREE-OF-FREEDOM
MOTION BASE SYSTEM FOR
FLIGHT SIMULATORS

Thesis Approved:

Martin T Hagan
Thesis Adviser

Richard L. Cummins

Ronald D. Childs

Norman N. Durham
Dean of the Graduate College

1019825

ACKNOWLEDGMENTS

I wish to express my gratitude to Dr. M. Hagan, my major adviser for his interest, guidance, patience and invaluable suggestions; and to Dr. Cummins and Dr. Rhoten for being members of the committee. My special thanks to Nidal Sammur for his help and encouragement. Special thanks to David Schooley who took the time in proof reading this thesis. And for all those who made this project possible.

I am most greatful to my husband Jose Vicente who brought me here for my Master's study and helped me so much in not beeing homesick. My thanks to my brother who helped us in understanding more the USA. And thank you mom and dad in making it possible for us to come, for your support and understanding of being away for a long period. Thank you sis for being a company to our parents.

I would also like to thank my brothers and sisters-in-law and my mother-in-law in their understanding of my husband's absence during our studies here.

I hope that my effort in the completion of this work corresponds to theirs.

TABLE OF CONTENTS

LIST OF TABLES

Figure Page

CHAPTER I

INTRODUCTION

This is part of an ongoing research project at OSU funded by Flight Safety International to design and analyze a digital motion base system.

A flight simulator is used to train pilots. The object is to fool them into thinking they are flying in a real aircraft without ever getting off the ground. With an accurate simulation training can take place in a very realistic environment, without danger to the pilot.

In some simulators the only indication to the pilot that the aircraft is in motion is through the visual display and the instruments [1]. But the acceleration cues felt by the pilot in flight also affect his response, so many simulators attempt to reproduce these cues by mounting the simulator cockpit on a moving motion base [1].

A moving-base flight simulator consists of a motion base which has an upper moving platform and a fixed lower platform. The upper platform is attached to the lower platform by hydraulic cylinders or actuators that extend and retract, making possible the motion of the simulator. The cockpit is mounted on top of the moving upper platform.

Since the simulator is driven by cylinders that have constrained movement, it is confined to a fixed volume and is not able to move as much as the real aircraft [2]. This is the major difficulty of moving-base simulators [2]. To make this simulator effective we need to use this volume as best as possible.

Simulators are built with different degrees-of-freedom of movement of the motion base. The six-degree-of-freedom simulators have been adopted for most training simulators, although it has not been experimentally established why [1]. The motion base that was used has six cylinders. Therefore it has six-degrees-of-freedom and it can move in the directions of three mutually perpendicular axes and also rotate about these axes. The rotations are called roll (x axis), pitch (y axis) and yaw (z axis). A six-degree-of-freedom motion base is shown in Figure 1.1.

The points where the hydraulic cylinders attach to the platforms form an equilateral triangle in each platform as it can be seen in Figure 1.2. In this figure it is also shown how the axes described above are defined with respect to the simulator platform.

For the pilot to have the same sensations as if he were in the real aircraft a simulator attempts to produce the same accelerations as the real aircraft would produce in response to the pilot's commands [1].

To achieve these accelerations we designed a motion system consisting of the following blocks:

- primary motion
- special effects
- motion control
- servo

The primary motion block transforms the accelerations coming from the host computer, which has the main aircraft simulation, into accelerations of the motion base, taking into account its constraints of movement.

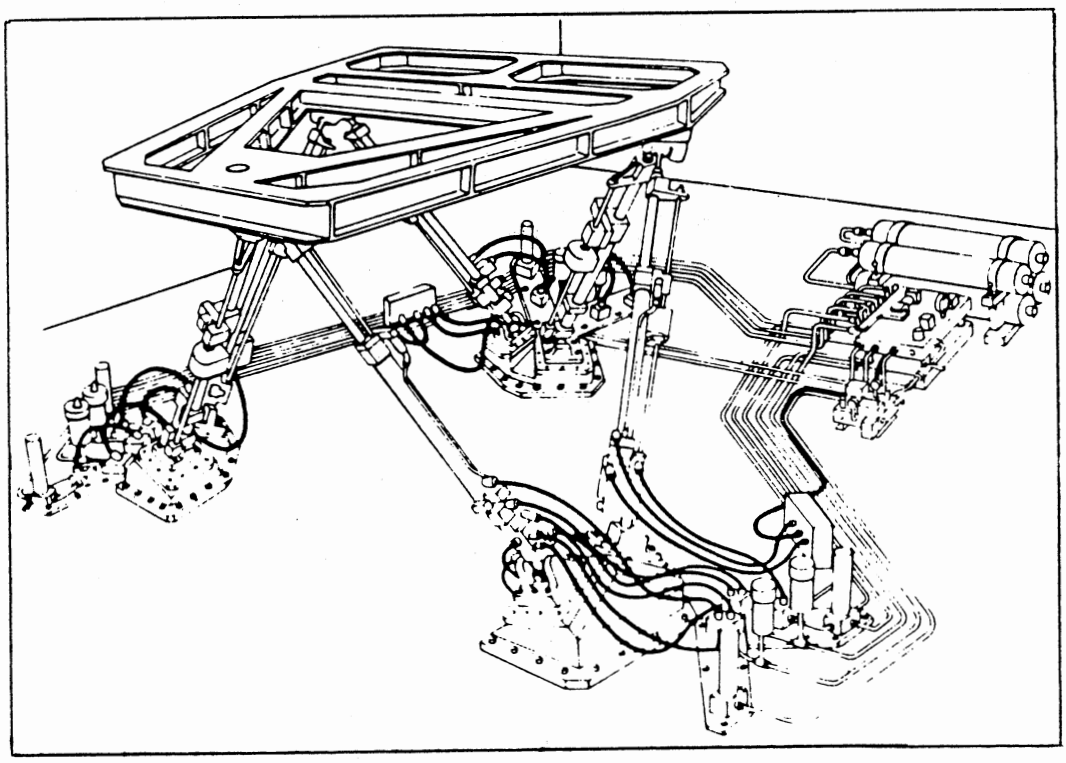


Figure 1.1. A Six Degree-of-Freedom Motion Base [1]

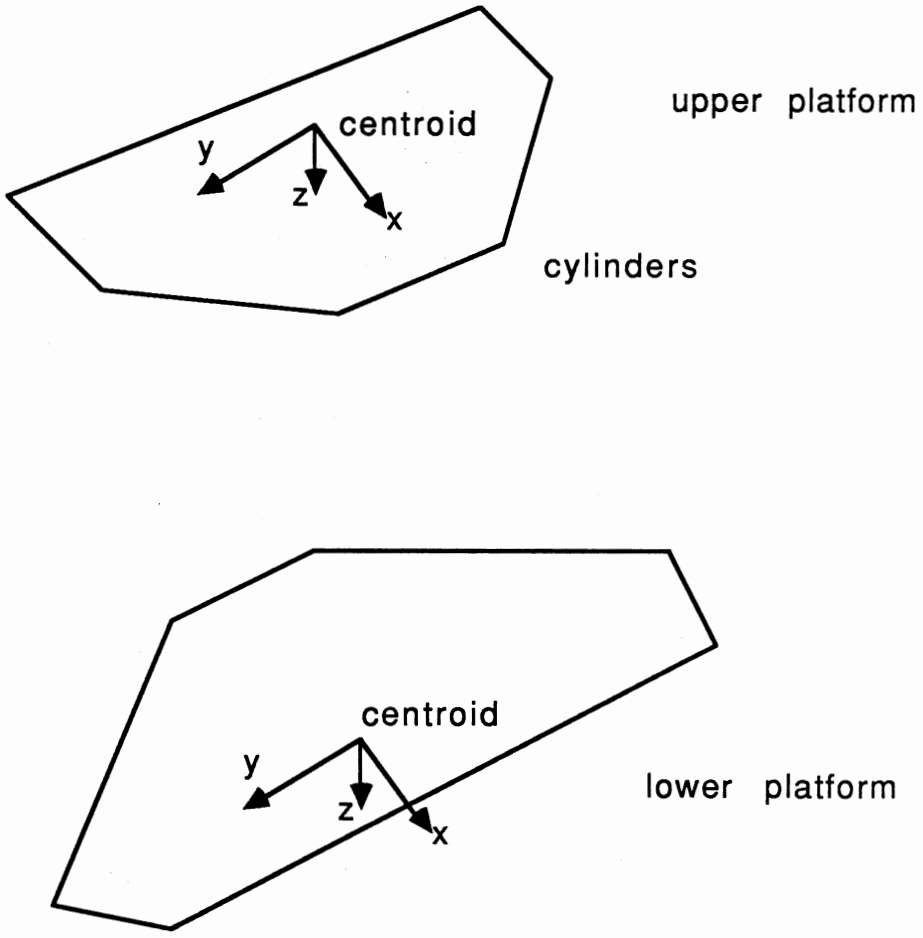


Figure 1.2. Definition of the Coordinate Systems

The special effects block creates additional accelerations which correspond to small disturbances that affect the aircraft, like runway bumps and landing gear movements.

The motion control block creates tests to be performed on the hydraulic cylinders, fades signals in and out on flight resets and crashes, brings the motion base to its neutral position, and also calibrates feedback position of the servo, given by position transducers, for better control. However, these functions of the motion control block do not need to be modeled. The parts of the motion control block that are of interest are the geometric transformation and the smoothing algorithm.

The geometric transformation, part of the motion control block, transforms upper platform positions and orientations into cylinder extensions. The smoothing algorithm prevents the cylinders from moving beyond their limits and also ensures that the cylinders reach their maximum extensions with zero acceleration.

The servo block takes the desired cylinder positions from the motion control block and attempts to move the cylinders smoothly to these positions.

The system operates with the host computer reading the pilot's commands given through controls such as the wheel, pedals and throttle. The host computer contains the flight simulation program, which produces aircraft attitudes, rates and accelerations. The host sends accelerations and rates to the primary motion block and flags to the special effects block. The special effects and primary motion, each independently, calculate a position and an angular orientation for the upper platform. The outputs of the special effects and the primary

motion are summed, filtered and transformed into cylinder extensions in the motion control block. The servo then commands the motion base with the desired cylinder extensions given by the motion control. A diagram of this system is shown in Figure 1.3.

A simulation of this motion system (MOTSIM) was developed in order to obtain a better understanding of the whole system, in order to be able to test the effects of various elements of the system on overall performance. With this simulation new designs can be tested without building hardware. This program will be used as a tool aiding engineers in the design of a motion base system for a flight simulator. The main object of this thesis is to describe the motion system simulation (MOTSIM).

To simulate the motion system it is necessary to simulate each of the blocks represented in Figure 1.3. A diagram of the simulated system (MOTSIM) is shown in Figure 1.4. Here we have almost the same diagram as Figure 1.3, except that the motion base is replaced by the valve/cylinder combination and the motion control block by the geometric transformation and smoothing algorithm block (the simulated functions of the motion control block). In addition,an inverse geometric transformation is performed to obtain true platform position and orientation from true cylinder lengths (as opposed to desired cylinder lengths).

In Chapter II the primary motion block is defined. In Chapter III the special effects are described, and a procedure for simulating them is developed. In Chapter IV the geometric transformation and the smoothing algorithm, which make up the motion control block, are discussed.

Chapter V explains the operation of the servo. Chapter VI discusses the simulation of the valve/cylinder combination. In Chapter VII the inverse geometric transformation is analyzed. Chapter VIII is a summary of the entire simulation. It explains how the various blocks interact with each other. A conclusion and suggestions for further improvements are presented in the final chapter.

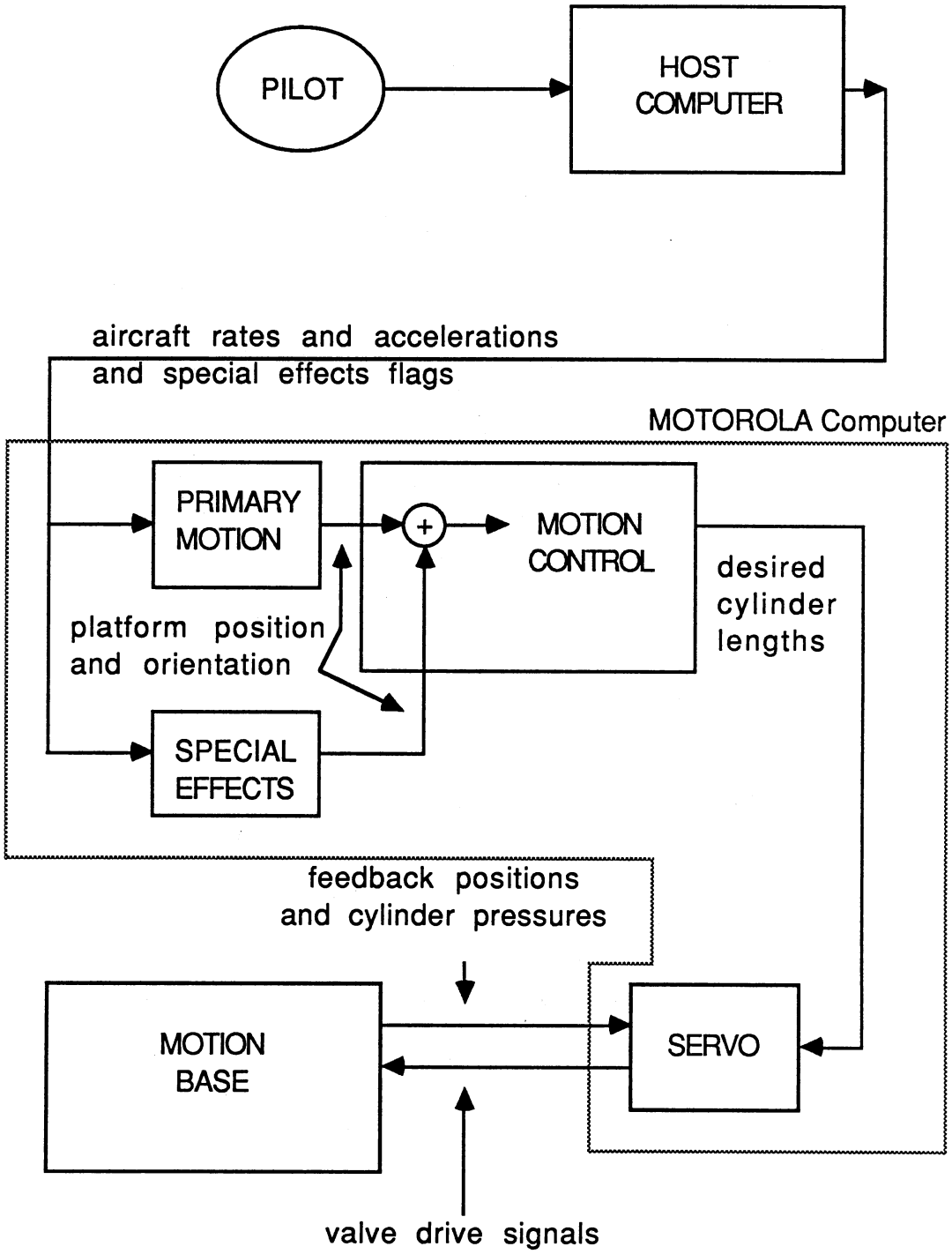
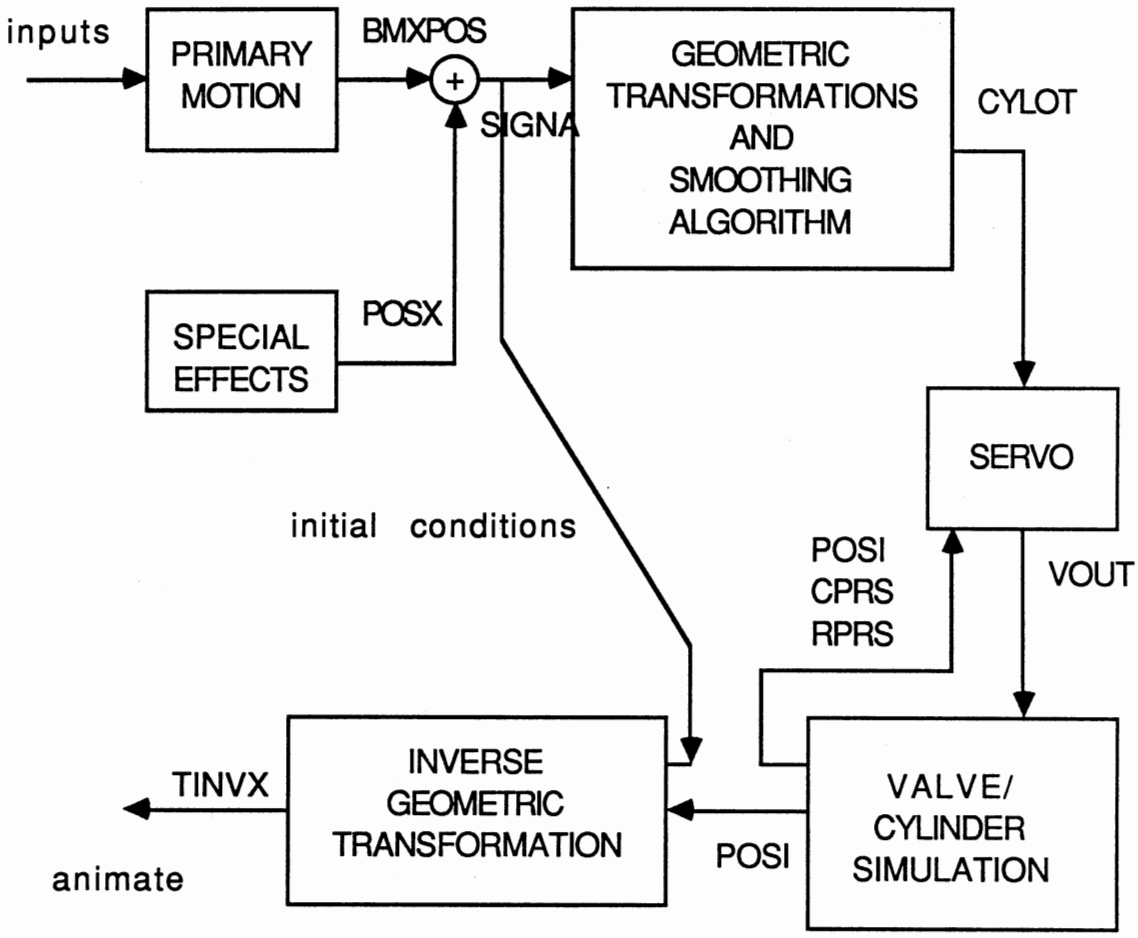


Figure 1.3. Digital Motion Base System



BMXPOS - position and angles of orientation of centroid of upper platform
POSX - position and angles of orientation of centroid of upper platform
SIGNA = BMXPOS + POSX
CYLOT - desired cylinder extensions
POSI - actual cylinder extensions
CPRS - cap pressure
RPRS - rod pressure
VOUT - voltage to drive valves
TINVX - position and angles of orientation of centroid of upper platform

Figure 1.4 Diagram of Simulated System

CHAPTER II

PRIMARY MOTION

As was seen in Chapter I the output of the main aircraft simulation consists of signals representing its acceleration and attitude. These signals must be converted to platform positions in order to drive the motion platform. An acceptable perception of the real aircraft's motion must be preserved. There is not a one-to-one relationship since the motion platform is limited in movement and the real aircraft is not [1].

The primary motion is the block that transforms the aircraft accelerations into motion base accelerations and positions. It is composed of a coordinate transformation, a washout, and a gravity alignment as shown in Figure 2.1.

The coordinate transformation transforms accelerations from the aircraft axes to the moving motion-base axes [1]. The host computer generally computes the aircraft accelerations in a vehicle-based axis system. However, the motion platform has an earth-based system. Therefore, the coordinate transformation is required to determine the accelerations in this earth-based axes system [1]. So, the outputs of the coordinate transformation are motion platform accelerations.

Next, the motion platform accelerations go through the washout algorithm. The washout is mainly a high-pass filter so it excludes undesired low frequency signals from motion base input [3]. Low

frequency signals or constant inputs require a large movement of the motion base which might exceed the cylinder's travel limits. The washout is used to limit the movement of the motion base while best reproducing the accelerations of the simulated aircraft [1].

Gravity alignment is used to simulate the constant inputs or sustained accelerations [3]. There is a reorientation of the pilot with respect to gravity. This can be simulated by pitch and roll of the motion base platform. But these angles should be small enough so as not to reproduce a significant z component [4].

After aircraft accelerations are transformed into motion-base accelerations, motion-base positions need to be obtained. To transform the accelerations into positions a double-integration is performed. This is necessary since the inputs to the geometric transformation are the position and angular orientation of the moving platform, which drive the motion base. These inputs are the primary motion output added with the special effects output.

The position of the moving platform obtained from simulated aircraft accelerations is added to the position of the moving platform given by the special effects and is transformed into cylinder extensions by the geometric transformation described later in Chapter IV.

Emphasis on special effects will be given in the next chapter.

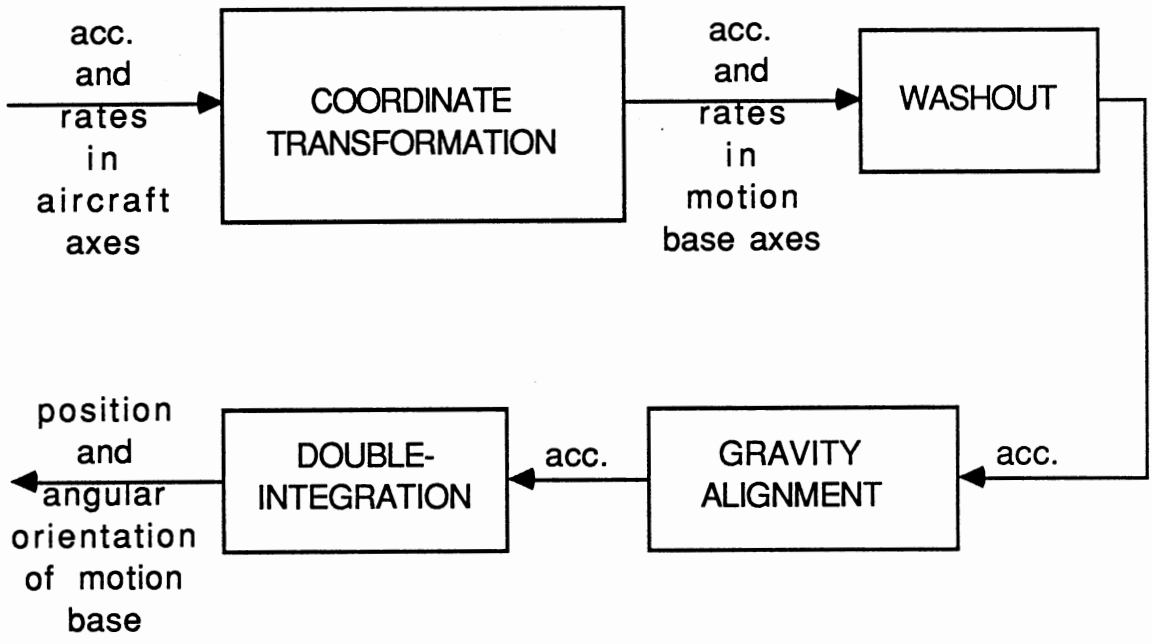


Figure 2.1. Primary Motion Block Diagram

CHAPTER III

SPECIAL EFFECTS

Special effects are any disturbances affecting the aircraft which are not caused by movement of the pilots controls. They range from simple bumps caused by cracks on the runway to turbulence vibrations. They are usually put into three categories: 1) bumps, 2) vibrations and 3) those effects that register on the instruments (turbulence and surge). There are many kinds of special effects and they are different for each aircraft in number and type.

Special effects vary depending on the type of aircraft being simulated and little documentation is available. The design of this package of subroutines was based on many discussions with personnel experienced in the simulation of special effects.

A traditional approach to create a special effect is empirical in nature. They are created on a trial and error basis which uses the subjective opinion of the pilot to reach an acceptable perception. This package will help in the design of special effects.

Special effects were all created with a small number of analog signal generators, making it difficult to achieve the complexity of the special effect being created. Currently sine wave generators with adjustable frequency and amplitude are used. In the digital implementation the number of generators that can be used for each special effect created is large. This gives more flexibility in tuning special effects.

The most complex special effect is turbulence because of its randomness and dependence on altitude and velocity of the aircraft, and also because it affects the readings on the instruments. A lot of research has been done on turbulence, but because of its complexity, it was not implemented in this project. Later it should be incorporated for improvements.

Another type of special effect is vibration, such as runway rumble and buffets. This is the type studied and implemented here. Vibrations are created by a combination of different generators, now digital generators, whose parameters can be changed quite easily.

The generators used here are of three kind: sine wave, first order filtered white noise and second order filtered white noise. As many as twelve generators can be combined together in any way to produce one special effect. Also, a combination of different special effects can be chosen to run simultaneously.

The special effects section of the motion base system simulation consists of two parts: set up and implementation. This chapter will describe the set up section first, then the implementation section.

Set Up Section

The structured diagram for the set up section of the special effects package is shown in Figure 3.1. It is menu driven, making it very easy to use. Next a description of each subroutine used in the package is given. The flowcharts of these subroutines can be found in Appendix A.

ADDSE: this subroutine is used to add special effects. It is mainly a menu that calls other specific subroutines as shown in Figure A.1.

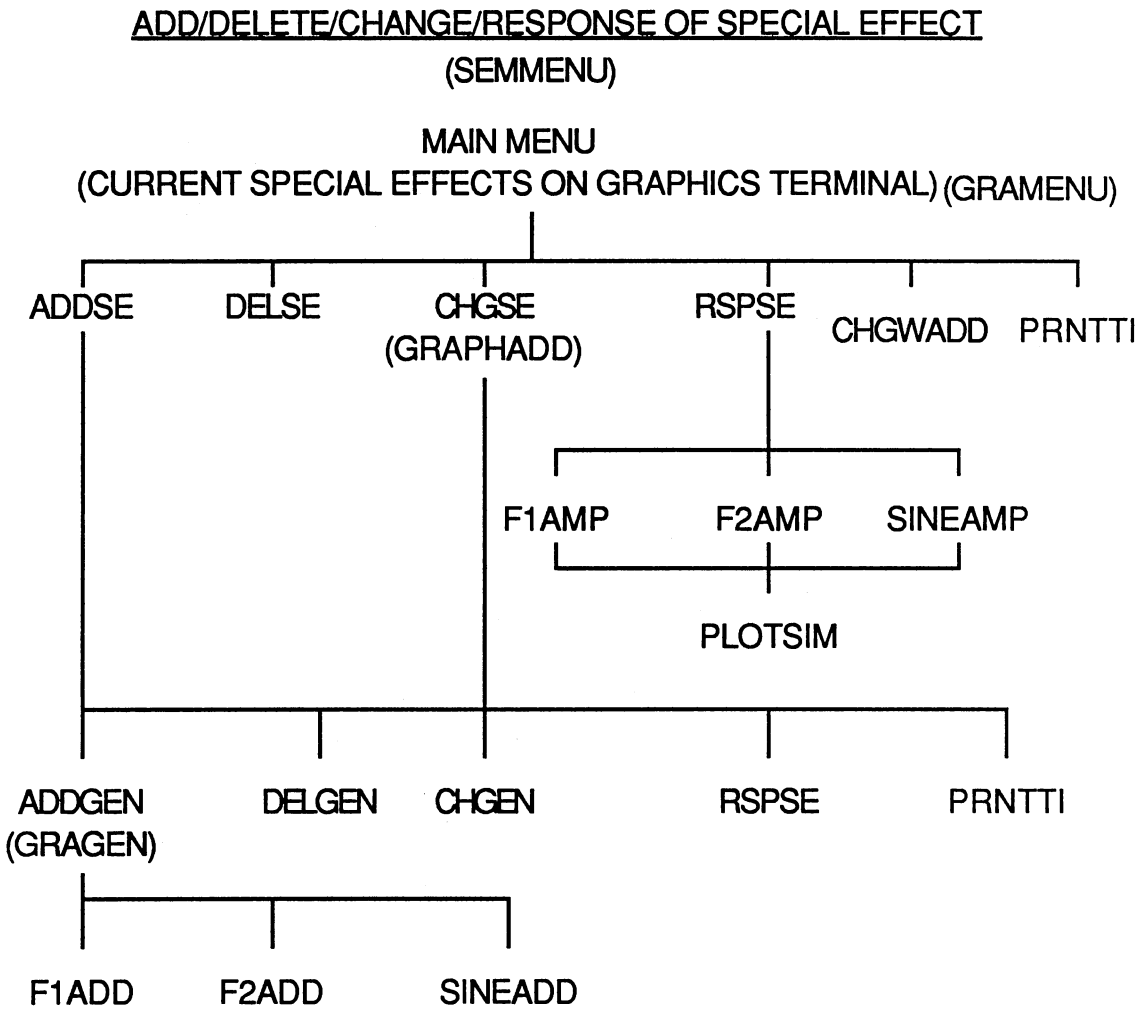


Figure 3.1. Special Effects Subroutine Simplified Diagram

DELSE: this subroutine deletes any unwanted special effects. Its flowchart is seen in Figure A.2.

CHGSE: this subroutine changes the specified special effect by adding generators, deleting generators, changing parameters of a generator and checking its response. Its flowchart is shown in Figure A.3.

GRAPHADD: this subroutine was written for the graphics terminal to show the type of generators that can be used to create a special effect.

RSPSE: this subroutine checks the special effect by plotting its response. Its flowchart is in Figure A.4. To calculate the response four other subroutines were called, they are:

F1AMP: calculates the first order filtered white noise. It is simply a first order filter with white noise input. In Figure A.5 the flowchart for the first order filtered white noise generator is shown. It calculates:

$$U = (X-0.5)*V + M$$

$$X_1 = e^{-DT*2\pi*BW} *X_1 + [1- e^{-DT*2\pi*BW}]*U \qquad (3.1)$$

$$Y = X_1$$

where: Y - amplitude of generator

X_1 - state variable

DT - sampling interval

BW - bandwidth of filter (Hz)

U - uniformly distributed white noise with mean M and variance V [5].

X - seed for the random number generator

F2AMP: calculates the second order filtered white noise. It is simply a second order filter with white noise input. In Figure A.7 the basic flowchart for the second order filtered white noise generator is shown. Two sets of equations are used since the damping ratio (ξ) of the second order filter is between 0 and 1. The value of the conditional frequency $\left(\omega=BW^{*}\sqrt{1-\xi^{2}}\right)$ [6], where BW is the bandwidth of the filter, determines which set of equations is used. These equations are shown in Figure A.6 and they form the matrix AD and BD such that:

$$X = AD^{*}X + BD^{*}U \qquad (3.2)$$

where: X - state vector variable = $[X_1 \;\; X_2]^T$

U - white noise defined as above

And the amplitude of the filter is given by: $Y = X_1$.

SINEAMP: calculates the amplitude of the sine wave. In Figure A.5 the flowchart for the sine wave generator is shown. It calculates:

$$Y = A * \sin(2\pi fT) \qquad (3.3)$$

where: Y - amplitude

A - maximum amplitude

f - specified frequency (Hz)

T - sampling interval

PLOTSIM: plots the values of the output on the graphics terminal.

CHGWADD: this subroutine changes the name of the special effect selected or changes where to add the special effect. The special effects can be added in each of the degrees-of-freedom discussed in Chapter I: x, y, z directions and angles roll, pitch and yaw. Its flowchart is shown in Figure A.8.

PRNTTI: is a subroutine written to get hardcopies of the graphics

terminal.

ADDGEN: this subroutine adds generators to the special effect being created or changed. Its flowchart is shown in Figure A.9.

GRAGEN: this subroutine was written for the graphics terminal and it displays the generators and the parameters of each generator for the special effect selected. Its flowchart is shown in Figure A.10.

DELGEN: this subroutine deletes unwanted generators created for the special effect selected. Its flowchart is shown in Figure A.3.

CHGEN: this subroutine changes the parameters of a chosen generator. Its flowchart is shown in Figure A.3.

F1ADD, F2ADD, SINEADD: these subroutines add a filtered white noise first order, filtered white noise second order and sine wave generator respectively to the special effect being created or changed. Their flowcharts are shown in Figure A.9 as part of the flowchart for the subroutine ADDGEN.

GRAMENU: this subroutine written for the graphics terminal shows the name of the special effects already created, if any, and where they are being added in the simulation. Its flowchart is shown in Figure A.11.

All of the above routines are called by a main menu called SEMMENU and they are all used to create and simulate a special effect. The variables used in these various subroutines are described in Table I. Below a better description of the arrays PSTO and PAR is made.
Initial condition vector PSTO is:

For sine wave: PSTO(I) = 0

For filtered white noise first order: PSTO(I) = 10000.

PSTO(I+1) = 0.

For filtered white noise second order: PSTO(I) = 10000.

PSTO(I+1) = 0.

PSTO(I+2) = 0.

Parameter vector PAR is:

For sine wave: PAR(I) = amplitude

PAR(I+1) = frequency

For filtered white noise first order: PAR(I) = mean

PAR(I+1) = variance

PAR(I+2) = bandwidth

For filtered white noise second order: PAR(I) = mean

PAR(I+1) = variance

PAR(I+2) = bandwidth

PAR(I+3) = damping

Implementation Section

The special effects created using this package will later be used in the simulation of the motion-base system (MOTSIM). The flowchart of the subroutine that implements the special effect block seen in Figure 1.4 is shown in Figure 3.2. The subroutine is called SMSPEF, and it calls the already described subroutines F1AMP, F2AMP and SINEAMP.

The output of this block, which is upper platform position and orientation, is added to the output of the previously described primary motion block and is input to the geometric transformation and smoothing algorithm block which will be described in the next chapter.

TABLE I

VARIABLES USED IN SPECIAL EFFECTS

Variable Name	Description	Type and Dimension	Units
NAME	name of special effect	character (20x1)	
IEFF	number of special effect currently in use	integer	
WADD	number indicating where to add the amplitude of the special effect given by index number 1. position x axis 2. position y axis 3. position z axis 4. pitch angle 5. roll angle 6. yaw angle	integer (20x1)	
KEEP	keeps number of the generators for a specified special effect (only used in CHANGE S.E.)	integer (1000x1)	
MAXKEEP	twice total number of generators used in specified special effect (only used in CHANGE S.E.)	integer	
MAXGEN	total number of generators	integer	
MAXPAR	total number of parameters	integer	
MAXPSTO	total number of initial conditions	integer	
NGEN	type of generator (1,2,3) 1. 1st order filter 2. 2nd order filter 3. sine wave	integer (240x1)	
SPECIAL	indicates number of special effect the generator (given by index number) belongs to	integer (240x1)	
PAR	parameters of all generators	real (1000x1)	

TABLE I (Continued)

Variable Name	Description	Type and Dimension	Units
PSTO	stores the variables needed for next calculations of amplitudes of the generators and initial conditions	real (1000x1)	
POSX	output of special effects in real time	real (6x1)	in
DTSPEF	sampling interval	real	sec
IEFFLG	if true the special effect is on and its amplitude should be calculated	flag[a] (20x1)	
IFLGEN	if true the generator is on and its amplitude should be calculated	flag[a] (240x1)	

[a]flag is true if equal to -1 and false if equal to 0

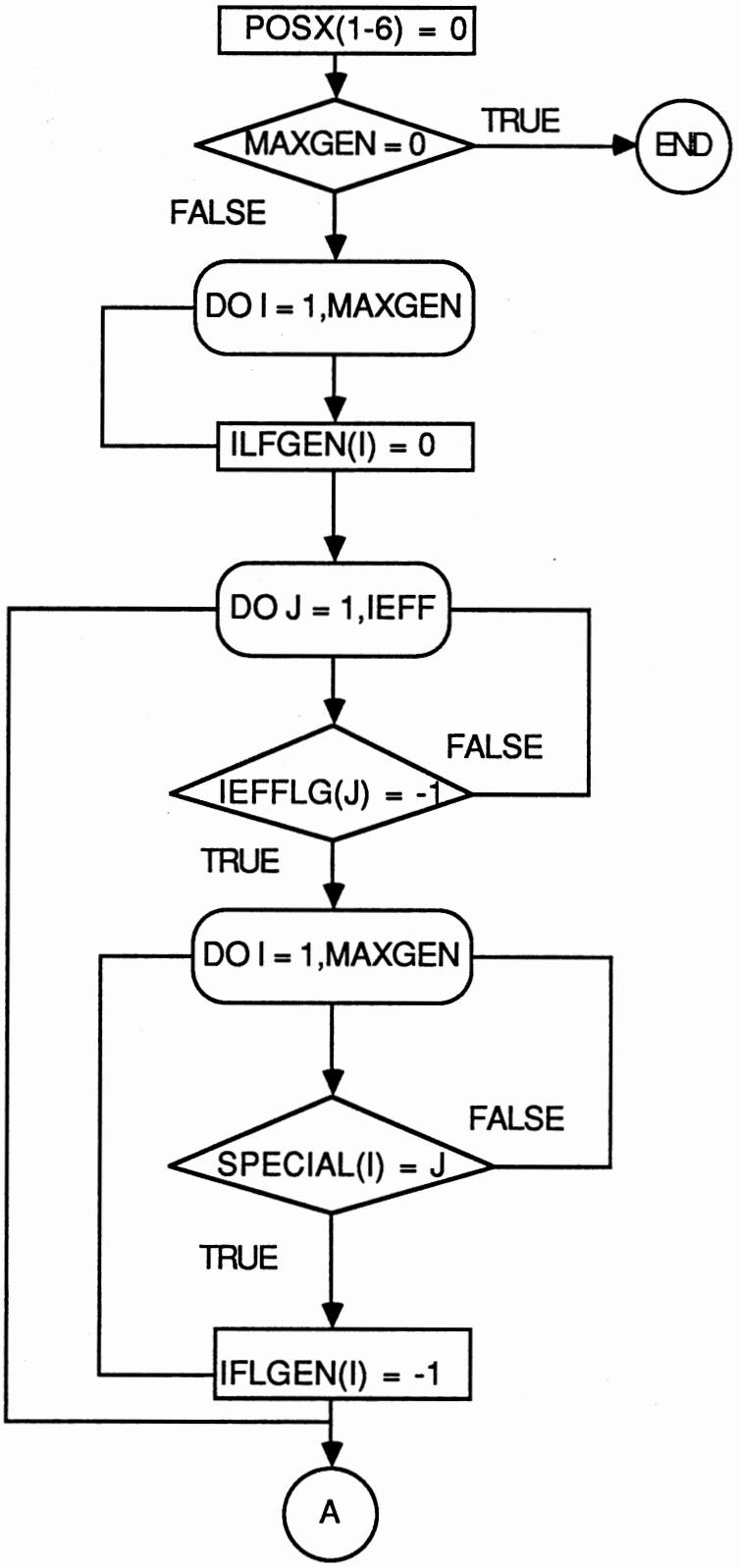


Figure 3.2. Flowchart for Subroutine SMSPEF

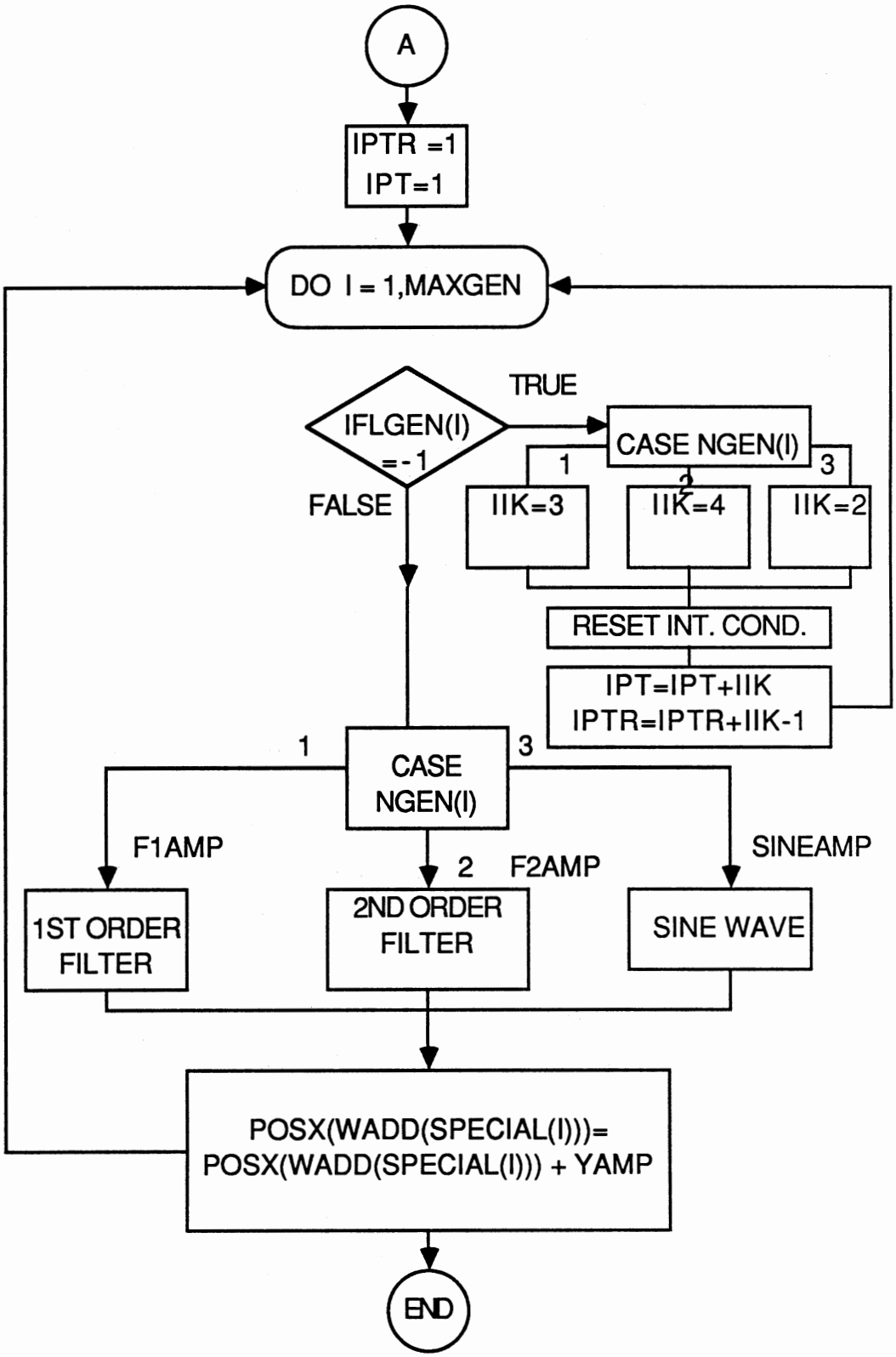


Figure 3.2. (Continued)

CHAPTER IV

GEOMETRIC TRANSFORMATION AND SMOOTHING ALGORITHM

The geometric transformation transforms the position and angular orientation of the upper platform into cylinder extensions, and the smoothing algorithm makes sure that the cylinders don't extend beyond their limits.

The geometric transformation is necessary since the motion base cannot be driven by the position and angular orientation of its center of gravity given by the aircraft simulation. It is driven by the actuators (cylinders) which from the combination of their extensions gives the position of the motion base [7].

The geometric transformation is based on the geometry of the motion base. Figure 4.1 shows the relationship between the centroid of each platform and the cylinder attachment point at each platform. The vectors are defined as:

z - from the centroid of the lower platform to the centroid of the upper platform

l_i - from cylinder i attachment point at the lower platform to cylinder i attachment point at the upper platform - cylinder i vector

B_i - form the centroid of the lower platform to cylinder i attachment point at the lower platform

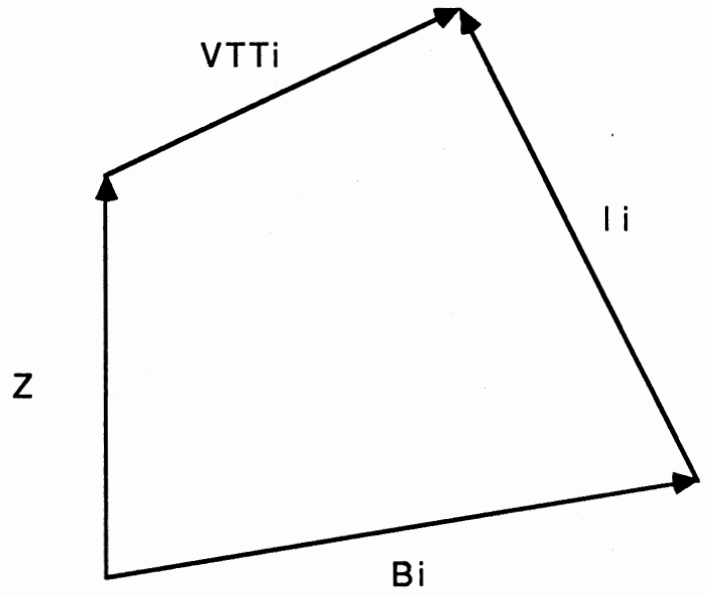


Figure 4.1. General Orientation Vector of Actuator i

VTT_i - from the centroid of the upper platform to cylinder i attachment point at the upper platform.

All coordinates are defined with respect to the lower platform coordinate system (defined in Chapter I Figure 1.2).

From Figure 4.1 the following vector equation is obtained:

$$l_i = VTT_i + Z - B_i \tag{4.1}$$

The vector VTT_i has its coordinates determined with respect to the lower platform coordinate system. The vectors given by special effects and primary motion are with respect to upper platform coordinate system. Therefore, a transformation must be performed in order to convert the vectors from the upper platform coordinate system to the lower platform coordinate system. This transformation is called an Euler angle transformation.

Let R be defined as the displacement vector of the upper platform and $A_{i,m}$ be defined as the vector going from the centroid of the upper platform to the ith cylinder attachment point at the upper platform. The following equation results:

$$VT_i = R + A_{i,m} \tag{4.2}$$

which is the total displacement vector of the ith cylinder attachment point of cylinder i.

In order to define this vector in the coordinate system of the lower platform (VTT_i) the Euler angle transformation must be applied. This is nothing more than a rotation of axes and the Euler's rotation matrix T is used. This transformation is defined as:

$$VTT_i = [T]^T VT_i \tag{4.3}$$

where vector VT_i is rotated by the movement of upper platform (pitch, roll and yaw angles). For more detail of this transformation see [7].

The order in which the given equations are calculated is:

1. $VT_i = R + A_{i,m}$
2. $VTT_i = [T]^T VT_i$
3. $l_i = VTT_i + Z - B_i$ (4.4)

And the cylinder extension is calculated as:

$$|l_i| = (l_i^T l_i)^{1/2} \quad (4.5)$$

Cylinder movement is limited to 18in extension or retraction (total cylinder length is 36in). Then, having the cylinder extensions, one must make sure that it was not overextended. This is the purpose of the smoothing algorithm. It is used as a safety device since the washout filter, in the primary motion block, should prevent the cylinders from hitting the end during normal maneuvers.

The smoothing algorithm input/output relationship (normalized) is shown in Figure 4.2. It is a linear relationship until the input reaches .83, after that it is a nonlinear (circular) relationship limiting the output to .9 of the maximum cylinder extension. This ensures that the cylinders stay within their limits.

The subroutine that simulates the geometric transformation and smoothing algorithm block is called GTRN2 and a simplified flowchart is shown in Figure 4.3. The variables used in the subroutine are described in Table II.

The output of this block - geometric transformation and smoothing algorithm - is input to the servo that commands the cylinder movement and stabilizes the system. A study of the servo is done in the next

chapter.

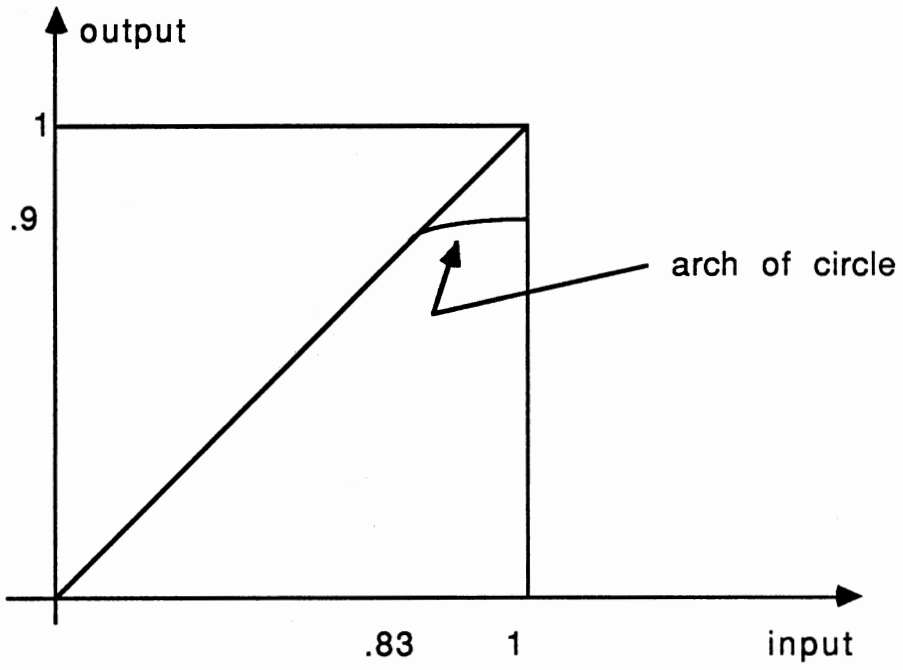


Figure 4.2. Input/Output Relationship of Smoothing Algorithm

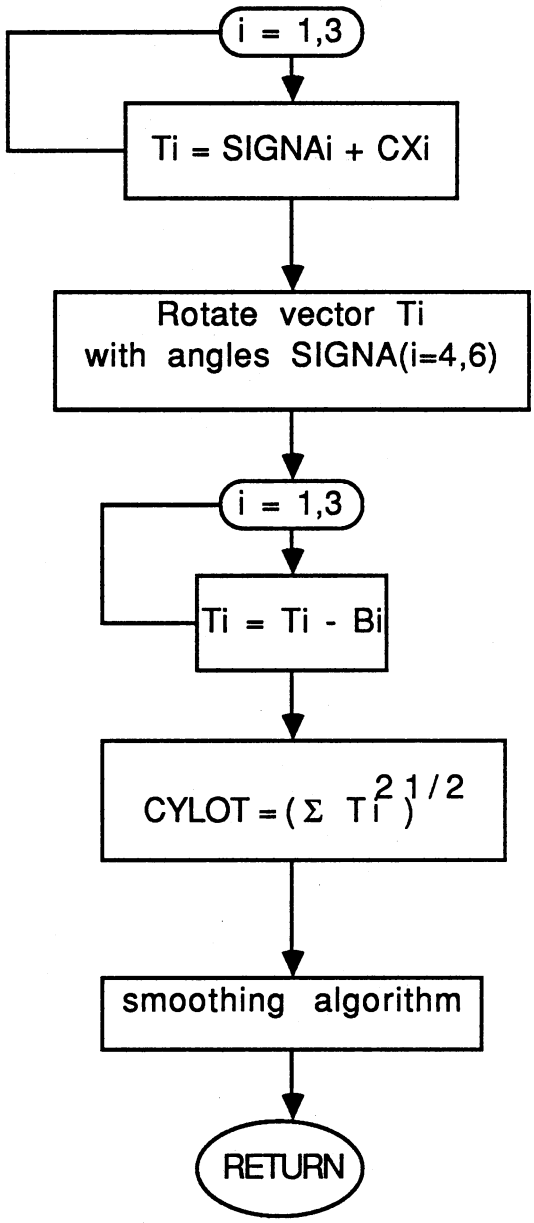


Figure 4.3. Flowchart of Subroutine GTRN2

TABLE II

VARIABLES USED IN GEOMETRIC TRANSFORMATION AND SMOOTHING ALGORITHM

Variable Name	Description	Type and Dimension	Units
SIGNA	position and angle of orientation of upper platform	real (6x1)	in,rad
CYLOT	cylinder extension	real (6x1)	in
CX	x direction component of upper platform attachment point	real (6x1)	in
CY	y direction component of upper platform attachment point	real (6x1)	in
BX	x direction component of lower platform attachment point	real (6x1)	in
BY	y direction component of lower platform attachment point	real (6x1)	in
BZ	height difference between upper and lower platform	real	in

CHAPTER V

SERVO

Servo, short for servomechanism, is a feedback control system [8]. It is the controller of the model, making it stable while increasing response and accuracy. Its operation depends upon the difference between the actual position of the object and the desired position. The servo acts to reduce this difference to zero and make the actual position equal to the desired position, so it is a model follower [9]. It is a device that moves an object while expending the smallest amount of energy [8].

In this sutdy the object to be controlled is the piston of the hydraulic cylinder and the servo will track its position.

A servo is composed of [8]: - error detector
- amplifiers
- compensation

The error detector measures the difference between the actual and desired positions. The amplifiers are used to better control the actual position. And the compensation is used to stabilize the system if cannot be stabilized by position feedback alone [6]. Usually mechanical systems become unstable because of resonances in the mechanical linkages [8].

A block diagram of the servo is shown in Figure 5.1. The feedback is composed of position feedback and compensation. If position

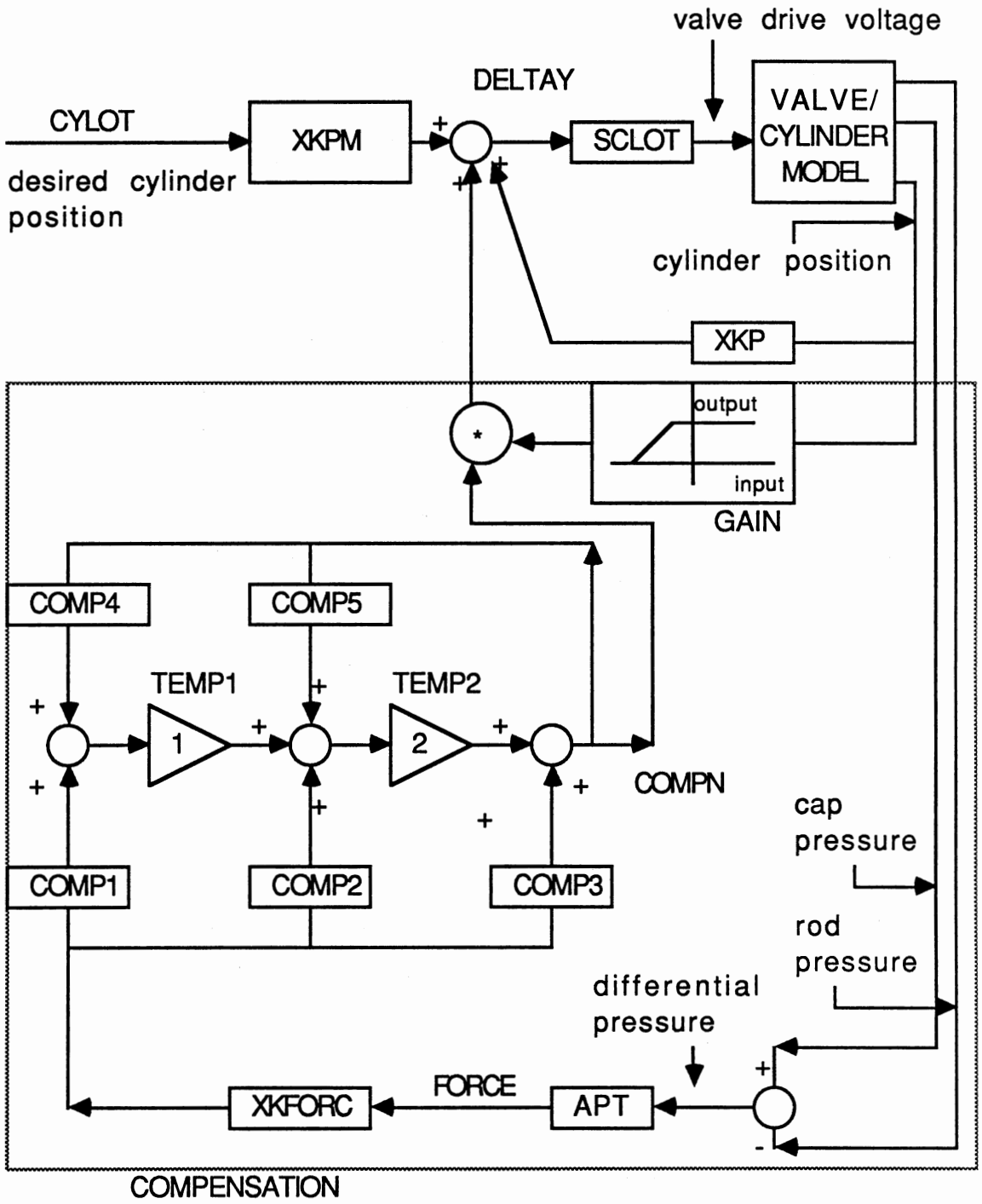


Figure 5.1. Servo Block Diagram

feedback alone is used and the gains are made high enough so that the servo will track the desired position the system becomes unstable because of resonances of the motion base. So the compensation is needed to make the system stable.

The compensation consists of pressure feedback. The pressure feedback is an approximate measure of the acceleration of the piston, which is the main compensation element. It goes through a second-order filter and is multiplied by a gain which is determined by piston position. This gain is necessary since the compensation is taken out when the piston is at the cushion, i.e., when the piston is totally retracted. When the cylinders are totally retracted the pressures measured do not represent acceleration of the piston, so if pressure feedback is used it will give an undesired output. The gain is determined by element GAIN in the servo block diagram, Figure 5.1.

The flowchart for the servo subroutine is shown in Figure 5.2. The description of the variables used in the subroutine are in Table III.

As can be seen from the servo block diagram, Figure 5.1, the output of the servo (which is the voltage to the valve that drives the cylinders) is input to the valve/cylinder model. Position and pressures obtained from the valve/cylinder model are input to the servo as feedback elements (compensation).

In the next chapter an analysis of the valve/cylinder model is made.

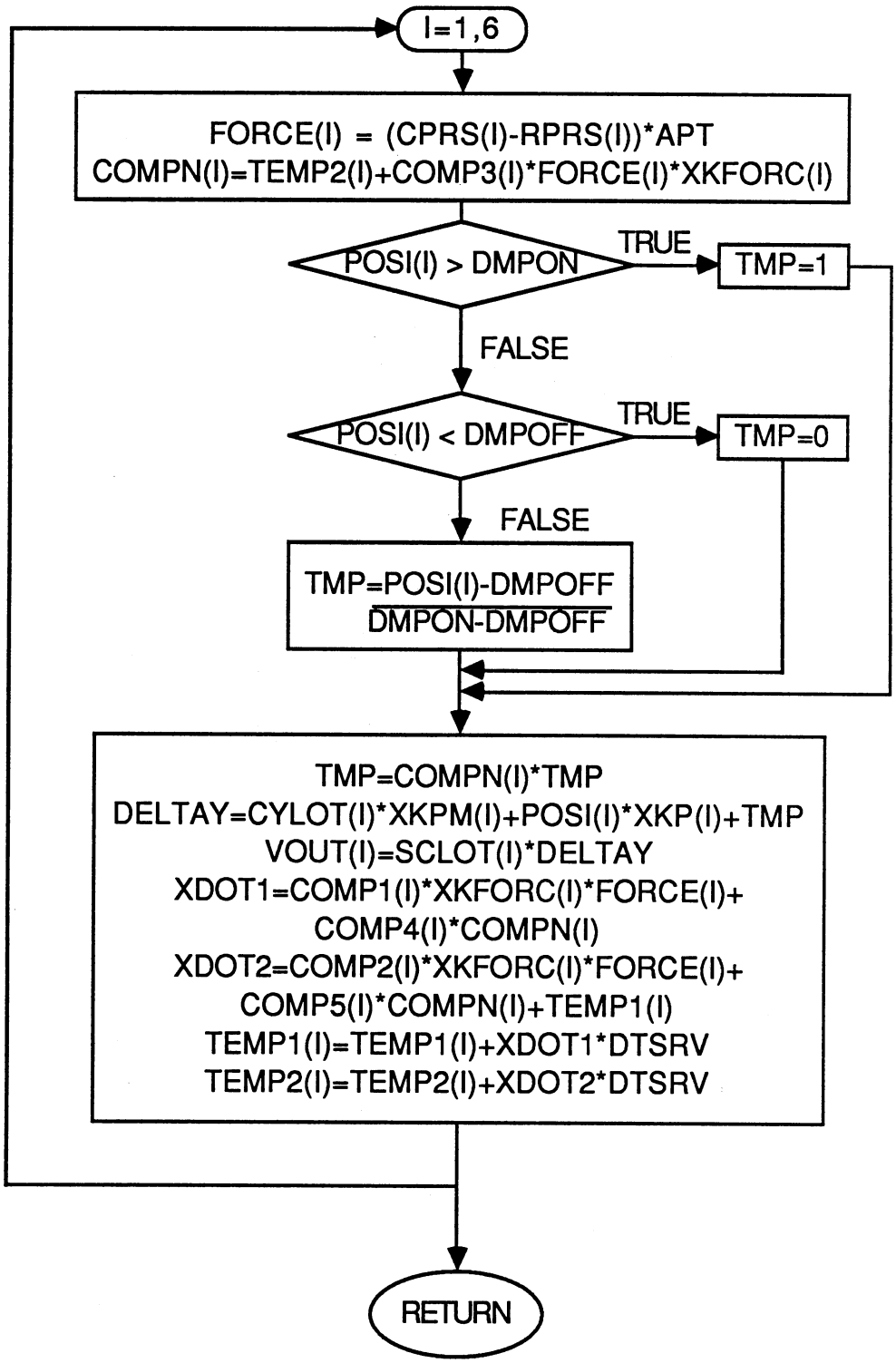


Figure 5.2. Flowchart of Subroutine SMSRV

TABLE III
VARIABLES USED IN SERVO

Variable Name	Description	Type and Dimension	Units
XKPM	desired position gain	real (6x1)	
XKP	actual (feedback) position gain	real (6x1)	
SCLOT	scale for valve voltage	real (6x1)	mV/in
XKFORC	differential pressure gain	real (6x1)	lb/psi
COMP1	compensation gain #1	real (6x1)	
COMP2	compensation gain #2	real (6x1)	
COMP3	compensation gain #3	real (6x1)	
COMP4	compensation gain #4	real (6x1)	
COMP5	compensation gain #5	real (6x1)	
DMPON	higher damping fade limit	real	in
DMPOFF	lower damping fade limit	real	in
FORCE	differential force	real (6x1)	lb
TEMP1	storage variable for integration initial condition of compensation	real (6x1)	
TEMP2	storage variable for integration initial condition of compensation	real (6x1)	
APT	area of piston	real	in^2
VOUT	input current to drive valve	real (6x1)	mA
CPRS	cap pressure	real (6x1)	psi
RPRS	rod pressure	real (6x1)	psi
DTSRV	sampling interval	real	sec

CHAPTER VI

VALVE/CYLINDER MODEL

The valve is commanded by the servo, which was described in the previous chapter, and measurements are transferred to the servo as feedback elements for compensation. Now a closer look at how the valve/cylinder model is made.

The valve/cylinder combination is a hydraulic system which is composed of the following: hydraulic fluids with a reservoir, valves, and an actuator (piston) to convert hydraulic energy into mechanical energy [10]. Ogata [10] explains many types of hydraulic systems, their equations and how they work.

The valves are used to control the direction of the fluid flow [10]. They are classified by types and by the number of ways that the flow can enter and leave the valve [10]. Having this in mind, the valve used is a sliding-spool four-way valve. The equations for the movement of the valve are nonlinear. The valve is assumed to be zero-lapped, i.e., there is no overlap when the port and the spool are aligned, so the area of the port is given by $A = k*x$ (x-spool displacement) as shown in Figure 6.1. The movement of the spool causes a flow through the valve orifices (see Figure 6.3).

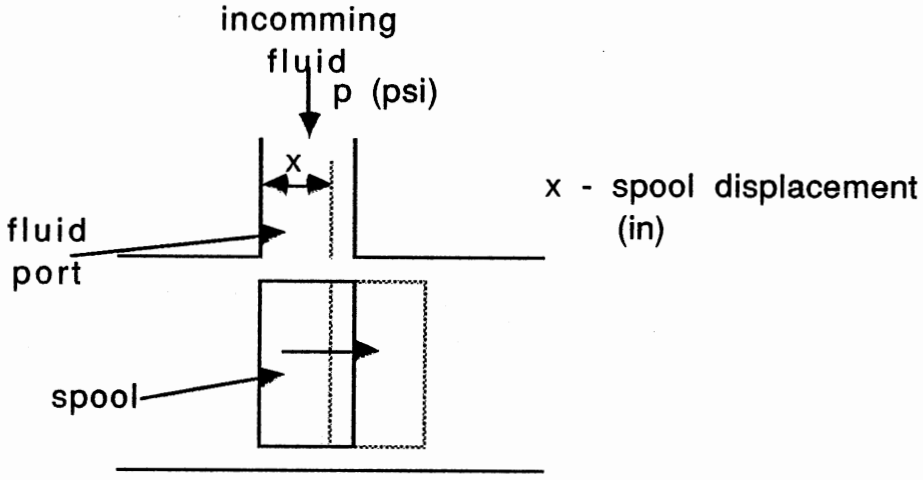


Figure 6.1. Spool Movement

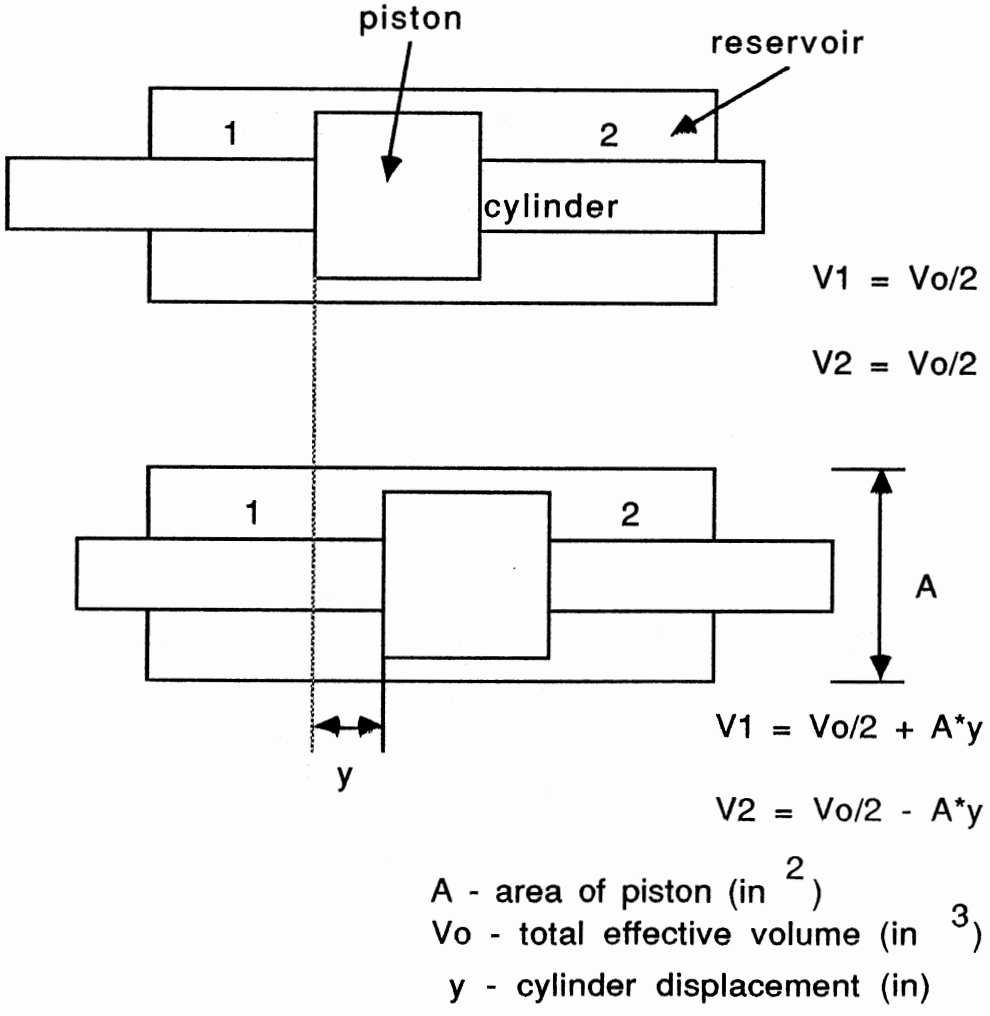


Figure 6.2. Piston Movement

These flows can be calculated by [10]:

$$\text{orifice 1: } Q_1 = cA\sqrt{(2g/\gamma)(p_s - p_1)} = ck\sqrt{2g/\gamma}\sqrt{p_s - p_1}\,x = C\sqrt{p_s - p_1}\,x$$
$$\text{orifice 2: } Q_2 = cA\sqrt{(2g/\gamma)(p_2 - p_r)} = ck\sqrt{2g/\gamma}\sqrt{p_2 - p_r}\,x = C\sqrt{p_2 - p_r}\,x \qquad (6.1)$$

where: k - proportional area constant

x - spool displacement

p_1 - cap pressure

p_2 - rod pressure

p_s - supply pressure

p_r - return pressure

g - gravity

$\gamma = g^*\rho$ - specific weight of oil

ρ - mass density

c - discharge coefficient

$C = ck\sqrt{2g/\gamma}$ (see Appendix B)

The flow of the hydraulic fluid into reservoir 1 compresses the hydraulic fluid in reservoir 1 (Figure 6.2). The compressibility is determined by the bulk modulus of the fluid [10]. As the fluid is compressed, the pressure increases which gives the compressibility flow as:

$$\frac{dV}{dt} = \frac{-V}{\beta}\frac{dp}{dt} \qquad (6.2)$$

where: p - pressure

V - volume

β - bulk modulus, which is defined as:

$$\beta = \frac{dp}{-dV/V} \tag{6.3}$$

The change in pressure creates a force which causes the piston to move. The change in volume is expressed by $V_1 = V_0/2 + A^*y$ and $V_2 = V_0/2 - A^*y$ where A - area of piston, y - displacement of piston and V_o - total effective volume. This is shown in Figure 6.2.

Because of imperfect fabrication of the cylinders some leakage of the oil occurs (Q_3, Q_4 and Q_5 in Figure 6.3). The leakage is proportional to the pressures of the two parts involved, so the leakage flow is defined as:

$$Q_L = L^*\Delta p \tag{6.4}$$

where: L - leakage proportional constant

Δp - differential pressure

When the piston moves an additional flow of A*v (area of piston * velocity of piston) is created as if it were flowing from side 1 to side 2 (see Figure 6.3). The total flow in each side of the cylinder will be:

$$\begin{aligned} &\text{in} \quad = \quad \text{out} \\ &\text{side 1: } Q_1 \quad = Q_3 + Q_4 + Q_6 + A^*v \\ &\text{side 2: } Q_3 + A^*v = Q_2 + Q_5 + Q_7 \end{aligned} \tag{6.5}$$

The equation characterizing the piston's movement comes from Newton's Law: F = ma. These equations are shown in Figure 6.4. The state variable representation of the system [11] is:

$$w' = A^*w + B \tag{6.6}$$

where: w' = dw/dt

$$w = [y_1 \;\; y'_1 \;\; y_2 \;\; y'_2]^T$$

$$w' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{K}{M_1} & -\frac{B}{M_1} & \frac{K}{M_1} & \frac{B}{M_1} \\ 0 & 0 & 0 & 1 \\ \frac{K}{M_2} & \frac{B}{M_2} & -\frac{K}{M_2} & -\frac{B}{M_2} \end{bmatrix} w + \begin{bmatrix} 0 \\ \frac{A}{M_1}(p_1 - p_2) \\ 0 \\ 0 \end{bmatrix} \tag{6.7}$$

The motor that moves the spool can also be mathematically modeled and linearized [6]. The motor equation is given in Figure 6.5. The state variable model representation of the motor system [11] is given by:

$$w' = dw/dt = A*w + B*u \tag{6.8}$$

where: $w = [x \;\; x']^T$

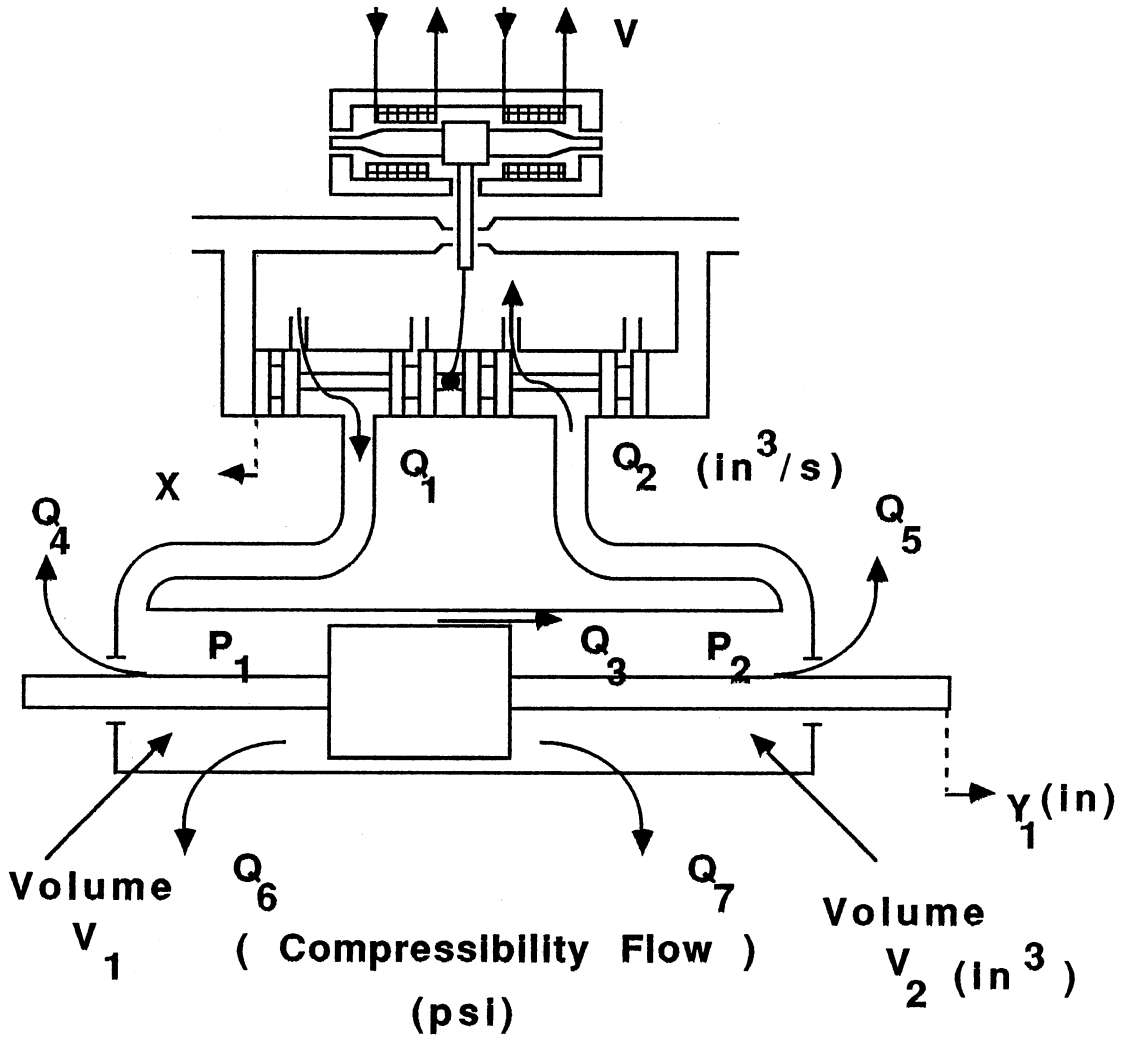
$$w' = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\xi\omega_n \end{bmatrix} w + \begin{bmatrix} 0 \\ \frac{K_m \omega_n^2}{R} \end{bmatrix} V \tag{6.9}$$

In the MOTSIM program the differential equations are integrated numerically using Euler's method:

$$w(t+\Delta t) = w(t) + w'(t)*\Delta t \tag{6.10}$$

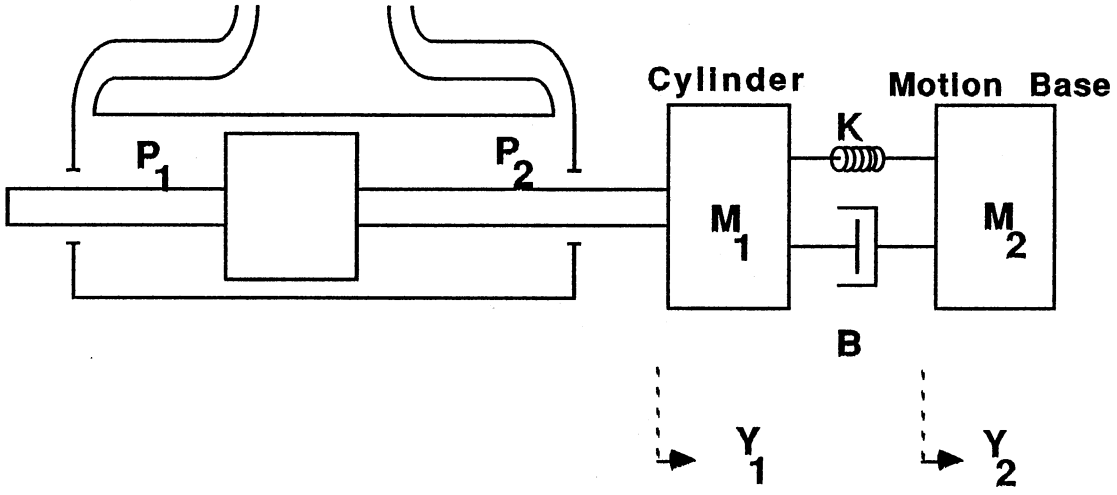
where Δt is the integration interval.

Now that all of the equations necessary to simulate the valve/cylinder model are known, the order in which they shall be executed must be determined. The following paragraph explains how the order, in which the equations are calculated, was obtained. This order is also given in Figure 6.6 and the equations referred in the paragraph below are from that figure. Table IV relates the variables seen in Figure 6.6 to the variables used in the subroutine that simulates the valve/cylinder combination.



Vo = V1 + V2 (total effective volume) (in^3)
A = APT - are of piston (in^2)
β = BETAO - bulk modulus of fluid
P1 = CPRS - cap pressure
P2 = RPRS - rod pressure

Figure 6.3. Valve/Cylinder



$$M_1 \frac{d^2}{dt^2} y_1 = A(p_1 - p_2) - K(y_1 - y_2) - B\left(\frac{d}{dt} y_1 - \frac{d}{dt} y_2\right)$$

$$M_2 \frac{d^2}{dt^2} y_2 = K(y_1 - y_2) + B\left(\frac{d}{dt} y_1 - \frac{d}{dt} y_2\right)$$

M_1 - mass of cylinder (slugs)

M_2 - mass of motion base on each cylinder (slugs)

g - gravity (=32.2 ft/s^2 or 32.2*12 in/s^2)
So the given mass in pounds (lb) has to be divided by g to be in mass units (slugs), the program does this.

K - spring constant (lb/in)

B - damping coefficient (lb/(in/s))

y_1 - displacement of cylinder (in)

y_2 - displacement of motion base (in)

Figure 6.4. Newton's Law for the Piston

TORQUEMOTOREQUATION

$$\frac{d^2}{dt^2} x + 2\xi\omega_n \frac{d}{dt} x + \omega_n^2 x = \frac{K_m \omega_n^2}{R} V$$

ξ = DAMPM– damping ratio

$\omega_n = 2\pi$BANDM– angular velocity

K_m = GAINM– gain (in/mA)

R = VRST –motor resistence (Ω)

V = VOUT– input voltage (mV)

x – output displacement (in)

Figure 6.5. Torque Motor Equation

The order in which the equations are executed is determined by the way the system operates. To drive the cylinders, a voltage is applied to the valve which moves the spool (equation 1). This opens the orifices 1 and 3 or 2 and 4 (see Figure 6.3) letting the fluid pass through the ports (equation 2). In addition to this flow, there is a leakage flow around the piston (equation 3). If orifices 1 and 3 are open $p_1 > p_2$, the piston moves forward. If orifices 2 and 4 are open $p_1 < p_2$, the piston moves backward. As was seen above the pressure difference causes the piston to move (equation 4). The flow into the cylinder creates a compressibility flow (equation 5) causing a change in pressures (equation 6). These are the steps used to simulate the valve/cylinder combination and the subroutine in which they are implemented is called SMVALVE. The variables used in the subroutine are described in Table V.

The servo block diagram (Figure 5.1) shows pressures and positions of the cylinders, derived from this block (valve/cylinder model block), being sent to the servo as feedback to stabilize the system.

To make sure that the actual cylinder lengths are correct, given the desired cylinder lengths (outputs of the geometric transformation), another transformation is used, the inverse geometric transformation. The inverse geometric transformation will be described in the next chapter.

(1) $\frac{d^2}{dt^2}X + 2\zeta\omega_n\frac{d}{dt}X + \omega_n^2 X = K_m\,\omega_n^2 i$ } Torque Motor Equation

(2) $Q_1 = cA_1\sqrt{\frac{2g}{\gamma}(p_s - p_1)} = C\sqrt{(p_s - p_1)}\,X$ (for positive X)

$Q_1 = C\sqrt{(p_1 - p_r)}\,X$ (for negative X)

$Q_2 = cA_2\sqrt{\frac{2g}{\gamma}(p_2 - p_r)} = C\sqrt{(p_2 - p_r)}\,X$ (for positive X)

$Q_2 = C\sqrt{(p_s - p_2)}\,X$ (for negative X)

} Orifice Pressure Drop

(3) $Q_3 = L_1(p_1 - p_2)$

$Q_4 = L_2(p_1 - p_r)$

$Q_5 = L_3(p_2 - p_r)$

} Leakage Flow

(4) Newton's Law for the Piston (See Figure 6.4)

(5) $Q_6 = Q_1 - Q_3 - Q_4 - A\frac{d}{dt}Y_1$

$Q_7 = Q_3 - Q_2 - Q_5 + A\frac{d}{dt}Y_1$

$V_1 = V_0/2 + A \times Y_1$

$V_2 = V_0/2 - A \times Y_1$

} Volumetric Flow

(6) $\frac{V_1}{\beta}\frac{d}{dt}p_1 = Q_6$

$\frac{V_2}{\beta}\frac{d}{dt}p_2 = Q_7$

[p_1 and p_2 saturate at p_s (high) and p_r (low)]

} Compressibility Flow

Figure 6.6. Valve/Cylinder Equations

TABLE IV

PROGRAM NAMES FOR VARIABLES
SEEN IN FIGURE 6.6

Variable Name in figure	Program Name
C	FRC
p_1	CPRS
p_2	RPRS
A	APT
y	PD
$\frac{d\ y}{dt}$	PDOT
β	BETAO
p_s	PS
p_r	PR
L_1	XL1
L_2	XL2
L_3	XL3
$Q_\#$	Q#
$V_\#$	V#
V_o	VO

TABLE V
VARIABLES USED IN VALVE/CYLINDER

Variable Name	Description	Type and Dimension	Units
MOTOR SPECIFICATIONS			
DAMPM	damping	real	
BANDM	bandwidth	real	Hz
GAINM	gain	real	in/mA
VRST	resistance	real	Ω
VALVE SPECIFICATIONS			
PS	supply pressure	real	psi
PR	return pressure	real	psi
XL1	leakage flow constant #1	real	$in^3/(s*psi)$
XL2	leakage flow constant #2	real	$in^3/(s*psi)$
XL3	leakage flow constant #3	real	$in^3/(s*psi)$
APT	area of piston	real	in^2
VO	total effective volume of cylinder	real	in^3
XLIMIT	limit of spool position	real	in
FRC	flow rate constant	real	see App. B
XM1	mass of cylinder	real	lb
OIL SPECIFICATIONS			
BETAO	bulk modulus	real	psi
MODEL OF MOTION BASE			
XM2	mass on top of each cylinder because of motion base	real (6x1)	lb
XK	spring const. for each cylinder	real (6x1)	lb/in
B	damping coeff. for each cylinder	real (6x1)	lb/(in/s)

TABLE V (Continued)

Variable Name	Description	Type and Dimension	Units
SIMULATION VARIABLES			
X	spool position	real (6x1)	in
XDOT	spool velocity	real (6x1)	in/s
VOUT	input current to drive valve	real (6x1)	mA
XMOTOR	continuous time motor matrix	real (2x3)	see eq. 6.9
Q1	orifice flow	real (6x1)	in^3/s
Q2	orifice flow	real (6x1)	in^3/s
Q3	leakage flow	real (6x1)	in^3/s
Q4	leakage flow	real (6x1)	in^3/s
Q5	leakage flow	real (6x1)	in^3/s
Q6	compressibility flow	real (6x1)	in^3/s
Q7	compressibility flow	real (6x1)	in^3/s
CPRS	cap pressure	real (6x1)	psi
RPRS	rod pressure	real (6x1)	psi
PISTON	continuous time piston matrix	real (4x5)	see eq. 6.7
UIN	differential pressure	real (6x1)	psi
PD	piston displacement	real (6x1)	in
PDOT	piston velocity	real (6x1)	in/s
V1	volume on cap side	real (6x1)	in^3
V2	volume on rod side	real (6x1)	in^3
POSI	cylinder position	real (6x1)	in

CHAPTER VII

INVERSE GEOMETRIC TRANSFORMATION

The inverse geometric transformation is used to check the simulation. Its inputs are the actual cylinder extensions and the outputs are the positions and angular orientations of the upper platform. They are compared to the positions and angular orientations given as input to the geometric transformations (described in Chapter IV) to measure how well the system behaved [1].

The objective is then to find the actual positions and angles of orientation of the centroid of the upper platform of the motion base knowing only the six cylinder extensions given by potentiometers. So the root of the following equation must be found:

$$f(x) = 0 \tag{7.1}$$

where: f - vector formed by $f_i(x) = {}^{c}l_i^{T}(x)\ {}^{c}l_i(x) - {}^{m}|l_i|^2$

${}^{c}l_i(x)$ - calculated cylinder position (3 dimensional vector)

${}^{m}|l_i|$ - measured cylinder lengths

x - positions and angles of orientation of the centroid of the upper platform (6 dimensional vector).

To find the root of equation 7.1 an iterative numerical method was applied. The first numerical method used was Newton-Raphson [12] whose iterative formula has the form:

$$x(n+1) = x(n) - J^{-1}f \tag{7.2}$$

where: J - Jacobian of f with respect to x

f - vector formed by $f_i(x) = {}^{c}l_i^{T}(x)\ {}^{c}l_i(x) - {}^{m}|l_i|^2$

x - positions and angles of orientation of the centroid of the upper platform (6 dimensional vector).

The Newton-Raphson method for iteration suggested by Dieudonne [7] gave a very slow convergence and problems occurred when the determinant of J was near zero. Since the algorithm depends on the inverse of J another algorithm is required. The Marquarot algorithm was chosen since the inverse matrix that it calculates always exists [13]. The iteration formula has the form:

$$x(n+1) = x(n) - [J^{T}J+\mu I]^{-1}J^{T}f \qquad (7.3)$$

where: J - Jacobian of f with respect to x

f - vector formed by $f_i(x) = {}^{c}l_i^{T}(x)\ {}^{c}l_i(x) - {}^{m}|l_i|^2$

${}^{c}l_i(x)$ - calculated cylinder position (3 dimensional vector)

${}^{m}|l_i|$ - measured cylinder lengths

x - positions and angles of orientation of the centroid of the upper platform (6 dimensional vector).

μ - adjustable constant

I - identity matrix.

This algorithm has a faster convergence and it certainly gives an inverse for the matrix in brackets (eq. 7.3) since the constant μ is adjustable.

With the appropriate initial conditions equation 7.3 is repeated until the criterion for convergence is met. Two criteria for convergence are used:

1. $\Sigma(J^T f)_i < \varepsilon$

2. $\Sigma[x(n+1)-x(n)]^2 < TOL$ (7.4)

where ε and TOL are tolerances defined by the user.

The Marquarot algorithm ajusts μ at each iteration. If the value of $\Sigma f_i^2(x(n+1))$ is less than $\Sigma f_i^2(x(n))$ then μ is decreased by some factor (1/r). Otherwise μ is increased by some factor (r). As μ gets small the algorithm approches the Gaus-Newton method. As μ gets large the algorithm approches the steepest descent method [13].

The detailed equations for the implementation of the inverse geometric transformation - IGT - can be found in [7]. The subroutine that implements IGT is called SMIGT and its flowchart is shown in Figure 7.1. The variables used in the subroutine are in Table VI.

The order of the positions and angles in array variables TINVX and TINVXI are different than those in array variables TINVXL and TINVXN so a rearrangement of the arrays is necessary in the beginning and end of the routine (see Table VII).

In the next chapter an analysis of the whole system, and the integration of the various subroutines described in the previous chapters is discussed.

TABLE VI

VARIABLES USED IN INVERSE GEOMETRIC TRANSFORMATION

Variable Name	Description	Type and Dimension	Units
EPSILON	tolerance for convergence of MARQUAROT algorithm	real	
TOL	tolerance for change in TINVX	real	
CEXT (POSI)	actual cylinder lengths	real (6x1)	in
TINVX	position and angular orientation of platform	real (6x1)	in,rad
TINVXI	initial condition of position and angular orientation (last saved position and angular orientation)	real (6x1)	in,rad
FI	identity matrix	real (6x6)	
TINVXN	new values of position and angular orientation	real (6x1)	in,rad
TINVXL	old values of position and angular orientation	real (6x1)	in,rad
F	difference between calculated length and actual length	real (6x1)	in
DF	gradient with respect to TINVX of vector F	real (6x6)	
DFINV	inverse of DF	real (6x6)	
DECTIV	decrement to approach correct value of TINVX	real (6x1)	in
CX	x direction component of upper platform attachment point	real (6x1)	in
CY	y direction component of upper platform attachment point	real (6x1)	in
BX	x direction component of lower platform attachment point	real (6x1)	in

TABLE VI (Continued)

Variable Name	Description	Type and Dimension	Units
BY	y direction component of lower platform attachment point	real (6x1)	in
BZ	height difference between upper and lower platform	real	in
ERRO	square of the norm of old F	real	
ERRN	square of the norm of new F	real	
RNORM	square of the norm of H (test for convergence)	real	
SUM	square of the norm of DECTIV (test for convergence)	real	

TABLE VII

ORDER OF POSITIONS AND ANGLES IN ARRAYS
TINVX, TINVXI, TINVXL AND TINVXN

TINVX and TINVXI	TINVXL and TINVXN	DIMENSION
x direction	x direction	in
y direction	y direction	in
z direction	z direction	in
pitch angle	yaw angle	rad
roll angle	pitch angle	rad
yaw angle	roll angle	rad

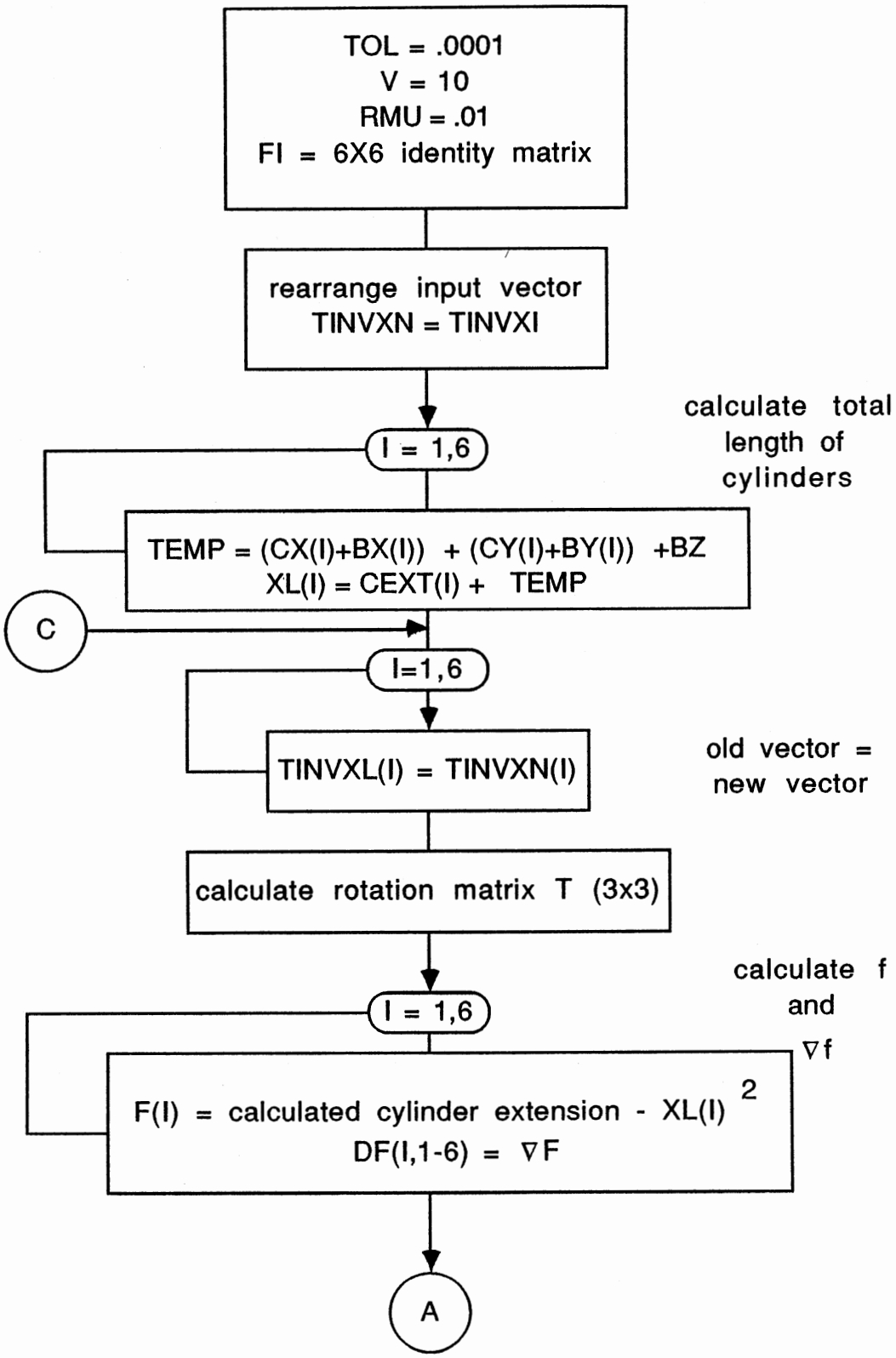


Figure 7.1. Flowchart for Subroutine SMIGT

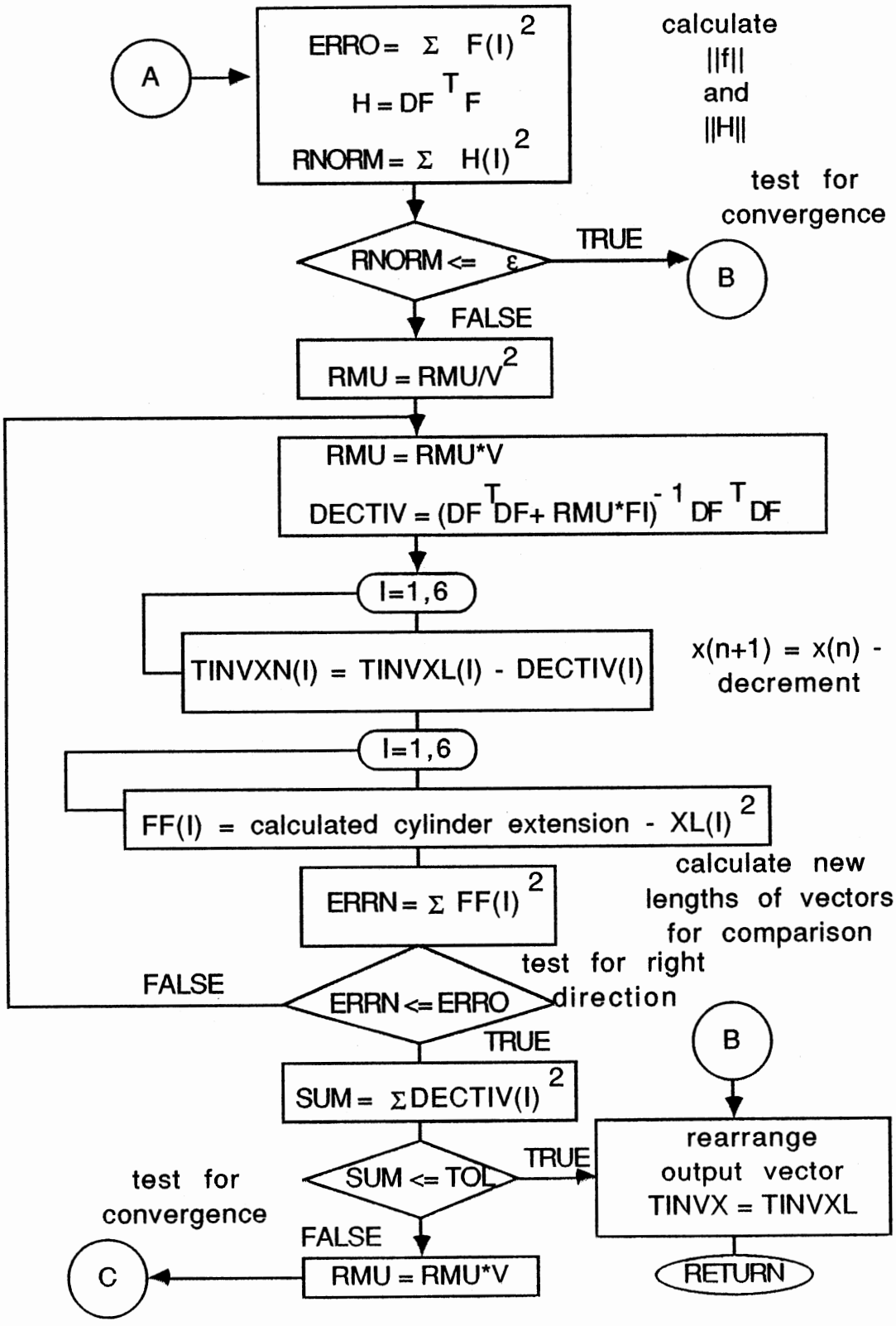


Figure 7.1. (Continued)

CHAPTER VIII

COMPUTER PROGRAM

In the previous chapters a description of the blocks that form the motion base system simulation (MOTSIM) was given. In this chapter the structure of MOTSIM is presented along, with a description of the subroutines and some sample runs.

The structured diagram of MOTSIM is shown in Figure 8.1. As one can see this program, not only executes the motion base system simulation, but also sets up and changes the parameters used in the blocks (see Figure 1.4), prepares the inputs, displays results, and shows an animation of the motion base system. Separate subroutines for each of these tasks were written. A brief description of the subroutines used in program MOTSIM is presented next. The program MOTSIM is menu driven, making it user-friendly.

Subroutine Descriptions

SEMMENU - this subroutine makes it possible to create special effects. It calls the various subroutines described in Chapter III and Appendix A.

SERVO - sets up the parameters for the servo.

VALVE - sets up the parameters that are used in the subroutine to simulate the valve/cylinder combination.

FILERD - reads the four files of parameters used in the program.

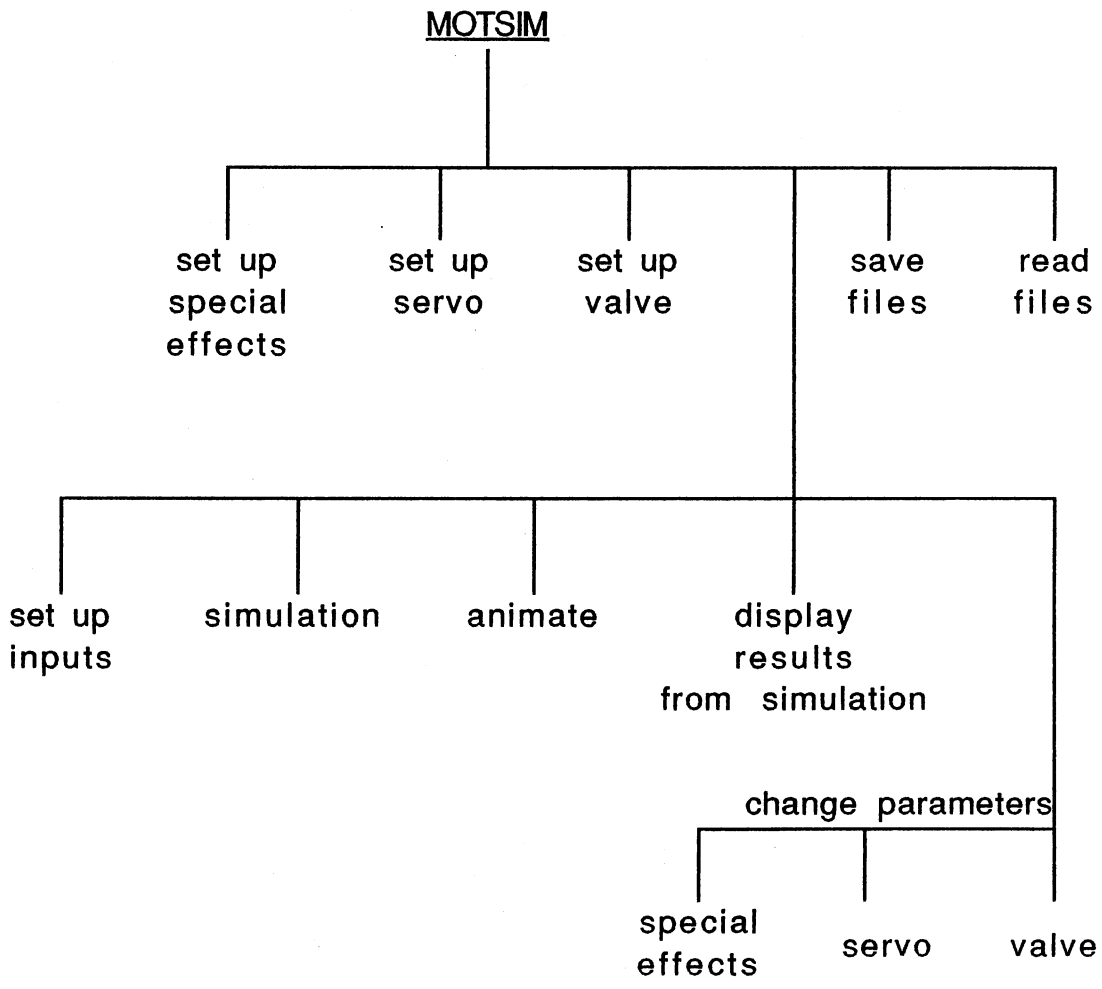


Figure 8.1. MOTSIM Structure Diagram

FILESV - saves three of the files of parameters used in the program.

INPUTS - this subroutine prepares the inputs to the simulation.

SIMULATE - this subroutine simulates the motion base system, and will be described later in this chapter.

ANIMATE - this subroutine shows the results of the simulation representing a motion base in movement in a graphics terminal.

DISRES - this subroutine is used to plot, display or print the results of the simulation.

CHGSE - this subroutine makes it possible to change or view parameters of the special effects.

CHGSRV - this subroutine makes it possible to change or view parameters of the servo.

CHGVAL - this subroutine makes it possible to change or view parameters used to simulate the valve/cylinder combination.

Subroutine SIMULATE

The most important subroutine is SIMULATE, the one that executes the simulation - the objective of the program MOTSIM. The basic flowchart for the SIMULATE subroutine is shown in Figure 8.2. As one can see it contains the blocks (see Figure 1.4) defined in the previous chapters, that is, primary motion, special effects, geometric transformation and smoothing algorithm, servo, valve/cylinder combination and inverse geometric transformation blocks. They are simulated by subroutines OSUONE, SMSPEF, GTRN2, SMSRV, SMVALVE and SMIGT respectively.

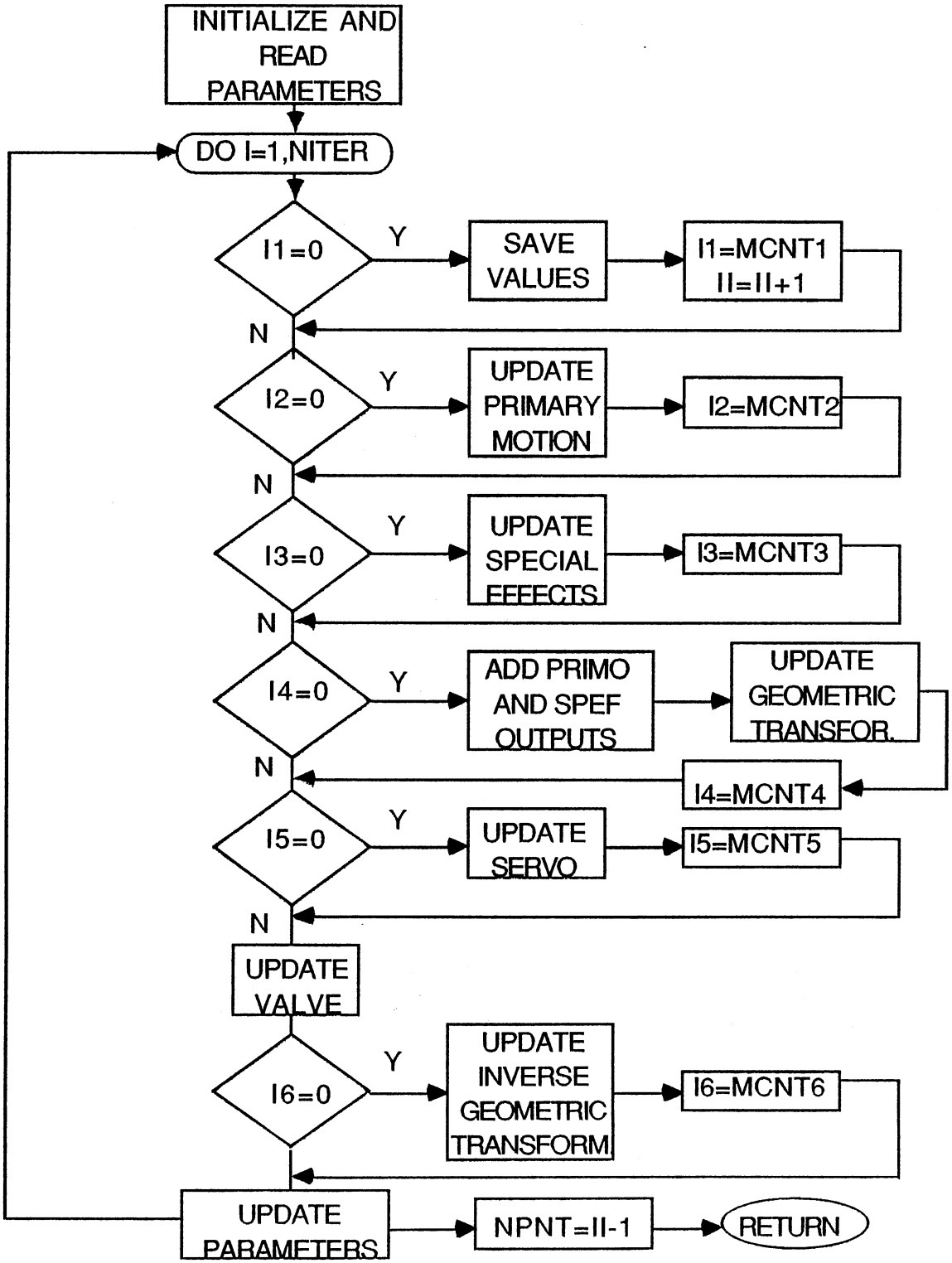


Figure 8.2. Basic Flowchart for Subroutine SIMULATE

SAVEVAL - save values- is the only subroutine seen in Figure 8.2 which has not been described yet. The SAVEVAL subroutine saves the variables chosen in subroutine SIMULATE in array variable PLOTV for plotting, displaying or printing after the simulation of the motion base system is completed. The variables can be chosen from file TABLE.SA. A maximum of 17 variables can be chosen per execution of the simulation.

All the subroutines used in SIMULATE have been described. A description of how the subroutine SIMULATE is executed follows now.

The first step to start execution of subroutine SIMULATE is to select option #2 (EXECUTE) from "SIMULATION MENU". The second step is to enter, in ascending order, the number of the variables the user wants to be saved, if any. These variables shall be entered by their corresponding number found in file TABLE.SA. The next step is to enter the sampling rates for the following SIMULATE blocks: valve/cylinder, primary motion, special effects, servo and save values. The rate for the geometric transformation and smoothing algorithm block is derived from the rates of primary motion and special effect blocks. The rate for the inverse geometric transformation block is derived from the rate of save values block. After these steps the integration intervals and the update counters will be calculated from the given rates. Table VIII below shows the blocks used in SIMULATE and the corresponding update variable and integration interval variable. These variables are also described in Table IX along with some other variables used in MOTSIM. Afterward, enter if the inverse geometric transformation block is to be executed. Finally, the total time for the simulation is entered.

TABLE VIII

COMMANDING VARIABLE PER PROGRAM UNIT

Block	Variable Update	Integration Variable
valve	every iteration	DT
servo	MCNT5	DTSRV
primary motion	MCNT2	DTPRMO
special effects	MCNT3	DTSPEF
IGT*	MCNT6	
GT*	MCNT4	
SAVE*	MCNT1	DTSAVE

*Where IGT stands for inverse geometric transformations, GT for geometric transformations and smoothing algorithm and SAVE for the routine that saves variables to be plotted, printed or displayed.

After the simulation is complete the results can be plotted, displayed or printed. An animation of the motion base can also be seen. The three dimensional picture of the moving motion base on the graphics terminal is drawn using the outputs of the inverse geometric transformation (positions and angular orientations of the centroid of the upper platform). However, the animation can only be performed when the inverse geometric transformation is executed. The animation of the motion base can be viewed from various angles: an angled view from the front, a straight front view, a straight top view and a straight right side view.

Additional observations must be made with respect to using the program. First, if a block is not to be executed zero must be entered as the RATE for that block, as described in Table VIII. Only the RATE for the valve cannot be zero. Second, the execution of some blocks depend

on other blocks. The inverse geometric transformation is only executed if the saving values (SAVEVAL) and the servo blocks are also executed. The geometric transformation and smoothing algorithm block is executed if the primary motion block and/or the special effects block are also executed. Next, the simulation is complete when the number of iterations, defined by TMAX/DT where TMAX is the total time of simulation and DT is the integration interval of the valve, is reached. Finally, if the primary motion block and the special effects block are not executed, then the desired position of the cylinders is a unit step. This way, the response of the valve/cylinder combination can be tested.

In the next section some examples are presented.

TABLE IX

VARIABLES USED IN SIMULATE

Variable Name	Description	Type and Dimension	Units
I1	counters for SAVEVAL to be executed	integer	
I2	counters for OSUONE to be executed	integer	
I3	counters for SMSPEF to be executed	integer	
I4	counters for GTRN2 to be executed	integer	
I5	counters for SMSRV to be executed	integer	
I6	counters for SMIGT to be executed	integer	
NVS	total number of chosen variables to save	integer	
NVAR	number of chosen variable to save	integer (17x1)	
MCNT1	number of iterations for SAVEVAL variables update	integer	
MCNT2	number of iterations for OSUONE variables update	integer	
MCNT3	number of iterations for SMSPEF variables update	integer	
MCNT4	number of iterations for GTRN2 variables update	integer	
MCNT5	number of iterations for SMSRV variables update	integer	
MCNT6	number of iterations for SMIGT variables update	integer	
NPNT	number of points saved for plotting	integer	
MITER	number of iterations	integer	
T	time counter	real	sec
DT	integration interval of valve	real	sec
DTPRMO	integration interval of primary motion	real	sec

TABLE IX (Continued)

Variable Name	Description	Type and Dimension	Units
DTSPEF	integration interval for special effects	real	sec
DTSRV	integration interval for servo	real	sec
DTSAVE	integration interval for SAVE	real	sec
TMAX	total time of simulation	real	sec

Examples

Several examples were prepared in order to illustrate some of the program features. The examples were divided into 6 categories: washout, special effects, geometric transformation, valve/cylinder model, inverse geometric transformation and animation.

Washout

The washout category illustrates the washout block (see Figure 2.1) part of the primary motion block.

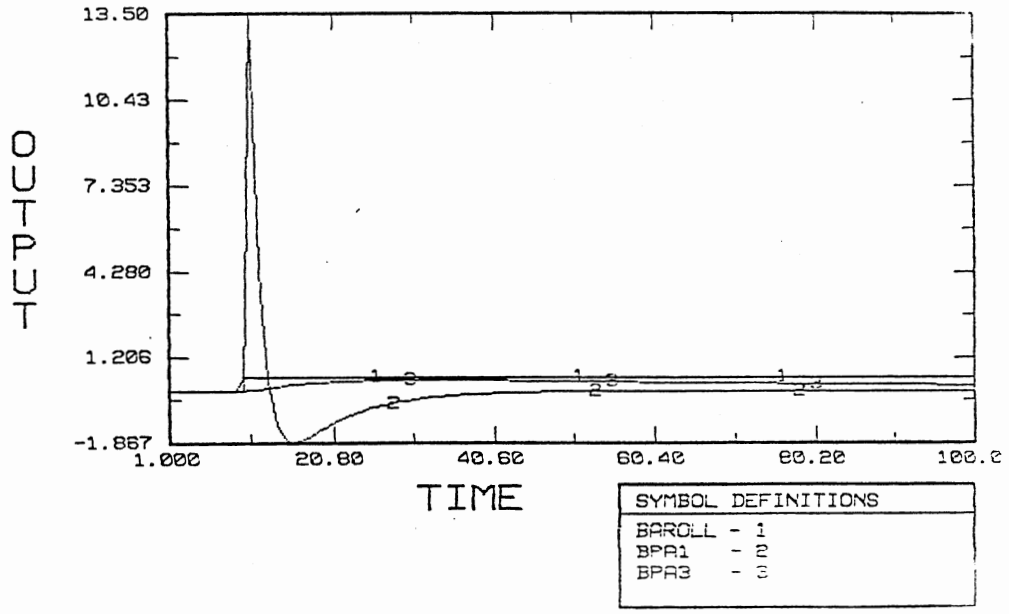


Figure 8.3. Input/Output of Washout

The figure above shows the input/output relationship of the washout block. The input BAROLL is a step roll acceleration in the motion base axes. The outputs are BPA1 (roll acceleration) and BPA3 (roll angle). The constant roll acceleration input is scaled, washed out and integrated to get the roll angle (BPA3). The roll angle decays to zero as the acceleration is washed out.

Special Effects

The special effects category illustrates the set up of special effects.

```
PAGE     1    LIST VER  122084 4  11/14/88  18:56:14  SYS:0064..TEST.SA

 9
SINEX                SINEY                SINEZ                SINE_PITCH
SINE_ROLL            SINE_YAW             FILT_1ST_ORDER       FILT_2ND_ORDER
COMBINATION
 1 2 3 4 5 6 2 4 6
 11
 1 2 3 4 5 6 7 8 9 9 9
 3 3 3 3 3 3 1 2 1 2 3
 28
 1.000 1.000 1.000 1.000 1.000 1.000 .1000 .5000 .1000 .5000 .1000
 .5000 .0000 1.000 10.00 .0000 1.000 15.00 .1000 .0000 1.000 10.00
 .0000 1.000 15.00 .2000 2.000 5.000
 17
 .0000 .0000 .0000 .0000 .0000 .0000 1.0000E+04 .0000 1.0000E+04 .0000 .0000
 1.0000E+04 .0000 1.0000E+04 .0000 .0000 .0000
```

Figure 8.4. Example of a Special Effect Data File

Figure 8.4 shows an example of a special effect data file after 9 special effects have been created. The values used here are for demonstration purposes only and do not represent true special effects.

```
CURRENT SPECIAL EFFECTS

01.  SINEX                 /AT:  POS  X
02.  SINEY                 /AT:  POS  Y
03.  SINEZ                 /AT:  POS  Z
04.  SINE_PITCH            /AT:  PITCH
05.  SINE_ROLL             /AT:  ROLL
06.  SINE_YAW              /AT:  YAW
07.  FILT_1ST_ORDER        /AT:  POS  Y
08.  FILT_2ND_ORDER        /AT:  PITCH
09.  COMBINATION           /AT:  YAW
```

Figure 8.5. "Current Special Effects" Screen

The above figure shows one of the graphics screens - "CURRENT SPECIAL EFFECTS" - when the data file of Figure 8.4 is used. The "/AT:" denotes where the output of the special effect is going. Example: Special effect 1, SINEX, is going to affect the position of the motion base in the x axis.

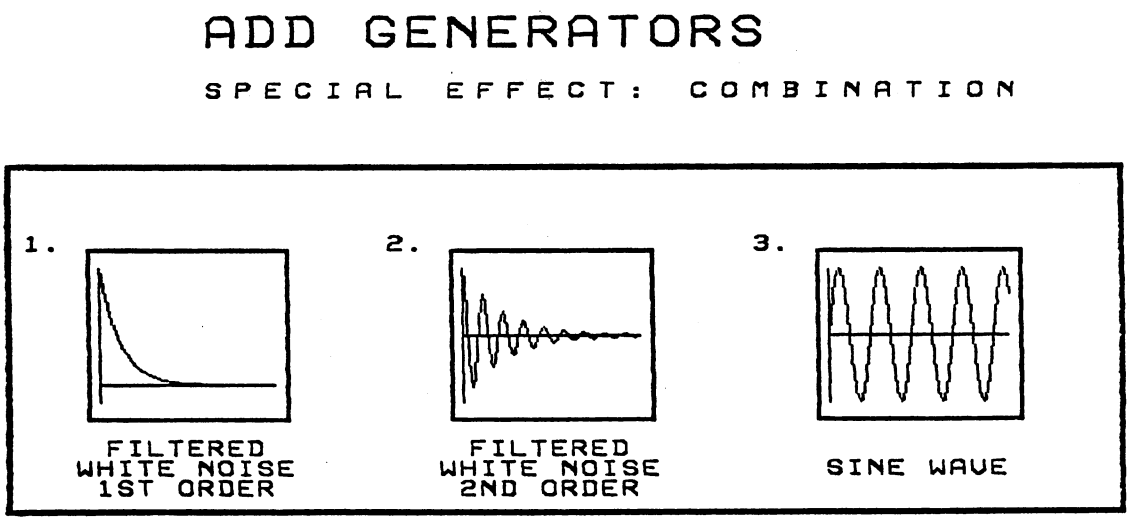


Figure 8.6. "Add Generators" Screen

Figure 8.6 shows the graphics screen that appears when the user wants to add generators to a specified special effect. The special effect specified here is "COMBINATION", selection 9 from Figure 8.5. Three types of generators can be chosen: first order filtered white noise, second order filtered white noise and sine wave. They are represented by an impulse response to a first order filter, an impulse response to a second order filter and a sine wave respectively.

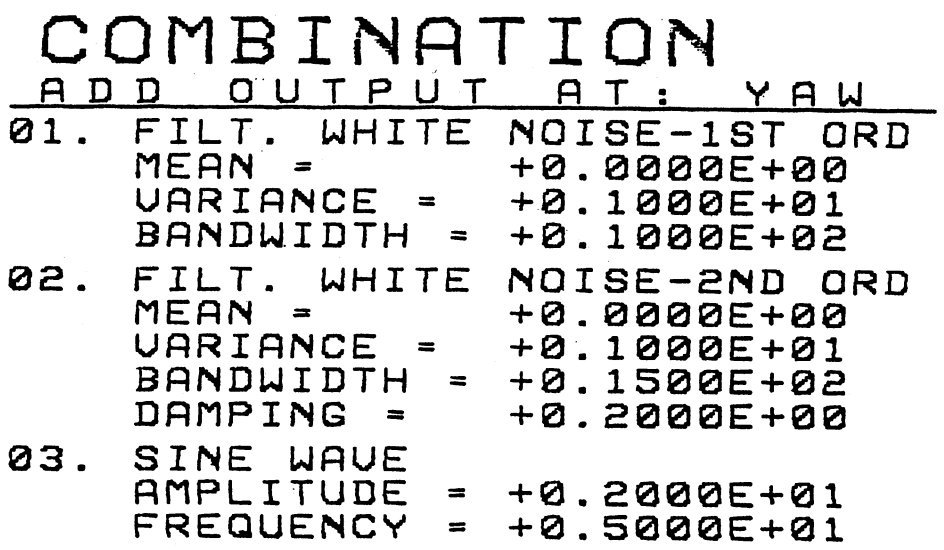


Figure 8.7. Special Effect "COMBINATION" with Generators

The figure above shows an example of the view option for a specified special effect. It illustrates the special effect "COMBINATION", selection 9 from Figure 8.5. From this screen one can tell that the special effect "COMBINATION" is going to affect the position of the motion base in the yaw angle. One can also see the generators that form the special effect. This special effect is composed of:

1) first order filtered white noise. The white noise has mean zero and variance 1. The bandwidth of the filter is 10 Hz.
2) second order filtered white noise. The white noise has mean zero and variance 1. The bandwidth of the filter is 15 Hz and the damping is 0.2.
3) sine wave with amplitude 2 and frequency 5 Hz.

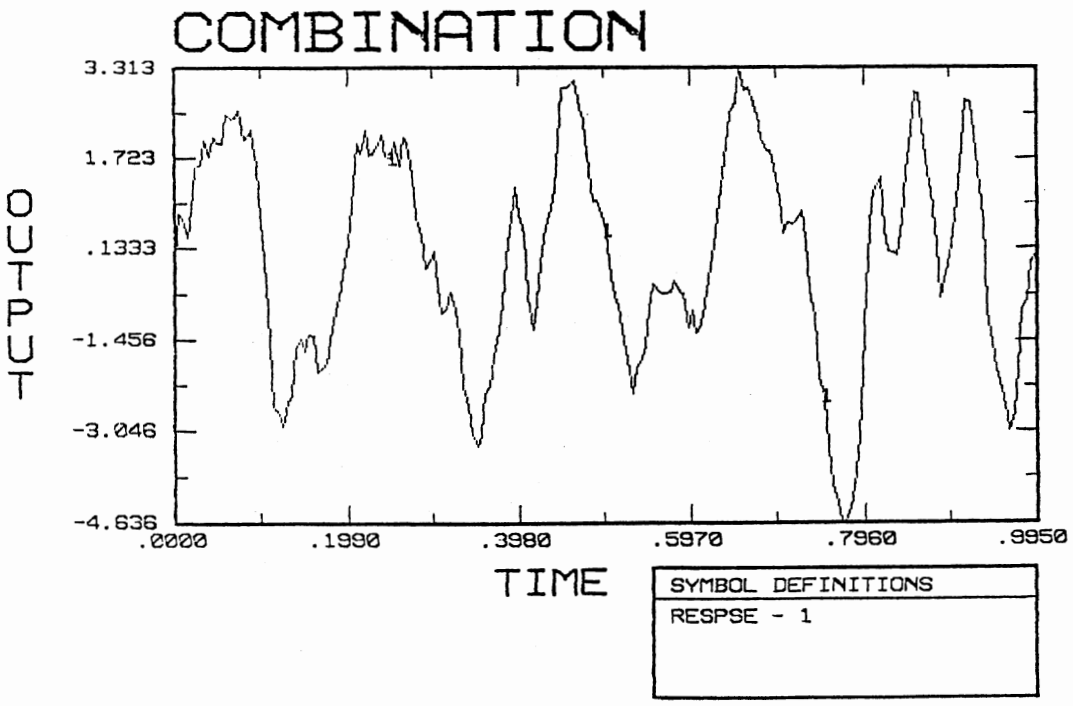


Figure 8.8. Response of Special Effect COMBINATION

The figure above illustrates the response of a special effect. This figure shows how the special effect COMBINATION is going to affect the yaw angle of the motion base in the time length of 1 second.

Geometric Transformation

The geometric transformation category illustrates the geometric transformation and smoothing algorithm block. The input to this block is SIGNA, which represents the position and angular orientation of the motion base. The output is CYLOT, the cylinder extensions.

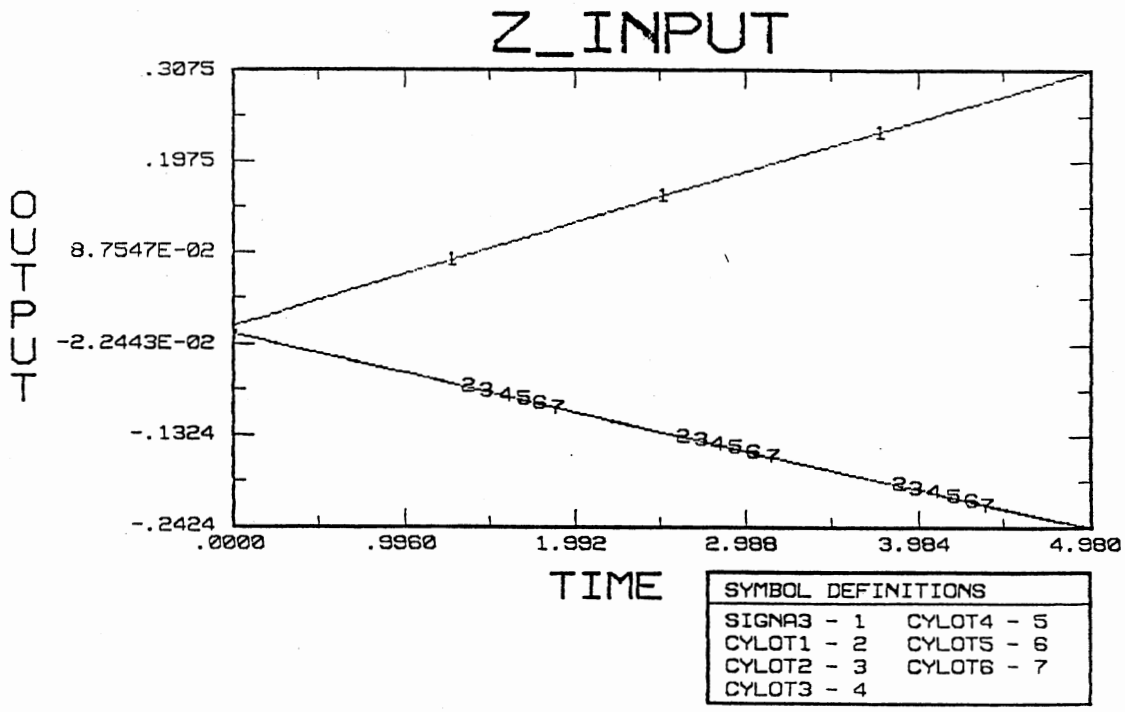


Figure 8.9. Input/Output of Geometric Transformation (Pure Z)

Figure 8.9 shows a pure z input (only SIGNA3). The outputs are cylinders 1-6 extension (CYLOT1 - CYLOT6). As expected all the cylinders move the same amount in the same time interval. This allows the motion base to move vertically. The position in the z axis is positive so all cylinders retract moving the motion base down (positive z direction).

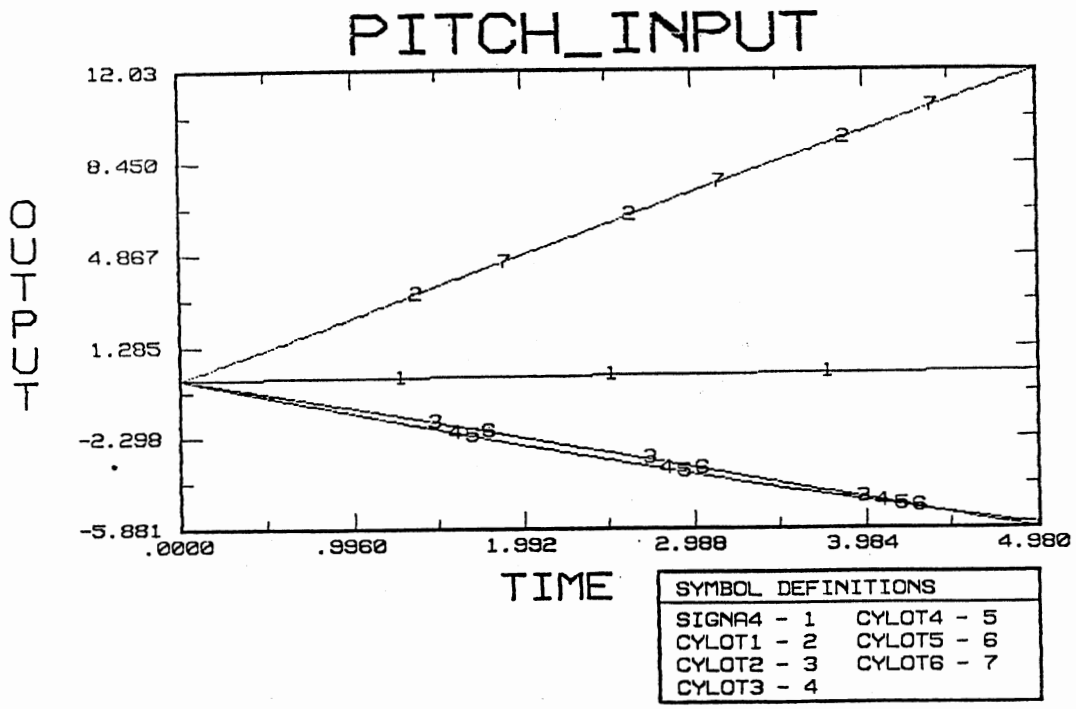


Figure 8.10. Input/Output of Geometric Transformation (Pure Pitch)

Figure 8.10 shows a pure pitch input (only SIGNA4). The outputs are cylinders 1-6 extension (CYLOT1 - CYLOT6). As expected the two front cylinders (1 and 6), the two back cylinders (3 and 4) and the two cylinders in the middle (2 and 5) (see Figure 1.1) move the same amount in the same time interval. The pitch angle is positive so the two front cylinders extend and the others retract making the motion base rotate up about the y axis (positive pitch).

Valve/Cylinder Model

The valve/cylinder model category illustrates the valve/cylinder combination response to a unit step as the desired cylinder extension.

```
PAGE     1    LIST VER  122084 4  11/16/88  13:47:48  SYS:0064..SRVPAR.SA

 2.1970E-03 2.1970E-03 2.1970E-03 2.1970E-03 2.1970E-03 2.1970E-03
 2.051 2.051 2.051 2.051 2.051 2.051
 4.351 4.351 4.351 4.351 4.351 4.351
 -5066. -5066. -5066. -5066. -5066. -5066.
 -1.000 -1.000 -1.000 -1.000 -1.000 -1.000
 1.000 1.000 1.000 1.000 1.000 1.000
 -2.2700E-05 -2.2700E-05 -2.2700E-05 -2.2700E-05 -2.2700E-05 -2.2700E-05
 .0000 .0000 .0000 .0000 .0000 .0000
 1064. 1064. 1064. 1064. 1064. 1064.
 .0000 .0000 .0000 .0000 .0000 .0000
 -531.9 -531.9 -531.9 -531.9 -531.9 -531.9
 -1064. -1064. -1064. -1064. -1064. -1064.
 -15.30 -16.30
PAGE     1    LIST VER  122084 4  11/17/88  11:43:23  SYS:0064..VALPAR.SA

  .7000 185.0 .1000 1000. .0000 .0000 .0000 .0000 7.000 210.0
  1.000 .6887 2.0000E+05 500.0 50.00
  1500. 1500. 1500. 1500. 1500. 1500.
  6.1300E+04 6.1300E+04 6.1300E+04 6.1300E+04 6.1300E+04 6.1300E+04
  487.8 487.8 487.8 487.8 487.8 487.8
```

Figure 8.11. Example of Servo and Valve/Cylinder Data Files

Figure 8.11 shows an example of a servo data file and a valve/cylinder combination data file. The values used here are for demonstration purposes only and do not represent a true valve/cylinder model or a servo.

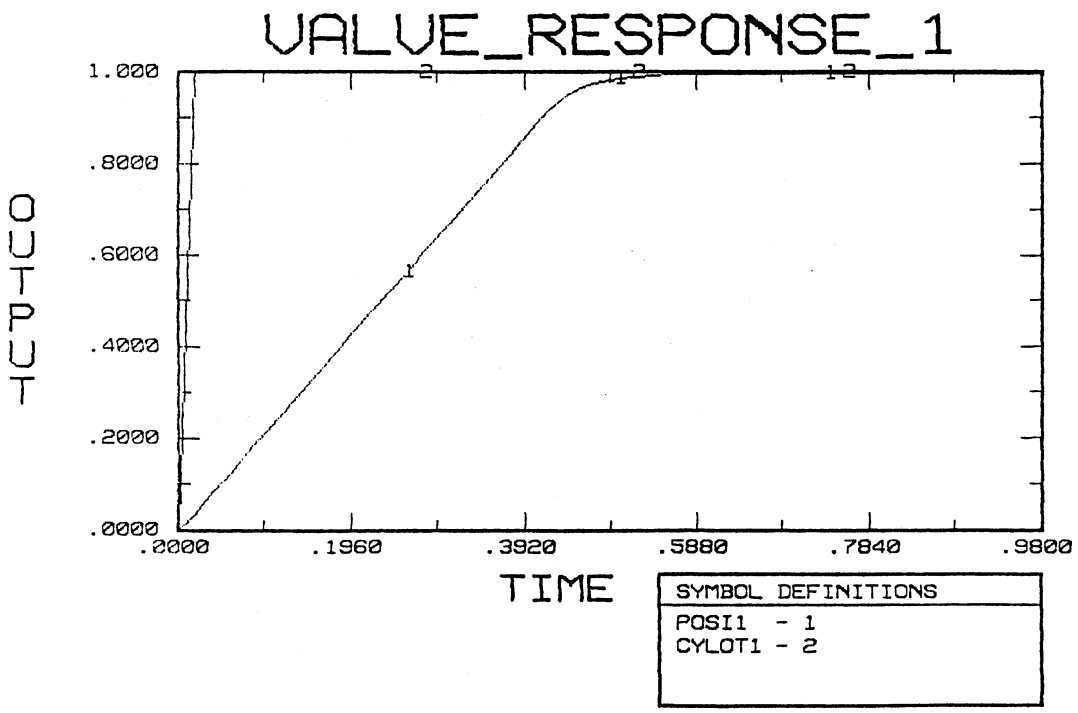


Figure 8.12. Valve/Cylinder/Servo Response 1

The figure above shows the response of the valve/cylinder model with position feedback. The purpose of this figure is to determine the settling time, about 0.5s, and also shows that the servo takes the cylinder smoothly to the desired position. The symbols are defined as:

- POSI1: the actual position of cylinder 1
- CYLOT1: the desired position of cylinder 1 (unit step)

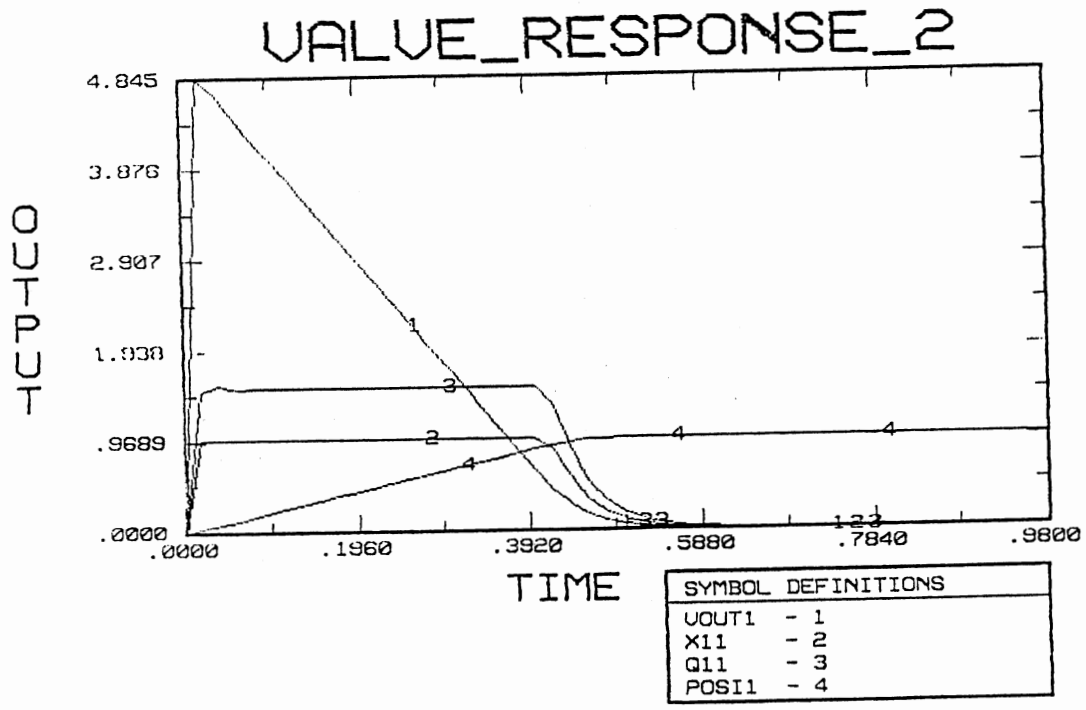


Figure 8.13. Valve/Cylinder/Servo Response 2

Figure 8.13 shows other variables in the unit step response. VOUT1 is the voltage applied to the motor to open the valve, derived from the differential position of the cylinder. X11 is the position of the spool and it reaches the maximum at 1 inch displacement. Q11 is the flow inside the valve. All the values of these variables decrease as the actual position of the cylinder (POSI1) reaches the desired position (1 in), as can be seen in the plot.

After the settling time for the response of the valve/cylinder model is obtained, an analysis of its response to a sine wave is performed. Figures 8.14 and 8.15 below show the response of the valve/cylinder model to a 0.01 Hz and 0.1 Hz sine wave respectively. A good response was obtained for both inputs even though there is some phase shift between commanded position and actual position.

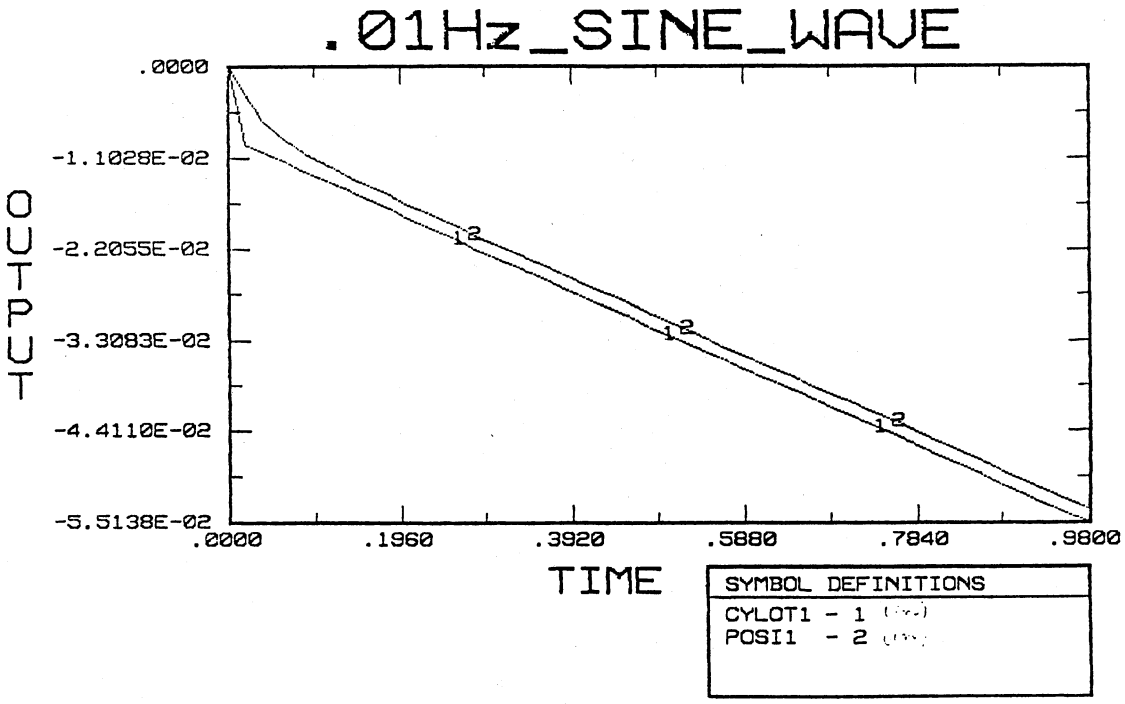


Figure 8.14. Valve/Cylinder Response to 0.01 Hz Sine Wave

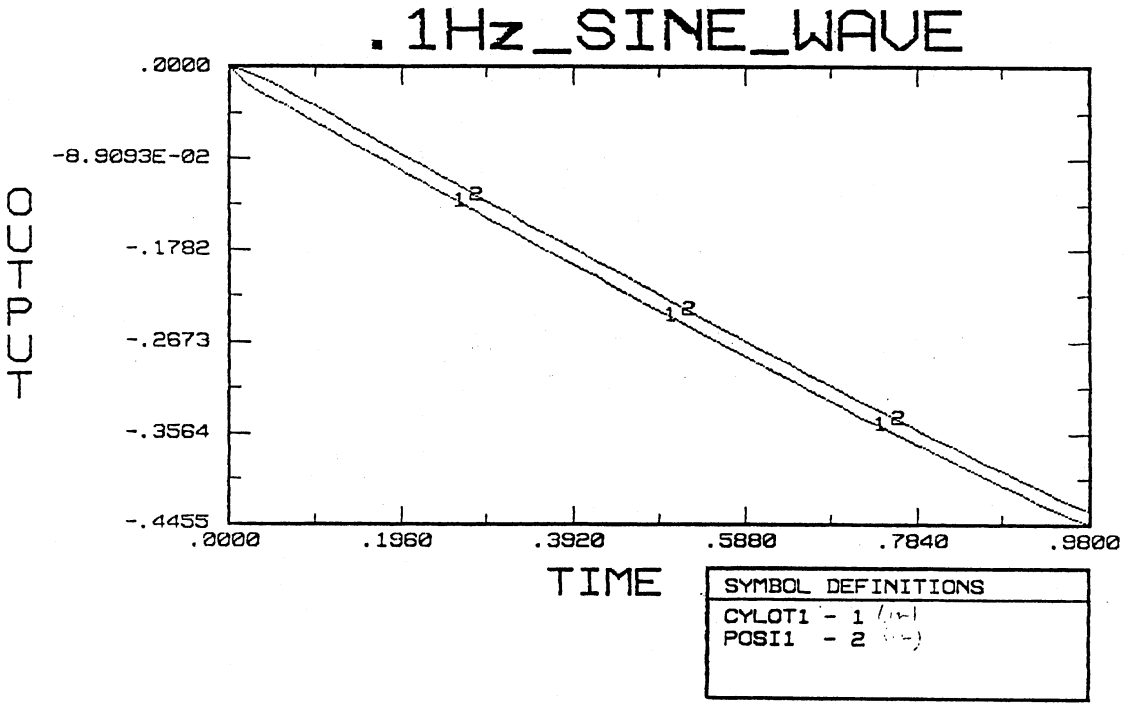


Figure 8.15. Valve/Cylinder Response to 0.1 Hz Sine Wave

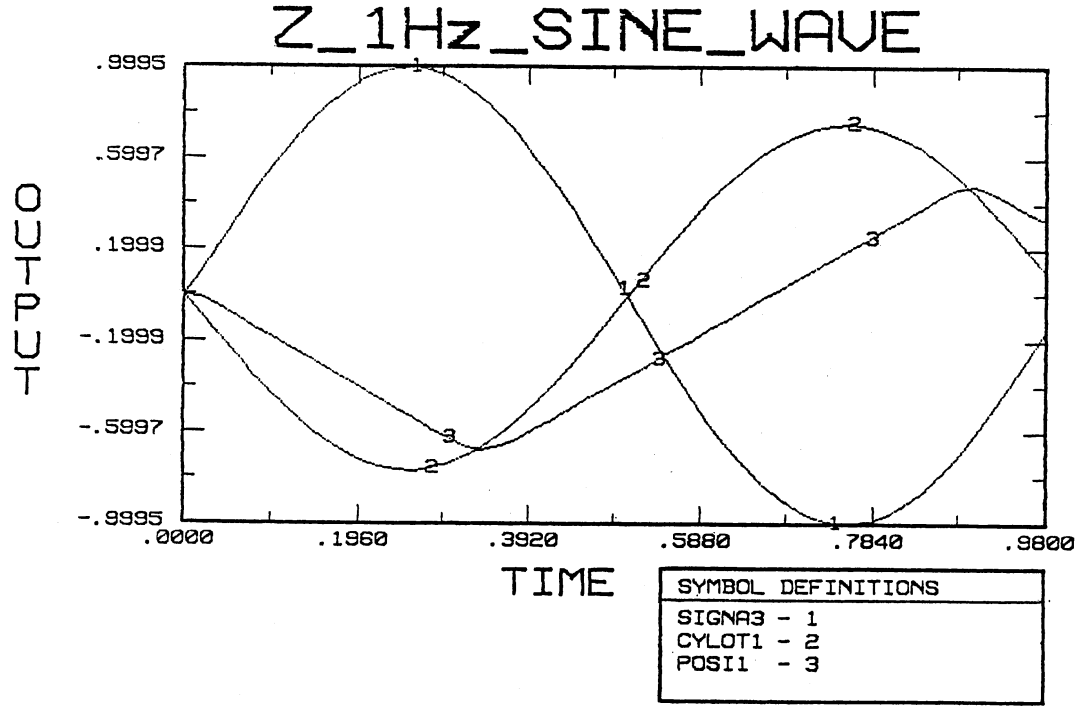


Figure 8.16. Valve/Cylinder Response to 1 Hz Sine Wave

Figure 8.16 shows the valve/cylinder model and servo response to a 1 Hz sine wave input. A bad response was obtained. SIGNA3 is the sine wave input to the z position of the motion base. CYLOT1 is the corresponding cylinder 1 extension, output of the geometric transformation and smoothing algorithm. POSI1 is the actual cylinder 1 extension which is further behind the desired cylinder extension. From the figure it can be concluded that the servo is not able to track the desired cylinder extension at 1 Hz. This occurs because the frequency of the sine wave is higher than the response of the valve/cylinder model.

Inverse Geometric Transformation

The inverse geometric transformation category illustrates the inverse geometric transformation block.

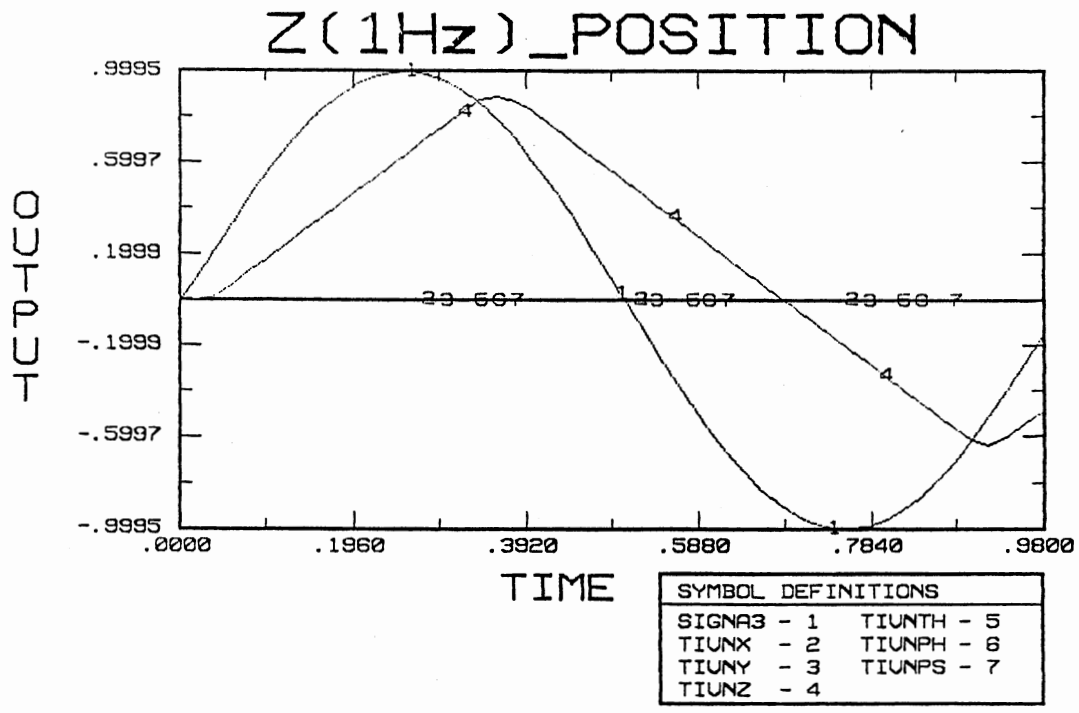


Figure 8.17. IGT of a 1 Hz Sine Wave

The figure above refers to Figure 8.16 where the servo does not track the desired cylinder position. This figure shows how the inverse transformation can provide the actual position and angular orientation of the motion base (TINVX-TINVPS). Also, the desired motion base position and angular orientation (SIGNA3) can be compared to actual values.

As an additional test, the forward geometric transformation was performed on the output of the inverse transformation. This forward transformation can be seen in Figure 8.18 below. The symbols in the figure are defined as:

- CYLOT1: desired cylinder 1 extension
- POSI1: actual cylinder 1 extension
- AFTER1: cylinder 1 extension derived from the output of the inverse geometric transformation.

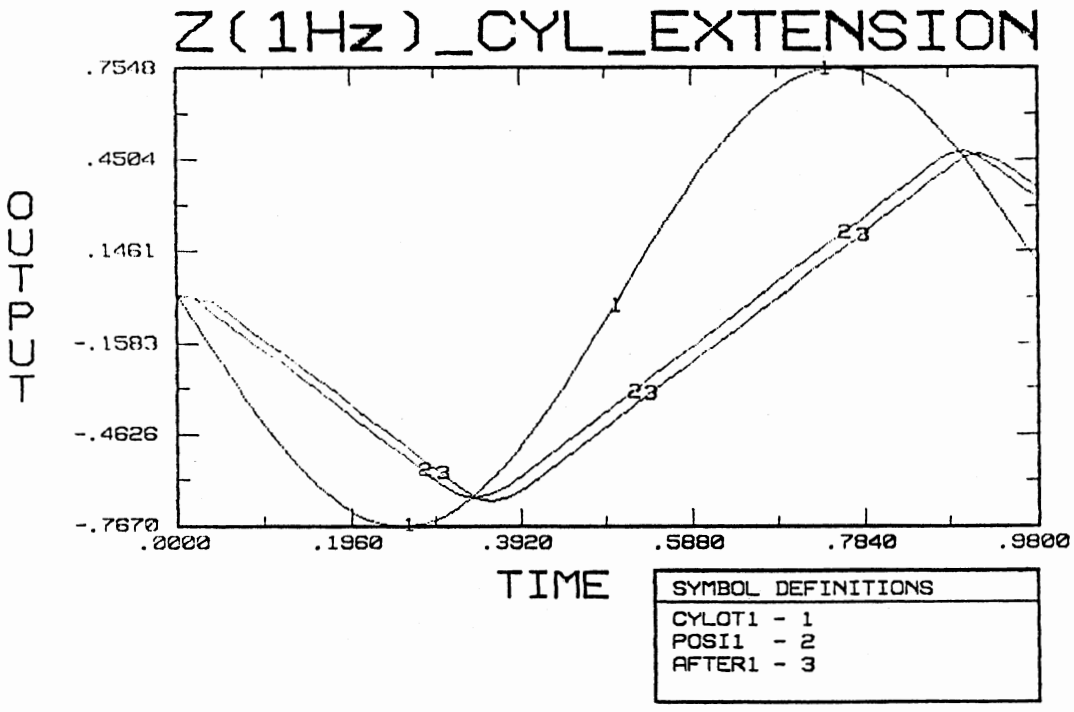


Figure 8.18. Cylinder Extensions to a 1 Hz Sine Wave

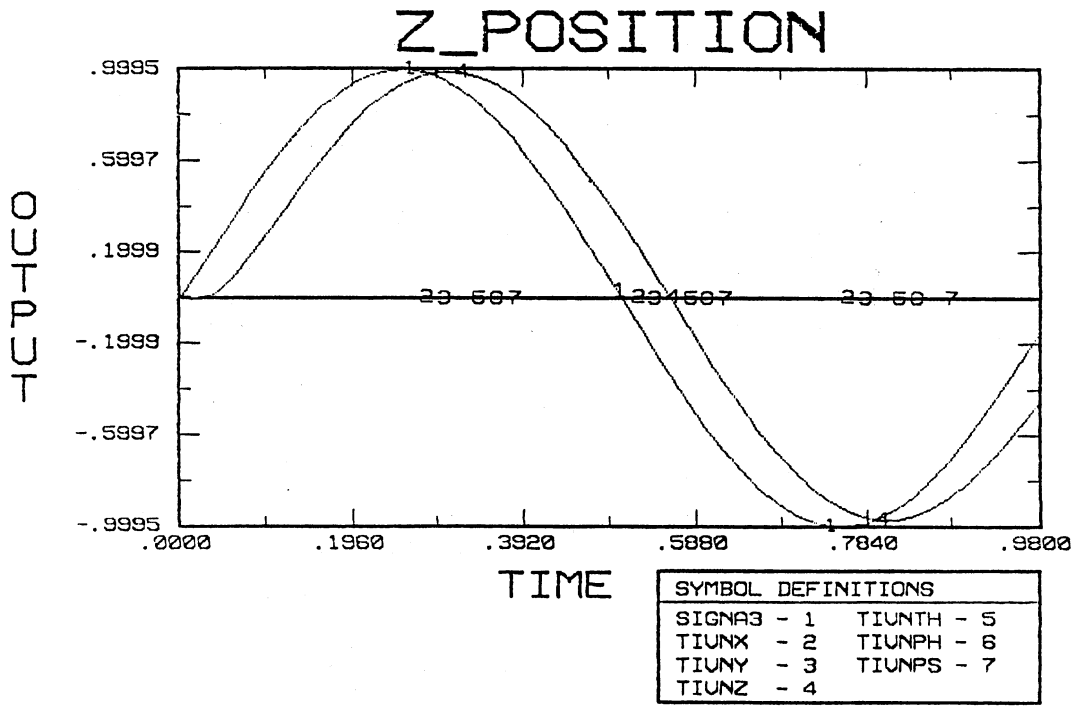


Figure 8.19. IGT Output to a Pure Z Input

Figure 8.19 shows the difference between desired and actual z position of the motion base when the servo is tracking the desired cylinder position. The difference occurs because of the phase shift described earlier. SIGNA3 is the desired z position and TINVZ is the actual z position given by the inverse geometric transformation with the actual cylinder extension. The difference between the actual cylinder extension (POSI1), the desired cylinder extension (CYLOT1) and the cylinder extension obtained from the output of the inverse geometric transformation (AFTER1) can be seen in Figure 8.20 below. The difference between actual cylinder extension (POSI) and desired cylinder extension (CYLOT) is caused by the phase shift in the servo.

The difference between actual cylinder extension (POSI) and cylinder extension obtained from the output of the inverse geometric transformation (AFTER) is due to approximations and tolerances for convergence in the algorithm for the inverse geometric transformation.

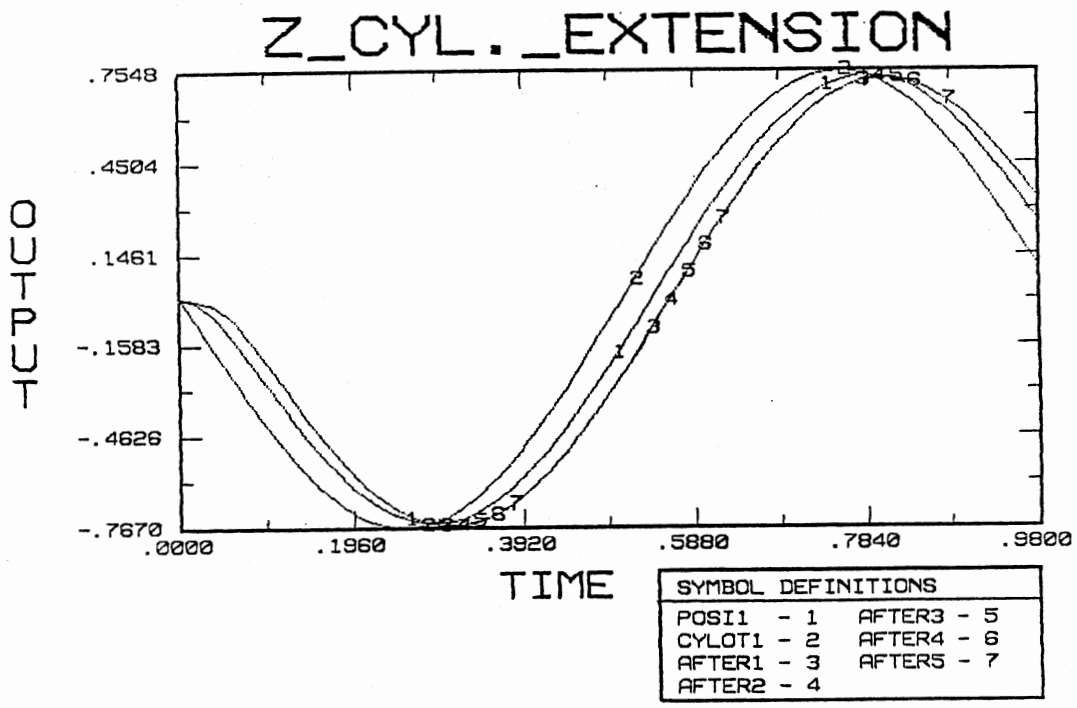


Figure 8.20. Cylinder Extension to a Pure Z Input

Another example was made to show the differences that occur between the actual and desired position and angular orientation of the motion base. The pitch angle was chosen. Figures 8.21 through 8.24 below give the same comparison plots as previous figures and include a printout showing the magnitude of the error.

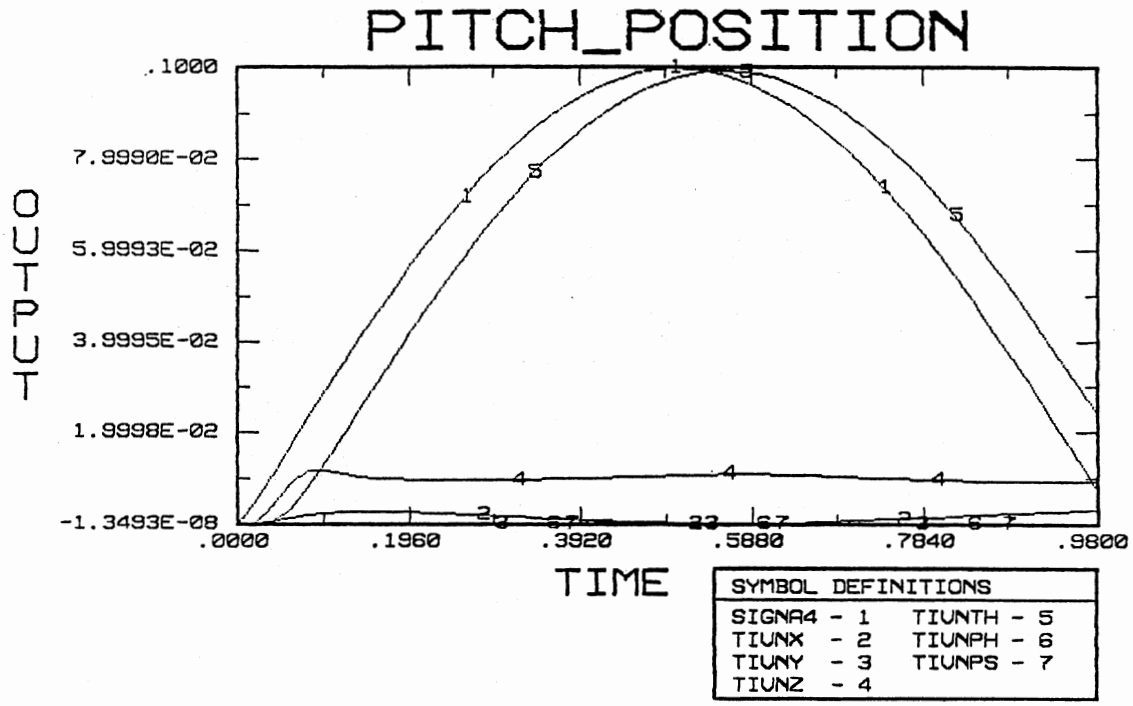


Figure 8.21. IGT Output to a Pure Pitch Input - Position

TIME	SIGNA4	TIVNX	TIVNY	TIVNZ	TIVNTH	TIVNPH	TIVNPS
0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00
0.20000E-01	0.47107E-02	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00
0.40000E-01	0.10973E-01	0.57682E-03	-0.60871E-08	0.40131E-02	0.65524E-03	-0.45550E-10	0.11438E-09
0.60000E-01	0.17193E-01	0.16129E-02	0.23997E-08	0.92316E-02	0.36229E-02	-0.82736E-10	0.84346E-10
0.80000E-01	0.23345E-01	0.22817E-02	-0.11335E-07	0.11757E-01	0.82719E-02	-0.57012E-10	0.16319E-09
0.10000E+00	0.29404E-01	0.26283E-02	0.40309E-07	0.11719E-01	0.13718E-01	0.11337E-09	-0.10406E-09
0.12000E+00	0.35348E-01	0.28467E-02	-0.60847E-08	0.11108E-01	0.19489E-01	0.25342E-09	-0.14373E-09
0.14000E+00	0.41152E-01	0.29435E-02	-0.12582E-07	0.10658E-01	0.25388E-01	0.64414E-10	0.21379E-09
0.16000E+00	0.46793E-01	0.29247E-02	-0.58645E-08	0.10315E-01	0.31293E-01	0.78762E-10	-0.11629E-09
0.18000E+00	0.52250E-01	0.28393E-02	-0.13493E-07	0.10048E-01	0.37126E-01	-0.13513E-09	0.15759E-10
0.20000E+00	0.57501E-01	0.27190E-02	0.26163E-08	0.98724E-02	0.42835E-01	0.17168E-10	-0.14426E-09
0.22000E+00	0.62524E-01	0.25811E-02	-0.26613E-09	0.97885E-02	0.48387E-01	-0.49334E-10	0.10388E-09
0.24000E+00	0.67301E-01	0.24170E-02	0.47249E-08	0.97504E-02	0.53753E-01	0.38820E-10	0.18879E-09
0.26000E+00	0.71813E-01	0.22432E-02	0.15583E-08	0.97504E-02	0.58909E-01	-0.60989E-10	0.78115E-10
0.28000E+00	0.76041E-01	0.20588E-02	-0.40801E-09	0.97656E-02	0.63833E-01	0.10005E-10	0.61812E-10

Figure 8.22. Printout of IGT Output to a Pure Pitch Input - Position

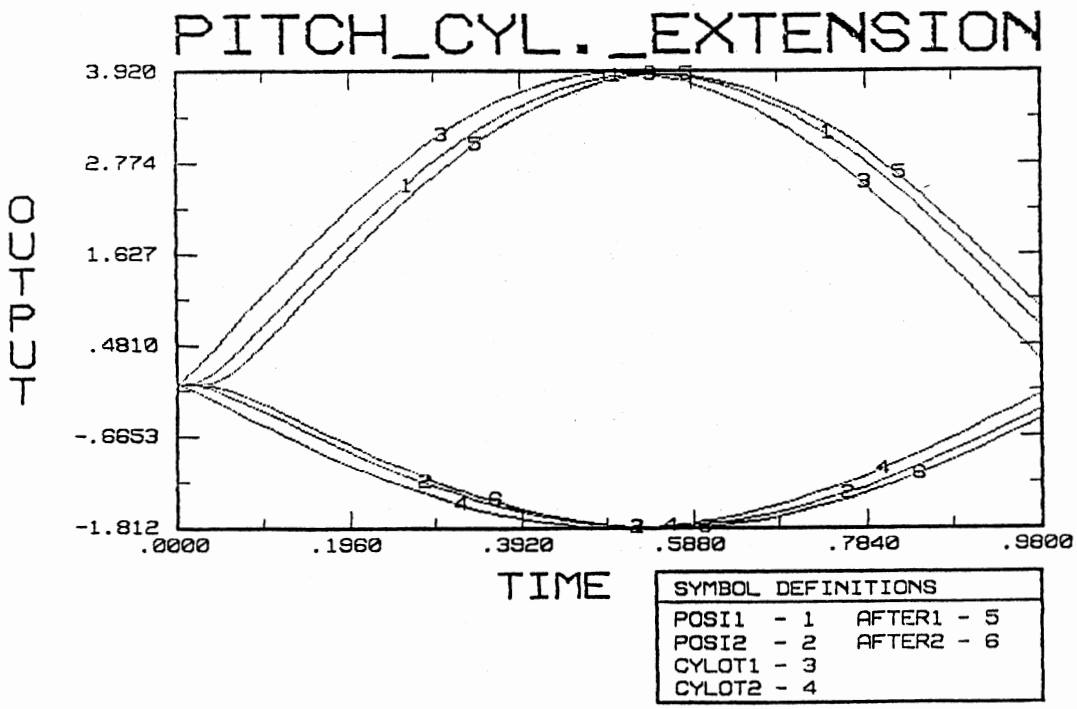


Figure 8.23. IGT Output to a Pure Pitch Input - Cylinder Extension

TIME	POSI1	POSI2	CYLOT1	CYLOT2	AFTER1	AFTER2
0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00
0.20000E-01	0.21207E-01	-0.14114E-01	0.17593E+00	-0.90904E-01	-0.85373E-02	-0.85449E-02
0.40000E-01	0.13165E+00	-0.69791E-01	0.42132E+00	-0.20081E+00	0.14160E-01	-0.23392E-01
0.60000E-01	0.31078E+00	-0.15317E+00	0.66515E+00	-0.31041E+00	0.12657E+00	-0.79910E-01
0.80000E-01	0.52380E+00	-0.24894E+00	0.90646E+00	-0.41925E+00	0.30688E+00	-0.16376E+00
0.10000E+00	0.75045E+00	-0.35040E+00	0.11443E+01	-0.52690E+00	0.52042E+00	-0.25974E+00
0.12000E+00	0.98218E+00	-0.45461E+00	0.13776E+01	-0.63289E+00	0.74726E+00	-0.36131E+00
0.14000E+00	0.12143E+01	-0.55940E+00	0.16056E+01	-0.73679E+00	0.97904E+00	-0.46555E+00
0.16000E+00	0.14437E+01	-0.66333E+00	0.18273E+01	-0.83814E+00	0.12111E+01	-0.57032E+00
0.18000E+00	0.16682E+01	-0.76550E+00	0.20418E+01	-0.93652E+00	0.14403E+01	-0.67422E+00
0.20000E+00	0.18866E+01	-0.86526E+00	0.22482E+01	-0.10315E+01	0.16648E+01	-0.77635E+00
0.22000E+00	0.20977E+01	-0.96205E+00	0.24457E+01	-0.11227E+01	0.18830E+01	-0.87606E+00
0.24000E+00	0.23007E+01	-0.10554E+01	0.26336E+01	-0.12096E+01	0.20939E+01	-0.97279E+00
0.26000E+00	0.24945E+01	-0.11448E+01	0.28111E+01	-0.12920E+01	0.22966E+01	-0.10660E+01
0.28000E+00	0.26785E+01	-0.12299E+01	0.29774E+01	-0.13693E+01	0.24902E+01	-0.11554E+01

Figure 8.24. Printout of IGT Output to a Pure Pitch Input - Cylinder Extensions

Animation

The animation category illustrates some of the screens seen on the graphics terminal during the animation of the motion base system simulation. The fixed lower base platform is represented by a filled-in triangle, the moving upper platform is represented by an open triangle. Figure 8.25 shows the motion base in the initial neutral position from a front angle view. Figure 8.26 shows the motion base in a positive pitch angle from a straight side view. Figure 8.27 shows the motion base in a positive yaw angle from a straight top view. Figure 8.28 shows the motion base in a negative y position from a straight front view.

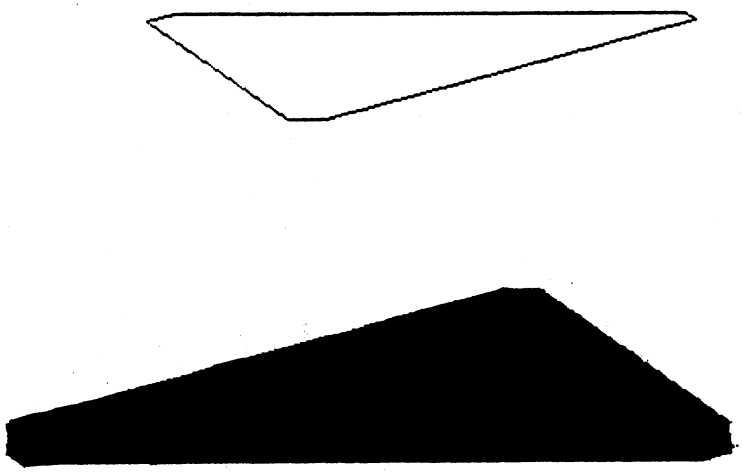


Figure 8.25. Motion Base in Initial Neutral Position - Angle View

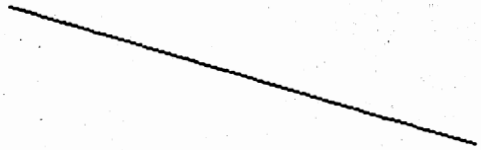


Figure 8.26. Motion Base in Pitch Angle - Side View

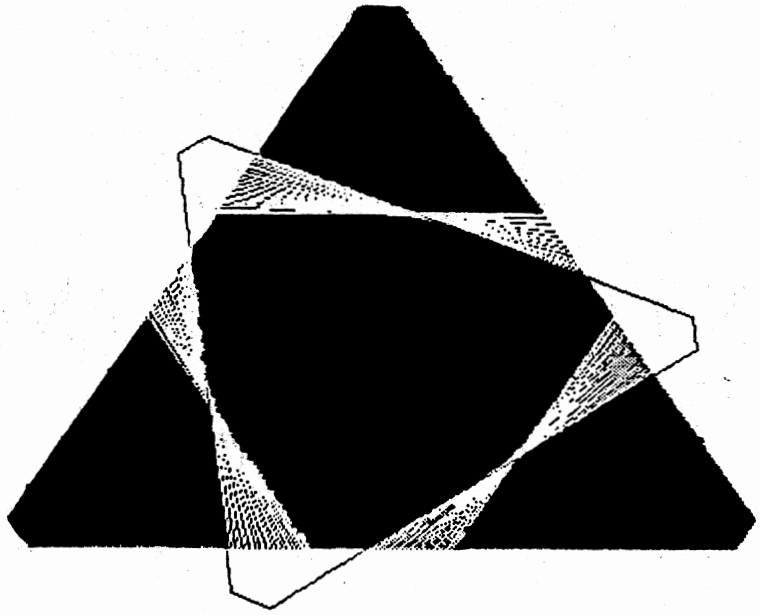


Figure 8.27. Motion Base in Yaw Angle - Top View

Figure 8.28. Motion Base in Y Displacement - Front View

This category concludes the examples of the program features. In the next chapter a conclusion is made and future improvements are suggested.

CHAPTER IX

CONCLUSION

A FORTRAN program to simulate a six degree-of-freedom motion base system for flight simulators was developed. A study of the system was performed prior to writing the program code. A general description of the system was given in Chapter I. In the following chapters all the blocks pertinent to the simulation were analyzed. A valve/cylinder model was defined in Chapter VI so the simulation would become possible. Finally, some examples were given in chapter VIII.

Currently, this type of program is not readily available to simulator designers. The program provides a means of testing the dimensions and parameter variations for a six degree-of-freedom motion base without ever building the hardware. It also permits the testing of new features of the actual motion base system software and helps engineers in the calibration process of the motion base system making it faster and less expensive. Therefore it should be helpful to most simulator designers.

In the current valve/cylinder simulation, the weight of the motion base on each cylinder is considered constant, although this is not what happens in reality. As the motion base moves the weight on the cylinders change. Further improvements can be made in the model to account for the different weights on the cylinders.

Also, a routine to enter and change the primary motion parameters might be written, making the program completely user-friendly.

Other integration methods could be made available for the spool displacement and the piston displacement calclulations. Perhaps to permit the change from the continuous time system to its equivalent discrete time system.

Finally, the animation can be improved by adding the six cylinders attached to lower and upper base platforms. This allows a nicer view of the motion base system animation.

REFERENCES

[1] J. M. Rolfe and K. J. Staples, *Flight Simulation*. Cambridge, U.K.: Cambridge University Press, 1986.

[2] B. Friedland, C. K. Ling and M. F. Hutton, "Quasi-Optimum Design of a Six Degree of Freedom Moving Base Simulator Control System," NASA CR-2312, Oct. 1973.

[3] S. F. Schmidt and B. Conrad, "The Calculation of Motion Drive Signals for Piloted Flight Simulators," NASA CR-73375, Oct. 1969.

[4] S. F. Schmidt and B. Conrad, "Motion Drive Signals for Piloted Flight Simulators," NASA CR-1601, May 1970.

[5] A. Papuolis, *Probability, Random Varialbes, and Stochastic Processes*. New York: McGraw-Hill, 1984.

[6] B. C. Kuo, *Automatic Control Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[7] J. E. Dieudonne, R.V. Parrish and R. E. Bardusch, "An Actuator Extension Transformation for a Motion Simulator and an Inverse Transformation Applying Newton-Raphson's Method," NASA TN D-7067, Nov. 1972.

[8] W. R. Ahrendt and C. J. Savant, Jr., *Servomechanism Practice*. York, PA: McGraw-Hill,1960.

[9] P. L. Taylor, *Servomechanisms*. London, U.K.: Longmans, 1969.

[10] K. Ogata, *System Dynamics*. Englewood Cliffs, NJ: Prentice-Hall, 1978.

[11] K. Ogata, *Discrete Time Control Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[12] C. F. Gerald and P. O. Wheatley, *Applied Numerical Analysis*. Menlo Park, CA: Addison-Wesley, 1984.

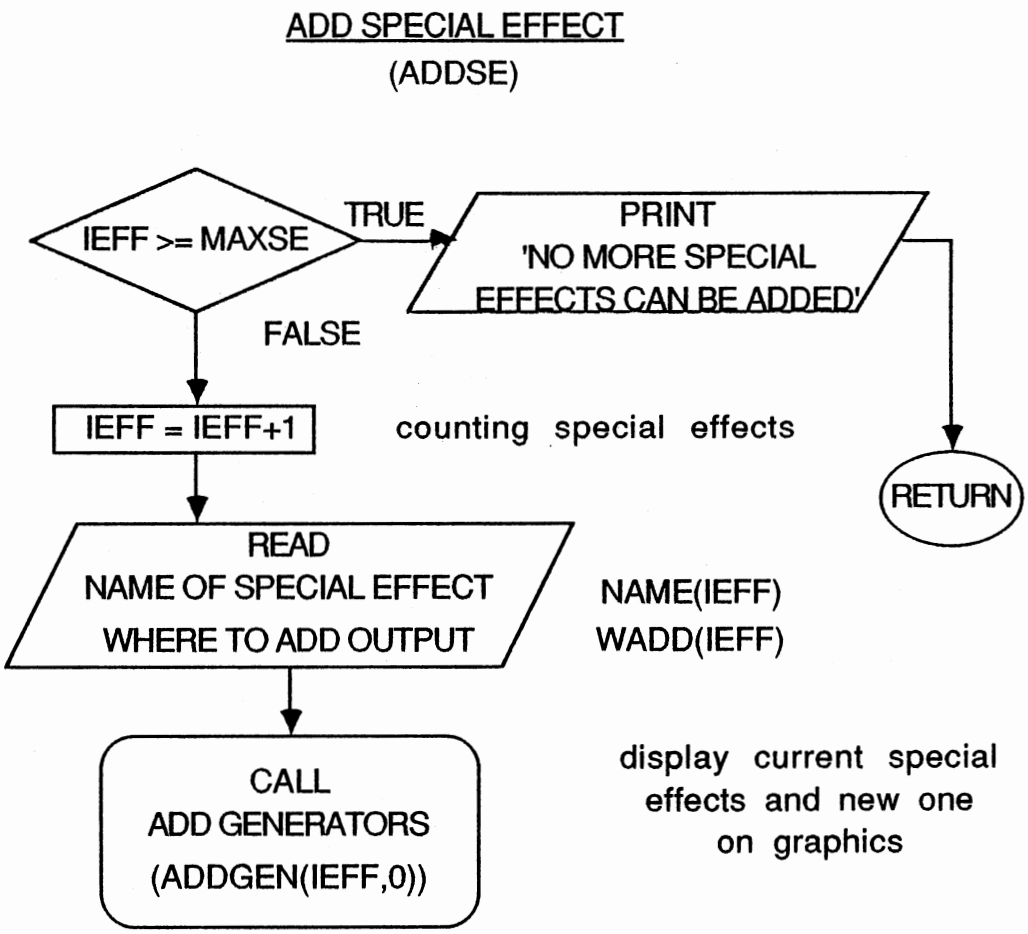
[13] L.E. Scales, *Introduction to Non-Linear Optimization.* New York: Springer-Verlag New York Inc., 1985.

[14] "Two-Stage Flow Control Servovalves." Catalog 762682. Noplace: MOOG.

APPENDIXES

APPENDIX A

ADDITIONAL FLOWCHARTS FOR SPECIAL EFFECTS SUBROUTINES



The maximum number of special effects is 20.
MAXSE = 20

Figure A.1. Flowchart of Subroutine ADDSE

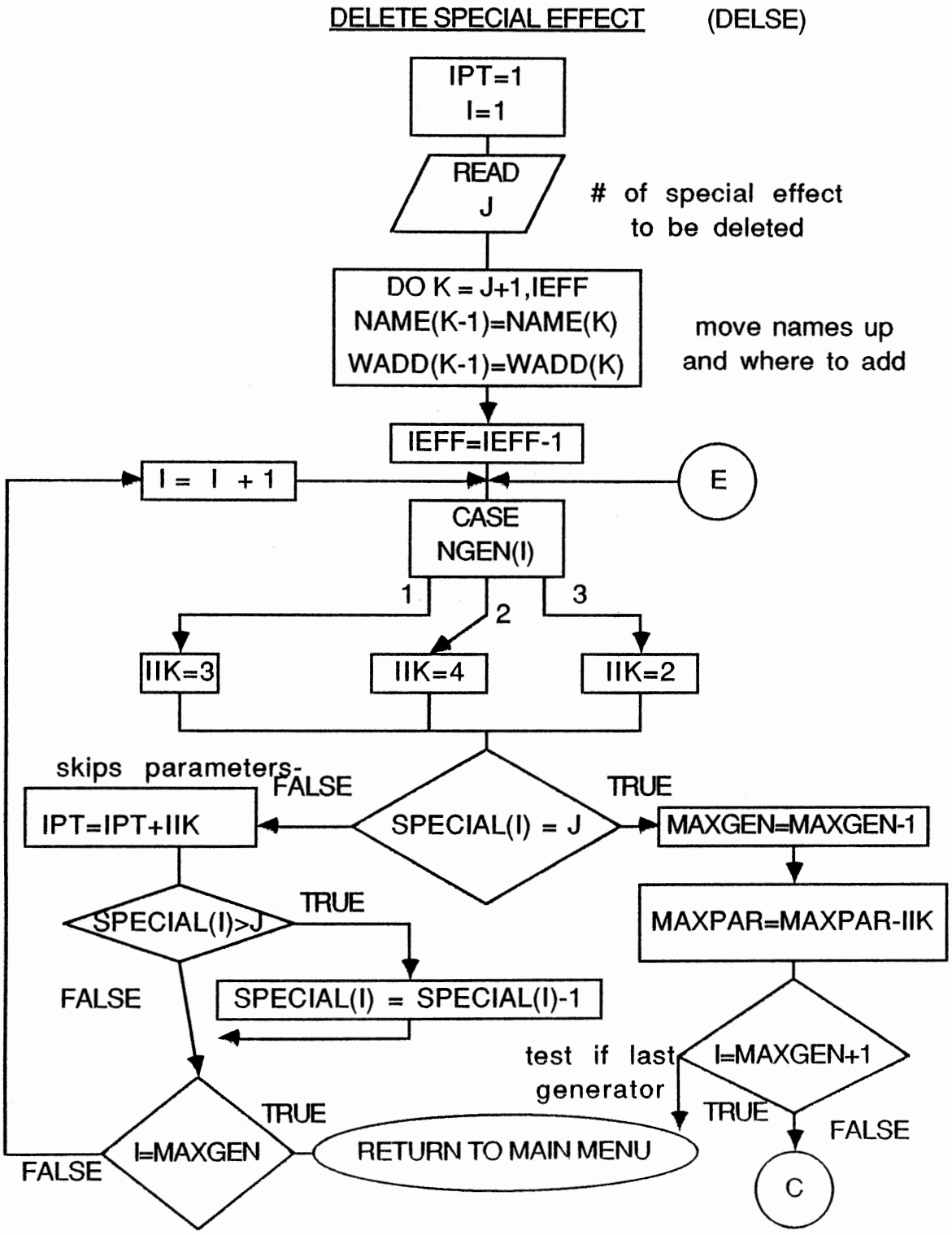


Figure A.2. Flowchart of Subroutine DELSE

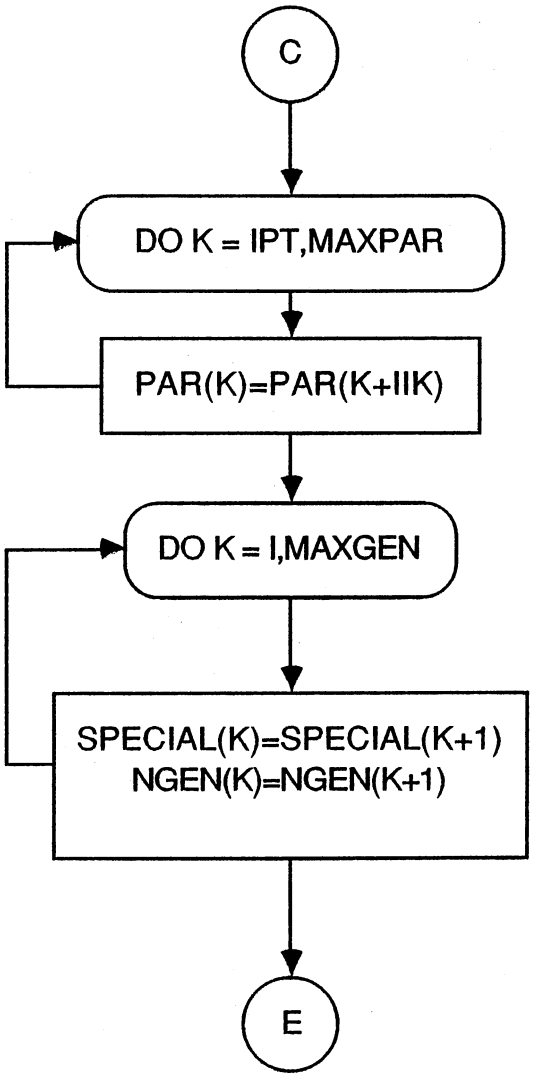


Figure A.2. (Continued)

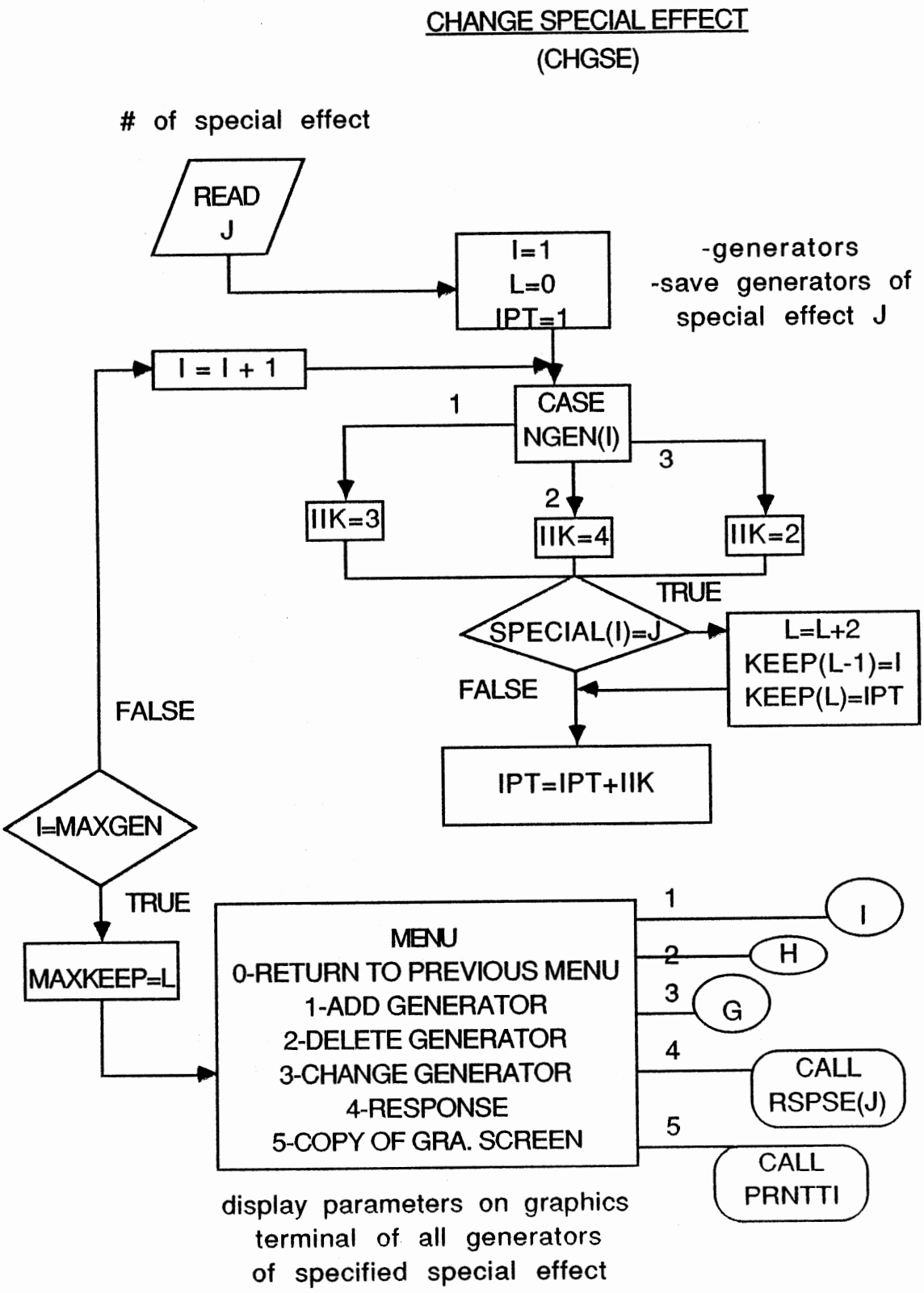


Figure A.3. Flowchart of Subroutine CHGSE

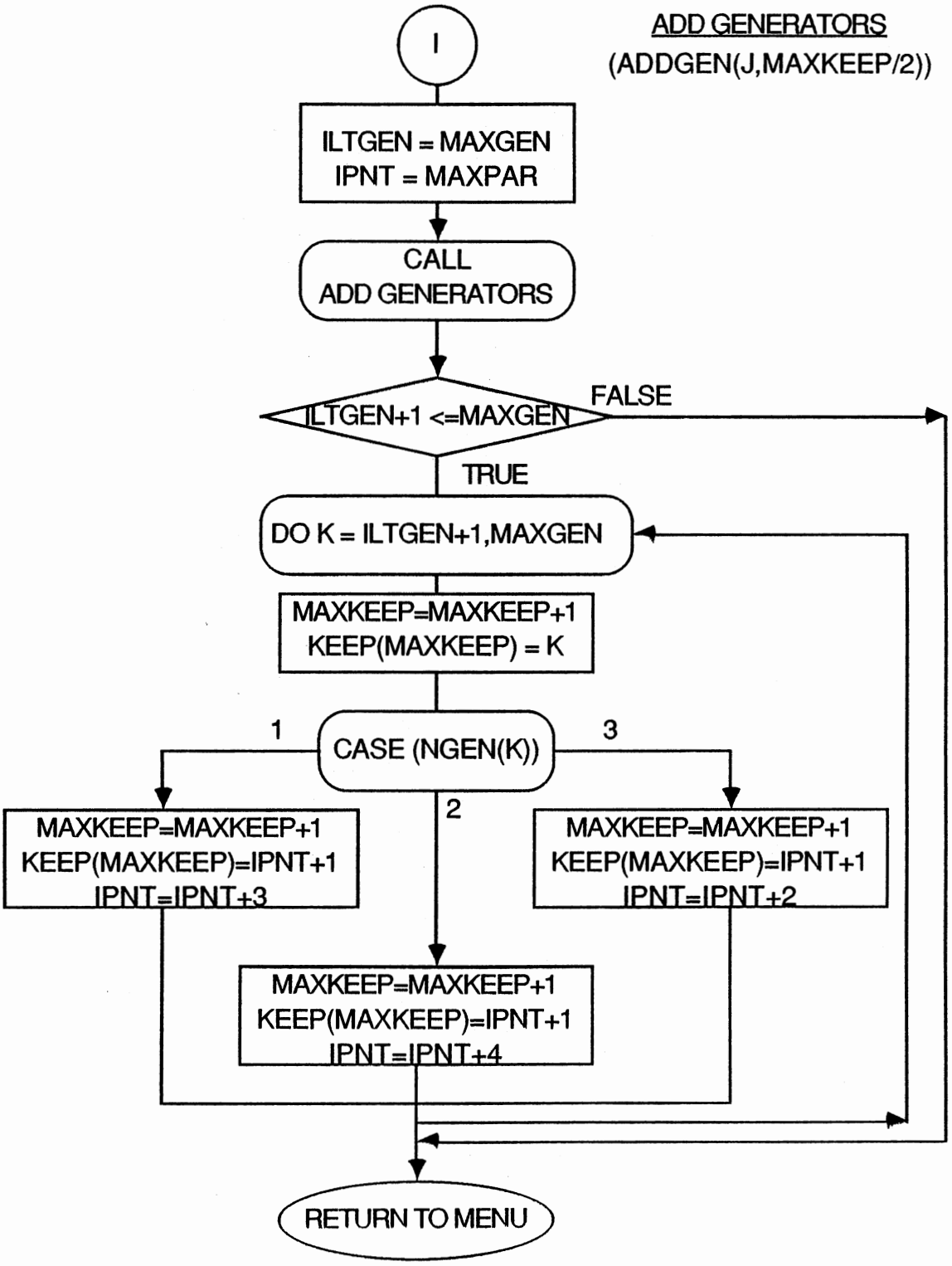


Figure A.3. (Continued)

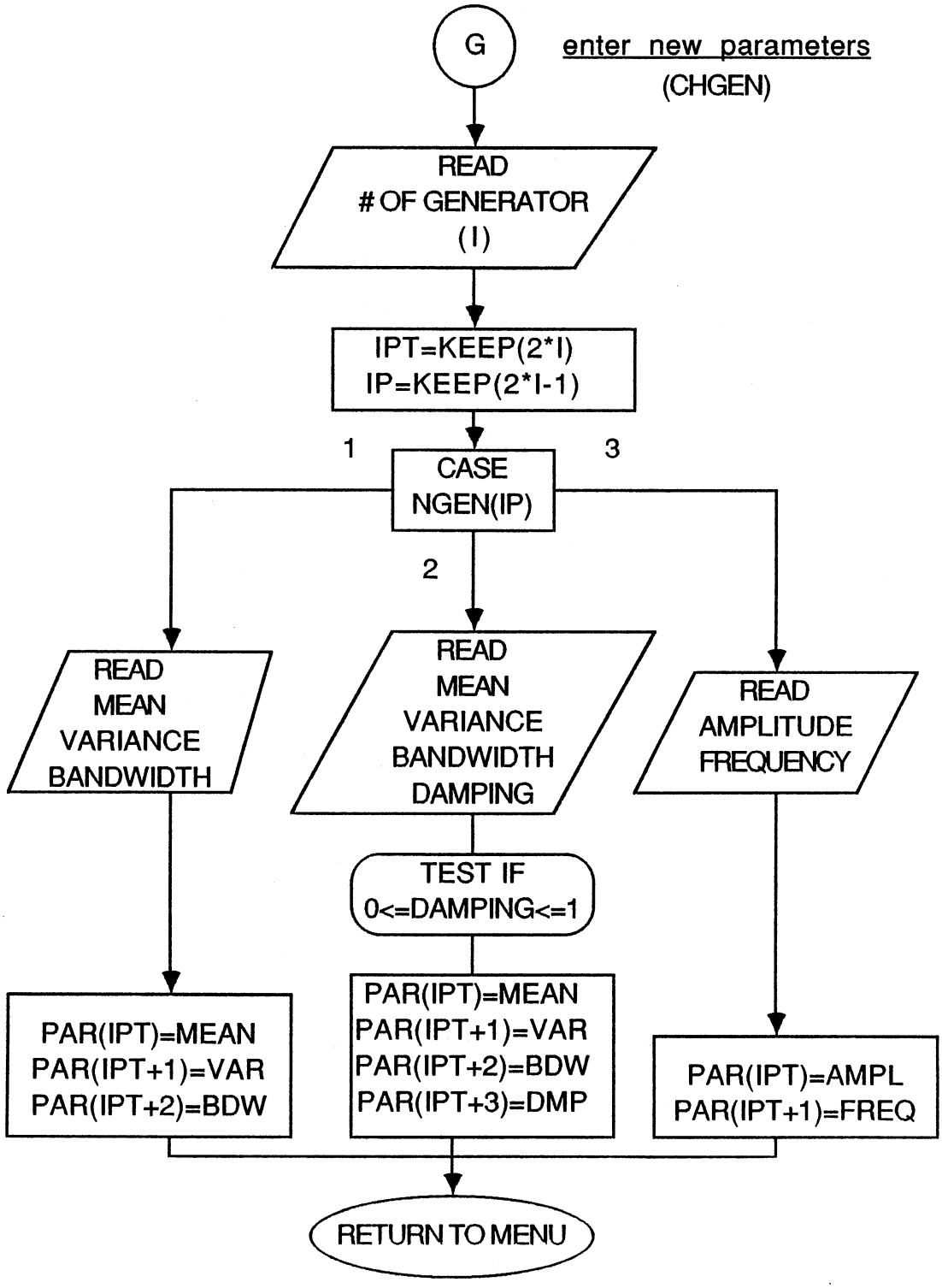


Figure A.3. (Continued)

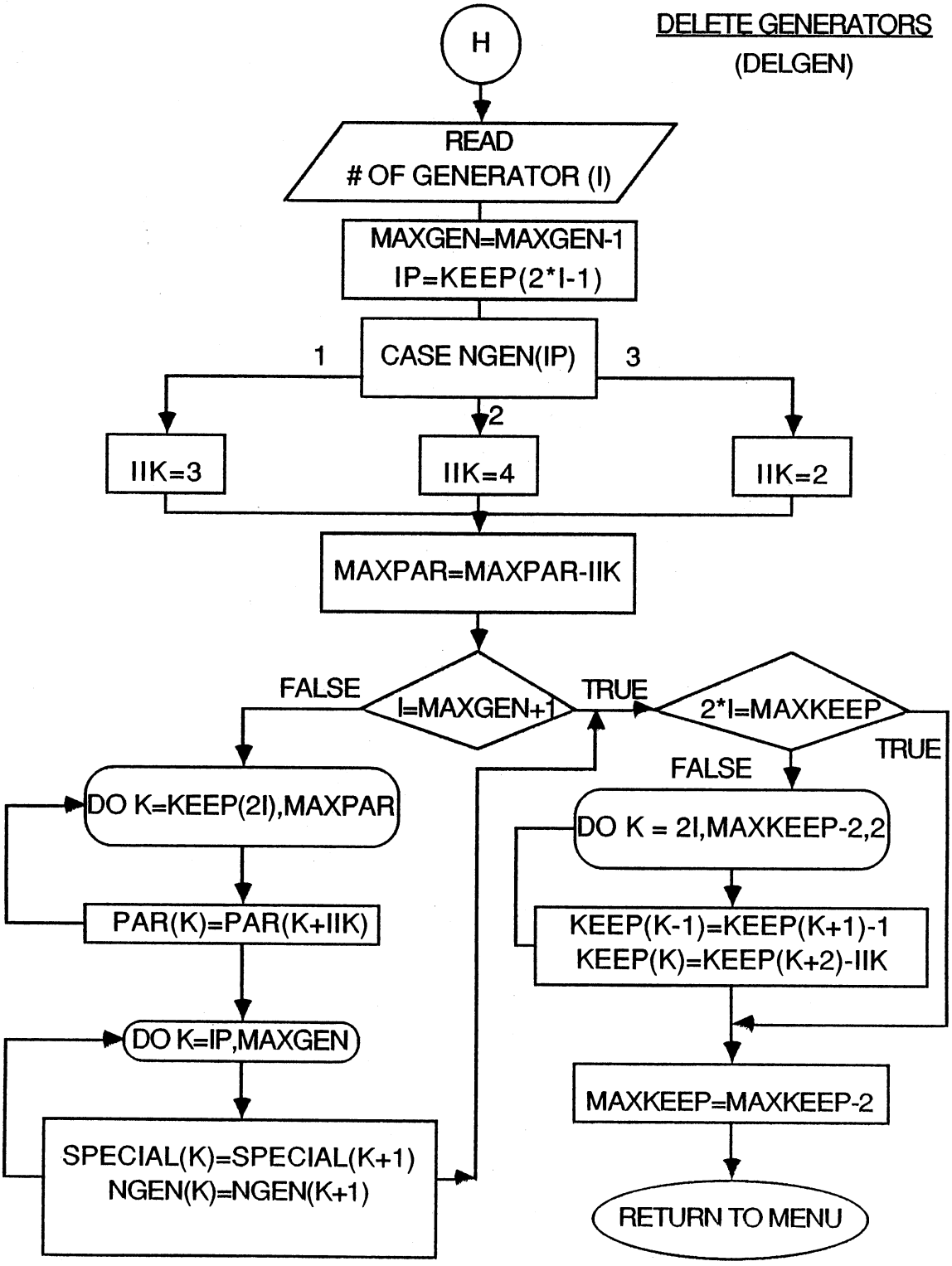


Figure A.3. (Continued)

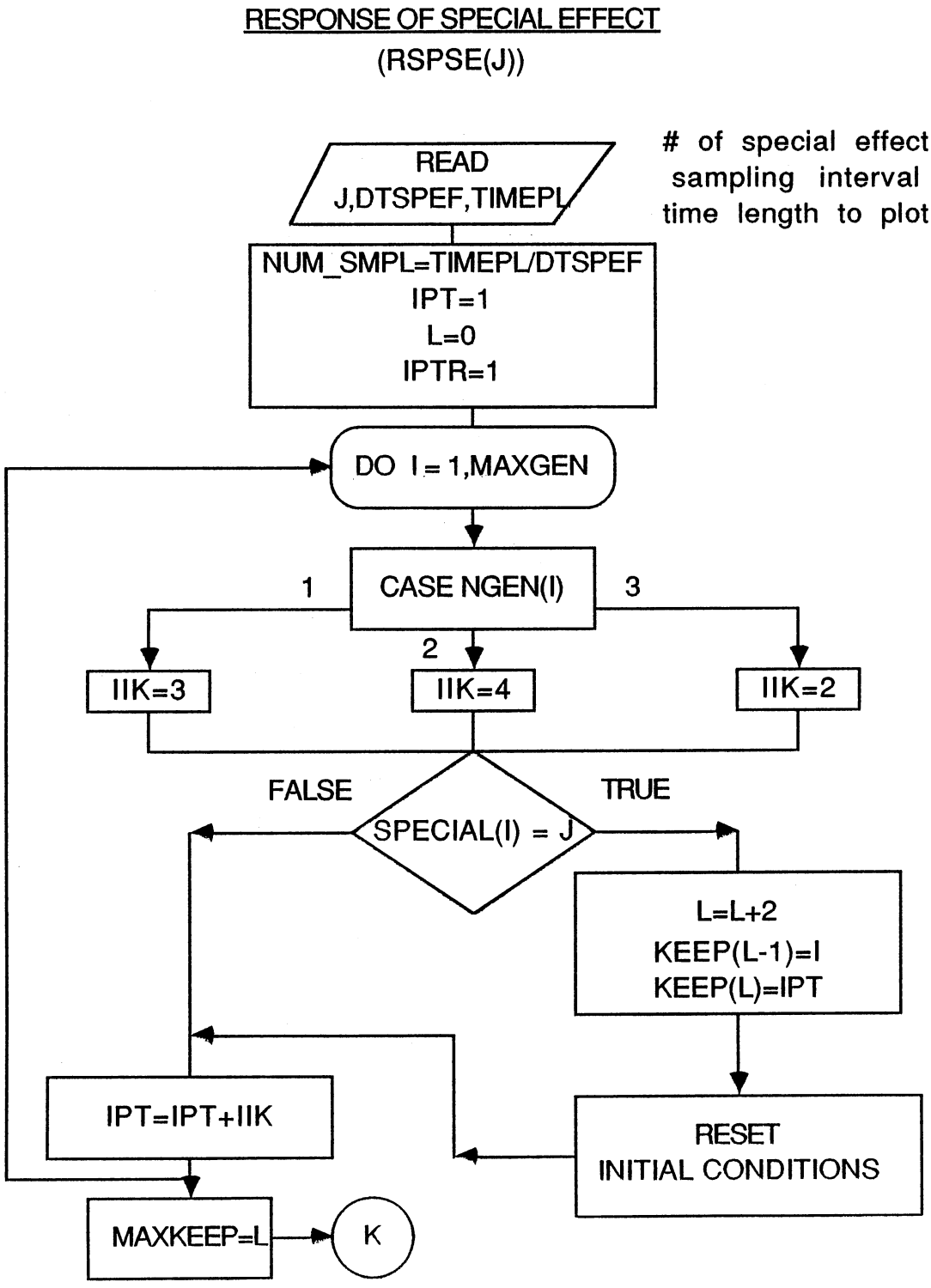


Figure A.4. Flowchart of Subroutine RSPSE

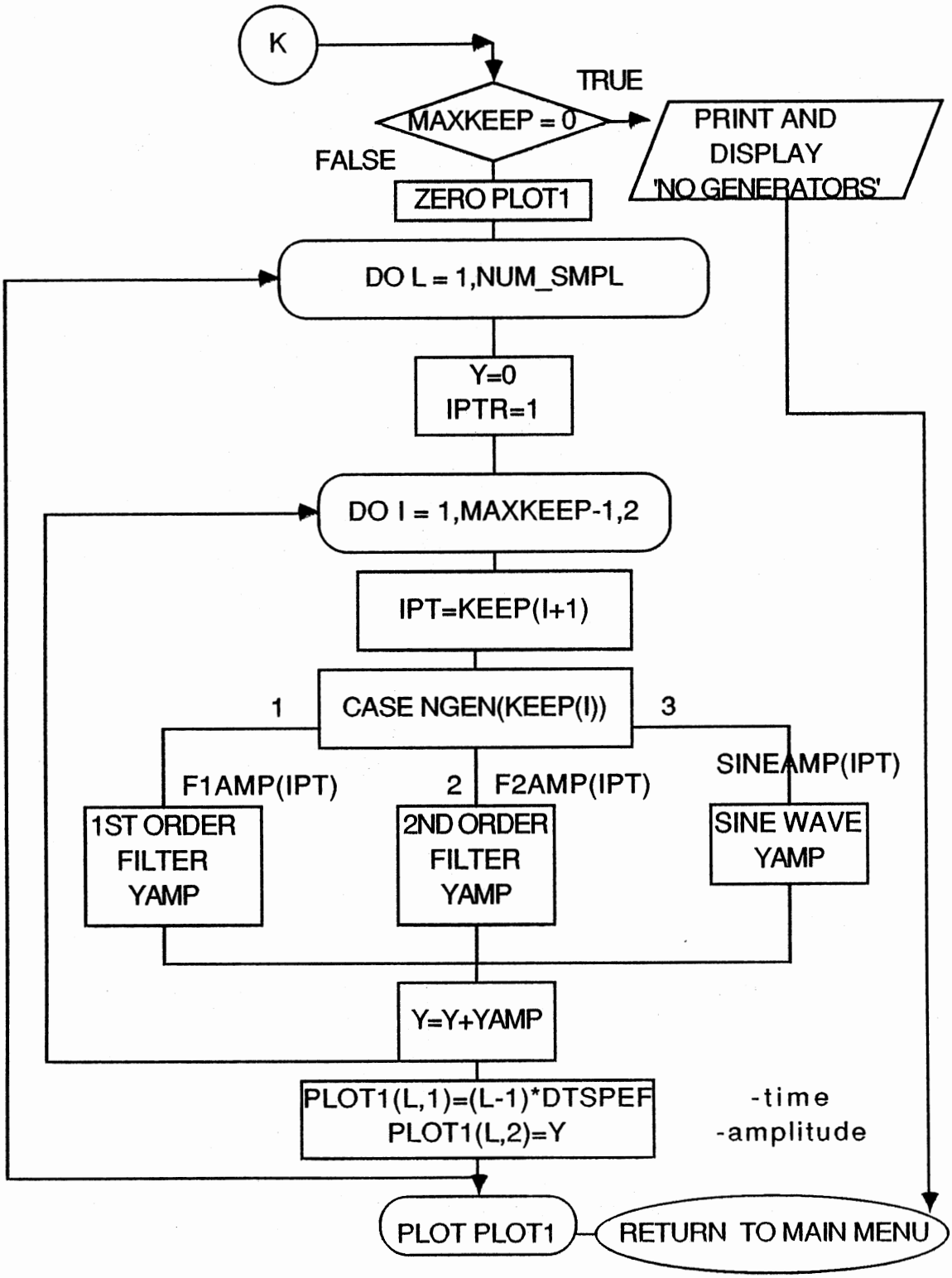
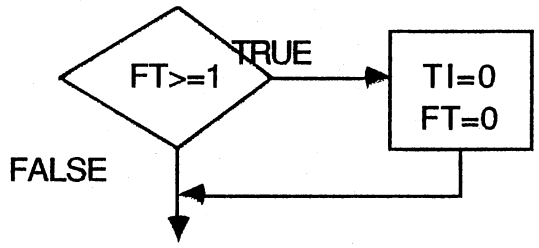


Figure A.4. (Continued)

SINE WAVE (SINEAMP(IPT))

A=PAR(IPT)
F=PAR(IPT+1)
IPT=IPT+2
TI=PSTO(IPTR)
FT=F*TI



YAMP=A*SIN(2*PI*FT)
TI=TI+DTSPEF
PSTO(IPTR)=TI
IPTR=IPTR+1

WHITE NOISE GENARATOR (WHITE)

U=WHITE(M,V)

INIT=PSTO(IPTR)
INIT=MOD(3125*INIT,65536)
X=FLOAT(INIT)/65536.
U=(X-0.5)*V+M
PSTO(IPTR)=INIT

Figure A.5. Flowchart of Subroutines SINEAMP and WHITE

1ST ORDER FILTER (F1AMP(IPT))

```
M=PAR(IPT)
V=SQRT(12*PAR(IPT+1))
A=EXP(-DTSPEF*2*PI*PAR(IPT+2))
B=1-A
IPT=IPT+3
X1=PSTO(IPTR+1)
YAMP=X1
U=WHITE(M,V)
X1=A*X1+B*U
PSTO(IPTR+1)=X1
IPTR=IPTR+2
```

2ND ORDER FILTER

EQUATIONS I

```
AD(1,1)=(1-BW*DTSPEF)*TEMP
AD(2,1)=-SEDT*TEMP
AD(1,2)=-AD(2,1)*(BW**2)
AD(2,2)=(1+BW*DTSPEF)*TEMP
BD(1)=-[AD(1,1)-1+2*(Z/BW)*AD(1,2)]
BD(2)=-[AD(2,1)+2*(Z/BW)*(AD(2,2)-1)]
```

EQUATIONS II

```
AD(1,1)=[COS(OMEGA*DTSPEF)-(ALPHA/OMEGA)*
SIN(OMEGA*DTSPEF)]*TEMP
AD(2,1)=-(1/OMEGA)*SIN(OMEGA*DTSPEF)*TEMP
AD(1,2)=-AD(2,1)*(BW**2)
AD(2,2)=[COS(OMEGA*DTSPEF)+(ALPHA/OMEGA)*
SIN(OMEGA*DTSPEF)]*TEMP
BD(1)=-[AD(1,1)-1+2*(ALPHA/(BW**2))*AD(1,2)]
BD(2)=-[AD(2,1)+2*(ALPHA/(BW**2))*(AD(2,2)-1)]
```

Figure A.6. Flowchart of Subroutine F1AMP and Equations for Subroutine F2AMP

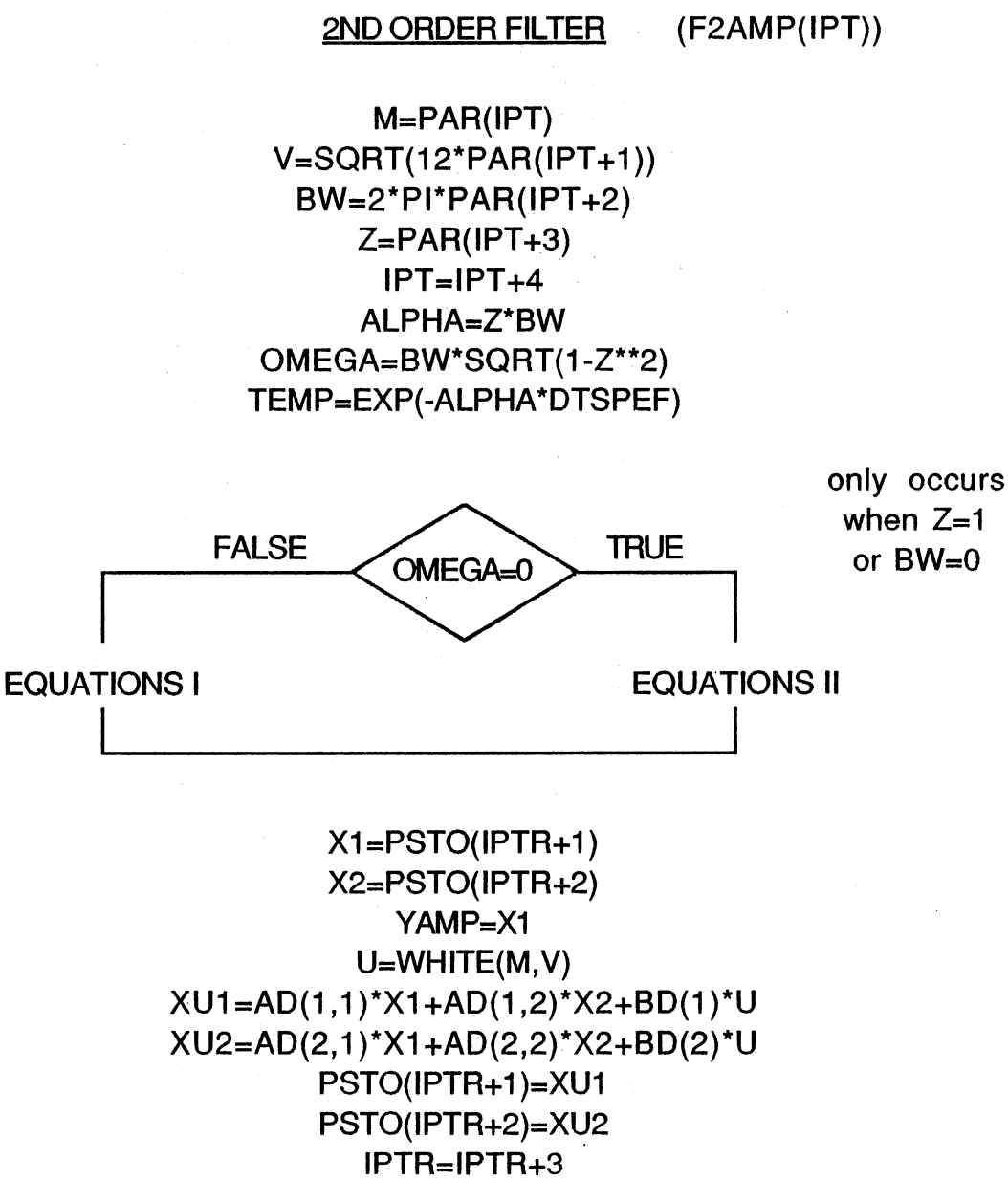


Figure A.7. Flowchart of Subroutine F2AMP

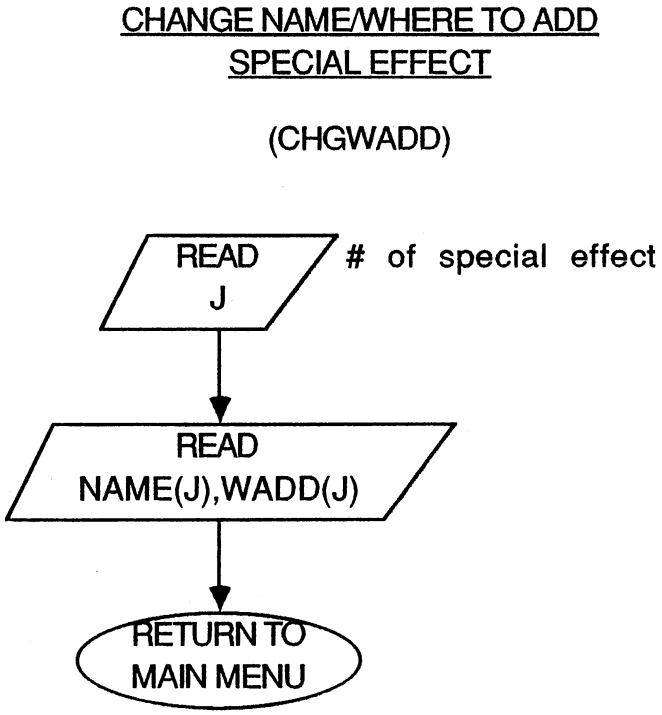


Figure A.8. Flowchart of Subroutine CHGWADD

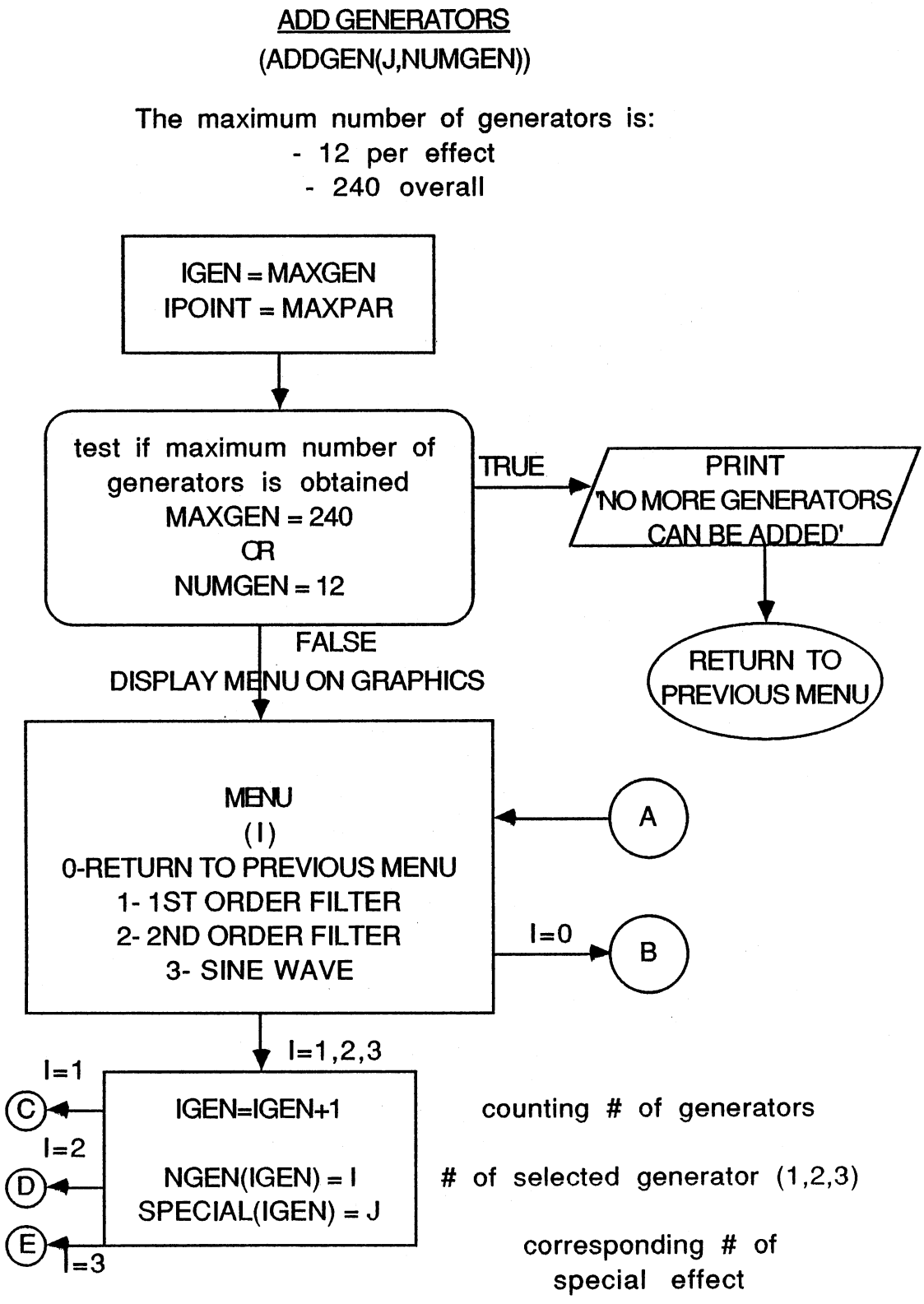


Figure A.9. Flowchart of Subroutine ADDGEN

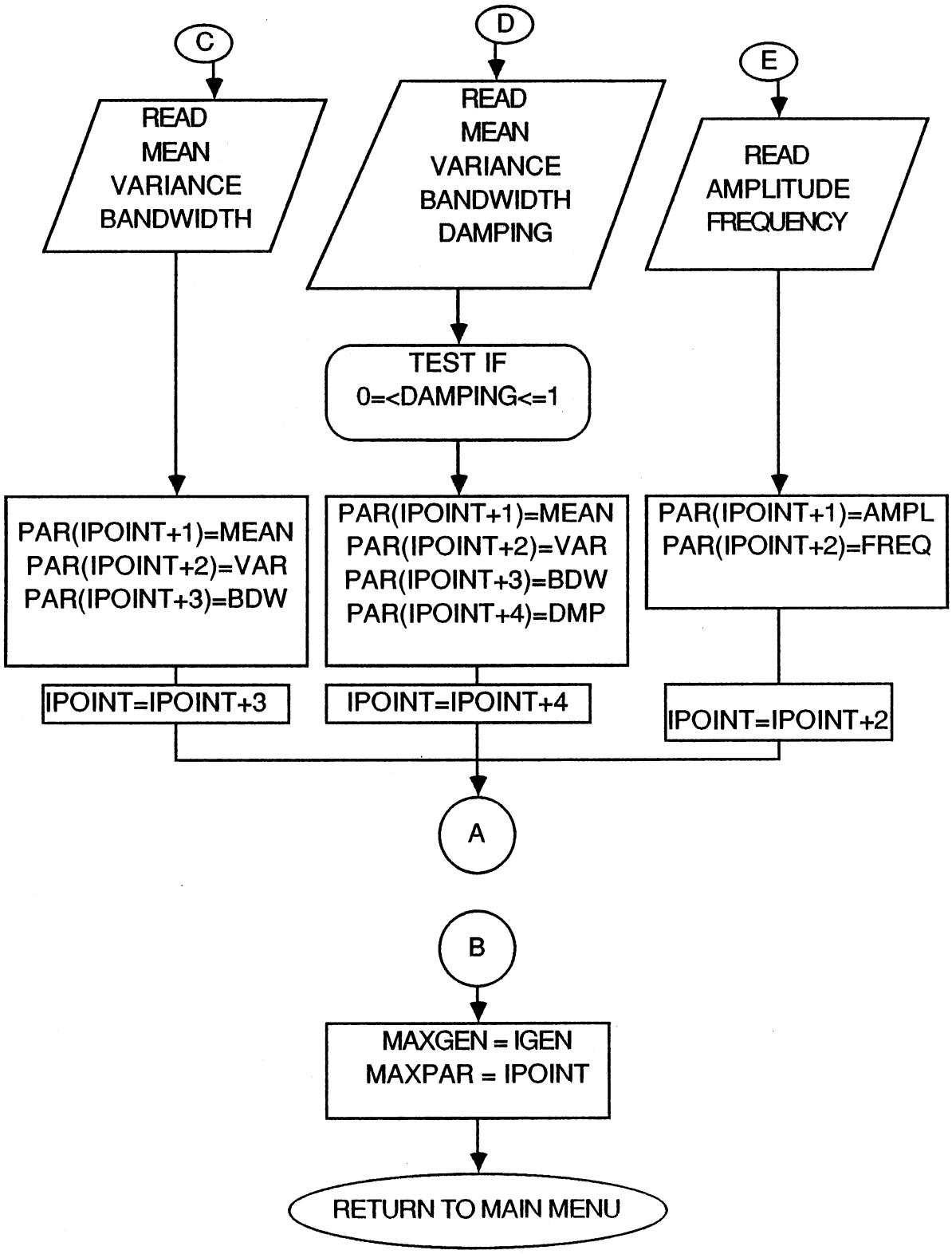


Figure A.9. (Continued)

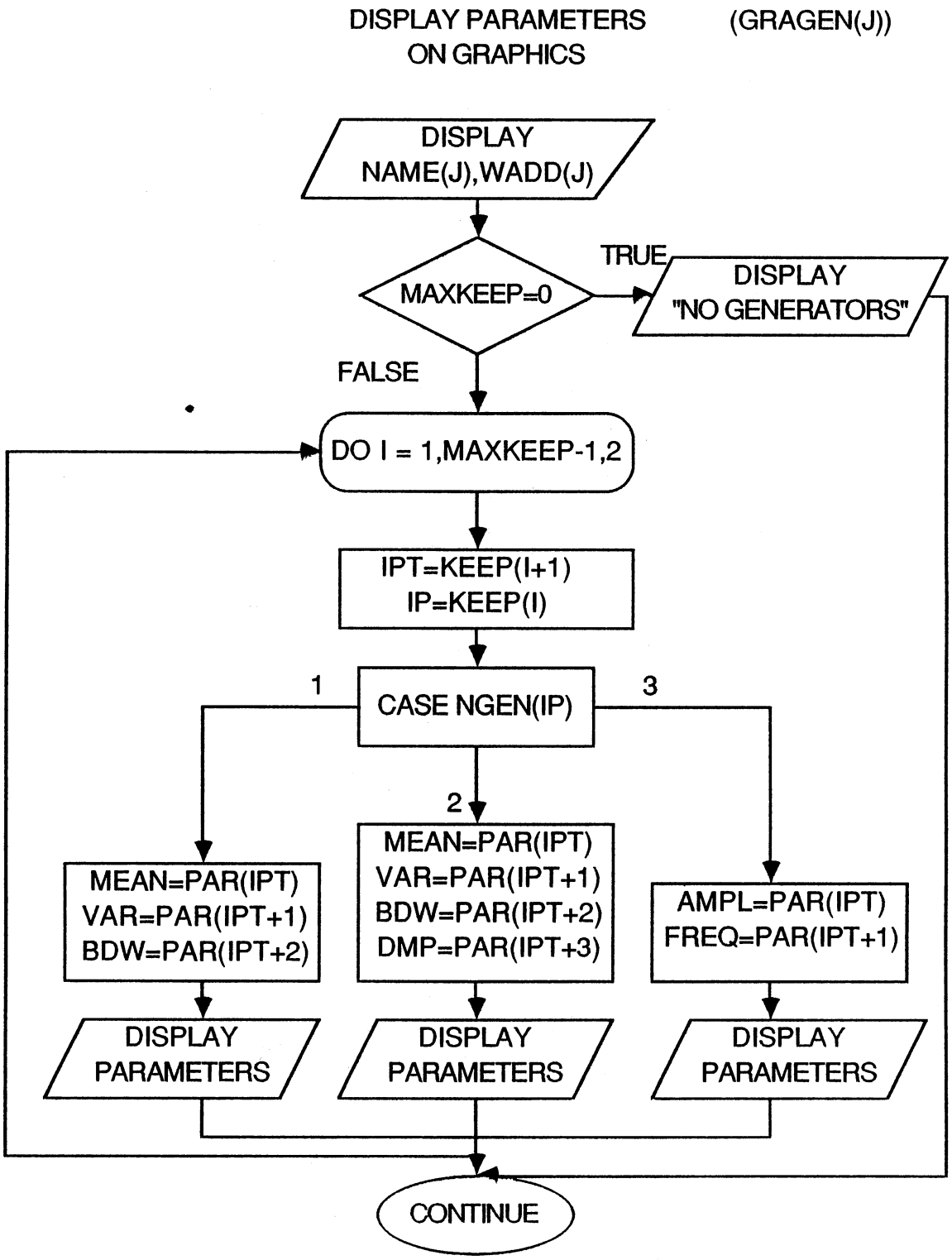


Figure A.10. Flowchart of Subroutine GRAGEN

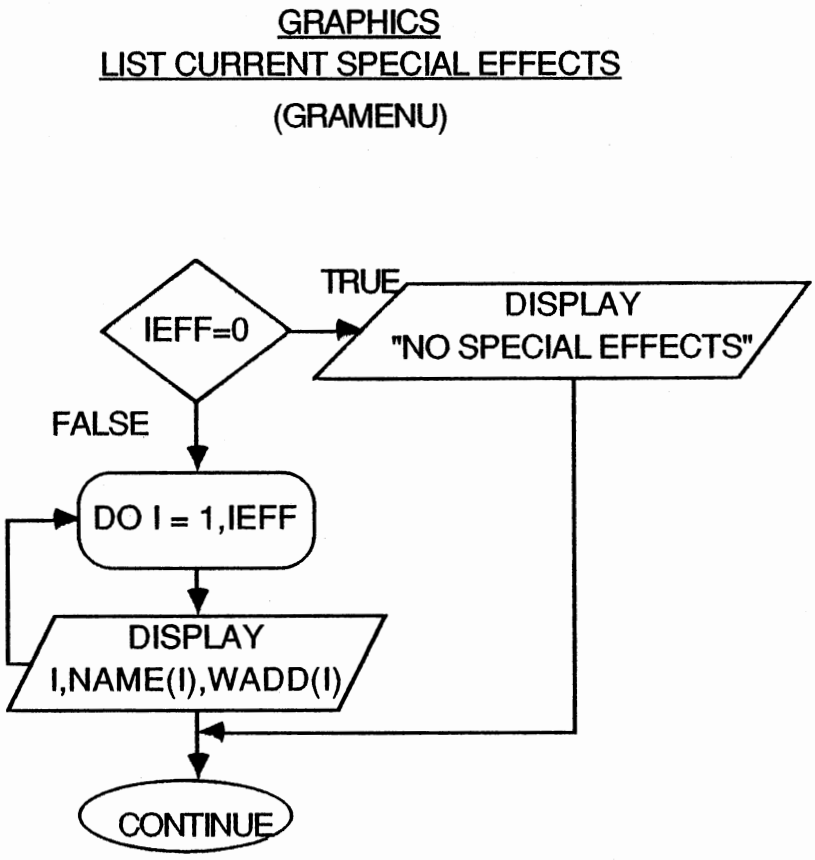


Figure A.11. Flowchart of Subroutine GRAMENU

APPENDIX B

EXAMPLE OF CALCULATION OF CONSTANT C FOR VALVE FLOW

In this appendix an example is given on how to find the constant C of equation 6.1 (Chapter VI). The following equations are given from a two-stage flow control servovalve catalog [14]. The flow is given as a function of pressures:

$$Q = k\, i \sqrt{p_v} \tag{B.1}$$

$$\text{So: } Q_r = k\, i_r \sqrt{p_s} \tag{B.2}$$

where: Q - orifice flow

Q_r - rated flow

i_r - rated current

i - current

p_V - pressure drop across both orifices

$= 2(p_S - p_1)$

p_S - supply pressure

k - proportional constant

Dividing Q by Q_r the constant k is eliminated. Which yields:

$$\frac{Q}{Q_r} = \frac{i}{i_r} \sqrt{\frac{p_v}{p_s}} \tag{B.3}$$

Replacing p_V in the above equation by 2(ps-p1) yields:

$$Q = Q_r \frac{i}{i_r} \sqrt{\frac{2(p_s - p_1)}{p_s}} \tag{B.4}$$

Letting $x = k_m * i$, k_m - proportional constant, yields:

$$Q = \frac{Q_r}{x_r \sqrt{p_s/2}} \sqrt{p_s - p_1}\ x \tag{B.5}$$

Comparing equation B.5 with $Q = C \sqrt{p_s - p_1}\ x$ (equation 6.1) yields:

$$C = \frac{Q_r}{x_r\sqrt{p_s/2}} \qquad \text{(B.6)}$$

Making: p_s = 1000 psi

Q_r = 4 gpm = 15.4 in^3/s

i_r = 10 mA and

if k_m = 0.1 (for artificial position) then x_r = 1 in, since $x_r = k_m * i_r$.

Substituting these values in the equation B.6 the constant C is found to be C = 0.6887 $in^2/s*(psi)^{1/2}$.

The numbers used here do not necessarily correspond to a specific valve, but are used for demonstration purposes only.

APPENDIX C

PROGRAM LISTING

```
**** MAIN PROGRAM

**** PRIMARY MOTION COMMON BLOCKS

      COMMON/CUEFILX/BXACCN,BAXA0,BAXA1,BAXA2,BAXA3
      COMMON/CUEFILY/BYACCN,BAYA0,BAYA1,BAYA2,BAYA3
      COMMON/CUEFILZ/BZACCN,BAZA0,BAZA1,BAZA2,BAZA3
      COMMON/CUEFILQ/BAPITH,BQA0,BQA1,BQA2,BQA3
      COMMON/CUEFILP/BAROLL,BPA0,BPA1,BPA2,BPA3
      COMMON/CUEFILR/BAYAW,BRA0,BRA1,BRA2,BRA3
      COMMON/USER1/BIN,NINP,INPNUM,NITER
      DIMENSION BIN(2000,9),INPNUM(9)
      COMMON/CTTRAN/FLTAXA,FLTAYA,FLTAZA
      COMMON/CTROT/FQADOT,FPADOT,FRADOT,FQA,FPA,FRA
      COMMON/GRAV1/BXALIN,BYALIN
      COMMON/MPCP1/BXPOS,BYPOS,BZPOS,BTHPOS,BPHPOS,BSIPOS
      COMMON/POSCON1/BMXPOS,BMYPOS,BMZPOS,BMTHEP,BMPHIP,BMPSIP
      COMMON/PRIM1/XENFL,YENFL,ZENFL,THENFL,PHENFL,SIENFL
      COMMON/PRIM2/XGRFL,YGRFL,MPCPFL,MPLPFL
      COMMON/PRIM3/BXAWTC,BYAWTC,BZAWTC,BTHWTC,BPHWTC,BSIWTC
      COMMON/PRIM4/BKAX1,BKAY1,BKAZ1,BKQA1,BKPA1,BKRA1
      COMMON/PRIM5/BKAX2,BKAY2,BKAZ2,BKQA2,BKPA2,BKRA2
      COMMON/PRIM6/ALAGX,AKGAX,DLAGX,DKGAX,ALAGY,AKGAY,DLAGY,DKGAY
      COMMON/PRIM7/BKQA4,BKPA4
      COMMON/PRIM8/BLAGX,BLAGY,BLAGZ,BLAGT,BLAGP,BLAGS
      INTEGER*4  XENFL,YENFL,ZENFL,THENFL,PHENFL,SIENFL
      INTEGER*4  XGRFL,YGRFL,MPCPFL,MPLPFL
      REAL   BXAWTC,BYAWTC,BZAWTC,BTHWTC,BPHWTC,BSIWTC
      REAL   BKAX1,BKAY1,BKAZ1,BKQA1,BKPA1,BKRA1
      REAL BKAX2,BKAY2,BKAZ2,BKQA2,BKPA2,BKRA2
      REAL   ALAGX,AKGAX,DLAGX,DKGAX,ALAGY,AKGAY,DLAGY,DKGAY
      REAL   BKQA4,BKPA4
      REAL   BLAGX,BLAGY,BLAGZ,BLAGT,BLAGP,BLAGS

**** SPECIAL EFFECTS COMMON BLOCKS

      COMMON/SMSPE1/MAXGEN,MAXPAR,SPECIAL
      COMMON/SMSPE2/IEFF,NAME,WADD
      COMMON/SMSPE3/NGEN,PAR,IPOINT
      COMMON/SMSPE4/MAXPSTO,IPTR,PSTO
      COMMON/SMSPE5/KEEP,MAXKEEP
      COMMON/SMSPE7/POSX,IEFFLG

**** VALVE COMMON BLOCKS

      COMMON/SMVAL1/DAMPM,BANDM,GAINM,XM1,XM2,XK,B,VRST
      COMMON/SMVAL2/PS,PR
      COMMON/SMVAL3/XL1,XL2,XL3,APT,VO,XLIMIT,FRC,BETAO
      COMMON/SMVAL4/X,XDOT,PD,PDOT,XMOTOR,PISTON
      COMMON/SMVAL5/VOUT
      COMMON/SMVAL6/CPRS,RPRS,POSI,Q6,Q7
      COMMON/SMVAL7/Q1,Q2,Q3,Q4,Q5,V
```

```
**** SERVO COMMON BLOCKS

      COMMON/SMSRV1/XKPM,XKP,SCLOT,XKFORC,DMPOFF,DMPON
      COMMON/SMSRV2/COMP1,COMP2,COMP3,COMP4,COMP5
      COMMON/SMSRV3/TEMP1,TEMP2,CYLOT,FORCE
      COMMON/SMSRV4/COMPN

**** SAVE COMMON BLOCKS

      COMMON/SMDIS1/TMAX,NVS,NPRNT,NVAR,PLOTV,P                    LOTA
      COMMON/SMDIS2/TINVX,AFTER

**** ANIMATE COMMON BLOCKS

      COMMON/SMANI1/SIGNA

      DIMENSION NGEN(240),PAR(1000),KEEP(1000),PSTO(1000)
      DIMENSION POSX(6),IEFFLG(20)
      INTEGER WADD(20),SPECIAL(240)
      CHARACTER NAME(20)*20
      DIMENSION X(6,2),XDOT(6,2),PD(6,4),PDOT(6,4)
      DIMENSION XMOTOR(4,5),PISTON(4,5)
      DIMENSION XM2(6),XK(6),B(6)
      DIMENSION VOUT(6)
      DIMENSION CPRS(6),RPRS(6),POSI(6),Q6(6),Q7(6)
      DIMENSION Q1(6),Q2(6),V(6,2),Q3(6),Q4(6),Q5(6)
      DIMENSION XKPM(6),XKP(6),SCLOT(6),XKFORC(6)
      DIMENSION COMP1(6),COMP2(6),COMP3(6),COMP4(6),COMP5(6)
      DIMENSION TEMP1(6),TEMP2(6),CYLOT(6),FORCE(6)
      DIMENSION NVAR(17),PLOTV(2000,18),PLOTA(2000,6)
      DIMENSION SIGNA(6),COMPN(6)
      DIMENSION TINVX(6),AFTER(6)
      CHARACTER CH*1

      WHILE (1>0)
       WRITE(9,*)
       WRITE(9,*)
       WRITE(9,*)
       WRITE(9,'("          MAIN MENU          ")')
       WRITE(9,*)
       WRITE(9,*)
                                          WRITE(9,"  0.END          ")
CC     WRITE(9,'("    1. PRIMARY MOTION (NOT AVAILABLE) ")')
       WRITE(9,'("    1. SPECIAL EFFECTS          ")')
       WRITE(9,'("    2. SET UP SERVO             ")')
       WRITE(9,'("    3. SET UP VALVE             ")')
       WRITE(9,'("    4. SIMULATION               ")')
       WRITE(9,'("    5. READ FILE                ")')
       WRITE(9,'("    6. SAVE FILE                ")')
       WRITE(9,*)
       WRITE(9,*)
       WRITE(9,'("      ENTER YOUR SELECTION      ")')
       WRITE(9,*)
```

```
      WRITE(9,*)
      ICH = 1000
      READ(9,*) ICH
      SELECT CASE (ICH)
       CASE (0)
        EXIT
CCC     CASE (1)
CCC       CH = '_'
CCC       WRITE(9,'("YOU HAVE SELECTED: "PRIMARY MOTION"")')
CCC       WRITE(9,'(" <CR> TO RETURN - ANYTHING TO CONTINUE")')
CCC       READ(9,*) CH
CCC       IF (CH .NE. '_') CALL WASHOUT2
       CASE (1)
        CALL SEMMENU
                          CASE(2)
        CALL SERVO
       CASE (3)
        CALL VALVE
       CASE (4)
        CALL TSIMUL
       CASE (5)
        CALL FILERD
       CASE (6)
        CALL FILESV
       CASE DEFAULT
        WRITE(9,'("   ILLEGAL CHOICE - SELECT AGAIN")')
      END SELECT
     REPEAT
     STOP
     END

*************************************************
******* SUBROUTINE FOR BASE MOVEMENT ANIMATION
*************************************************
     SUBROUTINE ANIMATE

     COMMON/SMDIS1/TMAX,NVS,NPNT,NVAR,PLOTV,PLOTA

     DIMENSION NVAR(17),PLOTV(2000,18),PLOTA(2000,6)

     WHILE (1>0)
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("          BASE ANIMATION MENU     ")')
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("       0. RETURN TO PREVIOUS MENU     ")')
      WRITE(9,'("       1. ANGLE VIEW OF BASE          ")')
      WRITE(9,'("       2. TOP VIEW OF BASE           ")')
                                  WRITE(9,'("    3.SIDE VIEW OF BASE     ")')
      WRITE(9,'("       4. FRONT VIEW OF BASE          ")')
      WRITE(9,*)
      WRITE(9,*)
```

```
      WRITE(9,'("       ENTER YOUR SELECTION       ")')
      WRITE(9,*)
      WRITE(9,*)

****** READ SELECTION

      ICH = 1000
      READ(9,*) ICH
      SELECT CASE (ICH)
        CASE (0)
         RETURN
        CASE (1)
         IXVW=-15 ; IYVW=15 ; I1=-40 ; I2=0 ; I3=0
        CASE (2)
         IXVW=-90 ; IYVW=0  ; I1=-40 ; I2=0 ; I3=120
        CASE (3)
         IXVW=0   ; IYVW=90 ; I1=0   ; I2=0 ; I3=135
        CASE (4)
         IXVW=0   ; IYVW=0  ; I1=-40 ; I2=0 ; I3=0
        CASE DEFAULT
         WRITE(9,'("ILLEGAL CHOICE")')
         WRITE(9,'("MAKE ANOTHER SELECTION")')
      END SELECT

**** CONVERTION FACTOR TO DRAWING COORDINATES

      CONDRW = 3.033

**** INITIALIZE GRAPHICS SCREEN

      CALL PLTINIT
 CALLP                          ROJCT(0)
      CALL VWIDEN

**** VIEWER REFERENCE POINT

      CALL VWRPT(I1,I2,I3)

**** VIEWER ROTATION

      CALL VWROTX(IXVW)
      CALL VWROTY(IYVW)

**** DRAWS LOWER BASE
**** FIRST POINT IN LOWER BASE - FRONT LEFT HAND VERTIX

      CALL MOVE3(70,15,0)
      CALL COLOR(131)

**** DRAW THE THICKNESS

      DO (J = 1,20)
       CALL POLYR3(6,0,0,0,348,0,0,358,0,-17,184,0,-319,164,0,-319,
```

```
+            -10,0,-17)
     CALL MDTRAN(0,1,0)
     REPEAT

     CALL PRMFIL(1)
     CALL COLOR(45)
     CALL POLYR3(6,0,0,0,348,0,0,358,0,-17,184,0,-319,164,0,-319,
+          -10,0,-17)
     CALL PRMFIL(0)

     DO (II=1,NPNT)

***  CONVERSION OF X,Y,Z TO DRAWING COORDINATES

     IX=NINT(PLOTA(II,2)*CONDRW)
     IY=NINT(-PLOTA(II,3)*CONDRW)+200
     IZ=NINT(PLOTA(II,1)*CONDRW)

     IAX=NINT((180./3.1416)*(-PLOTA(II,4)))
     IAY=NINT((180./3.1416)*(-PLOTA(II,6)))
  IAZ=NINT((180./3                                    .1416)*(PLOTA(II,5)))

**** DRAW UPPER BASE AND ERASE IT

     CALL MDIDEN

**** (244,35,-112) CENTER OF LOWER BASE
**** (IXO,IYO,IZO) CENTER OF UPPER BASE AFTER MOVEMENT

     IXO=IX+244
     IYO=IY+35
     IZO=IZ-112
     CALL MDORG(IXO,IYO,IZO)

**** ROTATE WRT ORIGIN IN UPPER BASE

     CALL MDROTX(IAX)
     CALL MDROTY(IAY)
     CALL MDROTZ(IAZ)
     CALL MOVE3(244,35,-112)      ! CENTER OF LOWER BASE
     CALL MOVER3(IX,IY,IZ)        ! CENTER OF UPPER BASE

**** FIRST POINT IN LOWER BASE - FRONT RIGHT HAND VERTIX

     CALL MOVER3(10,0,156)
     CALL COLOR(210)
     CALL POLYR3(6,0,0,0,130,0,-226,120,0,-242,-140,0,-242,
+          -150,0,-226,-20,0,0)
     CALL MOVE3(244,35,-112)
     CALL MOVER3(IX,IY,IZ)
     CALL MOVER3(10,0,156)
     CALL COLOR(0)
     CALL POLYR3(6,0,0,0,130,0,-226,120,0,-242,-140,0,-242,
```

```
+              -150,0,-226,-20,0,0)
      CALL PRMFIL(0)
                        REPEAT
    REPEAT
    END

************************************************
******* SUBROUTINE TO CHANGE PARAMETERS
************************************************
   SUBROUTINE CHGPAR

**** DISPLAY MENU FOR CHANGE IN PARAMETERS

    WHILE (1>0)
     WRITE(9,*)
     WRITE(9,*)
     WRITE(9,'("        CHANGE/VIEW PARAMETERS MENU   ")')
     WRITE(9,*)
     WRITE(9,*)
     WRITE(9,'("       0. RETURN TO PREVIOUS MENU     ")')
CCC   WRITE(9,'("        1. PRIMARY MOTION (NOT AVAILABLE)")')
     WRITE(9,'("       1. SPECIAL EFFECTS            ")')
     WRITE(9,'("       2. SERVO                      ")')
     WRITE(9,'("       3. VALVE                      ")')
     WRITE(9,*)
     WRITE(9,*)
     WRITE(9,'("       ENTER YOUR SELECTION       ")')
     WRITE(9,*)
     WRITE(9,*)
**** READ SELECTION
     ICH = 1000
     READ(9,*) ICH
     SELECT CASE (ICH)
                          CASE(0)
       EXIT
CCC      CASE (1)
CCC       CALL CHGPM
       CASE (1)
       CALL GRAMENU     !GRAPHICS-MENU OF CURRENT SPECIAL EFF.
        CALL CHGSE
       CASE (2)
        CALL CHGSRV
       CASE (3)
        CALL CHGVAL
       CASE DEFAULT
        WRITE(9,'("ILLEGAL CHOICE")')
        WRITE(9,'("MAKE ANOTHER SELECTION")')
     END SELECT
    REPEAT
    RETURN
    END

************************************************
```

```
******** SUBROUTINE TO CHANGE SERVO PARAMETERS
************************************************
      SUBROUTINE CHGSRV

      COMMON/SMSRV1/XKPM,XKP,SCLOT,XKFORC,DMPOFF,DMPON
      COMMON/SMSRV2/COMP1,COMP2,COMP3,COMP4,COMP5

      DIMENSION XKPM(6),XKP(6),SCLOT(6),XKFORC(6)
      DIMENSION COMP1(6),COMP2(6),COMP3(6),COMP4(6),COMP5(6)
      CHARACTER CH*1

      WRITE(9,'("YOU HAVE SELECTED: "CHANGE SERVO PARAMETERS"")')
      WRITE(9,'("<CR> TO RETURN - ANYTHING TO CONTINUE")')
                        CH='_'
      READ(9,*) CH
      IF (CH .EQ. '_') RETURN
      WRITE(9,*)
      WRITE(9,'("ENTER SERVO PARAMETERS - <CR> FOR NO CHANGE")')
      WRITE(9,*)
      WRITE(9,'("COMMANDED POSITION GAIN: ")')
      WRITE(9,'(6E12.5)') (XKPM(I),I=1,6)
      READ(9,*) (XKPM(I),I=1,6)
      WRITE(9,*)
      WRITE(9,'("FEEDBACK POSITION GAIN: ")')
      WRITE(9,'(6E12.5)') (XKP(I),I=1,6)
      READ(9,*) (XKP(I),I=1,6)
      WRITE(9,*)
      WRITE(9,'("SCALE FOR VALVE VOLTAGE (mV/in):")')
      WRITE(9,'(6E12.5)') (SCLOT(I),I=1,6)
      READ(9,*) (SCLOT(I),I=1,6)
      WRITE(9,*)
      WRITE(9,'("DIFFERENTIAL PRESSURE GAIN: ")')
      WRITE(9,'(6E12.5)') (XKFORC(I),I=1,6)
      READ(9,*) (XKFORC(I),I=1,6)
      WRITE(9,*)
      WRITE(9,'("COMPENSATION GAINS #1: ")')
      WRITE(9,'(6E12.5)') (COMP1(I),I=1,6)
      READ(9,*) (COMP1(I),I=1,6)
      WRITE(9,'("COMPENSATION GAINS #2: ")')
      WRITE(9,'(6E12.5)') (COMP2(I),I=1,6)
      READ(9,*) (COMP2(I),I=1,6)
      WRITE(9,'("                              MPENSATION GAINS #3:")')
      WRITE(9,'(6E12.5)') (COMP3(I),I=1,6)
      READ(9,*) (COMP3(I),I=1,6)
      WRITE(9,'("COMPENSATION GAINS #4: ")')
      WRITE(9,'(6E12.5)') (COMP4(I),I=1,6)
      READ(9,*) (COMP4(I),I=1,6)
      WRITE(9,'("COMPENSATION GAINS #5: ")')
      WRITE(9,'(6E12.5)') (COMP5(I),I=1,6)
      READ(9,*) (COMP5(I),I=1,6)
      WRITE(9,'("2 DAMPING FADE LIMITS (HIGHER FIRST)")')
      WRITE(9,'(2E12.5)') DMPON,DMPOFF
      READ(9,*) DMPON,DMPOFF
```

```
      RETURN
      END

*************************************************
******** SUBROUTINE TO CHANGE VALVE PARAMETERS
*************************************************
      SUBROUTINE CHGVAL

      COMMON/SMVAL1/DAMPM,BANDM,GAINM,XM1,XM2,XK,B,VRST
      COMMON/SMVAL2/PS,PR
      COMMON/SMVAL3/XL1,XL2,XL3,APT,VO,XLIMIT,FRC,BETAO

      DIMENSION XM2(6),XK(6),B(6)
      CHARACTER CH*1

      WRITE(9,'("YOU HAVE SELECTED: "CHANGE VALVE PARAMETERS"")')
     WRITE(9,'(<CR>TORETURN-ANYTHINGTOCONTIN                    UE')
      CH = '_'
      READ(9,*) CH
      IF (CH .EQ. '_') RETURN
      WRITE(9,'("ENTER VALVE PARAMETERS - <CR> FOR NO CHANGE")')
      WRITE(9,'("MOTOR SPECIFICATIONS")')
      WRITE(9,'("DAMPING: ",E11.5)') DAMPM
      READ(9,*) DAMPM
      WHILE ((DAMPM .LT. 0) .OR. (DAMPM .GT. 1.0))
       WRITE(9,*)
       WRITE(9,'("ILLEGAL DAMPING RATIO")')
       READ(9,*) DAMPM
      REPEAT
      WRITE(9,'("BANDWIDTH (Hz): ",E11.5)') BANDM
      READ(9,*) BANDM
      WRITE(9,'("GAIN (in/mA): ",E11.5)') GAINM
      READ(9,*) GAINM
      WRITE(9,'("RESISTANCE (ohms): ",E11.5)') VRST
      READ(9,*) VRST
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("VALVE SPECIFICATIONS")')
      WRITE(9,'("SUPPLY PRESSURE (psi): ",E11.5)') PS
      READ(9,*) PS
      WRITE(9,'("RETURN PRESSURE (psi): ",E11.5)') PR
      READ(9,*) PR
      WRITE(9,'("LEAKAGE FLOW CONSTANT (in3/(s*psi)): ")')
      WRITE(9,'(3E15.5)') XL1,XL2,XL3
      READ(9,*) XL1,XL2,XL3
     WRITE(9,'(AR                                    EACFPISTON(in2):',E11.5)) APT
      READ(9,*) APT
      WRITE(9,'("TOTAL EFFECTIVE VOLUME (in3): ",E11.5)') VO
      READ(9,*) VO
      WRITE(9,'("LIMIT OF SPOOL POSITION (in): ",E11.5)') XLIMIT
      READ(9,*) XLIMIT
      WRITE(9,'("FLOW RATE CONSTANT: ",E11.5)') FRC
      READ(9,*) FRC
```

```
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("OIL SPECIFICATIONS")')
      WRITE(9,'("BULK MODULUS (psi): ",E11.5)') BETAO
      READ(9,*) BETAO
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("MASS OF CYLINDER (lb): ",E11.5)') XM1
      READ(9,*) XM1
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("MODEL OF MOTION BASE")')
      WRITE(9,'("MASS OF BASE FOR EACH CYLINDER (lb): ")')
      WRITE(9,'(6E13.5)') (XM2(I),I=1,6)
      READ(9,*) (XM2(I),I=1,6)
      WRITE(9,'("SPRING CONSTANT FOR EACH CYLINDER (lb/in): ")')
      WRITE(9,'(6E13.5)') (XK(I),I=1,6)
      READ(9,*) (XK(I),I=1,6)
      WRITE(9,'("DAMPING COEFF. FOR EACH CYLINDER(lb/(in/s)): ")')
     WRITE(9,'(6E13.5)') (B(I),I=1,6)
      READ(9,*) (B(I),I=1,6)
      RETURN
      END

*************************************************
******** SUBROUTINE TO DISPLAY RESULTS
********   PLOT SELECTED VARIABLES
*************************************************
      SUBROUTINE DISRES

      COMMON/SMDIS1/TMAX,NVS,NPNT,NVAR,PLOTV,PLOTA

      DIMENSION NVAR(17),PLOT1(2000,9),PLOTV(2000,18),PLOTA(2000,6)
      DIMENSION IFLAG(8),VSCL(20)
      CHARACTER VNAME(20)*30,VSYMBOL(20)*6
      CHARACTER PNAME(8)*6,PSYMBOL(8)*1,XLABEL*4,YLABEL*6,HEADER*20

**** READ NAMES OF VARIABLES TO BE PLOTTED FROM FILE TABLE
      OPEN(UNIT=17,FILE='TABLE.SA')
      READ(17,*)
      READ(17,*)
      I = 1
      WHILE (I .LE. NVS)
       READ(17,'(I4,T10,A6,T20,A25,T50,F7.3)',END=10)
     +          IA,VSYMBOL(I),VNAME(I),VSCL(I)
       IF (IA .EQ. NVAR(I)) I = I + 1
      REPEAT
10    CONTINUE
      CLOSE (17)

      WHILE (1>0)
       WRITE(9,*)
       WRITE(9,*)
```

```
      WRITE(9,*)
      WRITE(9,                              ')
      WRITE(9,'("     DISPLAY RESULTS")')

*** ZERO FLAGS TO IDENTIFY VARIABLES TO BE PLOTTED

      DO (I = 1,8)
       IFLAG(I) = 0
      REPEAT

      WRITE(9,*)
      WRITE(9,'("  0. RETURN TO PREVIOUS MENU",T36,
     +          "(DEFAULT)")')
      WRITE(9,'("  1. TIME",T36,"(X AXIS)")')
      DO (J = 1,NVS)
        J1=J+1
        WRITE(9,'(I3,". ",A30,T36,"(",A6,")")')
     +            J1,VNAME(J),VSYMBOL(J)
      REPEAT
      WRITE(9,*)
      WRITE(9,'("ENTER YOUR CHOICES (MAX 8 VARIABLES):")')
      READ(9,*) (IFLAG(I),I=1,8)

      IF (IFLAG(1) .EQ. 0) RETURN

**** CHOOSE TIME INTERVAL

      WRITE(9,*)
      WRITE(9,'("ENTER TIME INTERVAL")')
      WRITE(9,'("    DEF = TOTAL TIME OF SIMULATION")')

**** TBEGIN: TIME OF BEGINNING
**** TEND: TIME OF END OF INTERVAL

      TBEGIN = 0
      TEND = TMAX
      READ(9,*) TBEGIN,TEND

**** NO INTERVALS OUT OF RANGE

      WHILE((TB                                   EGIN.LT.0).OR.(TEND.GT.TMAX)
     +                  .OR. (TBEGIN .GE. TEND))
        WRITE(9,'("INTERVAL OUT OF RANGE")')
        WRITE(9,'("ENTER TIME INTERVAL AGAIN")')
        TBEGIN = 0
        TEND = TMAX
        READ(9,*) TBEGIN,TEND
      REPEAT

**** CALCULATE - NPBEGIN: NUMBER OF POINT TO BEGIN AT
****             NPEND: NUMBER OF POINT TO END AT

      IF ((TBEGIN .EQ. 0) .AND. (TEND .EQ. TMAX)) THEN
```

```
      NPBEGIN = 1
      NPEND = NPNT
    ELSE
      NPBEGIN = NINT(TBEGIN*NPNT/TMAX) + 1
      NPEND = NINT(TEND*NPNT/TMAX)
    ENDIF

**** NPOINT: NUMBER OF POINTS IN THE INTERVAL

    NPOINT = NPEND - NPBEGIN + 1

    NVPLT = 0            !COUNTER OF VARIABLES TO PLOT
    DO (I=1,8)
     IF (IFLAG(I) .EQ. 0) EXIT
     NVPLT = NVPLT + 1
     TEMP = 1.

**** SET VARIABLE NAMES TO PLOT, PRINT OR DISPLAY

     IF ((IFLAG(I) .NE. 1) .OR. (I .NE. 1)) THEN
  PNAME                                  (I-1)=VSYMBOL(IFLAG(I)-1)
      PSYMBOL(I-1) = CHAR(I-1+ICHAR('0'))
      TEMP = VSCL(IFLAG(I)-1)
     ENDIF

**** SET NEW ARRAY TO PLOT

     IK = 1
     DO (K=NPBEGIN,NPEND)
       PLOT1(IK,I) = TEMP*PLOTV(K,IFLAG(I))
       IK = IK + 1
     REPEAT
    REPEAT

**** CHOOSE TO PLOT, PRINT OR DISPLAY DATA

    WRITE(9,*)
    WRITE(9,*)
    WRITE(9,*)
    WRITE(9,*)
    WRITE(9,*)
    WRITE(9,'(" 1. PLOT THE DATA (DEFAULT)")')
    WRITE(9,'(" 2. DISPLAY DATA ON SCREEN ")')
    WRITE(9,'(" 3. PRINT DATA")')
    WRITE(9,*)
    WRITE(9,'(" ENTER YOUR SELECTION ")')
    ICH = 1
    READ(9,*) ICH
    SELECT CASE (ICH)
     CASE (1)
       YLABEL = 'OUTPUT'
       XLABEL = 'TIME'
       CALL NAMEPLOT(HEADER)
```

```
      CALL PLOTSIM(PLOT1,NVPLT-1,NPOINT,HEADER,XLABEL,YLABEL,
+                  PNAME,PSYMBOL)
     CASE (2)
  IF(NVPLT.                              LE.5)THEN
       IK = 1
       ELSE
       IK = 2
       ENDIF
       DO (IJ=1,IK)
       IJL = IJ*5 ; IJF = IJL - 4
       IF (NVPLT .LT. IJL) IJL = NVPLT
       IF (IFLAG(1) .EQ. 1) THEN
       IJF = IJ*(IJ+1)
       WRITE(9,'("     TIME  ",4A13)')
+                        (PNAME(I),I=IJF-1,IJL-1)
       DO (K=1,NPOINT)
        WRITE(9,'(5E13.5)') PLOT1(K,1),(PLOT1(K,I),I=IJF,IJL)
       REPEAT
       ELSE
       WRITE(9,'(5A13)') (PNAME(I),I=IJF,IJL)
       DO (K=1,NPOINT)
        WRITE(9,'(5E13.5)') (PLOT1(K,I),I=IJF,IJL)
       REPEAT
       ENDIF
       WRITE(9,*)
       WRITE(9,*)
       WRITE(9,*)
      REPEAT
     CASE (3)
      N0 = 18
      OPEN(N0,FILE='CN04:',STATUS='NEW')
      WRITE(N0,'(A1)') CHAR(12)       !FORM FEED
      IF (IFLAG(1) .EQ. 1) THEN
  WRITE(N0,(1X                                    ,' TIME ',7A13))
+                          (PNAME(I),I=1,NVPLT-1)
      ELSE
       WRITE(N0,'(1X,8A13)') (PNAME(I),I=1,NVPLT)
      ENDIF
      DO (K=1,NPOINT)
       WRITE(N0,'(1X,8E13.5)') (PLOT1(K,I),I=1,NVPLT)
      REPEAT
      CLOSE (N0)
     CASE DEFAULT
      CONTINUE
   END SELECT

  REPEAT
  RETURN
  END

**************************************************
******* SUBROUTINE TO INTEGRATE USING EULER'S METHOD
*******
```

```
******* YYDOT = A*YY + B*U
******* YY = YY + YYDOT*XDT
*******
******* YY -> VARIABLE TO INTEGRATE
******* YYDOT -> DERIVATIVE
******* NORDER -> NUMBER OF STATES
******* AB -> MATRIX A/B
******* XDT -> SAMPLING INTERVAL
******* XINPUT -> INPUT U
******* II -> THE NUMBER CORRESPONDING TO THE CYLINDER
************************************************
   SUBROUTINE EULER(YY,YYDOT,NORDER,AB,XDT,XINPUT,II)

 DIMENSION YY(64),YYDOT(6                                    ),AB(45)

   DO (J=1,NORDER)
     TEMP = 0
     DO (K=1,NORDER)
       TEMP = TEMP+AB(J,K)*YY(II,K)
     REPEAT
     K = NORDER+1
     YYDOT(II,J) = TEMP+AB(J,K)*XINPUT
   REPEAT
   DO (J=1,NORDER)
     YY(II,J) = YY(II,J) + XDT*YYDOT(II,J)
   REPEAT
   RETURN
   END

************************************************
******* SUBROUTINE TO READ DATA FROM FILE
************************************************
   SUBROUTINE FILERD

   COMMON/SMSPE1/MAXGEN,MAXPAR,SPECIAL
   COMMON/SMSPE2/IEFF,NAME,WADD
   COMMON/SMSPE3/NGEN,PAR,IPOINT
   COMMON/SMSPE4/MAXPSTO,IPTR,PSTO
   COMMON/SMVAL1/DAMPM,BANDM,GAINM,XM1,XM2,XK,B,VRST
   COMMON/SMVAL2/PS,PR
   COMMON/SMVAL3/XL1,XL2,XL3,APT,VO,XLIMIT,FRC,BETAO
   COMMON/SMSRV1/XKPM,XKP,SCLOT,XKFORC,DMPOFF,DMPON
   COMMON/SMSRV2/COMP1,COMP2,COMP3,COMP4,COMP5

   DIMENSION NGEN(240),PAR(1000),PSTO(1000)
   INTEGER WADD(20),SPECIAL(240)
 CHARACTER*NAME(20)*                          20
   CHARACTER CH*1,FNAME*20
   DIMENSION XM2(6),XK(6),B(6)
   DIMENSION XKPM(6),XKP(6),SCLOT(6),XKFORC(6)
   DIMENSION COMP1(6),COMP2(6),COMP3(6),COMP4(6),COMP5(6)
   LOGICAL EX
```

```
      EX = .FALSE.

      FNAME = '_'
      WRITE(9,'("YOU HAVE SELECTED: "READ FILES"")')
      WRITE(9,'(" <CR> TO RETURN TO MAIN MENU ",
     +       " - ANYTHING TO CONTINUE")')
      READ(9,*) FNAME
      IF (FNAME(1:1) .EQ. '_') RETURN

***** READS INPUT FILE NAME - PRIMARY MOTION

      CALL READPRIM

      EX = .FALSE.

***** READS INPUT FILE NAME - SPECIAL EFFECTS

      FNAME = 'SPEDATA.SA'
      WRITE(9,'("WHAT IS SPECIAL EFFECT DATA FILE NAME?")')
      READ(9,*) FNAME

***** CHECK TO SEE IF FILE EXISTS

      INQUIRE(FILE=FNAME,EXIST=EX,IOSTAT=IOS,ERR=999)
      IF(.NOT.EX) THEN
        WRITE(9,'("DATA FILE ",(A)," DOES NOT EXIST")') FNAME
        RETURN
      ENDIF

***** OPEN FILE

      OPEN(UNIT=15,FILE=FNAME)

*** READ DATA FILE
***RE                              ADSPECIALEFFECTSDATA

      READ(15,*) IEFF
      IF (IEFF .GT. 0) THEN
        K = INT(IEFF/4.)
        IMOD = MOD(IEFF,4)
        IF (IMOD .NE. 0) K = K + 1
        DO (IJ = 1,K)
          IJL = IJ*4
          IJF = IJL - 3
          IF (IEFF .LT. IJL) IJL = IEFF
          READ(15,*) (NAME(J),J=IJF,IJL)
        REPEAT
        READ(15,*) (WADD(J),J=1,IEFF)
      ENDIF
      READ(15,*) MAXGEN
      IF (MAXGEN .GT. 0) THEN
        K = INT(MAXGEN/33.)
        IMOD = MOD(MAXGEN,33)
```

```
      IF (IMOD .NE. 0) K = K + 1
      DO (IJ = 1,K)
        IJL = IJ*33
        IJF = IJL - 32
        IF (MAXGEN .LT. IJL) IJL = MAXGEN
        READ(15,*) (SPECIAL(J),J=IJF,IJL)
        READ(15,*) (NGEN(J),J=IJF,IJL)
      REPEAT
    ENDIF
    READ(15,*) MAXPAR
    IF (MAXPAR .GT. 0) THEN
      K = INT(MAXPAR/11.)
      IMOD = MOD(MAXPAR,11)
      IF (IMOD .NE. 0) K = K + 1
      DO (IJ = 1,K)
        IJL = IJ*11
                                  IJF=IJL-10
        IF (MAXPAR .LT. IJL) IJL = MAXPAR
        READ(15,*) (PAR(J),J=IJF,IJL)
      REPEAT
    ENDIF
    READ(15,*) MAXPSTO
    IF (MAXPSTO .GT. 0) THEN
      K = INT(MAXPSTO/11.)
      IMOD = MOD(MAXPSTO,11)
      IF (IMOD .NE. 0) K = K + 1
      DO (IJ = 1,K)
        IJL = IJ*11
        IJF = IJL - 10
        IF (MAXPSTO .LT. IJL) IJL = MAXPSTO
      READ(15,*) (PSTO(J),J=IJF,IJL)
      REPEAT
    ENDIF

    CLOSE(15)

**** READ SERVO DATA

    EX = .FALSE.

**** READS INPUT FILE NAME - SERVO

    FNAME = 'SERVOPAR.SA'
    WRITE(9,'("WHAT IS SERVO DATA FILE NAME?")')
    READ(9,*) FNAME

**** CHECK TO SEE IF FILE EXISTS

    INQUIRE(FILE=FNAME,EXIST=EX,IOSTAT=IOS,ERR=999)
    IF(.NOT.EX) THEN
      WRITE(9,'("DATA FILE ",(A)," DOES NOT EXIST")') FNAME
      RETURN
    ENDIF
```

```
**** OPEN FILE

    OPEN(UNIT=15,FILE=FNAME)

**** READ DATA FILE

    READ(15,*) XJUMP
    READ(15,*) XJUMP
    READ(15,*) XJUMP
    READ(15,*) (SCLOT(I),I=1,6)
    READ(15,*) (XKP(I),I=1,6)
    READ(15,*) (XKPM(I),I=1,6)
    READ(15,*) (XKFORC(I),I=1,6)
    READ(15,*) (COMP1(I),I=1,6)
    READ(15,*) (COMP2(I),I=1,6)
    READ(15,*) (COMP3(I),I=1,6)
    READ(15,*) (COMP4(I),I=1,6)
    READ(15,*) (COMP5(I),I=1,6)
    READ(15,*) DMPON,DMPOFF

    CLOSE(15)

**** READ VALVE DATA

    EX = .FALSE.

**** READS INPUT FILE NAME - VALVE

    FNAME = 'VALVEPAR.SA'
    WRITE(9,'("WHAT IS VALVE DATA FILE NAME?")')
    READ(9,*) FNAME

**** CHECK TO SEE IF FILE EXISTS

    INQUIRE(FILE=FNAME,EXIST=EX,IOSTAT=IOS,ERR=999)
    IF(.NOT.EX) THEN
      WRITE(9,'("DATA FILE ",(A)," DOES NOT EXIST")') FNAME
      RETURN
    ENDIF

**** OPEN FILE

    OPEN(UNIT=15,FILE=FNAME)

**** READ DATA FILE

    READ(15,*) DAMPM,BANDM,GAINM,PS,PR,XL1,XL2,XL3,APT,VO
 READ(15,*)XLMT,FRC,BETAO                                    ,XM1,VRST
    READ(15,*) (XM2(I),I=1,6)
    READ(15,*) (XK(I),I=1,6)
    READ(15,*) (B(I),I=1,6)
```

```
      CLOSE(15)
      RETURN

999   WRITE(9,'("   ERROR CODE  ",I5)') IOS
      RETURN
      END

*************************************************
******* SUBROUTINE TO SAVE DATA INTO A FILE
*************************************************
      SUBROUTINE FILESV


      COMMON/SMSPE1/MAXGEN,MAXPAR,SPECIAL
      COMMON/SMSPE2/IEFF,NAME,WADD
      COMMON/SMSPE3/NGEN,PAR,IPOINT
      COMMON/SMSPE4/MAXPSTO,IPTR,PSTO
      COMMON/SMVAL1/DAMPM,BANDM,GAINM,XM1,XM2,XK,B,VRST
      COMMON/SMVAL2/PS,PR
      COMMON/SMVAL3/XL1,XL2,XL3,APT,VO,XLIMIT,FRC,BETAO
      COMMON/SMSRV1/XKPM,XKP,SCLOT,XKFORC,DMPOFF,DMPON
      COMMON/SMSRV2/COMP1,COMP2,COMP3,COMP4,COMP5

      DIMENSION NGEN(240),PAR(1000),PSTO(1000)
      INTEGER WADD(20),SPECIAL(240)
      CHARACTER NAME(20)*20
      CHARACTER CH*1,FNAME*20
      DIMENSION XM2(6),XK(6),B(6)
     DIMENSION XKPM(6),XKP(6),SCLOT(6)                                  ,XKFORC(6)
      DIMENSION COMP1(6),COMP2(6),COMP3(6),COMP4(6),COMP5(6)
      DIMENSION XJMP1(6),XJMP2(6),XJMP3(6)
      LOGICAL EX

      WHILE(1>0)
        EX = .TRUE.

**** READS OUTPUT FILE NAME

        FNAME = '_'
        WRITE(9,'("YOU HAVE SELECTED: "SAVE FILES"")')
        WRITE(9,'(" <CR> TO RETURN TO MAIN MENU -",
     +          " ANYTHING TO CONTINUE")')
        READ(9,*) FNAME
        IF (FNAME(1:1) .EQ. '_') RETURN

**** CHECK TO SEE IF FILE EXISTS

CCCC    INQUIRE(FILE=FNAME,EXIST=EX,IOSTAT=IOS,ERR=999)
CCCC    IF(EX) THEN
CCCC      WRITE(9,'("FILENAME IS: ",(A))') FNAME
CCCC      WRITE(9,'("DO YOU WANT TO WRITE OVER IT (Y/N)?")')
CCCC      CH = 'N'
CCCC      READ (9,*) CH
```

```
CCCC    IF (CH .NE. 'Y') EXIT
CCCC  ENDIF

CCCC   OPEN(UNIT=15,FILE=FNAME,STATUS='NEW')

**** WRITES DATA TO FILE

CCCC   ENFILE(15)
CCCC   CLOSE(15)
      EXIT
    REPEAT

**** READS OUTPUT FILE NAME - SPECIAL EFFECTS

    WHILE (1>0)
 EX=.TRU                          E

      FNAME = 'SPEDATA.SA'
      WRITE(9,'("WHAT IS THE OUTPUT FILE NAME ",
+                 " - SPECIAL EFFECTS?")')
      READ(9,*) FNAME

**** CHECK TO SEE IF FILE EXISTS

      INQUIRE(FILE=FNAME,EXIST=EX,IOSTAT=IOS,ERR=999)
      IF(EX) THEN
       WRITE(9,'("FILENAME IS: ",(A))') FNAME
       WRITE(9,'("DO YOU WANT TO WRITE OVER IT (Y/N)?")')
       CH = 'N'
       READ (9,*) CH
       IF (CH .NE. 'Y') EXIT
      ENDIF


*** INITIALIZE PARAMETERS FOR SPECIAL EFFECTS INITIAL CONDITIONS

      ITR = 1
      DO (K = 1,MAXGEN)
        SELECT CASE (NGEN(K))
          CASE (1)
            PSTO(ITR) = 10000.
            PSTO(ITR+1) = 0.
            ITR = ITR + 2
          CASE (2)
            PSTO(ITR) = 10000.
            PSTO(ITR+1) = 0.
            PSTO(ITR+2) = 0.
            ITR = ITR + 3
          CASE (3)
            PSTO(ITR) = 0.
            ITR = ITR + 1
 ENDSELECT
    REPEAT
```

```
***** OPEN FILE

      OPEN(UNIT=15,FILE=FNAME,STATUS='NEW')

**** WRITE DATA INTO FILE
**** WRITE SPECIAL EFFECTS DATA

      WRITE(15,*) IEFF
      IF (IEFF .GT. 0) THEN
        K = INT(IEFF/4.)
        IMOD = MOD(IEFF,4)
        IF (IMOD .NE. 0) K = K + 1
        DO (IJ = 1,K)
          IJL = IJ*4
          IJF = IJL - 3
          IF (IEFF .LT. IJL) IJL = IEFF
          WRITE(15,*) (NAME(J),J=IJF,IJL)
        REPEAT
        WRITE(15,*) (WADD(J),J=1,IEFF)
      ENDIF
      WRITE(15,*) MAXGEN
      IF (MAXGEN .GT. 0) THEN
        K = INT(MAXGEN/33.)
        IMOD = MOD(MAXGEN,33)
        IF (IMOD .NE. 0) K = K + 1
        DO (IJ = 1,K)
          IJL = IJ*33
          IJF = IJL - 32
          IF (MAXGEN .LT. IJL) IJL = MAXGEN
          WRITE(15,*) (SPECIAL(J),J=IJF,IJL)
          WRITE(15,*) (NGEN(J),J=IJF,IJL)
        REPEAT
      ENDIF
WRITE(15,*)MA                                XPAR
      IF (MAXPAR .GT. 0) THEN
        K = INT(MAXPAR/11.)
        IMOD = MOD(MAXPAR,11)
        IF (IMOD .NE. 0) K = K + 1
        DO (IJ = 1,K)
          IJL = IJ*11
          IJF = IJL - 10
          IF (MAXPAR .LT. IJL) IJL = MAXPAR
          WRITE(15,*) (PAR(J),J=IJF,IJL)
        REPEAT
      ENDIF
      MAXPSTO = ITR - 1
      WRITE(15,*) MAXPSTO
      K = INT(MAXPSTO/11.)
      IMOD = MOD(MAXPSTO,11)
      IF (IMOD .NE. 0) K = K + 1
      DO (IJ = 1,K)
        IJL = IJ*11
```

```
      IJF = IJL - 10
      IF (MAXPSTO .LT. IJL) IJL = MAXPSTO
     WRITE(15,*) (PSTO(J),J=IJF,IJL)
     REPEAT

     ENDFILE(15)
     CLOSE(15)
     EXIT
    REPEAT

**** WRITE SERVO DATA

    WHILE (1>0)
     EX = .TRUE.

**** READS OUTPUT FILE NAME - SERVO

     FNAME = 'SERVOPAR.SA'
     WRITE(9,'("WHAT IS THE OUTPUT FILE NAME - SERVO?")')
     READ(9,*) FNAME

****CHECKTOSEEIFFILEE                              XISTS

     INQUIRE(FILE=FNAME,EXIST=EX,IOSTAT=IOS,ERR=999)
     IF(EX) THEN
      WRITE(9,'("FILENAME IS: ",(A))') FNAME
      WRITE(9,'("DO YOU WANT TO WRITE OVER IT (Y/N)?")')
      CH = 'N'
      READ (9,*) CH
      IF (CH .NE. 'Y') EXIT
     ENDIF

**** OPEN FILE

     OPEN(UNIT=15,FILE=FNAME)

**** READ DATA TO WRITE OVER

      READ(15,*) (XJMP1(I),I=1,6)
      READ(15,*) (XJMP2(I),I=1,6)
      READ(15,*) (XJMP3(I),I=1,6)
     CLOSE(15)

**** OPEN FILE

     OPEN(UNIT=15,FILE=FNAME,STATUS='NEW')

**** WRITE DATA FILE

      WRITE(15,*) (XJMP1(I),I=1,6)
      WRITE(15,*) (XJMP2(I),I=1,6)
      WRITE(15,*) (XJMP3(I),I=1,6)
      WRITE(15,*) (SCLOT(I),I=1,6)
```

```
      WRITE(15,*) (XKP(I),I=1,6)
      WRITE(15,*) (XKPM(I),I=1,6)
      WRITE(15,*) (XKFORC(I),I=1,6)
      WRITE(15,*) (COMP1(I),I=1,6)
      WRITE(15,*) (COMP2(I),I=1,6)
      WRITE(15,*) (COMP3(I),I=1,6)
                                  WRITE(15,*)(COMP4(I),I=1,6)
      WRITE(15,*) (COMP5(I),I=1,6)
      WRITE(15,*) DMPON,DMPOFF

     ENDFILE(15)
     CLOSE(15)
     EXIT
    REPEAT

**** WRITE VALVE DATA

    WHILE (1>0)
     EX = .TRUE.

**** READS OUTPUT FILE NAME - VALVE

     FNAME = 'VALVEPAR.SA'
     WRITE(9,'("WHAT IS THE OUTPUT FILE NAME - VALVE?")')
     READ(9,*) FNAME

**** CHECK TO SEE IF FILE EXISTS

     INQUIRE(FILE=FNAME,EXIST=EX,IOSTAT=IOS,ERR=999)
     IF(EX) THEN
      WRITE(9,'("FILENAME IS: ",(A))') FNAME
      WRITE(9,'("DO YOU WANT TO WRITE OVER IT (Y/N)?")')
      CH = 'N'
      READ (9,*) CH
      IF (CH .NE. 'Y') EXIT
     ENDIF

**** OPEN FILE

     OPEN(UNIT=15,FILE=FNAME,STATUS='NEW')

**** WRITE DATA FILE

      WRITE(15,*) DAMPM,BANDM,GAINM,PS,PR,XL1,XL2,XL3,APT,VO
      WRITE(15,*) XLIMIT,FRC,BETAO,XM1,VRST
      WRITE(15,*) (XM2(I),I=1,6)
  WRITE(15,*)(XK(I),I=1,                              6)
      WRITE(15,*) (B(I),I=1,6)

     ENDFILE(15)
     CLOSE(15)
     EXIT
    REPEAT
```

```
      RETURN

999   WRITE(9,'("    ERROR CODE  ",I5)') IOS
      RETURN
      END

*************************************************
* SUBROUTINE TO PERFORM THE GEOMETRIC TRANSFORMATIONS (VERSION #2)
* THIS VERSION IS CLOSELY RELATED TO THE ASSEMBLY LANGUAGE IMP.
*************************************************
      SUBROUTINE GTRN2(ANGLE,CEXT)

**** INPUTS

      DIMENSION ANGLE(6)

**** OUTPUTS

      DIMENSION CEXT(6)
      DIMENSION TEMP1(6),TEMP2(6),FORCE(6)

      DIMENSION CX(6),CY(6),BX(6),BY(6),BOTX(6),BOTY(6),BOTZ(6)
      DIMENSION BCYLNT(6),BCYLN(6),BCYLNT1(6)

**** INITIALIZE THE DIFFERENT CONSTANTS USED

      DATA (CX(I),I=1,6)/51.428,-22.931,-28.497,-28.497,-22.931,51.428/
      DATA (CY(I),I=1,6)/3.214,46.146,42.931,-42.931,-46.146,-3.214/

      DATA (BX(I),I=1,6)/36.891,31.324,-68.215,-68.215,31.324,36.891/
     DATA (BY(I),I=1,                                          6)/57.469,60.683,3.214,-3.214,-60.683,-57.469/
      BZ=65.942

**** DETERMINE THE ACTUATOR LENGTH COMPONENTS

      DO (N=1,6)

        TX=ANGLE(1)+CX(N)
        TY=ANGLE(2)+CY(N)
        TZ=ANGLE(3)

        CALL ROTATE(ANGLE(5),TY,TZ)
        CALL ROTATE(ANGLE(4),TZ,TX)
        CALL ROTATE(ANGLE(6),TX,TY)

        BOTX(N)=TX-BX(N)
        BOTY(N)=TY-BY(N)
        BOTZ(N)=TZ-BZ
        CEXT(N)=SQRT(BOTX(N)**2+BOTY(N)**2+BOTZ(N)**2)

        IF(CEXT(N).GT.104.63)CEXT(N)=104.63
        IF(CEXT(N).LT.68.63)CEXT(N)=68.63
```

```
      BCYLNT(N)=(CEXT(N)-86.63)/18.0001
      IF (BCYLNT(N).GT.1.0) BCYLNT(N)=1.0
      IF (BCYLNT(N).LT.-1.0) BCYLNT(N)=-1.0

**** SMOOTHING FUNCTION

      IF (ABS(BCYLNT(N)).LT.0.8292837) THEN
        BCYLN(N)=BCYLNT(N)
      ELSE
        BCYLNT1(N)=0.658707+SQRT(0.058288106-(ABS(BCYLNT(N))
+               -1.0)**2)
        IF (BCYLNT(N).LT.0.0) THEN
          BCYLN(N)=-BCYLNT1(N)
        ELSE
                                  BCYLN(N)=BCYLNT1(N)
        ENDIF
      ENDIF

      CEXT(N)=BCYLN(N)*18.0001

    REPEAT

    RETURN
    END

*************************************************
******** SUBROUTINE TO INPUT FLAGS FOR SPECIAL EFFECTS
********    AND INPUTS TO PRIMARY MOTION
*************************************************
    SUBROUTINE INPUTS

    COMMON/USER1/BIN,NINP,INPNUM,NITER
    DIMENSION BIN(2000,9),INPNUM(9)

    COMMON/SMSPE2/IEFF,NAME,WADD
    COMMON/SMSPE7/POSX,IEFFLG

    DIMENSION POSX(6),IEFFLG(20)
    DIMENSION NCE(20)
    INTEGER WADD(20)
    CHARACTER NAME(20)*20


**** SET FLAGS TO FALSE

    DO (I=1,IEFF)
      IEFFLG(I) = 0
    REPEAT

**** ENTER FLAGS FOR SPECIAL EFFECTS

    WRITE(9,'("YOU HAVE SELECTED: "INPUTS"")')
    CALL GRAMENU
```

```
      WRITE(9,'("ENTER NUMBER OF SPECIAL EFFECTS TO RUN")')
      WRITE(9,'("<CR> TO RETURN TO PREVIOUS MENU")')
      N = -1
      READ(9,*) N
    WHILE                                    ((N.LE.-1).OR.(N.GT.IEFF))
       IF (N .EQ. -1) RETURN
       N = -1
       WRITE(9,*)
       WRITE(9,'("   ILLEGAL NUMBER - ENTER AGAIN")')
       READ(9,*) N
      REPEAT

**** IF N = 0 THEN NO SPECIAL EFFECTS TO RUN

      IF (N .GT. 0) THEN
       WRITE(9,'("ENTER NUMBER CORRESPONDING TO EACH EFFECT")')
       READ(9,*) (NCE(I),I=1,N)

**** SET FLAGS TO TRUE

       DO (I=1,N)
        IEFFLG(NCE(I)) = -1
       REPEAT
      ENDIF

**** SET INPUTS TO PRIMARY MOTION

      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'(" INPUT SET UP TO PRIMARY MOTION")')
      WRITE(9,*)
      DUM=.05
      CALL PREPINP(DUM)

      RETURN
      END
```

```
************************************************
******* SUBROUTINE TO SIMULATE MOTION BASE SYSTEM - SIMULATE
************************************************
   SUBROUTINE SIMULATE

   COMMON/CUEFILX/BXACCN,BAXA0,BAXA1,BAXA2,BAXA3
   COMMON/CUEFILY/BYACCN,BAYA0,BAYA1,BAYA2,BAYA3
   COMMON/CUEFILZ/BZACCN,BAZA0,BAZA1,BAZA2,BAZA3
   COMMON/CUEFILQ/BAPITH,BQA0,BQA1,BQA2,BQA3
   COMMON/CUEFILP/BAROLL,BPA0,BPA1,BPA2,BPA3
   COMMON/CUEFILR/BAYAW,BRA0,BRA1,BRA2,BRA3
   COMMON/USER1/BIN,NINP,INPNUM,NITER
   DIMENSION BIN(2000,9),INPNUM(9)
   COMMON/CTTRAN/FLTAXA,FLTAYA,FLTAZA
   COMMON/CTROT/FQADOT,FPADOT,FRADOT,FQA,FPA,FRA
   COMMON/GRAV1/BXALIN,BYALIN
   COMMON/MPCP1/BXPOS,BYPOS,BZPOS,BTHPOS,BPHPOS,BSIPOS
   COMMON/POSCON1/BMXPOS,BMYPOS,BMZPOS,BMTHEP,BMPHIP,BMPSIP
   COMMON/PRIM1/XENFL,YENFL,ZENFL,THENFL,PHENFL,SIENFL
   COMMON/PRIM2/XGRFL,YGRFL,MPCPFL,MPLPFL
   COMMON/PRIM3/BXAWTC,BYAWTC,BZAWTC,BTHWTC,BPHWTC,BSIWTC
   COMMON/PRIM4/BKAX1,BKAY1,BKAZ1,BKQA1,BKPA1,BKRA1
   COMMON/PRIM5/BKAX2,BKAY2,BKAZ2,BKQA2,BKPA2,BKRA2
   COMMON/PRIM6/ALAGX,AKGAX,DLAGX,DKGAX,ALAGY,AKGAY,DLAGY,DKGAY
   COMMON/PRIM7/BKQA4,BKPA4
   COMMON/PRIM8/BLAGX,BLAGY,BLAGZ,BLAGT,BLAGP,BLAGS
   INTEGER*4  XENFL,YENFL,ZENFL,THENFL,PHENFL,SIENFL
   INTEGER*4  XGRFL,YGRFL,MPCPFL,MPLPFL
   REAL   BXAWTC,BYAWTC,BZAWTC,BTHWTC,BPHWTC,BSIWTC
   REAL   BKAX1,BKAY1,BKAZ1,BKQA1,BKPA1,BKRA1
   REAL   BKAX2,BKAY2,BKAZ2,BKQA2,BKPA2,BKRA2
   REAL   ALAGX,AKGAX,DLAGX,DKGAX,ALAGY,AKGAY,DLAGY,DKGAY
   REAL   BKQA4,BKPA4
   REAL   BLAGX,BLAGY,BLAGZ,BLAGT,BLAGP,BLAGS
   COMMON/SMSPE3/NGEN,PAR,IPOINT
   COMMON/SMSPE4/MAXPSTO,IPTR,PSTO
   COMMON/SMSPE7/POSX,IEFFLG
   COMMON/SMVAL1/DAMPM,BANDM,GAINM,XM1,XM2,XK,B,VRST
   COMMON/SMVAL2/PS,PR
   COMMON/SMVAL4/X,XDOT,PD,PDOT,XMOTOR,PISTON
   COMMON/SMVAL5/VOUT
   COMMON/SMVAL6/CPRS,RPRS,POSI,Q6,Q7
   COMMON/SMSRV3/TEMP1,TEMP2,CYLOT,FORCE
   COMMON/SMSRV4/COMPN
                              COMMON/SMDIS1/TMAX,NVS,NPNT,NVAR,PLOTV,PLOTA
   COMMON/SMANI1/SIGNA
   COMMON/SMDIS2/TINVX,AFTER

   DIMENSION XM2(6),XK(6),B(6)
   DIMENSION NGEN(240),PAR(1000),PSTO(1000)
   DIMENSION POSX(6),IEFFLG(20)
   DIMENSION X(6,2),XDOT(6,2),PD(6,4),PDOT(6,4)
   DIMENSION XMOTOR(4,5),PISTON(4,5)
```

```
      DIMENSION CPRS(6),RPRS(6),POSI(6),Q6(6),Q7(6)
      DIMENSION TEMP1(6),TEMP2(6),CYLOT(6),FORCE(6)
      DIMENSION NVAR(17),PLOTV(2000,18),PLOTA(2000,6)
      DIMENSION SIGNA(6),VOUT(6),COMPN(6)
      DIMENSION TINVX(6),AFTER(6),PRIMPOS(6)
      CHARACTER CH*1
      DATA PI/3.1416/

**** INITIALIZE VARIABLES

      T=0 ; I1=0 ; I2=0 ; I3=0 ; I4=0 ; I5=0 ; I6=-1 ; II=1

**** SELECT VARIABLES TO BE SAVED

      WRITE(9,'("YOU HAVE SELECTED: "EXECUTE"")')
      WRITE(9,'("ENTER NUMBER IN ASCENDING ORDER OF THE VARIABLE TO
     + BE SAVED (<CR> TO RETURN)")')
      WRITE(9,'("   HIT <CR> AFTER EACH ONE - 0 TO CONTINUE")')
     NVS=0                                                    !NUMBER OF VARIABLES TO SAVE
      NTEST=-1
      WHILE (NVS .LT. 17)
       WRITE(9,'("ENTRY #",I3)') NVS+1
       READ(9,*) NTEST              !READ # OF VARIABLE TO SAVE
       IF (NTEST .EQ. -1) RETURN
       IF (NTEST .EQ. 0) EXIT
       NVS = NVS + 1                !# OF TOTAL VARIABLES TO SAVE
       NVAR(NVS) = NTEST            !SAVE # OF VARIABLE TO SAVE
       NTEST = -1
      REPEAT

**** ENTER RATES FOR EACH PART
***  IF RATE .LE. 0 THEN THAT PART IS NOT EXECUTED

      WRITE(9,*)
      WRITE(9,'(" SET TIMING PARAMETERS")')
      WRITE(9,'("ENTER RATE FOR VALVE UPDATE (in Hz)")')
      READ(9,*) RATE
      WHILE (RATE .LE. 0)
        WRITE(9,'(" RATE NOT VALID - ENTER AGAIN")')
        READ(9,*) RATE
      REPEAT
      WRITE(9,'("RATE WILL BE: ",E11.5," Hz")') RATE
      DT = 1./RATE
      WRITE(9,'("SAMPLING INTERVAL: ",E11.5," s")') DT

      WRITE(9,*)
      WRITE(9,'("ENTER RATE FOR PRIMARY MOTION UPDATE (in Hz)")')
     READ(9,                       )RATE
      IF (RATE .LE. 0) THEN
        DTPRMO = 0.
        MCNT2 = -1 ; I2 = -1
        MCNT4 = -1 ; I4 = -1
      ELSE
```

```
      DTPRMO = 1./RATE
      MCNT2 = NINT(DTPRMO/DT)
      MCNT4 = MCNT2
      DTPRMO = MCNT2*DT
      RATE = 1./DTPRMO
      WRITE(9,'("RATE WILL BE: ",E11.5," Hz")') RATE
      WRITE(9,'("SAMPLING INTERVAL: ",E11.5," s")') DTPRMO
    ENDIF

    WRITE(9,*)
    WRITE(9,'("ENTER RATE FOR SPECIAL EFFECTS UPDATE (in Hz)")')
    READ(9,*) RATE
    IF (RATE .LE. 0) THEN
      DTSPEF = 0.
      MCNT3 = -1 ; I3 = -1
    ELSE
      DTSPEF = 1./RATE
      MCNT3 = NINT(DTSPEF/DT)
      MCNT4 = MCNT3 ; I4 = 0
      DTSPEF = MCNT3*DT
      RATE = 1./DTSPEF
      WRITE(9,'("RATE WILL BE: ",E11.5," Hz")') RATE
      WRITE(9,'("SAMPLING INTERVAL: ",E11.5," s")') DTSPEF
    ENDIF

    MCNT6 = -1
    WRITE(9,*)
    WRITE(9,'("ENTER RATE FOR SERVO UPDATE (in Hz)")')
    READ(9,*) RATE
                              IF(RATE.LE.0)THEN
      DTSRV = 0.
      MCNT5 = -1 ; I5 = -1 ; MCNT6 = -1
    ELSE
      DTSRV = 1./RATE
      MCNT5 = NINT(DTSRV/DT)
      DTSRV = MCNT5*DT
      RATE = 1./DTSRV
      WRITE(9,'("RATE WILL BE: ",E11.5," Hz")') RATE
      WRITE(9,'("SAMPLING INTERVAL: ",E11.5," s")') DTSRV
      WRITE(9,*)
      WRITE(9,'("DO YOU WANT THE INVERSE GEOMETRIC TRANSFORMATIONS
     + TO BE EXECUTED (Y/N)?")')

**** INVERSE GEOM. TRANSF. IS EXECUTED IF SERVO
****    AND SAVING VALUES ARE ALSO

      CH = 'N'
      READ(9,*) CH
      IF (CH .EQ. 'Y') THEN
        MCNT6 = MCNT5 ; I6 = 0
        WRITE(9,*)
        WRITE(9,'("ENTER VALUE OF TOLERENCE FOR CONVERGENCE")')
        EPSILON = 2.
```

```
      READ(9,*) EPSILON
      WHILE (EPSILON .LT. 2)
       WRITE(9,*)
       WRITE(9,'("  VALUE NOT VALID - ENTER AGAIN")')
       READ(9,*) EPSILON
      REPEAT
     ENDIF
    ENDIF

    RATE = 0.
    IF((                                    NVS.GT.0).OR.(I6.NE.-1))THEN
     WRITE(9,*)
     WRITE(9,'("ENTER RATE FOR SAVING VALUES (in Hz)")')
     READ(9,*) RATE
     RT1 = 1./DT
     WHILE ((RATE .GT. RT1) .AND. (RATE .GT. 0))
      WRITE(9,'("RATE NOT VALID - ENTER AGAIN")')
      READ(9,*) RATE
     REPEAT
    ENDIF
    IF (RATE .LE. 0) THEN

**** IF SAVING VALUES IS NOT EXECUTED THEN
****    NEITHER IS INVESE GEOM. TRANSF.

     I1 = -1 ; MCNT1 = -1 ; DTSAVE = 1 ; I6 = -1 ; MCNT6 = -1
    ELSE
     DTSAVE = 1./RATE
     MCNT1 = NINT(DTSAVE/DT)
     DTSAVE = MCNT1*DT
     RATE = 1./DTSAVE
     WRITE(9,'("RATE WILL BE: ",E11.5," Hz")') RATE
     WRITE(9,'("SAMPLING INTERVAL: ",E11.5," s")') DTSAVE

**** RATE OF INVERSE GEOM. TRANSF. IS THE SAME AS SAVING VALUES RATE

     IF (I6 .EQ. 0) MCNT6 = MCNT1
    ENDIF
    IF (I6 .EQ. -1) THEN
     WRITE(9,*)
     WRITE(9,'("INVERSE GEOMETRIC TRANSFORMATIONS WILL NOT BE",
+                                    'EXECUTED")')
    ENDIF

    WRITE(9,*)
    WRITE(9,'("ENTER TOTAL TIME OF SIMULATION (in sec.)")')
    READ(9,*) TMAX

**** CALCULATES NUMBER OF ITERATIONS (MITER) AND
****    APROXIMATE NUMBER OF POINTS FOR PLOT (NPNT .LT. 2000)

    NPNT = NINT(TMAX/DTSAVE)
    WHILE ((NPNT .GE. 2000) .OR. (NPNT .LE. 0))
```

```
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'(" LENGTH NOT ACCEPTED - ENTER AGAIN")')
      READ(9,*) TMAX
      NPNT = NINT(TMAX/DTSAVE)
    REPEAT
    MITER = NINT(TMAX/DT)

**** MOTOR MATRIX - CONTINUOUS TIME

    TEMP = 2*PI*BANDM     !OMEGAN
    XMOTOR(1,1) = 0 ; XMOTOR(1,2) = 1 ; XMOTOR(2,1) = -(TEMP**2)
    XMOTOR(2,2) = -2*TEMP*DAMPM ; XMOTOR(1,3) = 0
    XMOTOR(2,3) = GAINM*(TEMP**2)/VRST

**** PISTON MATRIX - CONSTANT PART OF CONTINUOUS TIME

    PISTON(1,1)=0 ; PISTON(1,2)=1 ; PISTON(1,3)=0 ; PISTON(1,4)=0
    PISTON(3,1)=0 ; PISTON(3,2)=0 ; PISTON(3,3)=0 ; PISTON(3,4)=1
 PISTON(1,5)=0;PISTO                                    N(3,5)=0;PISTON(4,5)=0

**** INITIALIZING VARIABLES

    DO (I=1,6)
      DO (IJ=1,4)
       PD(I,IJ) = 0.          !CYLINDER EXTENSION
       PDOT(I,IJ) = 0.        !CYLINDER VELOCITY
      REPEAT
      X(I,1) = 0.             !SPOOL POSITION
      X(I,2) = 0.
      XDOT(I,1) = 0.          !SPOOL VELOCITY
      XDOT(I,2) = 0.
      VOUT(I) = 0.            !VOLTAGE TO VALVE
      Q6(I) = 0.              !FLOW
      Q7(I) = 0.              !FLOW
      POSI(I) = 0.            !CYLINDER POSITION
      TEMP1(I) = 0.           !SERVO INTEGRATOR
      TEMP2(I) = 0.           !SERVO INTEGRATOR
      COMPN(I) = 0.           !COMPENSATION OUTPUT
      CPRS(I) = (PS+PR)/2.    !CAP PRESSURE
      RPRS(I) = (PS+PR)/2.    !ROD PRESSURE
      FORCE(I) = 0.           !DIFFERENTIAL PRESSURE
      POSX(I) = 0.            !SPECIAL EFFECTS OUTPUT
      SIGNA(I) = 0.           !SUM PRMO+SPEF
                                    CYLOT(I)=0.    !COMMANDED CYLINDER POS
      TINVX(I) = 0.           !OUTPUT INV. GEOM. TRANSF.
      AFTER(I) = 0.           !CYL. EXTEN. AFTER IGT
      PRIMPOS(I) = 0.
    REPEAT

**** INITIALIZING PRIMARY MOTION VARIABLES

    III = 1
```

```
      BXACCN=0.0;BXN1=0.0;BAXA0=0.0;BAXA1=0.0;BAXA2=0.0;BAXA3=0.0
      BYACCN=0.0;BYN1=0.0;BAYA0=0.0;BAYA1=0.0;BAYA2=0.0;BAYA3=0.0
      BZACCN=0.0;BZN1=0.0;BAZA0=0.0;BAZA1=0.0;BAZA2=0.0;BAZA3=0.0
      BAROLL=0.0;BPN1=0.0;BPA0=0.0;BPA1=0.0;BPA2=0.0;BPA3=0.0
      BAYAW=0.0;BSN1=0.0;BRA0=0.0;BRA1=0.0;BRA2=0.0;BRA3=0.0

**** INITIALIZE VARIABLES FOR SUBROUTINE CTRAN1

      FLTAXA=0.0 ; FLTAYA=0.0 ; FLTAZA=0.0
      FQADOT=0.0 ; FPADOT=0.0 ; FRADOT=0.0
      FQA=0.0    ; FPA=0.0    ; FRA=0.0

**** INITIALIZE VARIABLES FOR SUBROUTINE GRAVAL1

      BXALIN=0.0
      BYALIN=0.0

**** INITIALIZE VARIABLES FOR SUBROUTINE PLATPOS1

      BXPOS=0.0
      BYPOS=0.0
      BZPOS=0.0
      BTHPOS=0.0
                        BPHPOS=0.0
      BSIPOS=0.0

**** INITIALIZE VARIABLES FOR SUBROUTINE POSCON1

      BMXPOS=0.0
      BMYPOS=0.0
      BMZPOS=0.0
      BMTHEP=0.0
      BMPHIP=0.0
      BMPSIP=0.0

**** INITIAL CONDITIONS FOR SPECIAL EFFECTS (PSTO ARRAY)

      ITR = 1
      K = 1
      WHILE (ITR .LE. MAXPSTO)
        SELECT CASE (NGEN(K))
         CASE (1)
           PSTO(ITR) = 10000.
           PSTO(ITR+1) = 0.
           ITR = ITR + 2
         CASE (2)
           PSTO(ITR) = 10000.
           PSTO(ITR+1) = 0.
           PSTO(ITR+2) = 0.
           ITR = ITR + 3
         CASE (3)
           PSTO(ITR) = 0.
           ITR = ITR + 1
```

```
      END SELECT
      K = K + 1
    REPEAT

**** SIMULATION STARTS HERE

    DO (I = 1,MITER)

**** SAVE VALUES

      IF (I1 .EQ. 0) THEN
        CALL SAVEVAL(T,II)
        II = II + 1
        I1 = MCNT1
      ENDIF

**** UPDATE PRIMARY MOTION

      IF (I2 .EQ. 0) THEN
  CALL OSUCNE(DTP                          RMO)
        III = III + 1
        PRIMPOS(1) = BMXPOS
        PRIMPOS(2) = BMYPOS
        PRIMPOS(3) = BMZPOS
        PRIMPOS(4) = BMTHEP
        PRIMPOS(5) = BMPHIP
        PRIMPOS(6) = BMPSIP
        I2 = MCNT2
      ENDIF

**** UPDATE SPECIAL EFFECTS

      IF (I3 .EQ. 0) THEN
        CALL SMSPEF(DTSPEF)
        I3 = MCNT3
      ENDIF

      IF (I4 .EQ. 0) THEN

**** ADD PRIMARY MOTION TO SPECIAL EFFECTS

        DO (J=1,6)
          SIGNA(J) = POSX(J)+PRIMPOS(J)
        REPEAT

**** UPDATE GEOMETRIC TRANSFORMATIONS

        CALL GTRN2(SIGNA,CYLOT)
        I4 = MCNT4
      ENDIF

**** UPDATE SERVO
```

```
      IF (I5 .EQ. 0) THEN
        CALL SMSRV(DTSRV)
        I5 = MCNT5

**** UNIT STEP INPUT TO VALVE IF SPECIAL EFFECTS AND PRIMARY MOTION
****  ARE NOT EXECUTED

        IF ((I2 .LE. -1) .AND. (I3 .LE. -1)) THEN
         DO (IIII = 1,6)
          CYLOT(IIII) = 1.
         REPEAT
                        ENDF
      ENDIF

**** UPDATE VALVE

      CALL SMVALVE(DT)

**** UPDATE INVERSE GEOMETRIC TRANSFORMATIONS

      IF (I6 .EQ. 0) THEN
        CALL SMIGT(POSI,SIGNA,TINVX,EPSILON)
        CALL GTRN2(TINVX,AFTER)
        I6 = MCNT6
      ENDIF

**** UPDATE VARIABLES

      I1 = I1 - 1
      I2 = I2 - 1
      I3 = I3 - 1
      I4 = I4 - 1
      I5 = I5 - 1
      I6 = I6 - 1
      T = T + DT

    REPEAT
    NPNT = II - 1    !TOTAL NUMBER OF POINTS OF SIMULATION
    RETURN
    END

*************************************************
******* SUBROUTINE TO ROTATE A VECTOR (2 ELEMENTS) BY THE GIVEN ANGLE:
*******
*******      | VAR1 | | COS(ANGLE)    -SIN(ANGLE) | | VAR1 |
*******      |      |=|                 ||          |
*******      | VAR2 | | SIN(ANGLE)     COS(ANGLE) | | VAR2 |
*******
*****                                                  ****************************************
    SUBROUTINE ROTATE(ANGLE,VAR1,VAR2)

    SINA=SIN(ANGLE)
    COSA=COS(ANGLE)
```

```
      TEMP1=VAR1*COSA-VAR2*SINA
      VAR2=VAR1*SINA+VAR2*COSA
      VAR1=TEMP1

      RETURN
      END

*************************************************
******* SUBROUTINE TO SAVE VALUES OF SELECTED VARIABLES
*************************************************
      SUBROUTINE SAVEVAL(T,I)

      COMMON/CUEFILX/BXACCN,BAXA0,BAXA1,BAXA2,BAXA3
      COMMON/CUEFILY/BYACCN,BAYA0,BAYA1,BAYA2,BAYA3
      COMMON/CUEFILZ/BZACCN,BAZA0,BAZA1,BAZA2,BAZA3
      COMMON/CUEFILQ/BAPITH,BQA0,BQA1,BQA2,BQA3
      COMMON/CUEFILP/BAROLL,BPA0,BPA1,BPA2,BPA3
      COMMON/CUEFILR/BAYAW,BRA0,BRA1,BRA2,BRA3
      COMMON/CTTRAN/FLTAXA,FLTAYA,FLTAZA
      COMMON/CTROT/FQADOT,FPADOT,FRADOT,FQA,FPA,FRA
      COMMON/GRAV1/BXALIN,BYALIN
      COMMON/MPCP1/BXPOS,BYPOS,BZPOS,BTHPOS,BPHPOS,BSIPOS
      COMMON/POSCON1/BMXPOS,BMYPOS,BMZPOS,BMTHEP,BMPHIP,BMPSIP
C     COMMON/SMSPE7/                    /POSX/IEFFLG
      COMMON/SMVAL4/X,XDOT,PD,PDOT,XMOTOR,PISTON
      COMMON/SMVAL5/VOUT
      COMMON/SMVAL6/CPRS,RPRS,POSI,Q6,Q7
      COMMON/SMVAL7/Q1,Q2,Q3,Q4,Q5,V
      COMMON/SMSRV3/TEMP1,TEMP2,CYLOT,FORCE
      COMMON/SMSRV4/COMPN
      COMMON/SMDIS1/TMAX,NVS,NPNT,NVAR,PLOTV,PLOTA
      COMMON/SMANI1/SIGNA
      COMMON/SMDIS2/TINVX,AFTER

      DIMENSION NVAR(17),PLOTV(2000,18)
      DIMENSION SAVE(500),PLOTA(2000,6)
      DIMENSION POSX(6),IEFFLG(20)
      DIMENSION X(6,2),XDOT(6,2),PD(6,4),PDOT(6,4)
      DIMENSION XMOTOR(4,5),PISTON(4,5)
      DIMENSION VOUT(6)
      DIMENSION CPRS(6),RPRS(6),POSI(6),Q6(6),Q7(6)
      DIMENSION Q1(6),Q2(6),V(6,2),Q3(6),Q4(6),Q5(6)
      DIMENSION TEMP1(6),TEMP2(6),CYLOT(6),FORCE(6)
      DIMENSION SIGNA(6),COMPN(6)
      DIMENSION TINVX(6),AFTER(6)

**** SAVING VALUES

      DO (J=1,6)

**** VALUES TO ANIMATE

        PLOTA(I,J) = TINVX(J)
```

```
**** OTHER VARIABLES

 SAVE(J)=POSX(J)      !SPECIALEFFECTSOUTPU                                T
    SAVE(J+6) = PD(J,1)          !CYLINDER POSITION
    SAVE(J+12) = PDOT(J,1)        !CYLINDER VELOCITY
    SAVE(J+18) = X(J,1)          !SPOOL POSITION
    SAVE(J+24) = XDOT(J,1)        !SPOOL VELOCITY
    SAVE(J+30) = VOUT(J)         !VOLTAGE TO VALVE
    SAVE(J+36) = CPRS(J)         !CAP PRESSURE
    SAVE(J+42) = RPRS(J)         !ROD PRESSURE
    SAVE(J+48) = POSI(J)         !CYLINDER POSITION (=PD)
    SAVE(J+54) = CYLOT(J)         !OUTPUT OF GEOM. TRANSF.
    SAVE(J+60) = FORCE(J)         !COMP. DIFF. FORCE (PRES)
    SAVE(J+66) = Q6(J)           !COMPRESSIBILITY FLOW
    SAVE(J+72) = Q7(J)           !COMPRESSIBILITY FLOW
    SAVE(J+78) = SIGNA(J)         !SPEFF + PRIMOT
    SAVE(J+84) = COMPN(J)         !SERVO COMPENSATION
    SAVE(J+90) = Q1(J)           !SPOOL FLOW
 SAVE(J+96)=Q2(J)      !SPOOLFLO                                          W
    SAVE(J+102)= V(J,1)          !VOLUME OF PISTON (CAP)
    SAVE(J+108)= V(J,2)          !VOLUME OF PISTON (ROD)
    SAVE(J+114)= Q3(J)           !LEAKAGE FLOW
    SAVE(J+120)= Q4(J)           !LEAKAGE FLOW
    SAVE(J+126)= Q5(J)           !LEAKAGE FLOW
    SAVE(J+132)= CPRS(J)-RPRS(J)     !DIFFERENTIAL PRESSURE
    SAVE(J+138)= TINVX(J)         !INVERSE TRANSF. OUTPUT
    SAVE(J+144)= AFTER(J)         !EXTEN. AFTER INV.TRANSF.
  REPEAT
  SAVE(151) = FLTAXA          !X ACC. INPUT
  SAVE(152) = FLTAYA          !Y ACC. INPUT
  SAVE(153) = FLTAZA          !Z ACC. INPUT
  SAVE(154) = FQA            !PITCH RATE INPUT
  SAVE(155) = FPA            !ROLL RATE INPUT
  SAVE(156) = FRA            !YAW RATE INPUT
  SAVE(157) = FQADOT           !PITCH ACC. INPUT
  SAVE(158) = FPADOT           !ROLL ACC. INPUT
  SAVE(159) = FRADOT           !YAW ACC. INPUT
                               SAVE(160)=BXACCN    !XCENT.ACC.
  SAVE(161) = BYACCN           !Y CENT. ACC.
  SAVE(162) = BZACCN           !Z CENT. ACC.
  SAVE(163) = BAROLL           !ROLL CENT. ACC.
  SAVE(164) = BAYAW            !YAW CENT. ACC.
  SAVE(165) = BAXA2           !X OUTPUT OF WASHOUT
  SAVE(166) = BAYA2           !Y OUTPUT OF WASHOUT
  SAVE(167) = BAZA2           !Z OUTPUT OF WASHOUT
  SAVE(168) = BPA2            !ROLL OUTPUT OF WASHOUT
  SAVE(169) = BRA2            !YAW OUTPUT OF WASHOUT
  SAVE(170) = BXALIN           !X GRAVITY ALIGN.
  SAVE(171) = BYALIN           !Y GRAVITY ALING.
  SAVE(172) = BXPOS            !X PLATF. POS.
  SAVE(173) = BYPOS            !Y PLATF. POS.
  SAVE(174) = BZPOS            !Z PLATF. POS.
```

```
      SAVE(175) = BTHPOS        !PITCH PLATF. POS.
      SAVE(176) = BPHPOS        !ROLL PLATF. POS
      SAVE(177) = BSIPOS        !YAW PLATF. POS.
      SAVE(178)=BMXPOS                          !X OUTPUT OF PRIM. MOTION
      SAVE(179) = BMYPOS        !Y OUTPUT OF PRIM. MOTION
      SAVE(180) = BMZPOS        !Z OUTPUT OF PRIM. MOTION
      SAVE(181) = BMTHEP        !PITCH OUTPUT OF PRIM. MOTION
      SAVE(182) = BMPHIP        !ROLL OUTPUT OF PRIM. MOTION
      SAVE(183) = BMPSIP        !YAW OUTPUT OF PRIM. MOTION
      SAVE(184) = BAXA0         !X ACC WASHOUT
      SAVE(185) = BAYA0         !Y ACC WASHOUT
      SAVE(186) = BAZA0         !Z ACC WASHOUT
      SAVE(187) = BPA0          !ROLL RATE WASHOUT
      SAVE(188) = BRA0          !YAW RATE WASHOUT
      SAVE(189) = BAXA1         !X VEL WASHOUT
      SAVE(190) = BAYA1         !Y VEL WASHOUT
      SAVE(191) = BAZA1         !Z VEL WASHOUT
      SAVE(192) = BPA1          !ROLL WASHOUT
      SAVE(193) = BRA1          !YAW WASHOUT
      SAVE(194) = BAXA3         !X INTER. WASHOUT
      SAVE(195) = BAYA3         !Y INTER. WASHOUT
      SAVE(196)=BAZA3                           !Z INTER. WASHOUT
      SAVE(197) = BPA3          !ROLL INTER. WASHOUT
      SAVE(198) = BRA3          !YAW INTER. WASHOUT

      IF (NVS .GT. 0) THEN

**** SAVING TIME TO PLOT

       PLOTV(I,1) = T

**** SAVING SELECTED VARIABLES TO PLOT

       DO (J=1,NVS)
         PLOTV(I,J+1) = SAVE(NVAR(J))
       REPEAT
      ENDIF
      RETURN
      END

*************************************************
******* SUBROUTINE TO SET UP SERVO
*******   INPUT SERVO PARAMETERS
*************************************************
      SUBROUTINE SERVO

      COMMON/SMSRV1/XKPM,XKP,SCLOT,XKFORC,DMPOFF,DMPON
      COMMON/SMSRV2/COMP1,COMP2,COMP3,COMP4,COMP5

      DIMENSION XKPM(6),XKP(6),SCLOT(6),XKFORC(6)
      DIMENSION COMP1(6),COMP2(6),COMP3(6),COMP4(6),COMP5(6)

      WRITE(9,'("YOU HAVE SELECTED: "SET UP SERVO"")')
```

```
      WRITE(9,*)
      WRITE(9,'("ENTER SERVO PARAMETERS")')
      WRITE(9,*)
      WRITE(9,'("COMMANDED POSITION GAIN (<CR> TO RETURN):" ")')
      XKPM(1) = 0
      READ(9,*) (XKPM(I),I=1,6)
      IF (XKPM(1) .EQ. 0) RETURN
      WRITE(9,*)
      WRITE(9,'("FEEDBACK POSITION GAIN: ")')
      READ(9,*) (XKP(I),I=1,6)
      WRITE(9,*)
      WRITE(9,'("SCALE FOR VALVE VOLTAGE (mV/in):")')
      READ(9,*) (SCLOT(I),I=1,6)
      WRITE(9,*)
      WRITE(9,'("DIFFERENTIAL PRESSURE GAIN: ")')
      READ(9,*) (XKFORC(I),I=1,6)
      WRITE(9,*)
      WRITE(9,'("COMPENSATION GAINS #1: ")')
      READ(9,*) (COMP1(I),I=1,6)
      WRITE(9,'("COMPENSATION GAINS #2: ")')
      READ(9,*) (COMP2(I),I=1,6)
      WRITE(9,'("COMPENSATION GAINS #3: ")')
      READ(9,*) (COMP3(I),I=1,6)
      WRITE(9,'("COMPENSATION GAINS #4: ")')
      READ(9,*) (COMP4(I),I=1,6)
      WRITE(9,'("COMPENSATION GAINS #5: ")')
      READ(9,*) (COMP5(I),I=1,6)
      WRITE(9,'("2 DAMPING FADE LIMITS (HIGHER FIRST)")')
      READ(9,*) DMPON,DMPOFF
      RETURN
      END

************************************************
***** SUBROUTINE TO                    EXECUTE INVERSE TRANSFORMATIONS
*******   FROM CYLINDER LENGHTS (CEXT) IT CALCULATES
*******   POSITION AND ANGULAR ORIENTATION OF UPPER PLATFORM (TINVX)
*******    USES MARQUAROT ALGORITHM FOR CONVERGENCE
*******    TINVXI: INITIAL CONDITION VECTOR
*******    EPSILON AND TOL: TOLERANCE FOR CONVERGENCE
************************************************
      SUBROUTINE SMIGT(CEXT,TINVXI,TINVX,EPSILON)

**** INPUT AND OUTPUT OF TRANSFORMATION - TINVX
****   X,Y,Z,THETA,PHI,PSI OF UPPER PLATFORM

      DIMENSION TINVX(6),CEXT(6),TINVXI(6)
      DIMENSION F(6),DF(6,6),DFINV(6,6),DECTIV(6)
      DIMENSION TINVXN(6),TINVXL(6)
      DIMENSION CX(6),CY(6),BX(6),BY(6),XL(6),FF(6)
      DIMENSION FI(6,6),DFT(6,6),H(6),HTH(6,6),DFI(6,6)

**** INITIALIZE THE DIFFERENT CONSTANTS USED
```

```
      DATA V,TOL/10.,0.00001/

      DATA (CX(I),I=1,6)/51.428,-22.931,-28.497,-28.497,-22.931,51.428/
      DATA (CY(I),I=1,6)/3.214,46.146,42.931,-42.931,-46.146,-3.214/

     DATA(B                                                X(I),I=1,6)/36.891,31.324,-68.215,-68.215,31.324,36.891/
      DATA (BY(I),I=1,6)/57.469,60.683,3.214,-3.214,-60.683,-57.469/
      BZ=65.942
      RMU = 0.01

**** FI IS THE IDENTITY MATRIX

      DO (I=1,6)
       DO (J=1,6)
        IF (I .EQ. J) THEN
         FI(I,J) = 1.
        ELSE
         FI(I,J) = 0.
        ENDIF
       REPEAT
      REPEAT

**** REARRANGE INPUT VECTOR - INITIAL CONDITION

      TINVXN(1) = TINVXI(1)
      TINVXN(2) = TINVXI(2)
      TINVXN(3) = TINVXI(3)-BZ
      TINVXN(4) = TINVXI(6)
      TINVXN(5) = TINVXI(4)
      TINVXN(6) = TINVXI(5)

**** CALCULATE TOTAL LENGTH OF VECTORS L

      DO (I=1,6)
       XL1 = CX(I) - BX(I)
       XL2 = CY(I) - BY(I)
       XL3 = BZ
       XL(I) = SQRT(XL1**2+XL2**2+XL3**2)
       XL(I) = XL(I) + CEXT(I)
      REPEAT

      WHILE (1>0)

**** MAKE NEW PARAMETERS EQUAL OLD PARAMETERS

       DO (I=1,6)
        TINVXL(I) = TINVXN(I)
       REPEAT

     T11=COS(TINVXL(4))*COS(T                                  INVXL(5))
      T12 = SIN(TINVXL(4))*COS(TINVXL(5))
      T13 = -1.*SIN(TINVXL(5))
      T21 = COS(TINVXL(4))*SIN(TINVXL(5))*SIN(TINVXL(6))
```

```
+       -SIN(TINVXL(4))*COS(TINVXL(6))
      T22 = SIN(TINVXL(4))*SIN(TINVXL(5))*SIN(TINVXL(6))
+       +COS(TINVXL(4))*COS(TINVXL(6))
      T23 = COS(TINVXL(5))*SIN(TINVXL(6))
      T31 = COS(TINVXL(4))*SIN(TINVXL(5))*COS(TINVXL(6))
+       +SIN(TINVXL(4))*SIN(TINVXL(6))
      T32 = SIN(TINVXL(4))*SIN(TINVXL(5))*COS(TINVXL(6))
+       -COS(TINVXL(4))*SIN(TINVXL(6))
      T33 = COS(TINVXL(5))*COS(TINVXL(6))

      DO (I=1,6)
       TX=CX(I)
       TY=CY(I)
       TZ=0

       CALL ROTATE(TINVXL(6),TY,TZ)
       CALL ROTATE(TINVXL(5),TZ,TX)
       CALL ROTATE(TINVXL(4),TX,TY)

       XL1=TX-BX(I)+TINVXL(1)
       XL2=TY-BY(I)+TINVXL(2)
       XL3=TZ+TINVXL(3)
       TEMP=XL1**2+XL2**2+XL3**2
       F(I) = TEMP - XL(I)**2

    DF(I,1)=2*(TINVXL(1)+CX(I)*T11+CY(I)*T                    21-BX(I))
      DF(I,2) = 2*(TINVXL(2)+CX(I)*T12+CY(I)*T22-BY(I))
      DF(I,3) = 2*(TINVXL(3)+CX(I)*T13+CY(I)*T23)
      DF(I,4) = -2*(TINVXL(1)-BX(I))*(CX(I)*T12+CY(I)*T22)
+             +2*(TINVXL(2)-BY(I))*(CX(I)*T11+CY(I)*T21)
      DF(I,5) = 2*(TINVXL(1)-BX(I))*(-CX(I)*SIN(TINVXL(5))*
+             COS(TINVXL(4))+CY(I)*SIN(TINVXL(6))*
+             COS(TINVXL(5))*COS(TINVXL(4)))
+             +2*(TINVXL(2)-BY(I))*(-CX(I)*SIN(TINVXL(5))
+             *SIN(TINVXL(4))+CY(I)*SIN(TINVXL(6))*
+             COS(TINVXL(5))*SIN(TINVXL(4)))-2*TINVXL(3)*(CX(I)
+             *COS(TINVXL(5))+CY(I)*SIN(TINVXL(6))
+             *SIN(TINVXL(5)))
      DF(I,6) = 2*(TINVXL(1)-BX(I))*CY(I)*T31
+             +2*(TINVXL(2)-BY(I))*CY(I)*T32
+             +2*TINVXL(3)*CY(I)*T33

      REPEAT

**** CALCULATE THE NORM OF F

      SUM = 0.
      DO (I=1,6)
       SUM = SUM + F(I)**2
      REPEAT
    ERRO                    =SUM

**** CALCULATE TRANSPOSE OF DF: DFT
```

```
      CALL MM_TRP(DF,DFT,6)

***** H = DFT*F

      CALL MVV_MUL(DFT,F,H,6)

***** CALCULATE THE NORM OF H

      SUM = 0.
      DO (I=1,6)
       SUM = SUM + H(I)**2
      REPEAT
      RNORM = SQRT(SUM)

***** TEST IF TOLERENCE IS REACHED

      IF (RNORM .LE. EPSILON) EXIT
      RMU = RMU/(V**2)

      ERRN = 2.*ERRO
      WHILE (ERRN .GT. ERRO)
       RMU = RMU*V

***** HTH = DFT*DF

       CALL MMM_MUL(DFT,DF,HTH,6)

***** DFI = DFT*DF + RMU*FI (RMU IS SCALAR)

       CALL MMM_SUM(HTH,FI,DFI,RMU,6)

***** DFINV: INVERSE OF MATRIX DFI

       CALL MAT_INV(DFI,DFINV,6,RCOND)

       IF (RCOND .EQ. 0) CYCLE

***** DFINV*H=DECTIV (DECREMENT IN PARAMETER VECTOR TINVX)

       CALL MVV_MUL(DFINV,H,DECTIV,6)

***** DECREMENT TINVX (POSITION AND
*****  ANGULAR ORIENTATION OF UPPER PLATFORM)
*****  AND CALCULATE ERROR FROM LAST DECREMENT (DELTA)

      DO(I=1,                          6)
        TINVXN(I) = TINVXL(I) - DECTIV(I)
       REPEAT

       DO (I=1,6)
        TX=CX(I)
        TY=CY(I)
```

```
      TZ=0

      CALL ROTATE(TINVXN(6),TY,TZ)
      CALL ROTATE(TINVXN(5),TZ,TX)
      CALL ROTATE(TINVXN(4),TX,TY)

      XL1=TX-BX(I)+TINVXN(1)
      XL2=TY-BY(I)+TINVXN(2)
      XL3=TZ+TINVXN(3)
      TEMP=XL1**2+XL2**2+XL3**2
      FF(I) = TEMP - XL(I)**2
     REPEAT

**** CALCULATE SUM OF SQUARES OF FF

     SUM = 0.
     DO (I=1,6)
      SUM = SUM + FF(I)**2
     REPEAT
     ERRN = SUM

    REPEAT

**** CALCULATE SUM OF SQUARES OF DECTIV

    SUM = 0.
    DO (I=1,6)
     SUM = SUM + DECTIV(I)**2
    REPEAT

    IF (SUM .LE. TOL) EXIT

    RMU = RMU*V

   REPEAT

**** REARRANGE OUTPUT

   TINVX(1) = TINVXL(1)
   TINVX(2) = TINVXL(2)
   TINVX(3) = TINVXL(3)+BZ
   TINVX(4) = TINVXL(5)
TINVX(5)                    =TINVXL(6)
   TINVX(6) = TINVXL(4)

   RETURN
   END

************************************************
******* SUBROUTINE FOR THE RESPONSE OF A SPECIAL EFFECT
************************************************
   SUBROUTINE SMSPEF(DTSE)
```

```
      COMMON/SMSPE1/MAXGEN,MAXPAR,SPECIAL
      COMMON/SMSPE2/IEFF,NAME,WADD
      COMMON/SMSPE3/NGEN,PAR,IPOINT
      COMMON/SMSPE4/MAXPSTO,IPTR,PSTO
      COMMON/SMSPE5/KEEP,MAXKEEP
      COMMON/SMSPE6/YAMP,DTSPEF
      COMMON/SMSPE7/POSX,IEFFLG

      DIMENSION NGEN(240),PAR(1000),KEEP(1000),PSTO(1000)
      DIMENSION POSX(6),IEFFLG(20)
      INTEGER WADD(20),SPECIAL(240)
      CHARACTER NAME(20)*20
      DIMENSION POSX(6),IFLGEN(240)

      DTSPEF = DTSE

***** ZERO ELEMENTS OF OUTPUT

      DO (I = 1,6)
        POSX(I) = 0.
      REPEAT

***** TEST FOR GENERATORS

      IF (MAXGEN .EQ. 0) RETURN

***** SET FLAGS TO FALSE

      DO (I = 1,MAXGEN)
        IFLGEN(I) = 0
      REPEAT

***SETFLAGSFORGENE                                    RATORSOFCORRESPONDINGSPECIALEFFECT

      DO (J = 1,IEFF)
        IF (IEFFLG(J) .EQ. -1) THEN
          DO (I = 1,MAXGEN)
            IF (SPECIAL(I) .EQ. J) THEN
              IFLGEN(I) = -1
            ENDIF
          REPEAT
        ENDIF
      REPEAT

***** INITIALIZE COUNTERS

      IPTR = 1
      IPT = 1

***** LOOP THROUGH ALL GENERATORS

      DO (I = 1,MAXGEN)
```

```
**** TEST IF GENERATOR IS ON

      IF (IFLGEN(I) .EQ. -1) THEN

**** SELECT GENERATOR TYPE AND CALCULATE AMPLITUDE

        SELECT CASE (NGEN(I))
         CASE (1)
          CALL F1AMP(IPT)
         CASE (2)
          CALL F2AMP(IPT)
         CASE (3)
          CALL SINEAMP(IPT)
        END SELECT

**** WHERE TO ADD OUTPUT OF GENERATOR

        POSX(WADD(SPECIAL(I))) = POSX(WADD(SPECIAL(I))) + YAMP
       ELSE

**** SKIP PARAMETERS AND RESET PSTO VECTOR
        SELECT CASE (NGEN(I))
         CASE (1)
          IIK = 3
                                    PSTO(IPTR)=10000.
          PSTO(IPTR+1) = 0.
         CASE (2)
          IIK = 4
          PSTO(IPTR) = 10000.
          PSTO(IPTR+1) = 0.
          PSTO(IPTR+2) = 0.
         CASE (3)
          IIK = 2
          PSTO(IPTR) = 0.
        END SELECT

**** INCREMENT POINTERS

      IPT = IPT + IIK
      IPTR = IPTR + IIK - 1
    ENDIF
    REPEAT

    RETURN
    END

************************************************
******** SUBROUTINE TO SIMULATE SERVO
************************************************
    SUBROUTINE SMSRV(DTSRV)

    COMMON/SMVAL3/XL1,XL2,XL3,APT,VO,XLIMIT,FRC,BETAO
    COMMON/SMVAL5/VOUT
```

```
      COMMON/SMVAL6/CPRS,RPRS,POSI,Q6,Q7
      COMMON/SMSRV1/XKPM,XKP,SCLOT,XKFORC,DMPOFF,DMPON
      COMMON/SMSRV2/COMP1,COMP2,COMP3,COMP4,COMP5
      COMMON/SMSRV3/TEMP1,TEMP2,CYLOT,FORCE
      COMMON/SMSRV4/COMPN

      DIMENSION VOUT(6)
      DIMENSION CPRS(6),RPRS(6),POSI(6),Q6(6),Q7(6)
                                        DIMENSION XKPM(6),XKP(6),SCLOT(6),XKFORC(6)
      DIMENSION COMP1(6),COMP2(6),COMP3(6),COMP4(6),COMP5(6)
      DIMENSION TEMP1(6),TEMP2(6),CYLOT(6)
      DIMENSION FORCE(6),COMPN(6)

      DO (I = 1,6)

***** CALCULATE DIFFERENTIAL FORCE

        FORCE(I) = (CPRS(I)-RPRS(I))*APT

***** COMPENSATION

        COMPN(I) = TEMP2(I) + COMP3(I)*FORCE(I)*XKFORC(I)
        IF (POSI(I) .GT. DMPON) THEN
          TMP = 1.
        ELSE
          IF (POSI(I) .LT. DMPOFF) THEN
            TMP = 0.
          ELSE
            TMP = (POSI(I)-DMPOFF)/(DMPON-DMPOFF)
          ENDIF
        ENDIF
        TMP = COMPN(I)*TMP
        DELTAY = CYLOT(I)*XKPM(I) + POSI(I)*XKP(I) + TMP

***** VOLTAGE TO VALVE

        VOUT(I) = -SCLOT(I)*DELTAY

***** INTEGRATING IN THE COMPENSATION LOOP

        XDOT1 = COMP1(I)*XKFORC(I)*FORCE(I)
     +        + COMP4(I)*COMPN(I)
        XDOT2 = COMP2(I)*XKFORC(I)*FORCE(I)
     +        + COMP5(I)*COMPN(I) + TEMP1(I)
                                        TEMP1(I) = TEMP1(I)+XDOT1*DTSRV
        TEMP2(I) = TEMP2(I) + XDOT2*DTSRV
      REPEAT

      RETURN
      END

************************************************
******** SUBROUTINE TO SIMULATE VALVE
```

```
************************************************
   SUBROUTINE SMVALVE(DT)

   COMMON/SMVAL1/DAMPM,BANDM,GAINM,XM1,XM2,XK,B,VRST
   COMMON/SMVAL2/PS,PR
   COMMON/SMVAL3/XL1,XL2,XL3,APT,VO,XLIMIT,FRC,BETAO
   COMMON/SMVAL4/X,XDOT,PD,PDOT,XMOTOR,PISTON
   COMMON/SMVAL5/VOUT
   COMMON/SMVAL6/CPRS,RPRS,POSI,Q6,Q7
   COMMON/SMVAL7/Q1,Q2,Q3,Q4,Q5,V

   DIMENSION XM2(6),XK(6),B(6)
   DIMENSION X(6,2),XDOT(6,2),PD(6,4),PDOT(6,4)
   DIMENSION XMOTOR(4,5),PISTON(4,5),PISTOND(4,5)
   DIMENSION VOUT(6)
   DIMENSION CPRS(6),RPRS(6),POSI(6)
   DIMENSION Q1(6),Q2(6),Q3(6),Q4(6),Q5(6),Q6(6),Q7(6),V(6,2)

   DO (I=1,6)

**** SOLVE MOTOR EQUATION

    CALL EULER(X,XDOT,2,XMOTOR,DT,VOUT(I),I)

***CHECK FOR SATURATION OF SPOOL                    POSITION

    IF (X(I,1) .LT. -1.*XLIMIT) THEN
      X(I,1) = -1.*XLIMIT
      X(I,2) = 0
    ENDIF
    IF (X(I,1) .GT. XLIMIT) THEN
      X(I,1) = XLIMIT
      X(I,2) = 0
    ENDIF

**** CALCULATE FLOW RATES

    IF (X(I,1) .GE. 0.) THEN
      Q1(I) = FRC*SQRT(PS-CPRS(I))*X(I,1)
      Q2(I) = FRC*SQRT(RPRS(I)-PR)*X(I,1)
    ELSE
      Q1(I) = FRC*SQRT(CPRS(I)-PR)*X(I,1)
      Q2(I) = FRC*SQRT(PS-RPRS(I))*X(I,1)
    ENDIF

**** CALCULATE LEAKAGE FLOW

    Q3(I) = XL1*(CPRS(I)-RPRS(I))
    Q4(I) = XL2*(CPRS(I)-PR)
    Q5(I) = XL3*(RPRS(I)-PR)

**** PISTON MATRIX - CONTINUOUS TIME (VARIABLE COEFFICIENTS)
```

```
      PISTON(2,1)=-(XK(I)/XM1) ; PISTON(2,2)=-(B(I)/XM1)
      PISTON(2,3)=-PISTON(2,1) ; PISTON(2,4)=-PISTON(2,2)
      PISTON(4,1)=XK(I)/XM2(I) ; PISTON(4,2)=B(I)/XM2(I)
      PISTON(4,3)=-PISTON(4,1) ; PISTON(4,4)=-PISTON(4,2)
      PISTON(2,5)= APT/XM1


***MULTIPLYING T                                    NEW WHOLE MATRIX BY G (GRAVITY)
****   BECAUSE OF MASS GIVEN IN lb

      DO (IJ1=1,5)
        PISTON(2,IJ1) = PISTON(2,IJ1)*32.2*12
        PISTON(4,IJ1) = PISTON(4,IJ1)*32.2*12
      REPEAT

**** CALCULATE PISTON DISPLACEMENT
**** INPUT TO PISTON EQUATION - DIFFERENTIAL PRESSURE

      UIN = CPRS(I)-RPRS(I)
      CALL EULER(PD,PDOT,4,PISTON,DT,UIN,I)

**** CALCULATE VOLUMES AND COMPRESSIBILITY FLOW

      V(I,1) = VO/2. + APT*PD(I,1)
      V(I,2) = VO/2. - APT*PD(I,1)
      Q6(I) = Q1(I) - Q3(I) - Q4(I) - APT*PD(I,2)
      Q7(I) = Q3(I) - Q2(I) - Q5(I) + APT*PD(I,2)

**** CALCULATE NEW PRESSURES

      CPRS(I) = CPRS(I) + (Q6(I)/V(I,1))*BETAO*DT
      RPRS(I) = RPRS(I) + (Q7(I)/V(I,2))*BETAO*DT

**** CHECK FOR SATURATION OF PRESSURES

      IF (CPRS(I) .GT. PS) CPRS(I) = PS
      IF (CPRS(I) .LT. PR) CPRS(I) = PR
      IF (RPRS(I) .GT. PS) RPRS(I) = PS
      IF (RPRS(I) .LT. PR) RPRS(I) = PR

**** NEW CYLINDER LENGTHS

                                  POS(I)=PD(I,1)

    REPEAT
    RETURN
    END

************************************************
******** SUBROUTINE MENU TO TOTAL SIMULATION
************************************************
    SUBROUTINE TSIMUL
```

```
**** DISPLAY MENU FOR TOTAL SIMULATION

    WHILE (1>0)
     WRITE(9,*)
     WRITE(9,*)
     WRITE(9,'("          SIMULATION  MENU     ")')
     WRITE(9,*)
     WRITE(9,*)
     WRITE(9,'("       0. RETURN TO PREVIOUS MENU     ")')
     WRITE(9,'("       1. INPUTS                      ")')
     WRITE(9,'("       2. EXECUTE                     ")')
     WRITE(9,'("       3. BASE MOVEMENT ANIMATION     ")')
     WRITE(9,'("       4. DISPLAY RESULTS             ")')
     WRITE(9,'("       5. CHANGE/VIEW PARAMETERS      ")')
     WRITE(9,*)
     WRITE(9,*)
     WRITE(9,'("       ENTER YOUR SELECTION       ")')
     WRITE(9,*)
     WRITE(9,*)

****READ                          SELECTION

      ICH = 1000
      READ(9,*) ICH
      SELECT CASE (ICH)
        CASE (0)
         EXIT
        CASE (1)
         CALL INPUTS
        CASE (2)
         CALL SIMULATE
        CASE (3)
         CALL ANIMATE
        CASE (4)
         CALL DISRES
        CASE (5)
         CALL CHGPAR
        CASE DEFAULT
         WRITE(9,'("ILLEGAL CHOICE")')
         WRITE(9,'("MAKE ANOTHER SELECTION")')
      END SELECT
     REPEAT
     RETURN
     END

************************************************
******** SUBROUTINE TO ENTER VALVE PARAMETERS
************************************************
     SUBROUTINE VALVE

     COMMON/SMVAL1/DAMPM,BANDM,GAINM,XM1,XM2,XK,B,VRST
     COMMON/SMVAL2/PS,PR
     COMMON/SMVAL3/XL1,XL2,XL3,APT,VO,XLIMIT,FRC,BETAO
```

```
      DIMENSION XM2(6),XK(6),B(6)

      WRITE(9,'("YOU HAVE SELECTED: "SET UP VALVE"")')
      WRITE(9,'("ENTER VALVE PARAMETERS")')
      WRITE(9,'("MOTORS                    PECIFICATIONS")')
      WRITE(9,'("DAMPING (<CR> TO RETURN): ")')
      DAMPM = -1
      READ(9,*) DAMPM
      WHILE ((DAMPM .LT. 0) .OR. (DAMPM .GT. 1.0))
       IF (DAMPM .EQ. -1) RETURN
       DAMPM = -1
       WRITE(9,*)
       WRITE(9,'("ILLEGAL DAMPING RATIO")')
       READ(9,*) DAMPM
      REPEAT
      WRITE(9,'("BANDWIDTH (Hz):")')
      READ(9,*) BANDM
      WRITE(9,'("GAIN (in/mA):")')
      READ(9,*) GAINM
      WRITE(9,'("RESISTANCE (ohms):")')
      READ(9,*) VRST
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("VALVE SPECIFICATIONS")')
      WRITE(9,'("SUPPLY PRESSURE (psi):")')
      READ(9,*) PS
      WRITE(9,'("RETURN PRESSURE (psi):")')
      READ(9,*) PR
      WRITE(9,'("LEAKAGE FLOW CONSTANTS (in3/(s*psi)):")')
      READ(9,*) XL1,XL2,XL3
      WRITE(9,'("AREA OF PISTON (in2):")')
      READ(9,*) APT
      WRITE(9,'("TOTAL EFFECTIVE VOLUME (in3):")')
      READ(9,*) VO
      WRITE(9,'("LIMIT OF SPOOL POSITION (in):")')
      READ(9,*)                    XLIMT
      WRITE(9,'("FLOW RATE CONSTANT:")')
      READ(9,*) FRC
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("OIL SPECIFICATIONS")')
      WRITE(9,'("BULK MODULUS (psi):")')
      READ(9,*) BETAO
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("MASS OF CYLINDER (lb):")')
      READ(9,*) XM1
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("MODEL OF MOTION BASE")')
      WRITE(9,'("MASS OF BASE FOR EACH CYLINDER (lb):")')
      READ(9,*) (XM2(I),I=1,6)
```

```
      WRITE(9,'("SPRING CONSTANT FOR EACH CYLINDER (lb/in):")')
      READ(9,*) (XK(I),I=1,6)
      WRITE(9,'("DAMPING COEFF. FOR EACH CYLINDER (lb/(in/s)):")')
      READ(9,*) (B(I),I=1,6)
      RETURN
      END

*************************************************
**** INCLUDE ROUTINES FOR GRAPHICS TERMINAL

      INCLUDE GRAPHLIB.SA

**** INCLUDE SPECIAL EFFECTS SET UP ROUTINES

      INCLUDE UFSPEFF.SA

**** INCLUDE MATRIX OPERATIONS ROUTINES

      INCLUDE MVUTIL.SA
      INCLUDE LINPAC.SA

**** INCLUDE PRIMARY MOTION SUBROUTINES

      INCLUDE SYS:65..PRIMFOR.SA
```

```
************************************************
******** PURPOSE: USER FRIENDLY FOR SPECIAL EFFECTS OF
********         DIGITAL MOTION BASE SYSTEM

************************************************
******** SUBROUTINE TO ADD GENERATORS
************************************************
   SUBROUTINE ADDGEN (J,NUMGEN)

**** J DEFINES NUMBER OF SPECIAL EFFECT

   COMMON/SMSPE1/MAXGEN,MAXPAR,SPECIAL
   COMMON/SMSPE3/NGEN,PAR,IPOINT
   INTEGER SPECIAL(240)
   DIMENSION NGEN(240),PAR(1000)
   CHARACTER CH*1

**** GENERATOR POINTER

   IGEN = MAXGEN
   XGEN = MAXGEN

**** PARAMETERS POINTER

   IPOINT = MAXPAR

**** TEST IF MAXIMUM NUMBER OF GENERATORS IS OBTAINED

   IF ((MAXGEN .GE. 240) .OR. (NUMGEN .GE. 12)) THEN
      WRITE(9,*)
      WRITE(9,'("NO MORE GENERATORS CAN BE ADDED")')
      WRITE(9,'("   <CR> TO CONTINUE")')
      PAUSE
      WRITE(9,*)
      RETURN
   ENDIF

**** DISPLAY MENU FOR TYPE OF GENERATOR

   WHILE (1>0)
   WRITE(9,*)
   WRITE(9,*)
   WRITE(9,'("      GENERATOR TYPE SELECTION MENU        ")')
   WRITE(9,*)
   WRITE(9,*)
   WRITE(9,'("        0. RETURN TO PREVIOUS MENU         ")')
 WRITE(9,'("    1.FILTERED WHITE NOISE(1ST                                    ORDER)")')
   WRITE(9,'("        2. FILTERED WHITE NOISE (2ND ORDER)")')
   WRITE(9,'("        3. SINE WAVE                       ")')
   WRITE(9,*)
   WRITE(9,*)
   WRITE(9,'("        ENTER YOUR SELECTION       ")')
   WRITE(9,*)
```

```
      WRITE(9,*)

**** DISPLAY MENU ON GRAPHICS

      CALL GRAPHADD(J)

**** READ SELECTION

      I = 1000
      READ(9,*) I

**** TYPE OF GENERATOR SELECTION

      SELECT CASE (I)
        CASE (0)

**** UPDATE MAX. PARAMETERS AND MAX. GENERATORS

          MAXGEN = IGEN
          MAXPAR = IPOINT
          EXIT
        CASE (1)

**** BLINK FIRST ORDER FILTER ON GRAPHICS

          CALL BLINKX(64,0,0,0,10,10)

**** INPUT PARAMETERS FOR FIRST ORDER FILTER

          CALL F1ADD(IGEN,J)

**** STOP BLINKING

          CALL BLINKX(64,0,0,0,0,0)
        CASE (2)

**** BLINK SECOND ORDER FILTER ON GRAPHICS

          CALL BLINKX(23,0,0,0,10,10)

****INPUTPARAMETER                              SFORSECONDORDERFILTER

          CALL F2ADD(IGEN,J)

**** STOP BLINKING

          CALL BLINKX(23,0,0,0,0,0)
        CASE (3)

**** BLINK SINE WAVE ON GRAPHICS

          CALL BLINKX(11,0,0,0,10,10)
```

```
**** INPUT PARAMETERS FOR SINE WAVE

        CALL SINEADD(IGEN,J)

**** STOP BLINKING

       CALL BLINKX(11,0,0,0,0,0)
      CASE DEFAULT
       WRITE(9,'("ILLEGAL CHOICE")')
       WRITE(9,'("MAKE ANOTHER SELECTION")')
   END SELECT

**** UPDATE NUMGEN

   IF (XGEN .NE. IGEN) THEN
     XGEN = XGEN + 1
     NUMGEN = NUMGEN + 1
   ENDIF

   IF (NUMGEN .GE. 12) THEN
      WRITE(9,*)
      WRITE(9,'("NO MORE GENERATORS CAN BE ADDED")')
      WRITE(9,'("   <CR> TO CONTINUE")')
      PAUSE
      WRITE(9,*)

**** UPDATE MAX. PARAMETERS AND MAX. GENERATORS

      MAXGEN = IGEN
      MAXPAR = IPOINT
      EXIT
   ENDIF
                    REPEAT
   RETURN
   END

************************************************
******* SUBROUTINE TO ADD A SPECIAL EFFECT
************************************************
   SUBROUTINE ADDSE

   COMMON/SMSPE2/IEFF,NAME,WADD
   INTEGER WADD(20)
   CHARACTER NAME(20)*20,C*20
   MAXSE = 20

**** TEST MAXIMUM NUMBER OF SPECIAL EFFECTS

   IF (IEFF .GE. MAXSE) THEN
      WRITE(9,*)
      WRITE(9,'("NO MORE SPECIAL EFFECTS CAN BE ADDED")')
      WRITE(9,'("   <CR> TO CONTINUE")')
      PAUSE
```

```
      WRITE(9,*)
      RETURN
    ENDIF

**** INPUT NAME OF SPECIAL EFFECT

    WRITE(9,*)
    WRITE(9,*)
    WRITE(9,*)
    WRITE(9,'("YOU HAVE SELECTED: "ADD A SPECIAL EFFECT"")')
    WRITE(9,'("  SPECIAL EFFECT NUMBER: ",I3)') IEFF+1
    WRITE(9,'("  ENTER NAME OF SPECIAL EFFECT (MAXIMUM 20 CHARARCT
+ERS): ")')
    WRITE(9,'("1___5____0____5____0   (<CR> TO RETURN TO SPECIAL E
+FFECTS MENU)")')
C='                    0
    READ(9,*) C
    IF (C(1:1) .EQ. '0') THEN
      RETURN
    ELSE
      IEFF = IEFF + 1
      NAME(IEFF) = C
    ENDIF
    WRITE(9,*)
    WRITE(9,*)
    WRITE(9,'("   WHERE TO ADD OUTPUT OF SPECIAL EFFECT")')
    WRITE(9,*)
    WRITE(9,'("    1. POSITION IN X AXIS")')
    WRITE(9,'("    2. POSITION IN Y AXIS")')
    WRITE(9,'("    3. POSITION IN Z AXIS")')
    WRITE(9,'("    4. PITCH ANGLE")')
    WRITE(9,'("    5. ROLL ANGLE")')
    WRITE(9,'("    6. YAW ANGLE")')
    WRITE(9,*)
    WRITE(9,*)
    WRITE(9,'("   ENTER SELECTION")')
    WADD(IEFF) = 10
    READ(9,*) WADD(IEFF)
    WHILE ((WADD(IEFF) .LT. 1) .OR. (WADD(IEFF) .GT. 6))
     WRITE(9,*)
     WRITE(9,'("WRONG SELECTION - SELECT AGAIN")')
     READ(9,*) WADD(IEFF)
    REPEAT

**** ADD GENERATORS FOR SPECIFIED SPECIAL EFFECT

    CALL ADDGEN (IEFF,0)
    RETURN
    END

*********************************************                    ***
******** SUBROUTINE TO CHANGE PARAMETERS OF A GENERATOR
*************************************************
```

```
      SUBROUTINE CHGEN

      COMMON/SMSPE3/NGEN,PAR,IPOINT
      COMMON/SMSPE5/KEEP,MAXKEEP
      DIMENSION NGEN(240),PAR(1000),KEEP(1000)

      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("YOU HAVE SELECTED: "CHANGE A GENERATOR"")')
      WRITE(9,'("ENTER NUMBER OF GENERATOR TO CHANGE:")')
      WRITE(9,'("  <CR> TO RETURN TO PREVIOUS MENU")')
      I = 0

***** TEST IF GENERATORS EXISTS

      READ(9,*) I
      WHILE ((I .LE. 0) .OR. (I .GT. MAXKEEP/2))
       IF (I .EQ. 0) RETURN
       I = 0
       WRITE(9,*)
       WRITE(9,*)
       WRITE(9,'("WRONG SELECTION - SELECT AGAIN")')
       READ(9,*) I
      REPEAT

***** INITIALIZE POINTERS

      IPT = KEEP(2*I)
      IP = KEEP(2*I-1)
      SELECT CASE (NGEN(IP))
       CASE (1)

***** FILTERED WHITE NOISE (1ST ORDER) - INPUT PARAMETERS

        WRITE(9,*)
      WRITE(9                          ,')
        WRITE(9,'("INPUT PARAMETERS FOR FILTERED WHITE NOISE (1ST
     +ORDER) GENERATOR")')
        WRITE(9,'("    <CR> FOR NO CHANGE")')
        WRITE(9,*)
        WRITE(9,'("   MEAN OF WHITE NOISE")')
        READ(9,*) PAR(IPT)
        WRITE(9,*)
        WRITE(9,'("   VARIANCE OF WHITE NOISE ")')
        READ(9,*) PAR(IPT+1)
        WRITE(9,*)
        WRITE(9,'("   BANDWIDTH OF FILTER (in Hz)")')
        READ(9,*) PAR(IPT+2)
        WRITE(9,*)
       CASE (2)

***** FILTERED WHITE NOISE (2ND ORDER) - INPUT PARAMETERS
```

```
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("INPUT PARAMETERS FOR FILTERED WHITE NOISE (2ND
 +ORDER) GENERATOR")')
      WRITE(9,'("    <CR> FOR NO CHANGE")')
      WRITE(9,*)
      WRITE(9,'("    MEAN OF WHITE NOISE")')
      READ(9,*) PAR(IPT)
      WRITE(9,*)
      WRITE(9,'("    VARIANCE OF WHITE NOISE")')
      READ(9,*) PAR(IPT+1)
      WRITE(9,*)
  W                                    RITE(9,'("  BANDWIDTH OF FILTER (nHz)")')
      READ(9,*) PAR(IPT+2)
      WRITE(9,*)
      WRITE(9,'("    DAMPING OF FILTER")')
      READ(9,*) PAR(IPT+3)
      WHILE ((PAR(IPT+3) .LT. 0) .OR. (PAR(IPT+3) .GT. 1))
       WRITE(9,*)
       WRITE(9,'("    INVALID DAMPING - ENTER AGAIN")')
       READ(9,*) DMP
      REPEAT
     CASE (3)

**** SINE WAVE - INPUT PARAMETERS

      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("INPUT PARAMETERS FOR SINE WAVE GENERATOR")')
      WRITE(9,'("    <CR> FOR NO CHANGE")')
      WRITE(9,*)
      WRITE(9,'("   AMPLITUDE OF SINE WAVE")')
      READ(9,*) PAR(IPT)
      WRITE(9,*)
      WRITE(9,'("   FREQUENCY OF SINE WAVE")')
      READ(9,*) PAR(IPT+1)
      WRITE(9,*)
   END SELECT
   RETURN
   END


************************************************
******** SUBROUTINE TO CHANGE A SPECIAL EFFECT                                    **********
**************************
   SUBROUTINE CHGSE

   COMMON/SMSPE1/MAXGEN,MAXPAR,SPECIAL
   COMMON/SMSPE2/IEFF,NAME,WADD
   COMMON/SMSPE3/NGEN,PAR,IPOINT
   COMMON/SMSPE5/KEEP,MAXKEEP
   DIMENSION NGEN(240),PAR(1000)
```

```
      DIMENSION KEEP(1000)
      INTEGER SPECIAL(240),WADD(20)
      CHARACTER NAME(20)*20,CH*1

***** READ NUMBER OF SPECIAL EFFECT

      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("YOU HAVE SELECTED: "CHANGE/VIEW A SPECIAL EFFECT"
     +        ")')
      WRITE(9,'("ENTER NUMBER OF SPECIAL EFFECT TO CHANGE/VIEW")')
      WRITE(9,'("  <CR> TO RETURN TO PREVIOUS MENU")')
      J = 0
      READ(9,*) J

***** TEST IF SPECIAL EFFECT EXISTS

      WHILE ((J .LE. 0) .OR. (J .GT. IEFF))
       IF (J .EQ. 0) RETURN
       J = 0
       WRITE(9,*)
       WRITE(9,*)
       WRITE(9,'("WRONG SELECTION - SELECT AGAIN")')
       READ(9,*) J
      REPEAT

***** INITIALIZING

      I = 1
      L = 0
      IPT = 1

WHILE(MAX                    GEN>0)

***** SELECTING INCREMENT TO POINTER

       SELECT CASE (NGEN(I))
         CASE (1)
          IIK = 3
         CASE (2)
          IIK = 4
         CASE (3)
          IIK = 2
       END SELECT

***** TEST IF GENERATOR BELONGS TO SPECIAL EFFECT

       IF (SPECIAL(I) .EQ. J) THEN

***** IF TRUE STORE POINTERS OF GENERATOR
*****   AND INCREMENT INDEX
```

```
      L = L + 2

****      INDEX OF GENERATOR

      KEEP(L-1) = I

****      POINTER TO PARAMETERS

      KEEP(L) = IPT
    ENDIF

**** INCREMENT POINTER

    IPT = IPT + IIK

**** TEST IF LAST GENERATOR

    IF (I .EQ. MAXGEN) THEN
      EXIT
    ELSE

**** IF NOT INCREMENT POINTER

      I = I + 1
    ENDIF
   REPEAT

**** STORE MAXIMUM NUMBERS OF GENERATORS IN SPECIAL EFFECT

   MAXKEEP = L

   WHILE (1>0)
    WRITE(9,*)
    WRITE(9,*)
    WRITE(9,*)
  WRITE(9,'(" MENU TO CHANGE V                                    IEW A SPECIAL EFFECT ")')
    WRITE(9,*)
    WRITE(9,*)
    WRITE(9,'("    0. RETURN TO PREVIOUS MENU      ")')
    WRITE(9,'("    1. ADD A GENERATOR              ")')
    WRITE(9,'("    2. DELETE A GENERATOR           ")')
    WRITE(9,'("    3. CHANGE A GENERATOR           ")')
    WRITE(9,'("    4. RESPONSE OF THIS SPECIAL EFFECT")')
    WRITE(9,'("    5. COPY OF GRAPHICS SCREEN      ")')
    WRITE(9,*)
    WRITE(9,*)
    WRITE(9,'("      ENTER YOUR SELECTION          ")')
    WRITE(9,*)
    WRITE(9,*)

**** LIST PARAMETERS OF GENERATORS ON GRAPHICS SCREEN

    CALL GRAGEN (J)
```

```
      ICH = 100
      READ(9,*) ICH
      SELECT CASE (ICH)
       CASE (0)
          EXIT
       CASE (1)

**** STORE MAXGEN AND MAXPAR BEFORE ADDING GENERATORS

          ILTGEN = MAXGEN
          IPNT = MAXPAR

**** ADD GENERATORS

     CALL ADDGEN(J,MAXK                                    EEP2)

**** TEST IF ANY GENERATOR WAS ADDED

          IF (ILTGEN+1 .LE. MAXGEN) THEN

**** STORE POINTERS OF GENERATORS ADDED

            DO (K = ILTGEN+1,MAXGEN)
              MAXKEEP = MAXKEEP + 1
              KEEP(MAXKEEP) = K
              SELECT CASE (NGEN(K))
                CASE (1)
                  MAXKEEP = MAXKEEP + 1
                  KEEP(MAXKEEP) = IPNT + 1
                  IPNT = IPNT + 3
                CASE (2)
                  MAXKEEP = MAXKEEP + 1
                  KEEP(MAXKEEP) = IPNT + 1
                  IPNT = IPNT + 4
                CASE (3)
                  MAXKEEP = MAXKEEP + 1
                  KEEP(MAXKEEP) = IPNT + 1
                  IPNT = IPNT + 2
              END SELECT
            REPEAT
          ENDIF
       CASE (2)
          CALL DELGEN
                          CASE(3)
          CALL CHGEN
       CASE (4)
          CALL RSPSE (J)
       CASE (5)
         CH = '_'
        WRITE(9,'("YOU HAVE SELECTED: "COPY GRAPHICS SCREEN"")
+             ')
        WRITE(9,'("  <CR> TO RETURN - ANYTHING TO CONTINUE")')
        READ(9,*) CH
```

```
      IF (CH .NE. '_') CALL PRNTTI
     CASE DEFAULT
      WRITE(9,'("ILLEGAL CHOICE")')
      WRITE(9,'("MAKE ANOTHER SELECTION")')
    END SELECT
   REPEAT
   RETURN
   END

************************************************
******* SUBROUTINE TO CHANGE NAME/WHERE TO ADD SPECIAL EFFECT
************************************************
   SUBROUTINE CHGWADD

   COMMON/SMSPE1/MAXGEN,MAXPAR,SPECIAL
   COMMON/SMSPE2/IEFF,NAME,WADD
   INTEGER SPECIAL(50),WADD(20)
   CHARACTER NAME(20)*20

**** INPUT NUMBER OF SPECIAL EFFECT TO CHANGE

   WRITE(9,*)
   WRITE(9,*)
   WRITE(9,*)
 WRITE(9,'(YOUHA                                    VESELECTED:"CHANGENAMEWHERETOADDSPECIA
+L EFFECT"")')
   WRITE(9,'("ENTER NUMBER OF SPECIAL EFFECT TO CHANGE")')
   WRITE(9,'(" <CR> TO RETURN TO SPECIAL EFFECTS MENU")')
   J = 0
   READ(9,*) J

**** TEST IF SPECIAL EFFECT EXISTS

   WHILE ((J .LE. 0) .OR. (J .GT. IEFF))
    IF (J .EQ. 0) RETURN
    J = 0
    WRITE(9,*)
    WRITE(9,'("  ILLEGAL SELECTION - SELECT AGAIN")')
    READ(9,*) J
   REPEAT

**** INPUT CORRECT NAME OF SPECIAL EFFECT

   WRITE(9,*)
   WRITE(9,'("  ENTER NAME OF SPECIAL EFFECT (MAXIMUM 20 CHARARCT
+ERS): ")')
   WRITE(9,'("1___5____0____5____0    (<CR> FOR NO CHANGE)")')
   READ(9,*) NAME(J)
   WRITE(9,*)
   WRITE(9,*)
   WRITE(9,'("   WHERE TO ADD OUTPUT OF SPECIAL EFFECT")')
   WRITE(9,'("    1. POSITION IN X AXIS")')
   WRITE(9,'("    2. POSITION IN Y AXIS")')
```

```
      WRITE(9,'("     3. POSITION IN Z AXIS")')
      WRITE(9,'("  4. PITCH ANGLE")')
      WRITE(9,'("     5. ROLL ANGLE")')
      WRITE(9,'("     6. YAW ANGLE")')
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("   ENTER SELECTION")')
      WRITE(9,'("  <CR> FOR NO CHANGE")')
      WADD(J) = 10
      READ(9,*) WADD(J)
      WHILE ((WADD(J) .LT. 1) .OR. (WADD(J) .GT. 6))
       WRITE(9,*)
       WRITE(9,'("WRONG SELECTION - SELECT AGAIN")')
       READ(9,*) WADD(J)
      REPEAT
      RETURN
      END

*************************************************
******* SUBROUTINE TO DELETE A GENERATOR
*************************************************
      SUBROUTINE DELGEN

      COMMON/SMSPE1/MAXGEN,MAXPAR,SPECIAL
      COMMON/SMSPE3/NGEN,PAR,IPOINT
      COMMON/SMSPE5/KEEP,MAXKEEP
      DIMENSION NGEN(240),PAR(1000)
      DIMENSION KEEP(1000)
      INTEGER SPECIAL(240)

**** ENTER GENERATOR TO BE DELETED

      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("YOU HAVE SELECTED: "DELETE A GENERATOR"")')
      WRITE(9,'("ENTER NUMBER OF GENERATOR TO DELETE")')
      WRITE(9,'("  <CR> TO RETURN TO PREVIOUS MENU")')
      I = 0

**** TEST IF GENERATOR EXISTS
      READ(9,*) I
      WHILE ((I .LE. 0) .OR. (I .GT. MAXKEEP/2))
       IF (I .EQ. 0) RETURN
       I = 0
       WRITE(9,*)
       WRITE(9,*)
       WRITE(9,'("WRONG SELECTION - SELECT AGAIN")')
       READ(9,*) I
      REPEAT

**** DECREMENT NUMBER OF GENERATORS
```

```
      MAXGEN = MAXGEN - 1

**** POINTER OF GENERATOR

      IP = KEEP(2*I-1)

**** SELECT DECREMENT OF PARAMETERS

      SELECT CASE (NGEN(IP))
       CASE (1)
        IIK = 3
       CASE (2)
        IIK = 4
       CASE (3)
        IIK = 2
      END SELECT
      MAXPAR = MAXPAR - IIK

**** TEST IF LAST GENERATOR

      IF (I .NE. MAXGEN+1) THEN

**** IF NOT MOVE PARAMETERS UP

        DO (K = KEEP(2*I),MAXPAR)
          PAR(K) = PAR(K+IIK)
        REPEAT
        DO (K = IP,MAXGEN)
          SPECIAL(K) = SPECIAL(K+1)
    NGEN(K)                              =NGEN(K+1)
        REPEAT
      ENDIF

**** TEST IF LAST GENERATOR

      IF (2*I .NE. MAXKEEP) THEN

**** IF NOT MOVE PARAMETERS UP
        DO (K = 2*I,MAXKEEP-2,2)
          KEEP(K-1) = KEEP(K+1) - 1
          KEEP(K) = KEEP(K+2) - IIK
        REPEAT
      ENDIF

**** DECREMENT NUMBER OF PARAMETERS STORED FOR GENERATOR

      MAXKEEP = MAXKEEP - 2
      RETURN
       END

*************************************************
******* SUBROUTINE TO DELETE A SPECIAL EFFECT
*************************************************
```

```
      SUBROUTINE DELSE

      COMMON/SMSPE1/MAXGEN,MAXPAR,SPECIAL
      COMMON/SMSPE2/IEFF,NAME,WADD
      COMMON/SMSPE3/NGEN,PAR,IPOINT
      DIMENSION NGEN(240),PAR(1000)
      INTEGER SPECIAL(240),WADD(20)
      CHARACTER NAME(20)*20

**** INITIALIZE POINTER

      IPT = 1
      I = 1

**** READ NUMBER OF SPECIAL EFFECT

      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("YOU HAVE SELECTED: "DELETE A SPECIAL EFFECT"")')
                                        WRITE(9,'(ENTER NUMBER OF SPECIAL EFFECT TO DELETE")
      WRITE(9,'(" <CR> TO RETURN TO SPECIAL EFFECTS MENU")')
      J = 0
      READ(9,*) J

**** TEST IF SPECIAL EFFECT EXISTS

      WHILE ((J .LE. 0) .OR. (J .GT. IEFF))
       IF (J .EQ. 0) RETURN
       J = 0
       WRITE(9,*)
       WRITE(9,*)
       WRITE(9,'("WRONG SELECTION - SELECT AGAIN")')
       READ(9,*) J
      REPEAT

**** MOVE NAMES AND WHERE TO ADD UP

      DO (K = J+1,IEFF)
        NAME(K-1) = NAME(K)
        WADD(K-1) = WADD(K)
      REPEAT

**** DECREMENT NUMBER OF SPECIAL EFFECTS

      IEFF = IEFF - 1

      WHILE (1>0)

**** SELECT DECREMENT OR INCREMENT IN NUMBER OF PARAMETERS

       SELECT CASE (NGEN(I))
        CASE (1)
         IIK = 3
```

```
      CASE (2)
        IIK = 4
      CASE (3)
        IIK = 2
    END SELECT

****TEST IF GENERATOR BELONGS TO SPECIAL EFFECT SPECIFIED

    IF (SPECIAL(I) .EQ. J) THEN

****IF TRUE DECREMENT NUMBER OF GENERATORS

      MAXGEN = MAXGEN - 1

**** DECREMENT NUMBER OF PARAMETERS

      MAXPAR = MAXPAR - IIK

**** TEST IF LAST GENERATOR

      IF (I .EQ. MAXGEN+1) THEN
        EXIT
      ELSE

**** IF NOT TRUE MOVE PARAMETERS UP

        DO (K = IPT,MAXPAR)
          PAR(K) = PAR(K+IIK)
        REPEAT
        DO (K = I,MAXGEN)
          SPECIAL(K) = SPECIAL(K+1)
          NGEN(K) = NGEN(K+1)
        REPEAT
      ENDIF
    ELSE

**** IF NOT TRUE SKIP PARAMETERS
**** INCREMENT POINTER

      IPT = IPT + IIK

**** TEST IF BELONGS TO A GREATER VALUE OF SPECIAL EFFECT

      IF (SPECIAL(I) .GT. J) THEN

**** SUBTRACT TO MAKE IT BELONG TO CORRECT SPECIAL EFFECT

        SPECIAL(I) = SPECIAL(I) - 1
      ENDIF

**** TEST IF LAST GENERATOR

      IF (I .EQ. MAXGEN) THEN
```

```
          EXIT
        ELSE

***INCR                          EVENTPOINTERFORGENERATORS

          I = I + 1
        ENDIF
      ENDIF
    REPEAT
    RETURN
    END

************************************************
******** SUBROUTINE TO INPUT PARAMETERS FOR
********   FIRST ORDER FILTER GENERATOR
************************************************
    SUBROUTINE F1ADD (IGEN,J)

    COMMON/SMSPE1/MAXGEN,MAXPAR,SPECIAL
    COMMON/SMSPE3/NGEN,PAR,IPOINT
    DIMENSION NGEN(240),PAR(1000)
    INTEGER SPECIAL(240)
    REAL MEAN

**** INPUT PARAMETERS

    WRITE(9,*)
    WRITE(9,*)
    WRITE(9,'("YOU HAVE SELECTED: "FILTERED WHITE NOISE (1ST ORDER
+) GENERATOR"")')
    WRITE(9,'("INPUT PARAMETERS  (<CR> TO RETURN TO PREVIOUS MENU)
+         ")')
    WRITE(9,*)
    WRITE(9,'("   MEAN OF WHITE NOISE")')
    MEAN = 1000
    READ(9,*) MEAN
    IF (MEAN .EQ. 1000) RETURN
    PAR(IPOINT+1) = MEAN
    WRITE(9,*)
    WRITE(9,'("   VARIANCE OF WHITE NOISE")')
READ(9,*                                  )PAR(IPOINT+2)
    WRITE(9,*)
    WRITE(9,'("   BANDWIDTH OF FILTER (in Hz)")')
    READ(9,*) PAR(IPOINT+3)
    WRITE(9,*)
    IPOINT = IPOINT + 3

**** INCREMENT NUMBER OF GENERATORS

    IGEN = IGEN + 1

**** SPECIFY WHICH SPECIAL EFFECT IT BELONGS TO
```

```
      SPECIAL(IGEN) = J

**** TYPE OF GENERATOR

      NGEN(IGEN) = 1
      RETURN
      END

************************************************
******* SUBROUTINE TO CALCULATE AMPLITUDE FOR
*******     FIRST ORDER FILTER GENERATOR
************************************************
      SUBROUTINE F1AMP (IPT)

      COMMON/SMSPE3/NGEN,PAR,IPOINT
      COMMON/SMSPE4/MAXPSTO,IPTR,PSTO
      COMMON/SMSPE6/YAMP,DTSPEF
      DIMENSION NGEN(240),PAR(1000)
      DIMENSION PSTO(1000)
      REAL M
      DATA PI/3.1416/

**** MEAN OF WHITE NOISE

      M = PAR(IPT)

**** VARIANCE OF WHITE NOISE

      V = SQRT(12*PAR(IPT+1))

**** DECAY

      A = EXP(-DTSPEF*2*PI*PAR(IPT+2))
B=1-A

**** INCREMENT POINTER

      IPT = IPT + 3

**** GET LAST VALUE CALCULATED

      X1 = PSTO(IPTR+1)

**** AMPLITUDE

      YAMP = X1

**** CALCULATE WHITE NOISE

      CALL WHITE (U,M,V)

**** CALCULATE UPDATE VALUE FOR FIRST ORDER FILTER
```

```
      X1 = A*X1 + B*U

**** STORE LAST VALUE CALCULATED

      PSTO(IPTR+1) = X1

**** INCREMENT POINTER

      IPTR = IPTR + 2
      RETURN
      END

************************************************
******* SUBROUTINE TO INPUT PARAMETERS FOR
*******    SECOND ORDER FILTER GENERATOR
************************************************
      SUBROUTINE F2ADD (IGEN,J)

      COMMON/SMSPE1/MAXGEN,MAXPAR,SPECIAL
      COMMON/SMSPE3/NGEN,PAR,IPOINT
      DIMENSION NGEN(240),PAR(1000)
      INTEGER SPECIAL(240)
      REAL MEAN

**** INPUT PARAMETERS

      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("YOU HAVE SELECTED: "FILTERED WHITE NOISE (2ND ORDER
     +) GENERATOR"")')
      WRITE(9,'("INPUT PARAMETE                                    RS (<CR> TO RETURN TO PREVIOUS MENU)
     +         ")')
      WRITE(9,*)
      WRITE(9,'("    MEAN OF WHITE NOISE")')
      MEAN = 1000
      READ(9,*) MEAN
      IF (MEAN .EQ. 1000) RETURN
      PAR(IPOINT+1) = MEAN

      WRITE(9,*)
      WRITE(9,'("    VARIANCE OF WHITE NOISE")')
      READ(9,*) PAR(IPOINT+2)
      WRITE(9,*)
      WRITE(9,'("    BANDWIDTH OF FILTER (in Hz)")')
      READ(9,*) PAR(IPOINT+3)
      WRITE(9,*)
      WRITE(9,'("    DAMPING OF FILTER")')
      READ(9,*) DMP
      WHILE ((DMP .LT. 0) .OR. (DMP .GT. 1))
       WRITE(9,*)
       WRITE(9,'("    INVALID DAMPING - ENTER AGAIN")')
       READ(9,*) DMP
      REPEAT
```

```
      PAR(IPOINT+4) = DMP
      IPOINT = IPOINT + 4

**** INCREMENT NUMBER OF GENERATORS

      IGEN = IGEN + 1

**** SPECIFY WHICH SPECIAL EFFECT IT BELONGS TO

      SPECIAL(IGEN) = J

**** TYPE OF GENERATOR

      NGEN(IGEN) = 2
      RETURN
      END
```

```
************************************************
******* SUBROUTINE TO CALCULATE AMPLITUDE FOR
*******     SECOND ORDER FILTER GENERATOR
************************************************
      SUBROUTINE F2AMP (IPT)

      COMMON/SMSPE3/NGEN,PAR,IPOINT
      COMMON/SMSPE4/MAXPSTO,IPTR,PSTO
      COMMON/SMSPE6/YAMP,DTSPEF
      DIMENSION NGEN(240),PAR(1000)
      DIMENSION PSTO(1000)
      DIMENSION AD(2,2),BD(2)
      REAL M
      DATA PI/3.1416/

**** MEAN OF WHITE NOISE

      M = PAR(IPT)

**** VARIANCE OF WHITE NOISE

      V = SQRT(12*PAR(IPT+1))

**** BANDWIDTH

      BW = 2*PI*PAR(IPT+2)

**** DAMPING

      Z = PAR(IPT+3)

**** INCREMENT POINTER

      IPT = IPT + 4

      ALPHA = Z*BW
      OMEGA = BW*SQRT(1-Z**2)
      TEMP = EXP(-ALPHA*DTSPEF)

**** COEFFICIENTS FOR SECOND ORDER FILTER

      IF (OMEGA .NE. 0) THEN
       AD(1,1)=(COS(OMEGA*DTSPEF)
     +       -(ALPHA/OMEGA)*SIN(OMEGA*DTSPEF))*TEMP
       AD(2,1)=-(1/OMEGA)*SIN(OMEGA*DTSPEF)*TEMP
       AD(1,2)=-AD(2,1)*(BW**2)
       AD(2,2)=(COS(OMEGA*DTSPEF)
     +       +(ALPHA/OMEGA)*SIN(OMEGA*DTSPEF))*TEMP
       BD(1)=-(AD(1,1)-1+2*(ALPHA/(BW**2))*AD(1,2))
       BD(2)=-(AD(2,1)+2*(ALPHA/(BW**2))*(AD(2,2)-1))
      ELSE
       AD(1,1)=(1-BW*DTSPEF)*TEMP
       AD(2,1)=-DTSPEF*TEMP
```

```
      AD(1,2)=-AD(2,1)*(BW**2)
      AD(2,2)=(1+BW*DTSPEF)*TEMP
      BD(1)=-(AD(1,1)-1+2*(Z/BW)*AD(1,2))
      BD(2)=-(AD(2,1)+2*(Z/BW)*(AD(2,2)-1))
     ENDIF

*                              **GETLASTVALUECALCULATED

     X1 = PSTO(IPTR+1)
     X2 = PSTO(IPTR+2)

**** AMPLITUDE

     YAMP = X1

**** CALCULATE WHITE NOISE

     CALL WHITE (U,M,V)

**** CALCULATE UPDATE VALUE FOR SECOND ORDER FILTER

     XU1 = AD(1,1)*X1 + AD(1,2)*X2 + BD(1)*U
     XU2 = AD(2,1)*X1 + AD(2,2)*X2 + BD(2)*U

**** STORE LAST VALUE CALCULATED

     PSTO(IPTR+1) = XU1
     PSTO(IPTR+2) = XU2

**** INCREMENT POINTER

     IPTR = IPTR + 3
     RETURN
     END

************************************************
******* SUBROUTINE TO DISPLAY PARAMETERS OF
*******    GENERATORS ON GRAPHICS
************************************************
     SUBROUTINE GRAGEN (J)

     COMMON/SMSPE2/IEFF,NAME,WADD
     COMMON/SMSPE3/NGEN,PAR,IPOINT
     COMMON/SMSPE5/KEEP,MAXKEEP
     DIMENSION NGEN(240),PAR(1000)
     DIMENSION KEEP(1000)
     CHARACTER NAME(20)*20,CK*4,CKT*2,TEMP*31,CM*12
     CHARACTER TWADD(6)*10
     REAL MEAN
     INTEGER WADD(20)
                                    DATA/YC,IY,IX,ISIZE,IXX/13,420,0,1040/

**** INITIALIZE TABLE TWADD
```

```
      TWADD(1) = 'POS X AXIS'
      TWADD(2) = 'POS Y AXIS'
      TWADD(3) = 'POS Z AXIS'
      TWADD(4) = 'PITCH'
      TWADD(5) = 'ROLL'
      TWADD(6) = 'YAW'

      CALL PLTINIT
      CALL COLOR(180)
      CALL SMOVE(10,450)

**** DISPLAY SPECIAL EFFECT NAME AND WHERE TO ADD

      CALL TEXT(NAME(J),0,0,20)
      CALL SMOVE(10,435)
      CALL COLOR(197)
      TEMP = 'ADD OUTPUT AT: '//TWADD(WADD(J))
      CALL TEXT(TEMP,0,0,15)
      CALL COLOR(180)
      CALL SMOVE(0,433)
      CALL SDRAW(630,433)

**** TEST FOR GENERATORS

      IF (MAXKEEP .EQ. 0) THEN
        CALL COLOR(104)
        CALL SMOVE(0,250)
        CALL TEXT('NO GENERATORS',0,0,20)
      ELSE

**** DISPLAY PARAMETERS OF GENERATORS

        DO (I = 1,MAXKEEP-1,2)
          K = (I+1)/2
          J1 = (-1)**K
          IF (J1 .LT. 0) THEN

**** ODD J - COLOR (GREEN)

            ICOL = 104
                        ELSE

**** EVEN J - COLOR (BLUE)

            ICOL = 101
          ENDIF
          CALL COLOR(ICOL)

**** INITIALIZE POINTERS

          IPT = KEEP(I+1)
          IP = KEEP(I)
```

```
        SELECT CASE (NGEN(IP))
         CASE (1)

**** TEST IF END OF GRAPHICS LINE

          IF ((IY-4*IYC) .LE. -12) THEN

**** IF TRUE LIST PARAMETERS ON RIGHT HALF OF SCREEN

             IX = 319
             IXX = 359
             IY = 420
          ENDIF

**** DISPLAY PARAMETERS

          MEAN = PAR(IPT)
          VAR = PAR(IPT+1)
          BDW = PAR(IPT+2)
          CALL ITOASC(K,CK)
          CKT = CK(3:4)
          TEMP = CKT//'. FILT. WHITE NOISE-1ST ORD'
          CALL SMOVE(IX,IY)
          CALL TEXT(TEMP,0,0,ISIZE)
          IY = IY - IYC
          CALL REALTO_ASCII(MEAN,CM)
          TEMP = 'MEAN =      '//CM
   CALL SMOVE(IXX                                  Y)
          CALL TEXT(TEMP,0,0,ISIZE)
          IY = IY - IYC
          CALL REALTO_ASCII(VAR,CM)
          TEMP = 'VARIANCE =  '//CM
          CALL SMOVE(IXX,IY)
          CALL TEXT(TEMP,0,0,ISIZE)
          IY = IY - IYC
          CALL REALTO_ASCII(BDW,CM)
          TEMP = 'BANDWIDTH = '//CM
          CALL SMOVE(IXX,IY)
          CALL TEXT(TEMP,0,0,ISIZE)
          IY = IY - IYC - 4
         CASE(2)

**** TEST IF END OF GRAPHICS LINE

          IF ((IY-5*IYC) .LE. -12) THEN

**** IF TRUE LIST PARAMETERS ON RIGHT HALF OF SCREEN

             IX = 319
             IXX = 359
             IY = 420
          ENDIF
```

```
**** DISPLAY PARAMETERS

        MEAN = PAR(IPT)
        VAR = PAR(IPT+1)
        BDW = PAR(IPT+2)
        DMP = PAR(IPT+3)
        CALL ITOASC(K,CK)
        CKT = CK(3:4)
    TEMP=CKT//.                                   FILT.WHITENOISE.2NDORD'
        CALL SMOVE(IX,IY)
        CALL TEXT(TEMP,0,0,ISIZE)
        IY = IY - IYC
        CALL REALTO_ASCII(MEAN,CM)
        TEMP = 'MEAN =     '//CM
        CALL SMOVE(IXX,IY)
        CALL TEXT(TEMP,0,0,ISIZE)
        IY = IY - IYC
        CALL REALTO_ASCII(VAR,CM)
        TEMP = 'VARIANCE =  '//CM
        CALL SMOVE(IXX,IY)
        CALL TEXT(TEMP,0,0,ISIZE)
        IY = IY - IYC
        CALL REALTO_ASCII(BDW,CM)
        TEMP = 'BANDWIDTH = '//CM
        CALL SMOVE(IXX,IY)
        CALL TEXT(TEMP,0,0,ISIZE)
        IY = IY - IYC
        CALL REALTO_ASCII(DMP,CM)
        TEMP = 'DAMPING =   '//CM
        CALL SMOVE(IXX,IY)
        CALL TEXT(TEMP,0,0,ISIZE)
        IY = IY - IYC - 4
      CASE (3)

**** TEST IF END OF GRAPHICS LINE

    IF((IY-3*IYC).LE.                             .-12)THEN

**** IF TRUE LIST PARAMETERS ON RIGHT HALF OF SCREEN

        IX = 319
        IXX = 359
        IY = 420
      ENDIF

**** DISPLAY PARAMETERS

        AMPL = PAR(IPT)
        FREQ = PAR(IPT+1)
        CALL ITOASC(K,CK)
        CKT = CK(3:4)
        TEMP = CKT//'. SINE WAVE'
        CALL SMOVE(IX,IY)
```

```
            CALL TEXT(TEMP,0,0,ISIZE)
            IY = IY - IYC
            CALL REALTO_ASCII(AMPL,CM)
            TEMP = 'AMPLITUDE = '//CM
            CALL SMOVE(IXX,IY)
            CALL TEXT(TEMP,0,0,ISIZE)
            IY = IY - IYC
            CALL REALTO_ASCII(FREQ,CM)
            TEMP = 'FREQUENCY = '//CM
            CALL SMOVE(IXX,IY)
            CALL TEXT(TEMP,0,0,ISIZE)
            IY = IY - IYC - 4
          END SELECT
        REPEAT
      ENDIF
      RETURN
      END

************************************                                    **********
******* SUBROUTINE TO LIST CURRENT SPECIAL EFFECTS ON GRAPHICS
************************************************
      SUBROUTINE GRAMENU

      COMMON/SMSPE2/IEFF,NAME,WADD
      CHARACTER NAME(20)*20,TEMP*35,CI*4,CIT*2,TWADD(6)*5
      INTEGER WADD(20)

**** INITIALIZE TABLE TWADD

      TWADD(1) = 'POS X'
      TWADD(2) = 'POS Y'
      TWADD(3) = 'POS Z'
      TWADD(4) = 'PITCH'
      TWADD(5) = 'ROLL'
      TWADD(6) = 'YAW '

**** CLEAR SCREEN

      CALL PLTINIT

**** TEST IF THERE ARE ANY SPECIAL EFFECTS

      IF (IEFF .EQ. 0) THEN
        CALL COLOR(104)
        CALL SMOVE(10,250)
        CALL TEXT('NO CURRENT SPECIAL EFFECTS',0,0,20)
      ELSE

**** DISPLAY CURRENT SPECIAL EFFECTS

        CALL COLOR(179)
        CALL SMOVE(30,430)
        CALL TEXT('CURRENT SPECIAL EFFECTS',0,0,20)
```

```
      CALL SMOVE(10,420)
      CALL SDRAW(610,420)
      CALL SMOVE(10,419)
      CALL SDRAW(610,419)
      CALL SMOVE(10,418)
                                CALLSDRAW(610,418)
      IY = 390
      CALL COLOR(104)
      DO (I = 1,IEFF)
       CALL ITOASC(I,CI)
       CIT = CI(3:4)
       CALL SMOVE(20,IY)
       TEMP = CIT//'. '//NAME(I)//'/'AT: '//TWADD(WADD(I))
       CALL TEXT(TEMP,0,0,15)
       IY = IY - 20
      REPEAT
    ENDIF
    RETURN
    END

************************************************
******* MENU FOR GRAPHICS TO ADD GENERATORS
************************************************
    SUBROUTINE GRAPHADD (J)

    COMMON/SMSPE2/IEFF,NAME,WADD
    CHARACTER NAME(20)*20,TEMP*36
    INTEGER WADD(20)

**** DISPLAYS MENU FOR TYPE OF GENERATOR ON GRAPHICS MONITOR
**** CLEAR SCREEN

    CALL PLTINIT

    CALL COLOR(180)
    CALL SMOVE(100,420)
    CALL TEXT('ADD GENERATORS',0,0,20)

**** WRITE NAME OF SPECIAL EFFECT THAT IS ADDING GENERATORS

    CALL COLOR(209)
    CALL SMOVE(100,390)
    TEMP = 'SPECIAL EFFECT: '//NAME(J)
                                CALLTEXT(TEMP,0,0,15)

**** DRAW A RECTANGULE

    CALL PRMFIL(0)
    CALL COLOR(31)
    CALL SMOVE(0,145)
    CALL SRECT(550,350)
    CALL SMOVE(1,146)
    CALL SRECT(549,349)
```

```
      CALL SMOVE(2,147)
      CALL SRECT(548,348)

***** DRAW A RECTANGULE FOR FILTERED WHITE NOISE (1ST ORDER)

      CALL SMOVE(10,300)
      CALL COLOR(179)
      CALL TEXT('1. ',0,0,10)
      CALL SMOVE(40,200)
      CALL COLOR(64)
      CALL SRECT(140,300)
      CALL SMOVE(41,201)
      CALL SRECT(139,299)

***** DRAWS A UNIT STEP RESPONSE

      CALL COLOR(179)
      CALL SMOVE(46,211)
      CALL SDRAWR(0,78)
      CALL SMOVE(46,221)
      CALL SDRAWR(88,0)
      CALL SMOVE(46,221)
      DO (IT = 0,88)
        Y = EXP(-0.1*IT)
        IY = 221 + NINT(Y*68)
        IX = 46 + IT
        CALL SDRAW(IX,IY)
      REPEAT

***** TITLE OF ABOVE RECTANGULE

      CALL COLOR(215)
      CALL SMOVE(50,180)
      CALL TEXT('FILTERED',0,0,10)
      CALL SMOVE(35,168)
CALL TEXT('WH                                    ITE NOISE',0,0,10)
      CALL SMOVE(45,156)
      CALL TEXT('1ST ORDER',0,0,10)

***** DRAW A RECTANGULE FOR FILTERED WHITE NOISE (2ND ORDER)

      CALL SMOVE(190,300)
      CALL COLOR(179)
      CALL TEXT('2. ',0,0,10)
      CALL SMOVE(220,200)
      CALL COLOR(23)
      CALL SRECT(320,300)
      CALL SMOVE(221,201)
      CALL SRECT(319,299)

***** DRAWS A UNIT STEP RESPONSE

      CALL COLOR(179)
```

```
      CALL SMOVE(226,211)
      CALL SDRAWR(0,78)
      CALL SMOVE(226,250)
      CALL SDRAWR(88,0)
      CALL SMOVE(226,250)
      DO (IT = 0,88)
        Y = EXP(-0.05*IT)*COS(2*3.1416*.1*IT)
        IY = 250 + NINT(Y*39)
        IX = 226 + IT
        CALL SDRAW(IX,IY)
      REPEAT

**** TITLE OF ABOVE RECTANGULE

      CALL COLOR(215)
      CALL SMOVE(230,180)
      CALL TEXT('FILTERED',0,0,10)
      CALL SMOVE(215,168)
      CALL TEXT('WHITE NOISE',0,0,10)
      CALL SMOVE(225,156)
      CALL TEXT('2ND ORDER',0,0,10)

**** DRAW A RECTANGULE FOR SINE WAVE

CALL                          SMOVE(370,300)
      CALL COLOR(179)
      CALL TEXT('3. ',0,0,10)
      CALL SMOVE(400,200)
      CALL COLOR(11)
      CALL SRECT(500,300)
      CALL SMOVE(401,201)
      CALL SRECT(499,299)

**** DRAWS A SINE WAVE

      CALL COLOR(179)
      CALL SMOVE(406,211)
      CALL SDRAWR(0,78)
      CALL SMOVE(406,250)
      CALL SDRAWR(88,0)
      CALL SMOVE(406,250)
      DO (IT = 0,88)
        Y = SIN(2*3.1416*0.05*IT)
        IY = 250 + NINT(Y*39)
        IX = 406 + IT
        CALL SDRAW(IX,IY)
      REPEAT

**** TITLE OF ABOVE RECTANGULE

      CALL COLOR(215)
      CALL SMOVE(405,168)
      CALL TEXT('SINE WAVE',0,0,10)
```

```
      CALL PRMFIL(1)
      RETURN
      END

*************************************************
[B*
******   ROUTINE TO READ IN SAMPLING RATE
******
******   READS IN FREQ, THEN CALCULATES TIMER2,PRESCA2,DTF
******   DTF HAS THE FIRST POSSIBLE VALUE GREATER THAN DT
******
*************************************************
 SUBROUTINER_                                  1FREQ(DT,TIMER,PRESCA)A_FLG)

      INTEGER*1 TIMER,PRESCA

      CHARACTER CHF*1

      INTEGER PS,PSX


*     enter the # of interrupt cycles, repeatition rate (main count &
*      prescaler). Rate= 1.23 MHz/ (Maincount*Prescaler)
*
*     maximum main count=255
*          prescaler= 4     for psch=1
*                   =10          =2
*                   =16          =3
*                   =50          =4
*                   =64          =5
*                   =100          =6
*                   =200          =7

      IF(IA_FLG.EQ.1)THEN
        WRITE(9,'(" ENTER DESIRED ITERATION RATE IN Hz (EX.1000)")')
        FREQ=1.0/DT
        READ(9,*)FREQ
        DT=1/(FREQ)
       ENDIF
       DTF=0

       PRESCA=1
       ITIME=0
       WHILE(DT-DTF.GT.1.5E-6)

         ITIME=ITIME+1          ! INCREMENT    MAIN COUNTER
    IF(ITIME.EQ.256)THEN !MAINCOUNTEROVERFLOWED,                                  RESETIT
          ITIME=1           !
          PRESCA=PRESCA+1     !        INCREMENT TIMER
          IF(PRESCA.EQ.8)THEN
           WRITE(9,'("CAN NOT HAVE SUCH A SLOW SAMPLING RATE")')
           IF(IA_FLG.EQ.1)THEN
```

```
            GOTO 50
           ELSE
            TIMER=0
            PRESCA=0
            RETURN
           ENDIF
          ENDIF
         ENDIF
         PSX=PRESCA

         IF(PSX.EQ.1) PS=4
         IF(PSX.EQ.2) PS=10
         IF(PSX.EQ.3) PS=16
         IF(PSX.EQ.4) PS=50
         IF(PSX.EQ.5) PS=64
         IF(PSX.EQ.6) PS=100
         IF(PSX.EQ.7) PS=200

         DTF=ITIME*PS/1.23E+6

       REPEAT

       DT=DTF
       TIMER=ITIME
       FREQ=1/(DT)

       IF(IA_FLG.EQ.1)THEN
        WRITE(9,'(" SAMPLING RATE",G12.4," Hz")')FREQ
        WRITE(9,'(" DT",G12.4," SEC")')DT
  WRIT                                          E(9,(" TIMER PRESCA",2I5))TIMER,PRESCA
        WRITE(9,'(" PRESS <CR> TO CONTINUE OR "R" TO RE-ENTER FREQ
     +          ")')
        CHF=' '
        READ(9,*)CHF
        IF((CHF.EQ.'R').OR.(CHF.EQ.'r')) GOTO 50
       ENDIF

       RETURN

       END

*****************************************************
******* SUBROUTINE FOR THE RESPONSE OF A SPECIAL EFFECT
*****************************************************
   SUBROUTINE RSPSE (J)

*** J DEFINES NUMBER OF SPECIAL EFFECT,
***   IF ZERO THEN SPECIAL EFFECT IS NOT CHOSEN YET

   COMMON/SMSPE1/MAXGEN,MAXPAR,SPECIAL
   COMMON/SMSPE2/IEFF,NAME,WADD
   COMMON/SMSPE3/NGEN,PAR,IPOINT
   COMMON/SMSPE4/MAXPSTO,IPTR,PSTO
```

```
      COMMON/SMSPE5/KEEP,MAXKEEP
      COMMON/SMSPE6/YAMP,DTSPEF
      DIMENSION NGEN(240),PAR(1000)
      DIMENSION PSTO(1000),PLOT1(2000,2),KEEP(1000)
      INTEGER SPECIAL(240),WADD(20)
      CHARACTER NAME(20)*20,CH*1
     CHARACTER VNAME*8,HEADER*20,XLABEL*4,Y                                    LABEL*6,VSYMBOL*1

**** ENTER NUMBER OF SPECIAL EFFECT TO SEE OUTPUT

      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("YOU HAVE SELECTED: "RESPONSE OF A SPECIAL EFFECT"
     +         ")')
      IF (J .EQ. 0) THEN
        WRITE(9,'("ENTER NUMBER OF SPECIAL EFFECT YOU WANT TO SEE
     +           ")')
        WRITE(9,'("   <CR> TO RETURN TO PREVIOUS MENU")')
        READ(9,*) J

**** TEST IF SPECIAL EFFECT EXISTS

       WHILE ((J .LE. 0) .OR. (J .GT. IEFF))
         IF (J .EQ. 0) RETURN
         J = 0
         WRITE(9,*)
         WRITE(9,*)
         WRITE(9,'("WRONG SELECTION - SELECT AGAIN")')
         READ(9,*) J
       REPEAT
      ENDIF

**** SEE IF SAMPLING INTERVAL IS CORRECT

      WRITE(9,'("SAMPLING INTERVAL IS: ",E11.5," sec")') DTSPEF
      WRITE(9,'("  IS IT CORRECT (Y/N)?")')
      CH = 'N'
      READ(9,*) CH
      IF (CH .NE. 'Y') CALL R_1FREQ(DTSPEF,SETIM,SEPSC,1)
     WRITE(9,'("ENTER TIME LENGTH (in Sec                                    ) TO BE PLOTTED")')
      READ(9,*) TIMEPL

**** CALCULATES NUMBER OF SAMPLES

      NUM_SMPL = NINT(TIMEPL/DTSPEF)
      WHILE ((NUM_SMPL .GT. 2000) .OR. (NUM_SMPL .LE. 0))
        WRITE(9,*)
        WRITE(9,*)
        WRITE(9,'(" LENGTH IS TOO BIG - ENTER AGAIN")')
        READ(9,*) TIMEPL
        NUM_SMPL = NINT(TIMEPL/DTSPEF)
      REPEAT
```

```
**** INITIALIZE COUNTERS

   IPT = 1
   L = 0
   IPTR = 1

**** LOOP THROUGH ALL GENERATORS

   DO (I = 1,MAXGEN)

**** SELECT INCREMENT OF PARAMETERS

      SELECT CASE (NGEN(I))
       CASE (1)
        IIK = 3
       CASE (2)
        IIK = 4
       CASE (3)
        IIK = 2
      END SELECT

**** TEST FOR WHICH GENERATOR BELONGS TO SPECIAL EFFECT SELECTED

      IF (SPECIAL(I) .EQ. J) THEN

**** IF TRUE STORE POINTERS

         L = L + 2

****       INDEX OF GENERATOR

         KEEP(L-1) = I

****       POINTER OF PARAMETER ARRAY PAR

                              KEEP(L)=IPT

**** SET INITIAL CONDITIONS

         SELECT CASE (NGEN(I))
          CASE (1)
           PSTO(IPTR) = 10000.
           PSTO(IPTR+1) = 0.
          CASE (2)
           PSTO(IPTR) = 10000.
           PSTO(IPTR+1) = 0.
           PSTO(IPTR+2) = 0.
          CASE (3)
           PSTO(IPTR) = 0.
         END SELECT

**** INCREMENT POINTER (PSTO)
```

```
        IPTR = IPTR + IIK - 1
      ENDIF

**** INCREMENT POINTER (PAR)

      IPT = IPT + IIK
    REPEAT

**** STORE NUMBER OF GENERATORS FOR SPECIAL EFFECT SELECTED

    MAXKEEP = L

**** TEST FOR GENERATORS

    IF (MAXKEEP .EQ. 0) THEN
      CALL CLS(0)
      CALL SMOVE(20,250)
      CALL TEXT('NO GENERATORS',0,0,20)
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("NO GENERATORS - RETURN TO CONTINUE")')
      PAUSE
      RETURN
    ENDIF

**** ZERO ELEMENTS OF OUTPUT

DO(I=1,                          NUM_SMPL)
     PLOT1(I,2) = 0.
    REPEAT


**** CALCULATE AMPLITUDE OF SPECIAL EFFECT

    DO (L = 1,NUM_SMPL)
      Y = 0.

**** POINTER FOR STORAGE VARIABLE

      IPTR = 1
      DO (I = 1,MAXKEEP-1,2)
        IPT = KEEP(I+1)

**** SELECT TYPE OF GENERATOR

        SELECT CASE (NGEN(KEEP(I)))
         CASE (1)
           CALL F1AMP (IPT)
         CASE (2)
           CALL F2AMP (IPT)
         CASE (3)
           CALL SINEAMP (IPT)
```

```
        END SELECT
        Y = Y + YAMP
      REPEAT
      PLOT1(L,1) = (L - 1)*DTSPEF
      PLOT1(L,2) = Y
    REPEAT

**** PLOT YOUT ON GRAPHICS SCREEN

    HEADER = NAME(J)
    XLABEL = 'TIME'
    YLABEL = 'OUTPUT'
    VNAME = 'RESPSE'
    VSYMBOL = '1'

    CALL PLOTSIM (PLOT1,1,NUM_SMPL,HEADER,XLABEL,YLABEL,VNAME,
+             VSYMBOL)

    CALL PLTINIT
    RETURN
    END

************************************************
***                                  *** MAIN MENU FOR SPECIAL EFECTS
************************************************
    SUBROUTINE SEMMENU

    COMMON/SMSPE1/MAXGEN,MAXPAR,SPECIAL
    COMMON/SMSPE2/IEFF,NAME,WADD
    COMMON/SMSPE3/NGEN,PAR,IPOINT
    COMMON/SMSPE4/MAXPSTO,IPTR,PSTO
    COMMON/SMSPE5/KEEP,MAXKEEP
    COMMON/SMSPE6/YAMP,DTSPEF
    DIMENSION NGEN(240),PAR(1000)
    DIMENSION PSTO(1000),PLOT1(2000,2),KEEP(1000)
    INTEGER SPECIAL(240),WADD(20)
    CHARACTER NAME(20)*20,CH*1
    CHARACTER VNAME*8,HEADER*20,XLABEL*4,YLABEL*6,VSYMBOL*1

**** TEST IF FILE WAS READ

    IF ((IEFF .LT. 0) .OR. (IEFF .GT. 20)) THEN
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("  ARRAY BOUNDARY ERROR  ")')
      RETURN
    ENDIF

    WHILE (1>0)
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("        SPECIAL EFFECTS          ")')
```

```
      WRITE(9,'("          MAIN MENU              ")')
      WRITE(9,*)
      WRITE(9,*)
                                        WRITE(9,*)" 0.RETURNTOPREVIOUSMENU  ")
      WRITE(9,'("    1. ADD A SPECIAL EFFECT          ")')
      WRITE(9,'("    2. DELETE A SPECIAL EFFECT       ")')
      WRITE(9,'("    3. CHANGE/VIEW A SPECIAL EFFECT  ")')
      WRITE(9,'("    4. RESPONSE OF A SPECIAL EFFECT  ")')
      WRITE(9,'("    5. CHANGE NAME/WHERE TO ADD SPECIAL EFFECT
     +        ")')
      WRITE(9,'("    6. COPY OF GRAPHICS SCREEN       ")')
      WRITE(9,*)
      WRITE(9,*)
      WRITE(9,'("      ENTER YOUR SELECTION         ")')
      WRITE(9,*)
      WRITE(9,*)

**** LIST CURRENT SPECIAL EFFECTS ON GRAPHICS SCREEN

      CALL GRAMENU

**** SELECT FROM MAIN MENU

      ICH = 1000
      READ(9,*) ICH
      SELECT CASE (ICH)
       CASE (0)
        EXIT
       CASE (1)
        CALL ADDSE
       CASE (2)
        CALL DELSE
       CASE (3)
        CALL CHGSE
       CASE (4)
                              IJ=0
        CALL RSPSE (IJ)
       CASE (5)
        CALL CHGWADD
       CASE (6)
        CH = '_'
        WRITE(9,'("YOU HAVE SELECTED: "COPY GRAPHICS SCREEN"")
     +          ')
        WRITE(9,'("<CR> TO RETURN - ANYTHING TO CONTINUE")')
        READ(9,*) CH
        IF (CH .NE. '_') CALL PRNTTI
       CASE DEFAULT
        WRITE(9,'("ILLEGAL CHOICE")')
        WRITE(9,'("MAKE ANOTHER SELECTION")')
      END SELECT
     REPEAT
     RETURN
     END
```

```
************************************************
******* SUBROUTINE TO INPUT PARAMETERS FOR
*******    SINE WAVE GENERATOR
************************************************
   SUBROUTINE SINEADD (IGEN,J)

   COMMON/SMSPE1/MAXGEN,MAXPAR,SPECIAL
   COMMON/SMSPE3/NGEN,PAR,IPOINT
   DIMENSION NGEN(240),PAR(1000)
   INTEGER SPECIAL(240)

**** INPUT PARAMETERS

   WRITE(9,*)
   WRITE(9,*)
 WRITE(9,'(\YOU HAVE SELECTED: "SINE WAVE GENERATOR"                    ')
   WRITE(9,'("INPUT PARAMETERS   (<CR> TO RETURN TO PREVIOUS MENU
+)")')
   WRITE(9,*)
   WRITE(9,'("   AMPLITUDE OF SINE WAVE")')
   AMP = 1000
   READ(9,*) AMP
   IF (AMP .EQ. 1000) RETURN
   PAR(IPOINT+1) = AMP
   WRITE(9,*)
   WRITE(9,'("   FREQUENCY OF SINE WAVE")')
   READ(9,*) PAR(IPOINT+2)
   IPOINT = IPOINT + 2

**** INCREMENT NUMBER OF GENERATORS

   IGEN = IGEN + 1

**** SPECIFY WHICH SPECIAL EFFECT IT BELONGS TO

   SPECIAL(IGEN) = J

**** TYPE OF GENERATOR

   NGEN(IGEN) = 3
   RETURN
   END


************************************************
******* SUBROUTINE TO CALCULATE AMPLITUDE
*******    OF SINE WAVE GENERATOR
************************************************
   SUBROUTINE SINEAMP (IPT)

   COMMON/SMSPE3/NGEN,PAR,IPOINT
   COMMON/SMSPE4/MAXPSTO,IPTR,PSTO
```

```
      COMMON/SMSPE6/YAMP,DTSPEF
      DIMENSION NGEN(240),PAR(1000)
      DIMENSION PSTO(1000)
DATAPI3.141                          6/

**** AMPLITUDE OF SINE WAVE

      A = PAR(IPT)

**** FREQUENCY OF SINE WAVE

      F = PAR(IPT+1)

**** INCREMENT POINTER

      IPT = IPT + 2

**** GET LAST VALUE OF TIME

      TI = PSTO(IPTR)
      FT = F*TI

**** RESETS TIME AFTER ONE CYCLE

      IF (FT .GE. 1) THEN
        TI = 0.
        FT = 0.
      ENDIF

**** CALCULATES AMPLITUDE

      YAMP = A*SIN(2*PI*FT)

**** UPDATE TIME

      TI = TI + DTSPEF

**** STORE UPDATED TIME

      PSTO(IPTR) = TI

**** INCREMENT POINTER

      IPTR = IPTR + 1
        RETURN
      END

************************************************
******* WHITE NOISE GENERATOR
************************************************
      SUBROUTINE WHITE (U,M,V)

      COMMON/SMSPE4/MAXPSTO,IPTR,PSTO
```

```
      DIMENSION PSTO(1000)
      REAL M

      INIT = PSTO(IPTR)
      INIT = MOD(3125*INIT,65536)
      X = FLOAT(INIT)/65536.
      U = (X - 0.5)*V + M
      PSTO(IPTR) = INIT
      RETURN
     EN   D
```

```
**************************************************
******
******  ROOT SUBROUTINE FOR IMPLEMENTATION OF
******        VERSION #1 PRIMARY MOTION
*********

   SUBROUTINE OSUONE(DTPRMO,ITR)

**** COMMON BLOCKS USED IN SUBROUTINES CUEFIL1,SETINP,SETPLOT

   COMMON/CUEFILX/BXACCN,BAXA0,BAXA1,BAXA2,BAXA3
   COMMON/CUEFILY/BYACCN,BAYA0,BAYA1,BAYA2,BAYA3
   COMMON/CUEFILZ/BZACCN,BAZA0,BAZA1,BAZA2,BAZA3
   COMMON/CUEFILQ/BAPITH,BQA0,BQA1,BQA2,BQA3
   COMMON/CUEFILP/BAROLL,BPA0,BPA1,BPA2,BPA3
   COMMON/CUEFILR/BAYAW,BRA0,BRA1,BRA2,BRA3

**** COMMON BLOCKS USED IN SUBROUTINES PREPINP,SETINP

   COMMON/USER1/BIN,NINP,INPNUM,NITER
   DIMENSION BIN(2000,9),INPNUM(9)

**** COMMON BLOCKS USED IN SUBROUTINE CTRAN1

   COMMON/CTTRAN/FLTAXA,FLTAYA,FLTAZA
   COMMON/CTROT/FQADOT,FPADOT,FRADOT,FQA,FPA,FRA

**** COMMON BLOCK USED IN SUBROUTINE GRAVAL1

   COMMON/GRAV1/BXALIN,BYALIN

**** COMMON BLOCK USED IN SUBROUTINE PLATPOS1

   COMMON/MPCP1/BXPOS,BYPOS,BZPOS,BTHPOS,BPHPOS,BSIPOS

**** COMMON BLOCK USED IN SUBROUTINE POSCON1

    COMMON/POSCON1/BMXPOS,BMYPOS,BMZPOS,BMTHEP,BMPHIP,BMPSIP

**** COMMON BLOCK USED IN SUBROUTINE READPRIM

 COMMON/PRIM1/XENFL,YENFL,ZENFL,THENFL,PHENFL,SIENFL
   COMMON/PRIM2/XGRFL,YGRFL,MPCPFL,MPLPFL
   COMMON/PRIM3/BXAWTC,BYAWTC,BZAWTC,BTHWTC,BPHWTC,BSIWTC
   COMMON/PRIM4/BKAX1,BKAY1,BKAZ1,BKQA1,BKPA1,BKRA1
   COMMON/PRIM5/BKAX2,BKAY2,BKAZ2,BKQA2,BKPA2,BKRA2
   COMMON/PRIM6/ALAGX,AKGAX,DLAGX,DKGAX,ALAGY,AKGAY,DLAGY,DKGAY
   COMMON/PRIM7/BKQA4,BKPA4
   COMMON/PRIM8/BLAGX,BLAGY,BLAGZ,BLAGT,BLAGP,BLAGS

   INTEGER*4  XENFL,YENFL,ZENFL,THENFL,PHENFL,SIENFL
   INTEGER*4  XGRFL,YGRFL,MPCPFL,MPLPFL
   REAL   BXAWTC,BYAWTC,BZAWTC,BTHWTC,BPHWTC,BSIWTC
```

```
      REAL  BKAX1,BKAY1,BKAZ1,BKQA1,BKPA1,BKRA1
      REAL  BKAX2,BKAY2,BKAZ2,BKQA2,BKPA2,BKRA2
      REAL  ALAGX,AKGAX,DLAGX,DKGAX,ALAGY,AKGAY,DLAGY,DKGAY
      REAL  BKQA4,BKPA4
      REAL  BLAGX,BLAGY,BLAGZ,BLAGT,BLAGP,BLAGS
*
**** MAIN LOOP - PRIMARY MOTION
*
****     SET AIRCRAFT ACCELERATIONS FROM PREPARED INPUT

         CALL SETINP(ITR)

****     PERFORM CENTROID TRANSFORMATIONS

         CALL CTRAN1

**** PERFORM WASHOUT                                T(VERSION #1)

         CALL CUEFIL1(DTPRMO)

****     PERFORM GRAVITY ALIGNMENT (VERSION #1)

         CALL GRAVAL1

****     PERFORM MOTION PLATFORM COMMENDED POSITIONS (VERSION #1)

         CALL PLATPOS1

****     PERFORM MOTION POSITION CONTROL (VERSION #1)

         CALL POSCON1

      RETURN
      END


**************************************************
*******
******* MOTION ONSET CUE FILTERS SUBROUTINE
*********

      SUBROUTINE CUEFIL1(FIC20)

      COMMON/CUEFILX/BXACCN,BAXA0,BAXA1,BAXA2,BAXA3
      COMMON/CUEFILY/BYACCN,BAYA0,BAYA1,BAYA2,BAYA3
      COMMON/CUEFILZ/BZACCN,BAZA0,BAZA1,BAZA2,BAZA3
      COMMON/CUEFILQ/BAPITH,BQA0,BQA1,BQA2,BQA3
      COMMON/CUEFILP/BAROLL,BPA0,BPA1,BPA2,BPA3
      COMMON/CUEFILR/BAYAW,BRA0,BRA1,BRA2,BRA3

      COMMON/PRIM1/XENFL,YENFL,ZENFL,THENFL,PHENFL,SIENFL
```

```
      COMMON/PRIM3/BXAWTC,BYAWTC,BZAWTC,BTHWTC,BPHWTC,BSIWTC
      COMMON/PRIM4/BKAX1,BKAY1,BKAZ1,BKQA1,BKPA1,BKRA1
     COMMON/PR                                  IM5/BKAX2,BKAY2,BKAZ2,BKQA2,BKPA2,BKRA2

      INTEGER*4  XENFL,YENFL,ZENFL,THENFL,PHENFL,SIENFL
      REAL   BXAWTC,BYAWTC,BZAWTC,BTHWTC,BPHWTC,BSIWTC
      REAL   BKAX1,BKAY1,BKAZ1,BKQA1,BKPA1,BKRA1
      REAL   BKAX2,BKAY2,BKAZ2,BKQA2,BKPA2,BKRA2


*
**** LINEAR THIRD ORDER WASHOUT FILTER FOR THE LATERAL AXIS (X)
*

      ALPHA=BXAWTC ; A=BKAX2+ALPHA ; B=BKAX1+BKAX2*ALPHA
      C=BKAX1*ALPHA ; D=BKAX1

      BAXA0=D*BXACCN-A*BAXA1-B*BAXA2-C*BAXA3
      BAXA1=BAXA1+FIC20*BAXA0
      BAXA2=BAXA2+FIC20*BAXA1
      BAXA3=BAXA3+FIC20*BAXA2

*
**** LINEAR THIRD ORDER WASHOUT FILTER FOR THE LATERAL AXIS (Y)
*

      ALPHA=BYAWTC ; A=BKAY2+ALPHA ; B=BKAY1+BKAY2*ALPHA
      C=BKAY1*ALPHA ; D=BKAY1

      BAYA0=D*BYACCN-A*BAYA1-B*BAYA2-C*BAYA3
      BAYA1=BAYA1+FIC20*BAYA0
      BAYA2=BAYA2+FIC20*BAYA1
      BAYA3=BAYA3+FIC20*BAYA2

*
**** LINEAR THIRD ORDER WASHOUT FILTER FOR THE VERTICAL AXIS (Z)
*

      ALPHA=BZAWTC ; A=BKAZ2+ALPHA ; B=BKAZ1+BKAZ2*ALPHA
                                 C=BKAZ1*ALPHA;D=BKAZ1

      BAZA0=D*BZACCN-A*BAZA1-B*BAZA2-C*BAZA3
      BAZA1=BAZA1+FIC20*BAZA0
      BAZA2=BAZA2+FIC20*BAZA1
      BAZA3=BAZA3+FIC20*BAZA2

*
**** LINEAR THIRD ORDER WASHOUT FILTER FOR THE PITCH AXIS (Q)
*                    (DISABLED)

*       ALPHA=BTHWTC ; A=BKQA2+ALPHA ; B=BKQA1+BKQA2*ALPHA
*       C=BKQA1*ALPHA ; D=BKQA1
*
```

```
*       BQA0=D*BAPITH-A*BQA1-B*BQA2-C*BQA3
*       BQA1=BQA1+FIC20*BQA0
*       BQA2=BQA2+FIC20*BQA1
******** BQA3=BQA3+FIC20*BQA2

*
******** LINEAR THIRD ORDER WASHOUT FILTER FOR THE ROLL AXIS (P)
*

    ALPHA=BPHWTC ; A=BKPA2+ALPHA ; B=BKPA1+BKPA2*ALPHA
    C=BKPA1*ALPHA ; D=BKPA1

    BPA0=D*BAROLL-A*BPA1-B*BPA2-C*BPA3
    BPA1=BPA1+FIC20*BPA0
    BPA2=BPA2+FIC20*BPA1
    BPA3=BPA3+FIC20*BPA2

*
**** LINEAR THIRD ORDER WASHOUT FILTER FOR THE YAW AXIS (R)
*

    ALPHA=BSIWTC ; A=BKRA2+ALPHA ; B=BKRA1+BKRA2*ALPHA
    C=BKRA1*ALPHA ; D=BKRA1

 BRA0=D*BAYAW-A*BRA1-B*BRA2-C*B                                    RA3
    BRA1=BRA1+FIC20*BRA0
    BRA2=BRA2+FIC20*BRA1
    BRA3=BRA3+FIC20*BRA2

    RETURN
    END

**************************************************
*******
******* SUBROUTINE TO EXECUTE THE CENTROID TRANSFORMATIONS
*******

    SUBROUTINE CTRAN1

    COMMON/CUEFILX/BXACCN,BAXA0,BAXA1,BAXA2,BAXA3
    COMMON/CUEFILY/BYACCN,BAYA0,BAYA1,BAYA2,BAYA3
    COMMON/CUEFILZ/BZACCN,BAZA0,BAZA1,BAZA2,BAZA3
    COMMON/CUEFILQ/BAPITH,BQA0,BQA1,BQA2,BQA3
    COMMON/CUEFILP/BAROLL,BPA0,BPA1,BPA2,BPA3
    COMMON/CUEFILR/BAYAW,BRA0,BRA1,BRA2,BRA3

    COMMON/CTTRAN/FLTAXA,FLTAYA,FLTAZA
    COMMON/CTROT/FQADOT,FPADOT,FRADOT,FQA,FPA,FRA

**** DEFINE THE CONSTANTS  (LATER ALLOW TO BE CHANGED BY USER)

    FRX=10.  ; FRY=0.  ; FRZ=0.
```

```
      FDCOSL3=0. ;FDCOSM3=0. ; FDCOSN3=0.0 ; G=32.174

*
**** THE TRANSFORMATIONS
*

**** LONGITUDINAL ACCELERATION

      BXACCN=  FLTAXA   -   (FQA**2+FRA**2)*FRX
     1       + (FQA*FPA                                    -FRADOT)*FRY
     1                 +   (FRA*FPA+FQADOT)*FRZ
     1                 +   FDCOSL3*G

**** LATERAL ACCELERATION

      BYACCN=  FLTAYA   +   (FPA*FQA+FRADOT)*FRX
     1                 -   (FPA**2+FRA**2)*FRY
     1                 +   (FRA*FQA-FPADOT)*FRZ
     1                 +   FDCOSM3*G

**** VERTICAL ACCELERATION

      BZACCN=  FLTAZA   +   (FPA*FRA-FQADOT)*FRX
     1                 +   (FQA*FRA+FPADOT)*FRY
     1                 -   (FPA**2+FQA**2)*FRZ
     1                 +   FDCOSN3*G

**** SELECT TO PASS ANGULAR RATES (NOT ACCELERATIONS) TO WASHOUT

CC    BAPITH=FQA
      BAROLL=FPA
      BAYAW=FRA

      RETURN
      END
************************************************
******
******  GRAVITY ALIGNMENT SUBROUTINE
*********

      SUBROUTINE GRAVAL1

      COMMON/CUEFILX/BXACCN,BAXA0,BAXA1,BAXA2,BAXA3
      COMMON/CUEFILY/BYACCN,BAYA0,BAYA1,BAYA2,BAYA3
     COMMON/CUEFILZ/BZ                                    ACCN,BAZA0,BAZA1,BAZA2,BAZA3
      COMMON/CUEFILQ/BAPITH,BQA0,BQA1,BQA2,BQA3
      COMMON/CUEFILP/BAROLL,BPA0,BPA1,BPA2,BPA3
      COMMON/CUEFILR/BAYAW,BRA0,BRA1,BRA2,BRA3

      COMMON/GRAV1/BXALIN,BYALIN

      COMMON/PRIM2/XGRFL,YGRFL,MPCPFL,MPLPFL
      COMMON/PRIM6/ALAGX,AKGAX,DLAGX,DKGAX,ALAGY,AKGAY,DLAGY,DKGAY
```

```
      INTEGER*4  XGRFL,YGRFL,MPCPFL,MPLPFL
      REAL   ALAGX,AKGAX,DLAGX,DKGAX,ALAGY,AKGAY,DLAGY,DKGAY

**** GRAVITY ALIGN LONGITUDINAL ACCELERATION

      IF (BXACCN.GT.0.0) THEN
        FLAGP=ALAGX
        BKGAP=AKGAX
      ELSE
        FLAGP=DLAGX
        BKGAP=DKGAX
      ENDIF
      BXALIN=BXALIN+FLAGP*(BKGAP*BXACCN-BXALIN)

**** GRAVITY ALIGN LATERAL ACCELERATION

      IF (BYACCN.GT.0.0) THEN
        FLAGY=ALAGY
        BKGAY=AKGAY
      ELSE
        FLAGY=DLAGY
        BKGAY=DKGAY
      ENDIF
      BYALIN=BYALIN+FLAGY*(BKGAY*BYACCN-BYALIN)

      RETURN
      END

*************************************************
******
*****                                  MOTION PLATFORM COMMANDED POSITIONS SUBROUTINE
*********

      SUBROUTINE PLATPOS1

      COMMON/CUEFILX/BXACCN,BAXA0,BAXA1,BAXA2,BAXA3
      COMMON/CUEFILY/BYACCN,BAYA0,BAYA1,BAYA2,BAYA3
      COMMON/CUEFILZ/BZACCN,BAZA0,BAZA1,BAZA2,BAZA3
      COMMON/CUEFILQ/BAPITH,BQA0,BQA1,BQA2,BQA3
      COMMON/CUEFILP/BAROLL,BPA0,BPA1,BPA2,BPA3
      COMMON/CUEFILR/BAYAW,BRA0,BRA1,BRA2,BRA3

      COMMON/GRAV1/BXALIN,BYALIN

      COMMON/MPCP1/BXPOS,BYPOS,BZPOS,BTHPOS,BPHPOS,BSIPOS

      COMMON/PRIM7/BKQA4,BKPA4
      REAL   BKQA4,BKPA4

**** INITIALIZE THE CONSTANTS
```

```
      BKXPOS=0.037
      BKYPOS=0.037
      BKZPOS=0.037

      BKTHP=0.037
      BKPIP=0.037
      BKSIP=0.037

**** SET ALL SPECIAL EFFECTS TO ZERO

      BSPEFX=0.0
      BSPEFY=0.0
      BTURBW=0.0

      FSINT=0.0
      FSINPH=0.0

      BSPEFQ=0.0
      BSPEFP=0.0
      BSPEFR=0.0

**** LONGITUDINAL

      BXPOS=BKXPOS*BAXA2+BSPEFX

**** LATERAL

      BYPOS=BKYPOS*BAYA2+BSPEFY

***V                          ERTICAL

      BZPOS=BKZPOS*BAZA2+BTURBW

**** PITCH

      BTHPOS=BKTHP*BQA2+BKQA4*FSINT+BXALIN+BSPEFQ

**** ROLL

      BPHPOS=BKPIP*BPA2+BKPA4*FSINPH+BYALIN+BSPEFP

**** YAW

      BSIPOS=BKSIP*BRA2+BSPEFR

      RETURN
      END

************************************************
******
******  MOTION POSITION CONTROL SUBROUTINE
*********
```

```
      SUBROUTINE POSCON1

      COMMON/MPCP1/BXPOS,BYPOS,BZPOS,BTHPOS,BPHPOS,BSIPOS

      COMMON/POSCON1/BMXPOS,BMYPOS,BMZPOS,BMTHEP,BMPHIP,BMPSIP

      COMMON/PRIM2/XGRFL,YGRFL,MPCPFL,MPLPFL
      COMMON/PRIM8/BLAGX,BLAGY,BLAGZ,BLAGT,BLAGP,BLAGS

      INTEGER*4  XGRFL,YGRFL,MPCPFL,MPLPFL
      REAL   BLAGX,BLAGY,BLAGZ,BLAGT,BLAGP,BLAGS

**** POSITION CONTROL OUTPUTS (THE SIX POSITIONS IN INCHES )

      BMXPOS=BMXPOS+(BXPOS*12.-BMXPOS)*BLAGX
      BMYPOS=BMYPOS+(BYPOS*12.-BMYPOS)*BLAGY
      BMZPOS=BMZPOS+(BZPOS*12.-BMZPOS)*BLAGZ

      BMTHEP=BMTHEP+(BTHPOS-BMTHEP)*BLAGT
BMPHIP=BMPHIP+(BPHPOS-B                    MPHIP)*BLAGP
      BMPSIP=BMPSIP+(BSIPOS-BMPSIP)*BLAGS

      RETURN
      END

************************************************
******  ROUTINE TO READ THE PRIMARY MOTION PARAMETERS FROM A FILE
*********
      SUBROUTINE READPRIM

      COMMON/PRIM1/XENFL,YENFL,ZENFL,THENFL,PHENFL,SIENFL
      COMMON/PRIM2/XGRFL,YGRFL,MPCPFL,MPLPFL
      COMMON/PRIM3/BXAWTC,BYAWTC,BZAWTC,BTHWTC,BPHWTC,BSIWTC
      COMMON/PRIM4/BKAX1,BKAY1,BKAZ1,BKQA1,BKPA1,BKRA1
      COMMON/PRIM5/BKAX2,BKAY2,BKAZ2,BKQA2,BKPA2,BKRA2
      COMMON/PRIM6/ALAGX,AKGAX,DLAGX,DKGAX,ALAGY,AKGAY,DLAGY,DKGAY
      COMMON/PRIM7/BKQA4,BKPA4
      COMMON/PRIM8/BLAGX,BLAGY,BLAGZ,BLAGT,BLAGP,BLAGS

      INTEGER*4  XENFL,YENFL,ZENFL,THENFL,PHENFL,SIENFL
      INTEGER*4  XGRFL,YGRFL,MPCPFL,MPLPFL
      REAL   BXAWTC,BYAWTC,BZAWTC,BTHWTC,BPHWTC,BSIWTC
      REAL   BKAX1,BKAY1,BKAZ1,BKQA1,BKPA1,BKRA1
      REAL   BKAX2,BKAY2,BKAZ2,BKQA2,BKPA2,BKRA2
      REAL   ALAGX,AKGAX,DLAGX,DKGAX,ALAGY,AKGAY,DLAGY,DKGAY
REALB                          KQA4,BKPA4
      REAL   BLAGX,BLAGY,BLAGZ,BLAGT,BLAGP,BLAGS

      CHARACTER MOTFNAM*16
      LOGICAL EX
      EX = .FALSE.
      MOTFNAM = 'PRIMPAR.SA'
      WRITE(9,*) ('WHAT IS PRIMARY MOTION DATA FILE NAME?')
```

```
      READ(9,*)  MOTFNAM
      INQUIRE(FILE=MOTFNAM,EXIST=EX,IOSTAT=IOS,ERR=1000)
      IF(.NOT. EX) THEN
        WRITE(9,'("DATA FILE ",(A)," DOES NOT EXIST")') MOTFNAM
        STOP
      ENDIF
      OPEN(UNIT=10,FILE=MOTFNAM)

      READ(10,*)XENFL,YENFL,ZENFL,THENFL,PHENFL,SIENFL
      READ(10,*)XGRFL,YGRFL,MPCPFL,MPLPFL
      READ(10,*)BXAWTC,BYAWTC,BZAWTC,BTHWTC,BPHWTC,BSIWTC
      READ(10,*)BKAX1,BKAY1,BKAZ1,BKQA1,BKPA1,BKRA1
      READ(10,*)BKAX2,BKAY2,BKAZ2,BKQA2,BKPA2,BKRA2
      READ(10,*)ALAGX,AKGAX,DLAGX,DKGAX,ALAGY,AKGAY,DLAGY,DKGAY
      READ(10,*)BKQA4,BKPA4
      READ(10,*)BLAGX,BLAGY,BLAGZ,BLAGT,BLAGP,BLAGS

      RETURN
1000  WRITE(9,'(" ERROR READING THE FILE ")')
      RETURN
      END


*************************************************
                      ******
****** SUBROUTINE TO PREPARE THE INPUT BASED ON USER REQUIREMENT
*********
      SUBROUTINE PREPINP(FIC20)

      COMMON/USER1/BIN,NINP,INPNUM,NITER
      DIMENSION BIN(2000,9),INPNUM(9)

      CALL INIT_PROG_VARS

      DO (I=1,9)
        INPNUM(I)=0
      REPEAT

      WRITE(9,'(" ENTER NUMBER OF ITERATIONS ")')
      WRITE(9,'("   ENTER 0 FOR NO INPUT")')
      NITER = 0
      READ(9,*)NITER
      IF (NITER .EQ. 0) RETURN
      DO (I=1,10)
        WRITE(9,*)
      REPEAT
      WRITE(9,'(" INPUT VARIABLES ")')
      WRITE(9,*)
      WRITE(9,'(" 1. LONGITUDINAL ACCELERATION (FLTAXA) ")')
      WRITE(9,'(" 2. LATERAL ACCELERATION     (FLTAYA) ")')
      WRITE(9,'(" 3. VERTICAL ACCELERATION    (FLTAZA) ")')
      WRITE(9,'(" 4. PITCH ACCELERATION       (FQADOT) ")')
      WRITE(9,'(" 5. ROLL ACCELERATION        (FPADOT) ")')
```

```
   WRITE(9,'(" 6. YAW ACCELERATION        (FRADOT) ")')
 WRITE(9,'(" 7. PITCH VELOCITY      (FQA) ")')
   WRITE(9,'(" 8. ROLL VELOCITY           (FPA) ")')
   WRITE(9,'(" 9. YAW VELOCITY            (FRA) ")')
   WRITE(9,*)
   WRITE(9,'(" ENTER VARIABLES ")')
   READ(9,*)(INPNUM(I),I=1,9)

**** DETERMINE NUMBER OF INPUTS

   I=1
   WHILE((INPNUM(I).GE.1).AND.(INPNUM(I).LE.9))
     I=I+1
     IF(I.EQ.10) EXIT
   REPEAT
   NINP=I-1

**** CONSTRUCT INPUT

   DO (I=1,NINP)

****   LONGITUDINAL ACCELERATION

      IF (INPNUM(I).EQ.1) THEN
        WRITE(9,'(" FOR THE LONGITUDINAL ACCELERATION ")')
        CALL SETBIN(1,FIC20)

****   LATERAL ACC

      ELSEIF (INPNUM(I).EQ.2) THEN
        WRITE(9,'(" FOR THE LATERAL ACCELERATION ")')
        CALL SETBIN(2,FIC20)

****   VERTICAL ACCELERATION

      ELSEIF (INPNUM(I).EQ.3) THEN
        WRITE(9,'(" FOR THE VERTICAL ACCELERATION ")')
        CALL SETBIN(3,FIC20)

****   PITCH ACCELERATION

 ELSEIF(INPNUM(I).EQ.4)THEN
        WRITE(9,'(" FOR THE PITCH ACCELERATION ")')
        CALL SETBIN(4,FIC20)

****   ROLL ACCELERATION

      ELSEIF (INPNUM(I).EQ.5) THEN
        WRITE(9,'(" FOR THE ROLL ACCELERATION ")')
        CALL SETBIN(5,FIC20)

****   YAW ACCELERATION
```

```
      ELSEIF (INPNUM(I).EQ.6) THEN
        WRITE(9,'(" FOR THE YAW ACCELERATION ")')
        CALL SETBIN(6,FIC20)

****    PITCH VELOCITY

      ELSEIF (INPNUM(I).EQ.7) THEN
        WRITE(9,'(" FOR THE PITCH VELOCITY ")')
        CALL SETBIN(7,FIC20)

****    ROLL VELOCITY

      ELSEIF (INPNUM(I).EQ.8) THEN
        WRITE(9,'(" FOR THE ROLL VELOCITY ")')
        CALL SETBIN(8,FIC20)

****    YAW VELOCITY

      ELSEIF (INPNUM(I).EQ.9) THEN
        WRITE(9,'(" FOR THE YAW VELOCITY ")')
        CALL SETBIN(9,FIC20)
      ENDIF
    REPEAT

    RETURN
    END


************************************************
******
                                  ***** SUBROUTINE TO SETUP ARRAY BIN
*********
    SUBROUTINE SETBIN(M,FIC20)

    COMMON/USER1/BIN,NINP,INPNUM,NITER
    DIMENSION BIN(2000,9),INPNUM(9)

    DIMENSION BRKPT(10),AMP(10)

    WRITE(9,*)
    WRITE(9,'(" SELECT INPUT SIGNAL TYPE ")')
    WRITE(9,*)
    WRITE(9,'(" 1- STAIR STEP          ")')
    WRITE(9,'(" 2- SINE WAVE           ")')
    WRITE(9,'(" 3- READ FROM A FILE    ")')
    WRITE(9,*)
    READ(9,*)ISEL

    IF (ISEL.EQ.1) THEN

      WRITE(9,'(" ENTER # INPUT AMPLITUDE CHANGES")')
      READ(9,*)NUMCHG
      DO (J=1,NUMCHG)
```

```
        WRITE(9,'(" ENTER BREAK POINT AND AMPLITUDE ")')
        READ(9,*)BRKPT(J),AMP(J)
      REPEAT
      DO (K=1,NITER)
        DO (JJ=1,NUMCHG-1)
          IF ((K.GE.BRKPT(JJ)).AND.(K.LT.BRKPT(JJ+1)))
     1       BIN(K,M)=AMP(JJ)
        REPEAT
        IF (K.LT.BRKPT(1)) BIN(K,M)=0.0
        IF(K.GE.BRKPT(NUMCHG)) BIN(K,M)=AMP(NUMCH                    G)
      REPEAT

    ELSEIF(ISEL.EQ.2) THEN

      WRITE(9,'(" ENTER THE AMPLITUDE & FREQ OF WAVE ")')
       READ(9,*)AAMP,FREQ

**** SIN WAVE WITH FIC20

      DO (K=1,NITER)
        BIN(K,M)=AAMP*SIN(2.*3.1415927*FREQ*FLOAT(K)*FIC20)
      REPEAT

    ENDIF

    RETURN
    END
************************************************
******
******  SUBROUTINE TO SET THE DIFFERENT INPUTS IN EACH ITERATION.
******  THE ARRAY BIN IS USED TO SET THE PROPER INPUT VALUE FOR THE
******  RIGHT VARIABLE.
*********
    SUBROUTINE SETINP(IT)

    COMMON/CTTRAN/FLTAXA,FLTAYA,FLTAZA
    COMMON/CTROT/FQADOT,FPADOT,FRADOT,FQA,FPA,FRA

    COMMON/USER1/BIN,NINP,INPNUM,NITER
    DIMENSION BIN(2000,9),INPNUM(9)

    FLTAXA=BIN(IT,1)
    FLTAYA=BIN(IT,2)
    FLTAZA=BIN(IT,3)
    FQADOT=BIN(IT,4)
    FPADOT=BIN(IT,5)
    FRADOT=BIN(IT,6)
    FQA=BIN(IT,7)
    FPA=BIN(IT,8)
    FRA=BIN(IT,9)

    RETURN
    END
```

```
******                                                    ****************************************
******
******  SUBROUTINE TO INITIALIZE THE PROGRAM VARIABLES
*********

   SUBROUTINE INIT_PROG_VARS

   COMMON/USER1/BIN,NINP,INPNUM,NITER
   DIMENSION BIN(2000,9),INPNUM(9)

   DO (J=1,9)
     DO (I=1,2000)
       BIN(I,J)=0.0
     REPEAT
   REPEAT

   RETURN
   END

************************************************
```

NUMBER	SYMBOL	NAME - DESCRIPTION	SCALE
1	POSX	SPECIAL EFF - X AXIS	1.0
2	POSY	SPECIAL EFF - Y AXIS	1.0
3	POSZ	SPECIAL EFF - Z AXIS	1.0
4	POSTHE	SPECIAL EFF - PITCH	1.0
5	POSPHI	SPECIAL EFF - ROLL	1.0
6	POSPSI	SPECIAL EFF - YAW	1.0
7	PD11	PISTON1 DISPLACEMENT	1.0
8	PD21	PISTON2 DISPLACEMENT	1.0
9	PD31	PISTON3 DISPLACEMENT	1.0
10	PD41	PISTON4 DISPLACEMENT	1.0
11	PD51	PISTON5 DISPLACEMENT	1.0
12	PD61	PISTON6 DISPLACEMENT	1.0
13	PDOT11	PISTON1 VELOCITY	1.0
14	PDOT21	PISTON2 VELOCITY	1.0
15	PDOT31	PISTON3 VELOCITY	1.0
16	PDOT41	PISTON4 VELOCITY	1.0
17	PDOT51	PISTON5 VELOCITY	1.0
18	PDOT61	PISTON6 VELOCITY	1.0
19	X11	SPOOL1 DISPLACEMENT	1.0
20	X21	SPOOL2 DISPLACEMENT	1.0
21	X31	SPOOL3 DISPLACEMENT	1.0
22	X41	SPOOL4 DISPLACEMENT	1.0
23	X51	SPOOL5 DISPLACEMENT	1.0
24	X61	SPOOL6 DISPLACEMENT	1.0
25	XDOT11	SPOOL1 VELOCITY	1.0
26	XDOT21	SPOOL2 VELOCITY	1.0
27	XDOT31	SPOOL3 VELOCITY	1.0
28	XDOT41	SPOOL4 VELOCITY	1.0
29	XDOT51	SPOOL5 VELOCITY	1.0
30	XDOT61	SPOOL6 VELOCITY	1.0
31	VOUT1	VOLTAGE TO VALVE1	0.001
32	VOUT2	VOLTAGE TO VALVE2	1.0
33	VOUT3	VOLTAGE TO VALVE3	1.0
34	VOUT4	VOLTAGE TO VALVE4	1.0
35	VOUT5	VOLTAGE TO VALVE5	1.0
36	VOUT6	VOLTAGE TO VALVE6	1.0
37	CPRS1	CAP PRESSURE1	0.002
38	CPRS2	CAP PRESSURE2	1.0
39	CPRS3	CAP PRESSURE3	1.0
40	CPRS4	CAP PRESSURE4	1.0
41	CPRS5	CAP PRESSURE5	1.0
42	CPRS6	CAP PRESSURE6	1.0
43	RPRS1	ROD PRESSURE1	0.002.
44	RPRS2	ROD PRESSURE2	1.0
45	RPRS3	ROD PRESSURE3	1.0
46	RPRS4	ROD PRESSURE4	1.0
47	RPRS5	ROD PRESSURE5	1.0
48	RPRS6	ROD PRESSURE6	1.0
49	POSI1	CYLINDER1 LENGTH	1.0
50	POSI2	CYLINDER2 LENGTH	1.0
51	POSI3	CYLINDER3 LENGTH	1.0

52	POSI4	CYLINDER4 LENGTH	1.0
53	POSI5	CYLINDER5 LENGTH	1.0
54	POSI6	CYLINDER6 LENGTH	1.0
55	CYLOT1	COMM CYL1 LENGTH	1.0
56	CYLOT2	COMM CYL2 LENGTH	1.0
57	CYLOT3	COMM CYL3 LENGTH	1.0
58	CYLOT4	COMM CYL4 LENGTH	1.0
59	CYLOT5	COMM CYL5 LENGTH	1.0
60	CYLOT6	COMM CYL6 LENGTH	1.0
61	FORCE1	DIFFERENTIAL FORCE1	1.0
62	FORCE2	DIFFERENTIAL FORCE2	1.0
63	FORCE3	DIFFERENTIAL FORCE3	1.0
64	FORCE4	DIFFERENTIAL FORCE4	1.0
65	FORCE5	DIFFERENTIAL FORCE5	1.0
66	FORCE6	DIFFERENTIAL FORCE6	1.0
67	Q61	COMPRESS. FLOW1	1.0
68	Q62	COMPRESS. FLOW2	1
69	Q63	COMPRESS. FLOW3	1.0
70	Q64	COMPRESS. FLOW4	1.0
71	Q65	COMPRESS. FLOW5	1.0
72	Q66	COMPRESS. FLOW6	1.0
73	Q71	COMPRESS. FLOW1	0.1
74	Q72	COMPRESS. FLOW2	1.0
75	Q73	COMPRESS. FLOW3	1.0
76	Q74	COMPRESS. FLOW4	1.0
77	Q75	COMPRESS. FLOW5	1.0
78	Q76	COMPRESS. FLOW6	1.0
79	SIGNA1	SPEFX+PRIMOX	1.0
80	SIGNA2	SPEFY+PRIMOY	1.0
81	SIGNA3	SPEFZ+PRIMOZ	1.0
82	SIGNA4	SPEFTHE+PRIMOTHE	1.0
83	SIGNA5	SPEFPHI+PRIMOPHI	1.0
84	SIGNA6	SPEFPSI+PRIMOPSI	1.0
85	COMPN1	COMPENSATION 1	1.0
86	COMPN2	COMPENSATION2	1.0
87	COMPN3	COMPENSATION 3	1.0
88	COMPN4	COMPENSATION 4	1.0
89	COMPN5	COMPENSATION 5	1.0
90	COMPN6	COMPENSATION 6	1.0
91	Q11	FLOW11	0.1
92	Q12	FLOW12	1.0
93	Q13	FLOW13	1.0
94	Q14	FLOW14	1.0
95	Q15	FLOW15	1.0
96	Q16	FLOW16	1.0
97	Q21	FLOW21	1.0
98	Q22	FLOW22	1.0
99	Q23	FLOW22	1.0
100	Q24	FLOW25	1.0
101	Q25	FLOW24	1.0
102	Q26	FLOW26	1.0
103	V11	CAP VOLUME1	1.0
104	V12	CAPVOLUME2	1.0

105	V13	CAP VOLUME3	1.0
106	V14	CAP VOLUME4	1.0
107	V15	CAP VOLUME5	1.0
108	V16	CAP VOLUME6	1.0
109	V21	ROD VOLUME1	1.0
110	V22	ROD VOLUME2	1.0
111	V23	ROD VOLUME3	1.0
112	V24	ROD VOLUME4	1.0
113	V25	ROD VOLUME5	1.0
114	V26	ROD VOLUME6	1.0
115	Q31	LEAKAGE FLOW31	1.0
116	Q32	LEAKAGE FLOW32	1.0
117	Q33	LEAKAGE FLOW33	1.0
118	Q34	LEAKAGE FLOW34	1.0
119	Q35	LEAKAGE FLOW35	1.0
120	Q36	LEAKAGE FLOW36	1.0
121	Q41	LEAKAGE FLOW41	1.0
122	Q42	LEAKAGE FLOW42	1.0
123	Q43	LEAKAGE FLOW43	1.0
124	Q44	LEAKAGE FLOW44	1.0
125	Q45	LEAKAGE FLOW45	1.0
126	Q46	LEAKAGE FLOW46	1.0
127	Q51	LEAKAGE FLOW51	1.0
128	Q52	LEAKAGE FLOW52	1.0
129	Q53	LEAKAGE FLOW53	1.0
130	Q54	LEAKAGE FLOW54	1.0
131	Q55	LEAKAGE FLOW55	1.0
132	Q56	LEAKAGE FLOW56	1.0
133	UIN1	DIFFERENCIAL PRESS1	0.01
134	UIN2	DIFFERENCIAL PRESS2	1.0
135	UIN3	DIFFERENCIAL PRESS3	1.0
136	UIN4	DIFFERENCIAL PRESS4	1.0
137	UIN5	DIFFERENCIAL PRESS5	1.0
138	UIN6	DIFFERENCIAL PRESS6	1.0
139	TINVX	INV. TRANSF. X	1.0
140	TINVY	INV. TRANSF. Y	1.0
141	TINVZ	INV. TRANSF. Z	1.0
142	TINVTH	INV. TRANSF. THE	1.0
143	TINVPH	INV. TRANSF. PHI	1.0
144	TINVPS	INV. TRANSF. PSI	1.0
145	AFTER1	CEXT AFTER INVTRANS	1.0
146	AFTER2	CEXT AFTER INVTRANS	1.0
147	AFTER3	CEXT AFTER INVTRANS	1.0
148	AFTER4	CEXT AFTER INVTRANS	1.0
149	AFTER5	CEXT AFTER INVTRANS	1.0
150	AFTER6	CEXT AFTER INVTRANS	1.0
151	FLTAXA	X ACC. INPUT	1.0
152	FLTAYA	Y ACC. INPUT	1.0
153	FLTAZA	Z ACC. INPUT	1.0
154	FQA	PITCH RATE INPUT	1.0
155	FPA	ROLL RATE INPUT	1.0
156	FRA	YAW RATE INPUT	1.0
157	FQADOT	PITCH ACC. INPUT	1.0

158	FPADOT	ROLL ACC. INPUT	1.0
159	FRADOT	YAW ACC. INPUT	1.0
160	BXACCN	X CENT. ACC.	1.0
161	BYACCN	Y CENT. ACC.	1.0
162	BZACCN	Z CENT. ACC.	1.0
163	BAROLL	ROLL CENT. ACC.	1.0
164	BAYAW	YAW CENT. ACC.	1.0
165	BAXA2	X OUTPUT OF WASHOUT	1.0
166	BAYA2	Y OUTPUT OF WASHOUT	1.0
167	BAZA2	Z OUTPUT OF WASHOUT	1.0
168	BPA2	ROLL OUTPUT OF WASHOUT	1.0
169	BRA2	YAW OUTPUT OF WASHOUT	1.0
170	BXALIN	X GRAVITY ALIGN.	1.0
171	BYALIN	Y GRAVITY ALING.	1.0
172	BXPOS	X PLATF. POS.	1.0
173	BYPOS	Y PLATF. POS.	1.0
174	BZPOS	Z PLATF. POS.	1.0
175	BTHPOS	PITCH PLATF. POS.	1.0
176	BPHPOS	ROLL PLATF. POS.	1.0
177	BSIPOS	YAW PLATF. POS.	1.0
178	BMXPOS	X OUTPUT OF PRIM. MOTION	1.0
179	BMYPOS	Y OUTPUT OF PRIM. MOTION	1.0
180	BMZPOS	Z OUTPUT OF PRIM. MOTION	1.0
181	BMTHEP	PITCH OUTPUT PRIM. MOTION	1.0
182	BMPHIP	ROLL OUTPUT PRIM. MOTION	1.0
183	BMPSIP	YAW OUTPUT PRIM. MOTION	1.0
184	BAXA0	X ACC WASHOUT	1.0
185	BAYA0	Y ACC WASHOUT	1.0
186	BAZA0	Z ACC WASHOUT	1.0
187	BPA0	ROLL RATE WASHOUT	1.0
188	BRA0	YAW RATE WASHOUT	1.0
189	BAXA1	X VEL WASHOUT	1.0
190	BAYA1	Y VEL WASHOUT	1.0
191	BAZA1	Z VEL WASHOUT	1.0
192	BPA1	ROLL WASHOUT	1.0
193	BRA1	YAW WASHOUT	1.0
194	BAXA3	X INT WASHOUT	1.0
195	BAYA3	Y INT WASHOUT	1.0
196	BAZA3	Z INT WASHOUT	1.0
197	BPA3	ROLL INT WASHOUT	1.0
198	BRA3	YAW INT WASHOUT	1.0

VITA

Gisele Guimarães

Candidate for the Degree of

Master of Science

Thesis: SIMULATION AND 3D ANIMATION OF A SIX-DEGREE-OF-FREEDOM MOTION BASE SYSTEM FOR FLIGHT SIMULATORS

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Goiânia, Goiás, Brasil, January 31, 1961, the daughter of Jerson D. Guimarães and Balbina A. S. Guimarães.

Education: Graduated from Colégio de Aplicação da Universidade Federal de Goiás, Goiânia, Goiás, Brasil, in December 1978; received the title of Engenheiro Eletricista from Universidade Federal de Goiás, Goiânia, Goiás, Brasil, in October 1984; completed requirements for the Master of Science degree at Oklahoma State University in December, 1988.

Professional Experience: Software implementation and student assistance for a graduate course at Universidade Católica de Goiás, Goiânia, Goiás, Brasil, March, 1985, to December, 1985.

Membership in Honorary Societies: Tau Beta Pi and Eta Kappa Nu.