Dynamic Systems Analysis of a Vehicle Suspension System

Bruce M. Skeldon
University of Central Florida

DYNAMIC SYSTEMS ANALYSIS OF A VEHICLE SUSPENSION SYSTEM

BY

BRUCE MARK SKELDON
B.S., University of Notre Dame, 1978

RESEARCH REPORT

Submitted in partial fulfillment of the requirements
for the degree of Master Of Science in Engineering in
the Graduate Studies Program of the College of Engineering
University of Central Florida
Orlando, Florida

Summer Term
1986

ABSTRACT

This research report deals with one system in a Computer Aided Instruction package in the Dynamic Systems and Control Theory fields for college undergraduate students. It uses the computer to numerically solve the two degree of freedom equations of motion for a vehicle suspension system. Numerical solutions to the system differential equations are used to drive an animated display of the vehicle's motion (vertical displacement and rotation about the center of gravity) on the video display terminal.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

The Computer Engineering Department is developing a set of software programs to be used for Computer Aided Instruction in the control theory and dynamic systems analysis fields. This research report is a software program in the dynamic systems analysis field. This program describes a mechanical system consisting of an automobile suspension system. Other projects in the Computer Aided Instruction package include a two tank fluid system and a pendulum on a cart mechanical system.

The software programs are designed to run on an IBM AT computer equipped with graphics hardware. The software design goals were to use a high-level design language and an off-the-shelf software graphics package to perform the animation. The TURBO PASCAL language was chosen as the design language. A high-level language was chosen over the IBM assembly language so that other students could design future systems more easily. It was assumed that more students know a high-level design language than know the IBM assembly language.

Dynamic systems analysis of a vehicle suspension system is intended to provide a visual representation of the effects of initial conditions, external loading and choice of system parameters on the motion of a stationary automobile. Computer graphics are used to perform an animation of the system as the system equations are solved. A student can define the automobile with an initial displacement and an initial angle and then watch the automobile movement as it reaches steady state. Plots can be generated to show how the system reached steady state.

Chapter 1 defines the problem statement. Chapter 2 discusses the different numerical methods tested and the method for algorithm verification. Chapter 3 discusses different graphics packages that were investigated along with the animation method selected. Chapter 4 discusses the various ways the animation process was optimized. A sample simulation is discussed in Chapter 5.

Software listings are contained in Appendix 1 and Appendix 2. A user's manual and sample output are contained in Appendix 3.

CHAPTER I

PROBLEM DESCRIPTION

The design for this research report consists of three stages. The first stage involves drawing the free body diagram of the automobile and defining the system equations. The next stage in the design is the use of a numerical analysis method to solve the system equations. The third step consists of taking the output from the numerical analysis and driving the computer graphics animation.

System Model

Equations were defined to describe the motion of an automobile resting on the road with no horizontal motion. The automobile has two degrees of freedom. There is movement in the vertical direction and movement about the center of gravity. The free body diagram in Figure 1 defines the forces acting on the front and rear of the automobile. The equations are defined assuming small rotation TH(t). The following definitions apply throughout this paper:

TH(t)	: Angular Rotation Of Vehicle (radians)
X(t)	: Vertical Displacement of Vehicle (IN)

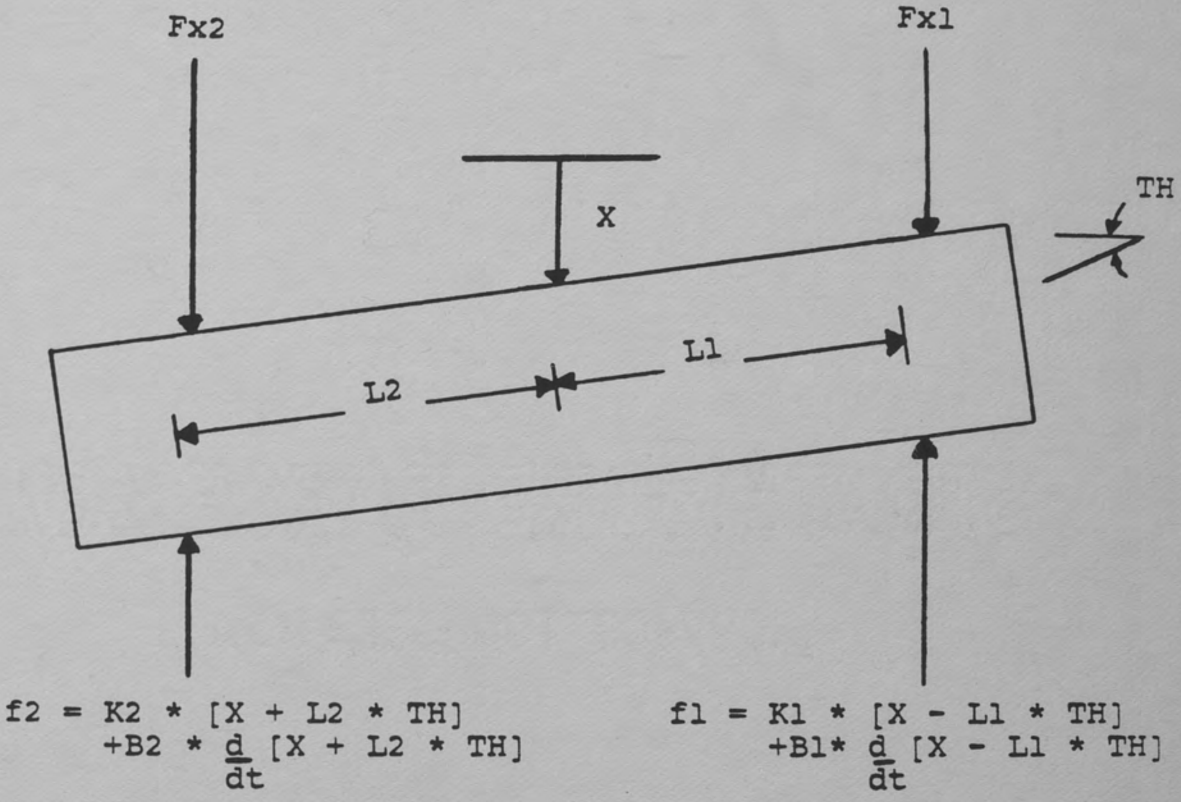


Figure 1. Free Body Diagram of Forces Acting on an Automobile

Fx1	:	External Force on Rear (LBS)
Fx2	:	External Force on Front (LBS)
K1	:	Spring Constant on Rear of Vehicle (LBS/IN)
K2	:	Spring Constant on Front of Vehicle (LBS/IN)
B1	:	Damping Constant on Rear Shock (LBS-SEC/IN)
B2	:	Damping Constant on Front Shock (LBS-SEC/IN)
L1	:	Length of Vehicle from Center to Rear (IN)
L2	:	Length of Vehicle from Center to Front (IN)
M	:	Mass Of Automobile (SLUGS)
I	:	Moment of Inertia

The two differential equations are derived by first summing the forces in the X direction and then summing the torques about the center of mass. The positive X direction is down and the positive TH direction is counter-clockwise. Adding the forces in the X direction yields the following equation:

$$M*\ddot{X} = - f1 - f2 + Fx1 + Fx2$$

where

$$f1 = K1*(X - L1*TH) + B1*(\dot{X} - L1*\dot{TH})$$

$$f2 = K2*(X + L2*TH) + B2*(\dot{X} + L2*\dot{TH})$$

Adding the torques around the center of mass yields the following equation:

$$I*\ddot{TH} = -f2*L2 + f1*L1 + Fx2*l2 - Fx1*L1$$

where f1 and f2 are defined above. After combining like terms the two second-order differential equations that describe the automobile suspension system are defined as follows:

$$\ddot{X} = (1/M)*[Fx1 + Fx2 - (B1 + B2)*\dot{X} - (K1 + K2)*X -$$
$$(B2*L2 - B1*L1)*\dot{TH} - (K2*L2 - K1*L1)*TH]$$

$$\ddot{TH} = (1/I)*[Fx2*L2 - Fx1*L1 - (B1*L1**2 + B2*L2**2)*\dot{TH}$$
$$- (K1*L1**2 + K2*L2**2)*TH - (B2*L2 - B1*L1)*\dot{X}$$
$$- (K2*L2 - K1*L1)*X]$$

Solve System Equations

The next step in the design was using a numerical method to solve the two second-order differential equations derived in the System Model section. By using a numerical analysis method an infinite combination of initial conditions, external loading, and system parameters can be used as inputs and a solution to the equations still obtained. The numerical method chosen was the fourth-order Runge-Kutta algorithm. This algorithm is the most widely used algorithm for solving differential equations.

Animation

The output from the numerical analysis method is used to drive an animation on the video display terminal. Output from the numerical analysis can rotate and translate the automobile depending on the system parameters. A design was required that can use the calculated X and TH values from the numerical method to accuratly redraw the automobile relatively fast.

CHAPTER II

NUMERICAL SOLUTION OF SYSTEM EQUATIONS

The fourth-order Runge-Kutta algorithm was chosen as the numerical analysis method. This algorithm was chosen because it is the most widely accepted algorithm used for solving differential equations. Two different fourth-order Runge-Kutta methods were attempted. The first method tried is known as the classical fourth-order Runge-Kutta method. The second fourth-order algorithm tried is known as the Runge-Kutta-Gill algorithm.

The Runge-Kutta algorithms are used to solve first-order differential equations or sets of first-order differential equations. The two differential equations defined in the problem description section are both second-order. Therefore, a set of first-order differential equations must be defined before the Runge-Kutta algorithm can be applied. The two second-order equations were converted to four first-order differential equations.

A new state variable, called Z1, was defined and set equal to X. A second state variable, called Z2, was also defined and equated to the first derivative of Z1. The two equations were defined as follows:

$$Z1 = X$$

$$Z2 = \dot{Z1} = \dot{X}$$

The desired two first-order differential equations were obtained by taking the derivative of both Z1 and Z2. These two first-order equations resulted from the one second-order equation defining X. The resulting two first-order equations were defined as follows:

$$\dot{Z1} = \dot{X} = Z2$$

$$\dot{Z2} = \ddot{Z1} = \ddot{X}$$

$$= (1/M)*[Fx1 + Fx2 - (B1 + B2)*Z2 - (K1 + K2)*Z3$$

$$- (B2*L2 - B1*L1)*Z4 - (K2*L2 - K1*L1)*Z3]$$

In a similiar manner the new state variables Z3 and Z4 were derived from the one second-order equation defining theta (TH). The resulting four first-order differential equations that were used in the Runge-Kutta algorithm are defined as follows:

$$\dot{Z1} = Z2$$

$$\dot{Z2} = (1/M)*[Fx1 + Fx2 - (B1 + B2)*Z2 - (K1 + K2)*Z3 -$$

$$(B2*L2 - B1*L1)*Z4 - (K2*L2 - K1*L1)*Z3]$$

$$\dot{Z3} = Z4$$

$$\dot{Z4} = (1/I)*[Fx2*L2 - Fx1*L1 - (B1*L1**2 + B2*L2**2)*Z4$$

$$- (K1*L1**2 + K2*L2**2)*Z3 - (B2*L2 - B1*L1)*Z2$$

$$- (K2*L2 - K1*L1)*Z1]$$

Classical Runge-Kutta Algorithm

As mentioned previously two different fourth-order Runge-Kutta algorithms were tried. The first method was the classical Runge-Kutta algorithm. This fourth-order algorithm defines a new slope as the sum of the old slope and a weighted sum of four new intermediate slopes. The intermediate slope values were defined as K1#, K2#, K3# and K4# where # is the associated state variable Z1, Z2, Z3 or Z4. The classical fourth-order Runge-Kutta algorithm for the Z1 state variable is defined as follows:

```
K1Z1 = Z2[i]
K2Z1 = Z2[i] + 0.5*H*K1Z2
K3Z1 = Z2[i] + 0.5*H*K2Z2
K4Z1 = Z2[i] + H*K3Z2
Z1[i + 1] = Z1[i] + (H/6)*[K1Z1 + 0.5*K2Z1 +
                             0.5*K3Z1 + K4Z1]
```

In the above, H is defined as the step size. This algorithm was expanded to generate similiar equations for the three other state variables. Refer to Appendix 1 for the PASCAL program written to test the classical fourth-order Runge-Kutta algorithm.

This algorithm was tested using a step size of both H = 0.001 and H = 0.01. There was no difference in output between the different step sizes therefore, a step size of H = 0.01 was used in the final design.

Algorithm Verification

An extensive amount of time was spent to assure that the algorithm solved the set of differential equations correctly. Two different methods were used to verify the results. The first method consisted of generating a TUTSIM (APPLIED i 1985) model to simulate the equations. TUTSIM is a simulation language that can be used to solve systems described by differential equations or sets of differential equations. The TUTSIM model is contained in Figure 2. The second method consisted of assigning a set of known parameters and initial conditions and solving the differential equations. This equation was then used to generate the analytical solution. The output from the analytical solution was compared to the output generated by the Runge-Kutta algorithm for both the Z1 and Z2 state variables. The results are contained in Table 1 and Table 2 respectively. The output consists of the time value (H), the two state variables of interest Z1 and Z2, the analytical solution (re) and the difference (dif) between the state variable, Z1 or Z2, and the analytical solution (re). It was shown that for the compared points the difference was near zero. For the example contained in Appendix 1 the following values were assigned:

```
Model File: car.sim
Date:       6 /     23 /  1986
Time:      16 :     29
Timing:    0.0100000  ,DELTA  ;    4.0000      ,RANGE
PlotBlocks and Scales:
Format:
     BlockNo,  Plot-MINimum,   Plot-MAXimum;  Comment
Horz:     0 ,   0.0000      ,    4.0000     ; Time
  Y1:    33 ,  -5.0000      ,    5.0000     ; X
  Y2:    32 ,  -5.0000      ,    5.0000     ; X1DOT
  Y3:       ,               ,               ;
  Y4:       ,               ,               ;

   0.0000           1 CON                           ;FX1
   0.0000           2 CON                           ;FX2
   0.0000           3 CON                           ;B1
   0.0000           4 CON                           ;B2
 400.0000           5 CON                           ;K1
 400.0000           6 CON                           ;K2
  50.0000           7 CON                           ;L1
  50.0000           8 CON                           ;L2
   7.7720           9 CON                           ;M
  15.000E+03       10 CON                           ;I
                   11 SUM      3      4             ;B1+B2
                   12 SUM      5      6             ;K1+K2
                   13 MUL      4      8             ;B2*L2
                   14 MUL      3      7             ;B1*L1
                   15 MUL      6      8             ;K2*L2
                   16 MUL      5      7             ;K1*L1
                   17 MUL      2      8             ;FX2*L2
                   18 MUL      1      7             ;FX1*L1
                   19 MUL      7     14             ;B1*L1**2
                   20 MUL      8     13             ;B2*L2**2
                   21 MUL      7     16             ;K1*L1**2
                   22 MUL      8     15             ;K2*L2**2
                   23 SUM     13    -14             ;B2*L2-B1*L1
                   24 SUM     15    -16             ;K2*L2-K1*L1
                   25 SUM     19     20             ;B1*L1**2+B2*L2**2
                   26 SUM     21     22             ;K1*L1**2+K2*L2**2
                   30 SUM      1      2    -34
                         -35    -36    -37
   7.7720          31 ATT     30
   0.0000          32 INT     31                    ;X1DOT
   5.0000          33 INT     32                    ;X
                   34 MUL     11     32             ;(B1+B2)*X1DOT
                   35 MUL     12     33             ;(K1+K2)*X
                   36 MUL     23     40             ;(B2*L2-B1*L1)*THE2D
                   37 MUL     24     41             ;(K2*L2-K1*L1)*THE
                   38 SUM     17    -18    -42
                         -43    -44    -45
  15.000E+03       39 ATT     38
   0.0000          40 INT     39                    ;THE1DOT
   0.0000          41 INT     40                    ;THE
                   42 MUL     25     40             ;(B1*L1**2+B2*L2**2)
                   43 MUL     26     41             ;(K1*L1**2+K2*L2**2)
                   44 MUL     23     32             ;(B2*L2-B1*L1)*X1DOT
                   45 MUL     24     33             ;(K2*L2-K1*L1)*X
```

Figure 2.   TUTSIM Model

TABLE 1

COMPARISON OF CLASSICAL RUNGE-KUTTA ALGORITHM AND ANALYTICAL SOLUTION [Z1]

H	Z1	Re	Dif
0.00	5.0000	5.0000	0.0000
0.10	2.6400	2.6400	0.0000
0.20	-2.2122	-2.2122	0.0000
0.30	-4.9761	-4.9761	0.0000
0.40	-3.0424	-3.0424	0.0000
0.50	1.7633	1.7633	0.0000
0.60	4.9044	4.9044	0.0000
0.70	3.4157	3.4157	0.0000
0.80	-1.2975	-1.2975	0.0000
0.90	-4.7859	-4.7859	0.0000
1.00	-3.7563	-3.7563	0.0000
1.10	0.8193	0.8193	0.0000
1.20	4.6214	4.6214	0.0000
1.30	4.0609	4.0608	0.0000
1.40	-0.3332	-0.3333	0.0001
1.50	-4.4127	-4.4128	0.0000
1.60	-4.3266	-4.3265	0.0000
1.70	-0.1561	-0.1560	0.0001
1.80	4.1618	4.1618	0.0001
1.90	4.5508	4.5508	0.0000
2.00	0.6439	0.6438	0.0001
2.10	-3.8709	-3.8710	0.0001
2.20	-4.7315	-4.7315	0.0000
2.30	-1.1255	-1.1254	0.0001
2.40	3.5430	3.5431	0.0001
2.50	4.8669	4.8668	0.0000
2.60	1.5963	1.5962	0.0001
2.70	-3.1812	-3.1813	0.0001
2.80	-4.9556	-4.9556	0.0000
2.90	-2.0518	-2.0517	0.0001
3.00	2.7889	2.7890	0.0001
3.10	4.9968	4.9969	0.0000
3.20	2.4877	2.4876	0.0001
3.30	-2.3698	-2.3700	0.0001
3.40	-4.9902	-4.9903	0.0000
3.50	-2.8998	-2.8997	0.0001
3.60	1.9281	1.9283	0.0002
3.70	4.9358	4.9359	0.0000
3.80	3.2841	3.2839	0.0001
3.90	-1.4679	-1.4681	0.0002
4.00	-4.8342	-4.8342	0.0001

TABLE 2

COMPARISON OF CLASSICAL RUNGE-KUTTA
ALGORITHM AND ANALYTICAL SOLUTION [Z2]

H	Z2	Re	Dif
0.00	0.0000	0.0000	0.0000
0.10	-43.0808	-43.0808	0.0000
0.20	-45.4927	-45.4926	0.0000
0.30	-4.9588	-4.9587	0.0001
0.40	40.2562	40.2563	0.0001
0.50	47.4688	47.4687	0.0001
0.60	9.8701	9.8699	0.0003
0.70	-37.0461	-37.0463	0.0002
0.80	-48.9902	-48.9902	0.0001
0.90	-14.6869	-14.6865	0.0004
1.00	33.4811	33.4814	0.0004
1.10	50.0424	50.0424	0.0000
1.20	19.3630	19.3625	0.0005
1.30	-29.5954	-29.5959	0.0005
1.40	-50.6153	-50.6153	0.0000
1.50	-23.8536	-23.8531	0.0006
1.60	25.4262	25.4269	0.0007
1.70	50.7033	50.7034	0.0001
1.80	28.1158	28.1151	0.0006
1.90	-21.0135	-21.0143	0.0008
2.00	-50.3057	-50.3059	0.0002
2.10	-32.1086	-32.1079	0.0007
2.20	16.3995	16.4004	0.0010
2.30	49.4262	49.4265	0.0003
2.40	35.7939	35.7932	0.0007
2.50	-11.6284	-11.6295	0.0011
2.60	-48.0733	-48.0737	0.0005
2.70	-39.1363	-39.1356	0.0007
2.80	6.7459	6.7472	0.0013
2.90	46.2599	46.2605	0.0006
3.00	42.1038	42.1031	0.0007
3.10	-1.7989	-1.8003	0.0014
3.20	-44.0034	-44.0042	0.0008
3.30	-44.6681	-44.6675	0.0006
3.40	-3.1655	-3.1639	0.0015
3.50	41.3254	41.3264	0.0010
3.60	46.8044	46.8039	0.0005
3.70	8.0994	8.0978	0.0016
3.80	-38.2516	-38.2528	0.0012
3.90	-48.4925	-48.4921	0.0004
4.00	-12.9558	-12.9541	0.0017

```
Fx1 = Fx2 = 0                 B1 = B2 = 0

K1 = K2 = 400                 L1 = L2 = 50

Z1[0] = 5                     Z2[0] = Z3[0] = Z4[0] =0
```

With the above parameters it can be shown that

$$X(t) = 5 * \cos [(K1 + K2)/M] * t$$

is a solution to the second-order differential equation for X with the given initial conditions.

The above equation was used to generate the analytical values for the Z1 state variable that was then compared to the Z1 output from the classical Runge-Kutta algorithm. To compare the output for the Z2 state variable we need the first derivative of the above equation. The first derivative results in the following equation:

$$\dot{X}(t) = - 5 * [(K1 + K2)/M] * \sin [(K1 + K2)/M] * t$$

This equation was used to generate the analytical values for the Z2 state variable that were compared to the Z2 values generated by the classical Runge-Kutta algorithm.

The above choice of parameters and initial conditions were selected because they describe a system that is oscillatory and will never achieve steady state. It was felt that after an extended period of time, if the analytical solution still agreed with the output of the Runge-Kutta algorithm then the algorithm was correctly solving the set of differential equations.

The second method for the algorithm verification consisted of generating a TUTSIM simulation program and comparing the output of the TUTSIM simulation to the output of the Runge-Kutta algorithm. The two differential equations defined in the problem description section were used to generate a TUTSIM block diagram. The block diagram was converted to the TUTSIM model contained in Figure 2. The TUTSIM simulation output in Table 3 was generated using the TUTSIM program in Figure 2. The program used the same set of parameters and the same set of initial conditions that were used in the analytical solution. The output of the TUTSIM simulation was similiar to the output of the Runge-Kutta algorithm. The difference was as much as .04. This means that the TUTSIM output was also different from the analytical solution by as much as .04. The outputs were reasonably close and the TUTSIM output was considered additional verification of the classical Runge-Kutta solutions. The TUTSIM model was run using a step size of both H = 0.001 and H = 0.01. The TUTSIM output is included in Table 3.

Runge-Kutta-Gill Algorithm

The Runge-Kutta-Gill algorithm was the second numerical method tried. This method is similiar to the classical approach with the differences being the weight factors assigned to the intermediate slopes. The algorithm

TABLE 3

TUTSIM OUTPUT

```
Model File: car.sim
Date:       6 /    23 /  1986
Time:      16 :    30
Timing:    0.0100000  ,DELTA  ;   4.0000        ,RANGE
PlotBlocks and Scales:
Format:
     BlockNo,  Plot-MINimum,   Plot-MAXimum;  Comment
Horz:      0 ,   0.0000       ,   4.0000      ; Time
  Y1:     33 ,  -5.0000       ,   5.0000      ; X
  Y2:     32 ,  -5.0000       ,   5.0000      ; X1DOT
  Y3:        ,                ,               ;
  Y4:        ,                ,               ;

   0.0000           5.0000          0.0000
   0.1000000        2.6379        -43.4192
   0.2000000       -2.2619        -45.5661
   0.3000000       -5.0117         -4.3478
   0.4000000       -2.9932         41.0306
   0.5000000        1.8750         47.3839
   0.6000000        4.9614          8.6442
   0.7000000        3.3275        -38.3432
   0.8000000       -1.4741        -48.8640
   0.8999990       -4.8754        -12.8855
   0.9999990       -3.6385         35.3759
   1.1000           1.0619         49.9954
   1.2000           4.7542         17.0411
   1.3000           3.9238        -32.1494
   1.4000          -0.6414040     -50.7692
   1.5000          -4.5987        -21.0809
   1.6000          -4.1814         28.6866
   1.7000           0.2156010      51.1793
   1.8000           4.4098         24.9757
   1.9000           4.4094        -25.0120
   2.0000           0.2124470     -51.2222
   2.1000          -4.1890        -28.6974
   2.2000          -4.6060         21.1516
   2.3000          -0.6396580      50.8971
   2.4000           3.9376         32.2187
   2.5000           4.7698        -17.1330
   2.6000           1.0629        -50.2055
   2.7000          -3.6576        -35.5142
   2.8000          -4.8997         12.9847
   2.9000          -1.4793         49.1520
   3.0000           3.3508         38.5597
   3.1000           4.9945         -8.7365
   3.2000           1.8856        -47.7436
   3.3000          -3.0194        -41.3331
   3.4000          -5.0536          4.4185
   3.5000          -2.2791         45.9897
   3.6000           2.6658         43.8139
   3.7000           5.0764         -0.0617771
   3.8000           2.6569        -43.9023
   3.9000          -2.2925        -45.9837
   4.0000          -5.0628         -4.3025
```

used to find the state variable Z1 was defined as follows:

```
    K1Z1 = Z2[i]

    K2Z1 = Z2[i] + 0.5*H*K1Z2

    K3Z1 = Z2[i] + c3*H*K1Z2 + c1*H*K2Z2

    K4Z1 = Z2[i] - (1 / SQRT(2))*H*K2Z2 + c2*H*K3Z2

    Z1[i + 1] = Z1[i] + (H/6)*[K1Z1 + 2*c1*K2Z1 +
                                   2*c2*K3Z1 + K4Z1]

  where

      c1 = 1 - (1 / SQRT(2))

      c2 = 1 + (1 / SQRT(2))

      c3 = - 0.5 + (1 / SQRT(2))
```

The PASCAL program used to run the Runge-Kutta-Gill algorithm is contained in Appendix 1. The same verification methods used to verify the classical Runge-Kutta algorithm were used to verify the Runge-Kutta-Gill algorithm. The results were similiar to the results obtained by the classical method and are contained in tables 4 and 5. Therefore, for reasons that are discussed in a later section, the classical Runge-Kutta algorithm was decided upon as the algorithm to solve the set of four first-order differential equations.

TABLE 4

COMPARISON OF RUNGE-KUTTA-GILL
ALGORITHM AND ANALYTICAL SOLUTION [Z1]

H	Z1	Re	Dif
0.00	5.0000	5.0000	0.0000
0.10	2.6400	2.6400	0.0000
0.20	-2.2122	-2.2122	0.0000
0.30	-4.9761	-4.9761	0.0000
0.40	-3.0424	-3.0424	0.0000
0.50	1.7633	1.7633	0.0000
0.60	4.9044	4.9044	0.0000
0.70	3.4157	3.4157	0.0000
0.80	-1.2975	-1.2975	0.0000
0.90	-4.7859	-4.7859	0.0000
1.00	-3.7563	-3.7563	0.0000
1.10	0.8193	0.8193	0.0000
1.20	4.6214	4.6214	0.0000
1.30	4.0609	4.0608	0.0000
1.40	-0.3332	-0.3333	0.0001
1.50	-4.4127	-4.4128	0.0000
1.60	-4.3266	-4.3265	0.0000
1.70	-0.1561	-0.1560	0.0001
1.80	4.1618	4.1618	0.0001
1.90	4.5508	4.5508	0.0000
2.00	0.6439	0.6438	0.0001
2.10	-3.8709	-3.8710	0.0001
2.20	-4.7315	-4.7315	0.0000
2.30	-1.1255	-1.1254	0.0001
2.40	3.5430	3.5431	0.0001
2.50	4.8669	4.8668	0.0000
2.60	1.5963	1.5962	0.0001
2.70	-3.1812	-3.1813	0.0001
2.80	-4.9556	-4.9556	0.0000
2.90	-2.0518	-2.0517	0.0001
3.00	2.7889	2.7890	0.0001
3.10	4.9968	4.9969	0.0000
3.20	2.4877	2.4876	0.0001
3.30	-2.3698	-2.3700	0.0001
3.40	-4.9902	-4.9903	0.0000
3.50	-2.8998	-2.8997	0.0001
3.60	1.9281	1.9283	0.0002
3.70	4.9358	4.9359	0.0000
3.80	3.2841	3.2839	0.0001
3.90	-1.4679	-1.4681	0.0002
4.00	-4.8342	-4.8342	0.0001

TABLE 5

COMPARISON OF RUNGE-KUTTA-GILL ALGORITHM AND ANALYTICAL SOLUTION [Z2]

H	Z2	Re	Dif
0.00	0.0000	0.0000	0.0000
0.10	-43.0808	-43.0808	0.0000
0.20	-45.4927	-45.4926	0.0000
0.30	-4.9588	-4.9587	0.0001
0.40	40.2562	40.2563	0.0001
0.50	47.4688	47.4687	0.0001
0.60	9.8701	9.8699	0.0003
0.70	-37.0461	-37.0463	0.0002
0.80	-48.9902	-48.9902	0.0001
0.90	-14.6869	-14.6865	0.0004
1.00	33.4811	33.4814	0.0004
1.10	50.0424	50.0424	0.0000
1.20	19.3630	19.3625	0.0005
1.30	-29.5954	-29.5959	0.0005
1.40	-50.6153	-50.6153	0.0000
1.50	-23.8536	-23.8531	0.0006
1.60	25.4262	25.4269	0.0007
1.70	50.7033	50.7034	0.0001
1.80	28.1158	28.1151	0.0006
1.90	-21.0135	-21.0143	0.0008
2.00	-50.3057	-50.3059	0.0002
2.10	-32.1086	-32.1079	0.0007
2.20	16.3995	16.4004	0.0010
2.30	49.4262	49.4265	0.0003
2.40	35.7939	35.7932	0.0007
2.50	-11.6284	-11.6295	0.0011
2.60	-48.0733	-48.0737	0.0005
2.70	-39.1363	-39.1356	0.0007
2.80	6.7459	6.7472	0.0013
2.90	46.2599	46.2605	0.0006
3.00	42.1038	42.1031	0.0007
3.10	-1.7989	-1.8003	0.0014
3.20	-44.0034	-44.0042	0.0008
3.30	-44.6681	-44.6675	0.0006
3.40	-3.1655	-3.1639	0.0015
3.50	41.3254	41.3264	0.0010
3.60	46.8044	46.8039	0.0005
3.70	8.0994	8.0978	0.0016
3.80	-38.2516	-38.2528	0.0012
3.90	-48.4925	-48.4921	0.0004
4.00	-12.9558	-12.9541	0.0017

CHAPTER III

COMPUTER GRAPHICS

Three different graphics packages were investigated before deciding on the TURBO TOOLBOX graphics package. The three packages were LENIPEN, TURBO PASCAL and the TURBO PASCAL TOOLBOX.

LENIPEN

The LENIPEN package consists of software that allows drawing in a free hand style of pictures on the video terminal. The LENIPEN package was designed to interface directly with the BASIC software package purchased with the IBM AT. The LENIPEN package could save the graphic screen in a binary file that could be redisplayed using standard BASIC commands (BSAVE, BLOAD). The research was intended to be designed using the TURBO PASCAL language. In order to use LENIPEN with the TURBO PASCAL language, a program to read and display the binary file stored using BASIC was required. This was considered a minor task. Therefore, the LENIPEN package was investigated further.

The automobile animation design required the ability to redisplay an automobile depending on the X and TH values calculated by the Runge-Kutta algorithm. To perform this animation the current automobile position must be known and the new position must be calculated. The LENIPEN package easily allowed whole screens or "windows" to be saved as binary files. The original intent for LENIPEN was to draw the car in different positions and then save the various displays to different binary files. This process would only allow a discrete number of new car positions to be drawn. This was unacceptable for the chosen design. A possible alternative for performing the automobile animation using LENIPEN was to determine on a pixel-by-pixel basis where every point was within the binary file and then change the binary value to either draw or undraw the automobile. The amount of time and effort involved in this task was considered too extensive; therefore, an alternate approach was desired.

The LENIPEN software also contained a package called LENIMATION. This package allowed the user to use the screen for animation. This was similiar to what was required by this design. Therefore, the LENIMATION package was investigated. It was determined that the LENIMATION package would take a finite number of user drawn screens and display them at an interval defined by the user. The set of screens was repeated as many times as desired. This method would be acceptable for animating a bird in flight where the bird was required to be shown in only a finite number of positions and then repeated. A flying bird could be animated by displaying the bird with approximately six different wing and body positions. These six screens would be displayed in sequence an infinite number of times. In the vehicle suspension system the number of different automobile positions was infinite. Therefore, the entire LENIPEN package was no longer considered a viable option for this design.

TURBO PASCAL Graphics Commands

The next approach was to use the graphics capabilities of the TURBO PASCAL language. The package consists of commands to place the screen in graphics mode, draw line commands, and draw point commands. These routines, along with a combination of assembly language routines, could have been used in the design. It was decided not to use this method because a design goal of this research report was to use a high-level design language and a standard off-the-shelf software graphics package that other graduate students could easily adapt to future research reports or thesis work. Therefore, the TURBO TOOLBOX package was chosen as the graphics software for the design.

TURBO TOOLBOX Package

The graphics design goal was to display an automobile as accurately and quickly as possible depending on the calculated X and TH positions. The automobile was defined by nineteen (19) pairs of X-Y points. Figure 3 shows the 19 X-Y points and their location on the automobile. The wheels and inner wheels (hubcaps) were defined by 109 and 52 pairs of X-Y points respectively. The WHLX, WHLY, HUBX and HUBY arrays were the wheel X and Y points and the hubcap X and Y points and are defined in the software listing contained in Appendix 2.

Standard TURBO TOOLBOX graphics routines were used to display the headings and borders that are contained on the graphics screen. A modified drawline procedure was used to draw the lines between the X-Y points. A new X and TH value was calculated from the Runge-Kutta algorithm. These new values could both rotate and translate the automobile's initial position. A capability was needed to perform both operations simultaneously and then redisplay the new position. The following equations when appled to all 19 X-Y points would accomplish both functions.

```
Pxl = centerx + cos(TH) * (Pxl[0] - centerx)
                - sin(TH) * (Pyl[0] - centery)
Pyl = centery + X + sin(TH) * (Pxl[0] - centerx)
                    + cos(TH) * (Pyl[0] - centery)
```

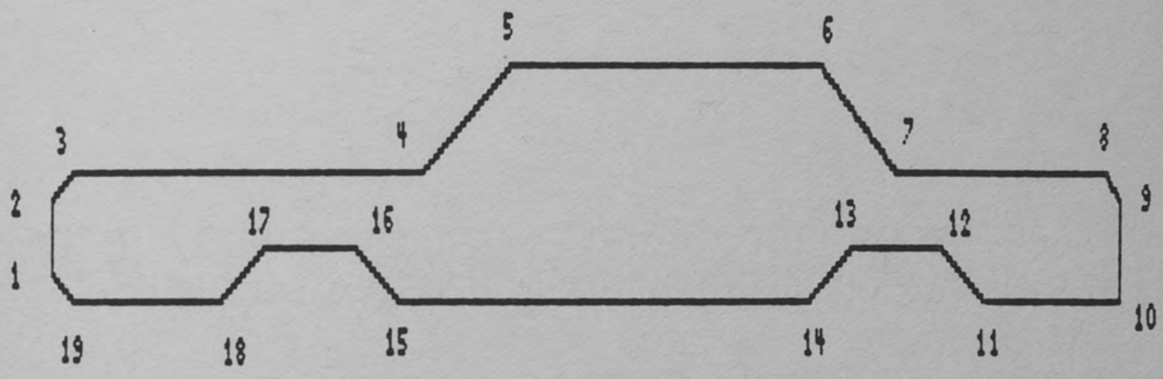


Figure 3. X-Y Data Points (19) that Define the Automobile

The values centerx and centery were the center point of the automobile and the values Px1[0], Py1[0] were the initial values for the automobile x1, y1 point. The rotation was about the center of the automobile and was performed by adding and/or subtracting the sine and cosine values of the new angle TH from the initial points. When the angle was equal to zero no rotation was performed which was the expected result. The translation was in the vertical direction and was accomplished by adding the new value X to the original Y value. For the above equations the following directions were defined and pertain to the graphic display. A positive TH rotated the automobile in a counter clockwise direction. A positive X value lowered the automobile closer to the road surface.

Graphics Animation

The animation was performed by first drawing the automobile in RAM memory and then swapping the RAM memory into the video display area. The initial automobile was drawn using the user defined ititial conditions set on the two state variables Z1 [X] and Z3 [TH]. The automobile was then undrawn by either clearing the RAM memory or by first setting the background color to black and then redrawing the lines using the previous set of 19 X-Y points. The automobile was then displayed in its new position by first calculating the 19 new X-Y points using the new X and TH values and then drawing the lines to connect the points. The RAM memory was agian swapped into the video display area. This process was repeated for as long as desired and gave the impression of continuous movement.

The procedure of having the new X and TH values drive the animation was one of two methods studied.

An alternate approach was tried where the X and TH data points of the Runge-Kutta algorithm were stored in a file until the simulation reached a steady state or N number of data points were calculated. The steady state condition was assumed when both the old and new X and TH values differed by .0001 respectively. N was defined as 3000 points. This method had its advantages and disadvantages over the chosen approach. The advantage of this method was a faster animation, because all the X and TH points were already calculated and the only overhead in the animation process was in the drawing and undrawing the lines that outline the automobile. It was later determined that the actual Runge-Kutta algorithm was not a significant factor in the animation process. This is left for discussion in the next section.

A disadvantage to this approach was the simulation was now a two-step approach, collecting the data and then performing the animation. A second disadvantage was in trying to display an oscillating system for an extended period of time. In the oscillating system the calculation of 3000 data points was time-consuming and required the user to wait before the animation could be displayed. This was considered unsatisfactory.

The second disadvantage resulted when the system had no damping, B1 = B2 = 0. In this situation the system would oscillate forever and never reach steady state. The chosen number of data points, 3000, allowed the animation to run for a long time, but a mechanism was needed to allow the animation to run forever. These two disadvantages along with determining the math did not add a significant amount of overhead resulted in deciding upon the first approach for the final design.

In addition to the software animation, plots were desired of all state variables versus time and the phase plots Xdot versus X and THdot versus TH. This was accomplished using the DRAWAXIS and DRAWPOLYGON procedures contained within the TURBO TOOLBOX package. It was a matter of supplying the two plotted data points in an array to the DRAWPOLYGON procedure. The design defines five arrays of 800 points each to store the first 800 calculated points for time, Z1, Z2, Z3 and Z4. The two desired plot values were used to build an array A that was passed to DRAWPOLYGON for plotting. It was decided that

800 points was sufficient amount of data to generate plots that accurately described the system.

CHAPTER IV

SOFTWARE OPTIMIZATION

The optimum design was having the graphics emulate the automobile in real time. This means that if the set of parameters chosen would reach steady state in N seconds then the graphical display would be at steady state in N seconds. This was the design goal.

The graphics software was designed using two different methods. The first method consisted of using the math algorithm output for the new X and TH values to directly drive the animation process. The second method consisted of saving the X and TH data in files and then using this data in the animation process after the simulation reached steady state or 3000 data points were calculated depending on which occurred first. This design was considered because it was first believed that the math algorithm was a time-consuming process. It was eventually determined that the math algorithm added a minimal amount of overhead to the animation process. This was determined by running the animation with no math algorithm and two hard coded values for X. It was noted that this animation process took as much time as an animation using the math algorithm. It was determined that a large amount of time was spent in the drawing, swapping, undrawing and then swapping portion of the animation. Therefore, a considerable amount of time went into attempting to optimize the graphics portion of the software.

Math Algorithm Selection

As stated previously, two different Runge-Kutta algorithms were tested. A comparison of their outputs contained in tables 1 and 4 indicate that the algorithms have identical outputs. The difference between the two algorithms was that the Runge-Kutta-Gill algorithm had additional math calculations in the calculation of the K3 and K4 terms along with the Z[i+1] term. This additional math overhead was considered unnecessary considering that the outputs were identical. It was for this reason that the classical Runge-Kutta algorithm was chosen as the algorithm to calculate the new X and TH values.

Modified TURBO TOOLBOX Commands

The TURBO TOOLBOX commands CLEARSCREEN and COPYSCREEN were used in the initial design of the graphics display. The TOOLBOX software was investigated and it was determined that the CLEARSCREEN command was a procedure that consisted of the TURBO PASCAL FILMEM command. This command will write into memory a user-specified value. In the TOOLBOX software the FILMEM command stored zeroes in memory. The zeroes, when displayed, will clear the screen. The COPYSCREEN command is a procedure that consists of the TURBO PASCAL MOVEMEM command. This command moves a block of memory from one address to another address. The procedure was used to copy data from the RAM memory into the video display area. This data was then displayed on the screen. The automobile was limited to where it could be displayed on the screen. It could never be displayed below the road surface. It also could not be lifted above a point that was twenty-five (25) inches above its steady state position. Therefore, only about one-fourth of the screen changed when the car was redrawn. It was decided to use the FILMEM and MOVEMEM commands and only clear or copy one-fourth of the screen. This was a considerable time savings and increased the animation speed considerably. The FILMEM command was chosen to clear the memory to undraw the car instead of changing the background color to black and then using DRAWLINEDIRECT to undraw the car because it was determined to be faster. This was determined by using both methods to undraw the car using the video display area only.

The DRAWLINE procedure was also investigated. It was determined that only a portion of this procedure was actually required by the graphics in this project. The additional software was considered overhead and unnecessary. Therefore, a portion of the DRAWLINE procedure was condensed and placed in the main software package. The new procedure was called DRAWLINEDIRECTT and was used to draw all the lines outlining the automobile.

The wheels were first drawn using the DRAWCIRCLE procedure. There were two problems when using this procedure. The first problem was the time it takes to draw a circle using this procedure. Four different circles were needed every time the automobile was redrawn. Therefore, this procedure added a considerable amount of time to the animation process. The second problem was in trying to draw the wheels when the top portion should be hidden by a fender. The TOOLBOX procedure only allowed a full circle. Therefore, a different procedure was needed to draw the wheels. A pixel-by-pixel outline of the tire was designed and the X-Y points are stored in the WHL, HUB arrays. The DP (drawpoint) TOOLBOX procedure was then used to draw the four circles. This method was much quicker than the DRAWCIRCLE procedure and easily allowed for drawing any desired portion of the wheel. The wheels were shown disappearing into the fenders by not drawing the points if the Y value from the array was less than the points Py17 for the front wheel or Py12 for the rear wheel. This was accomplished using the DRAWWHEEL and DRAWHUB procedures in the main software body.

Considerations for Displaying Automobile

Another consideration in speeding up the animation was how often the automobile should be displayed. It was determined that the user could visualize a single pixel change in the automobiles position. Therefore, the automobile was redisplayed only when it was determined that it had changed by at least one pixel. The entire automobile could not be checked and so two points were determined to be the guiding factors for the single pixel change. These two points were chosen as the bottom front point Py19 and the bottom rear point Py10. The initial Py10 and Py19 points were calculated using the initial conditions on Z1 [X] and Z3 [TH] and stored as OLDPY10 and OLDPY19. The math algorithm was then run to generate the new X and TH values. These new values were used to calculate Py10 and Py19. The new Py10 and Py19 values were compared to the old values and if the absolute difference was greater than or equal to one (1) the automobile was redrawn and the new values for Py10 and Py19 were stored for the next comparison. A similiar approach was used to ensure that the automobile was never displayed if it was above or below a certain value. The bottom points Py10 and Py19 were compared to the value for the road surface. If the values would place the automobile below the road surface the car was not redisplayed. The top points Py5 and Py6 were compared to a point that was twenty five pixels above their initial values. The automobile was not redisplayed if the values for Py5 or Py6 would place the automobile outside this window.

CHAPTER V

RESULTS

An example of the output from this research report is contained in this section. This specific example was chosen to show the effects of initial conditions on the solution to the system equations. The parameter values were selected such that the two second-order differential equations were non-coupled. This was achieved by selecting B1 = B2 = 5, K1 = K2 = 400, and L1 = L2 = 50. With this set of parameters the solution for X was not dependent on the solution for TH and the solution for TH was not dependent on the solution for X. The B1 = B2 = 5 parameter was chosen because the desired output was to achieve steady state after at least 700 data points were collected. This was desired for better plots.

The initial conditions were selected on the X [Z1] and TH [Z3] values such that the car was displaced 14 inches closer to the road, X(0) = 14, and with an initial angle of 4 degrees, TH(0) = 4/57.3, in the counter-clockwise direction. The positive X direction is down or closer to the road and the positive TH direction is counter-clockwise or the front end is closer to the road. The parameter values for this example are contained in Figure 4.

This set of initial conditions on X and TH positions the front end of the automobile below the road surface. This position cannot be displayed and an error message was displayed. The error message is DISPLAY LIMIT EXCEEDED. This message was not removed until the automobile was within the display limits. The limits were defined as the road surface and twenty-five inches above the starting center point. The error message was displayed whenever the automobile was calculated as being beyond the display limits. Sample animation outputs are contained in Figure 5 and Figure 6.

The simulation/animation was allowed to run and collect at least 700 data points for the plots. The plot contained in Figure 7 is X versus TIME and shows X starting with an initial condition of X = 14 and eventually reaching a steady state value of X = 0. A similiar plot of TH versus TIME is contained in Figure 9. In Figure 9 the TH value starts with an initial condition of TH = 4 and eventually reaches a steady state value of TH = 0. The system exponentially decays in both plots due to the damping constants B1 and B2. The plots of Xdot and THdot are included as Figure 8 and Figure 10 respectively. Both plots start from initial conditions of zero and both eventually reach a steady state of zero. This was expected because the initial conditions on Xdot and THdot were zero.

The phase plots X versus Xdot and TH versus THdot are contained in Figure 11 and Figure 12 respectively. Both plots spiral into the origin from the associated initial condition. This was the expected result for a system that achieves a steady state condition of X = 0 and TH = 0 when starting from non zero initial conditions for X and TH.

The numerical output in Table 6 is printed and displays the first 350 collected data points in H = .1 increments.

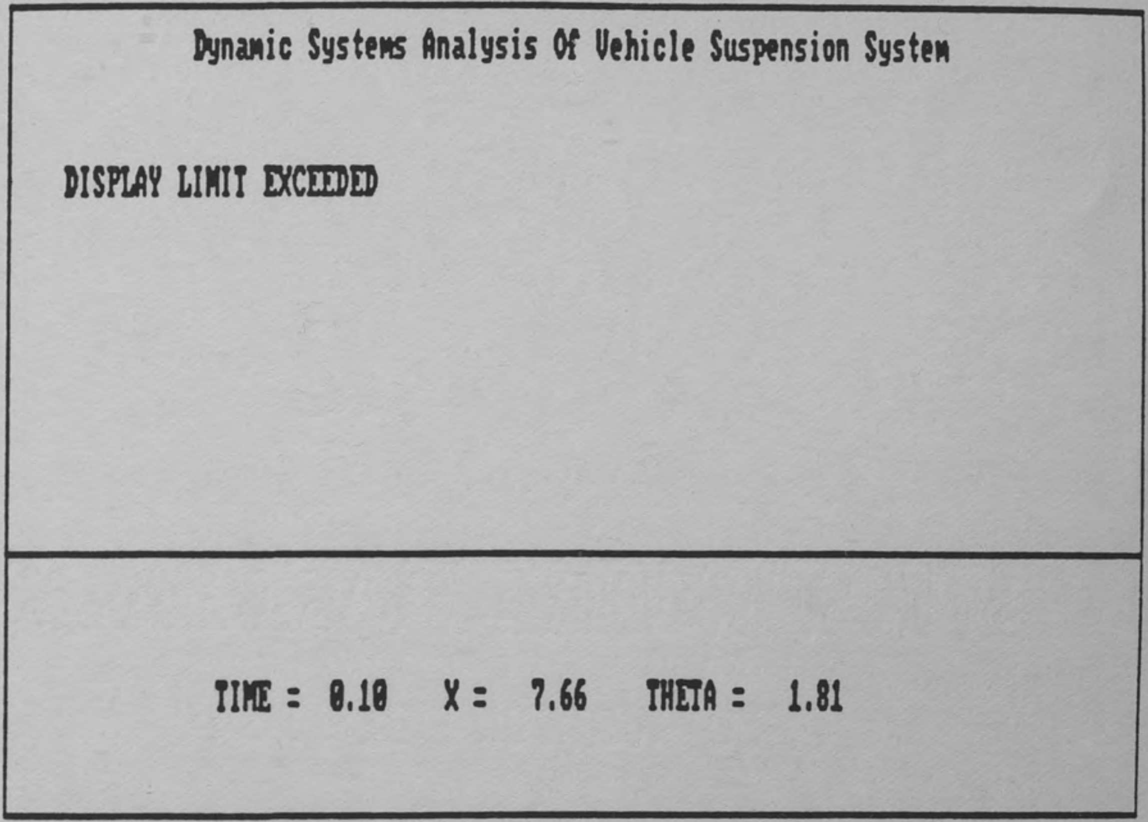
```
This section allows you to change certain variables used
in the automobile simulation. The following variables and
their values are changeable [units are in () ] :

   0: NO CHANGES REQUIRED
   1: Fx1 (force on rear (lbs) [-1500 to 1500])=       0.00
   2: Fx2 (force on front(lbs) [-1500 to 1500])=       0.00
   3: B1(dmping cnst rear(lb-sec/in)[0 to 500])=       5.00
   4: B2(dmping cnst frnt(lb-sec/in)[0 to 500])=       5.00
   5: K1(spring const rear (lb/in)[10 tO 1000])=     400.00
   6: K2(spring const frnt (lb/in)[10 to 1000])=     400.00
   7: L1  (center to rear     (in)   [10 to 75])=      50.00
   8: L2  (center to front    (in)   [10 to 75])=      50.00
   9: M   (auto mass  (lbs)      [1500 to 4000])=    3000.00
  10: X(0) (x init cond   (in)      [-25 to 15])=      14.00
  11: XDOT(0) (x dot init cond (in/sec)         =       0.00
  12: T(0) (theta init cond  (deg)    [-4 to 4])=       4.00
  13: THEDOT(0) (theta dot init con)            =       0.00
  14: CLEAR ALL INITIAL CONDITIONS

      J   (moment of inertia)                   =   15608.81

              ENTER SELECTION AND RETURN >
```

Figure 4.   Case 1 : Parameter Selection Menu

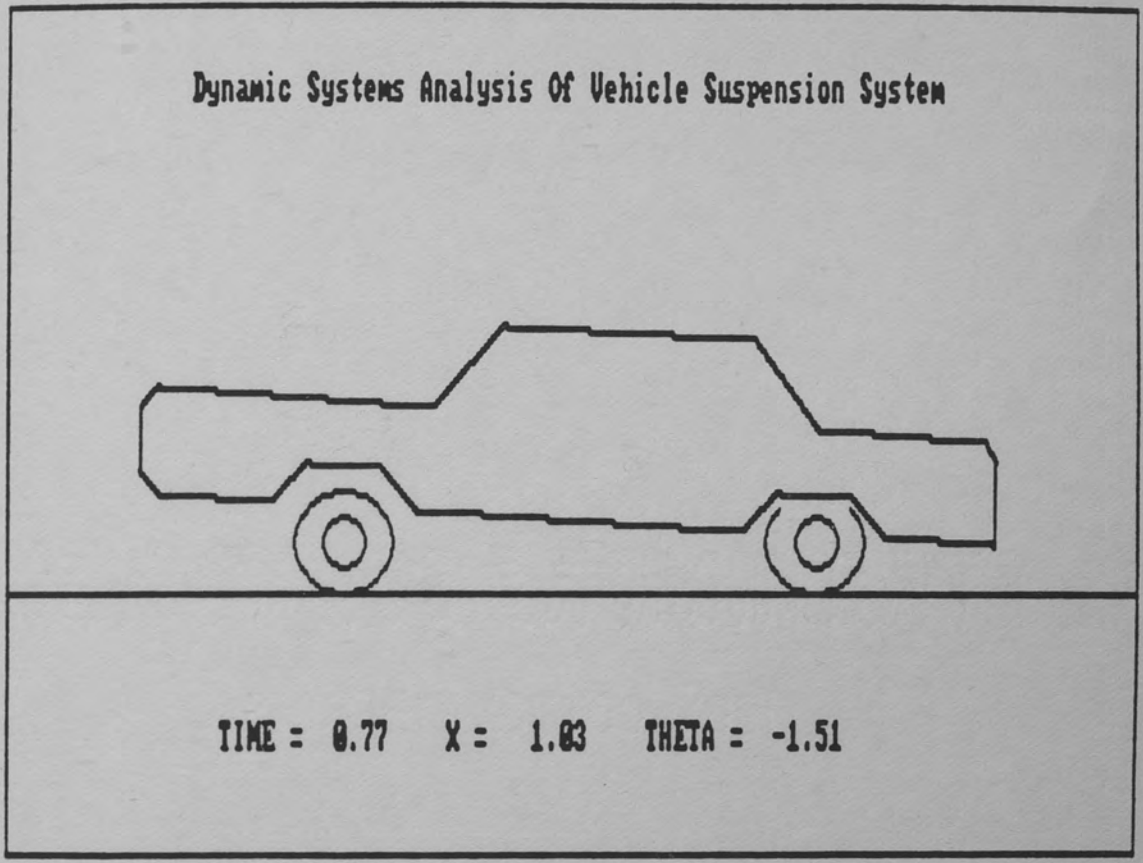


Dynamic Systems Analysis Of Vehicle Suspension System

DISPLAY LIMIT EXCEEDED

```
TIME = 0.10   X =  7.66   THETA =  1.81
```

Figure 5. Case 1 : Automobile Exceeding Display Limit



```
Dynamic Systems Analysis Of Vehicle Suspension System

TIME =  0.77    X =  1.03    THETA =  -1.51
```

Figure 6.   Case 1 : Automobile

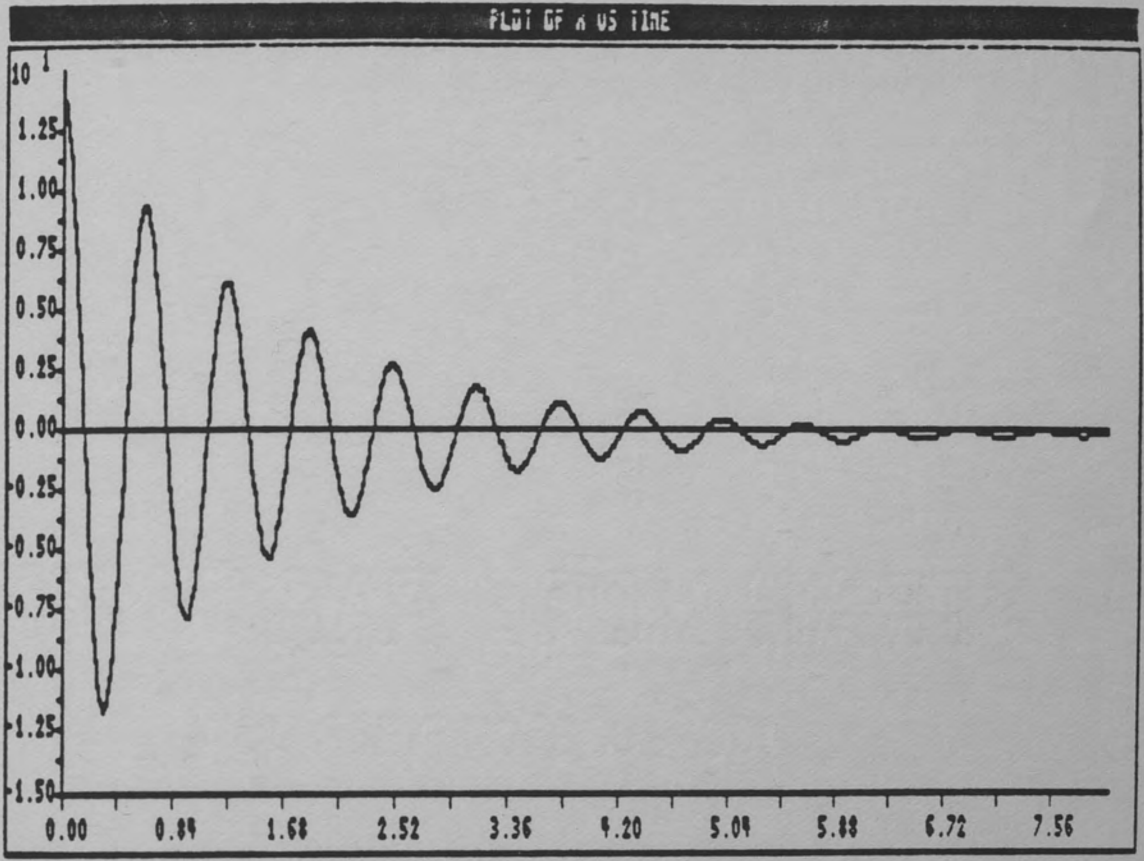


Figure 7. Case 1 : Plot of X versus TIME

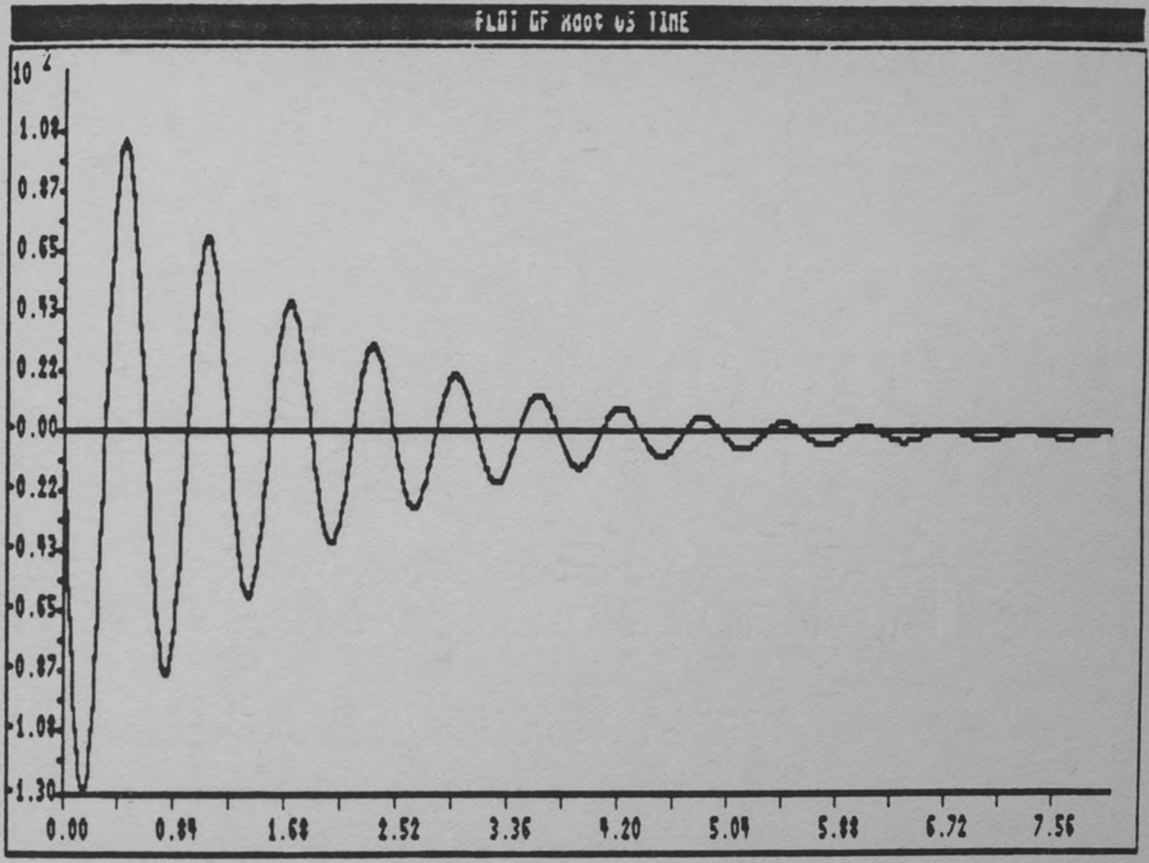


Figure 8. Case 1 : Plot of Xdot versus TIME

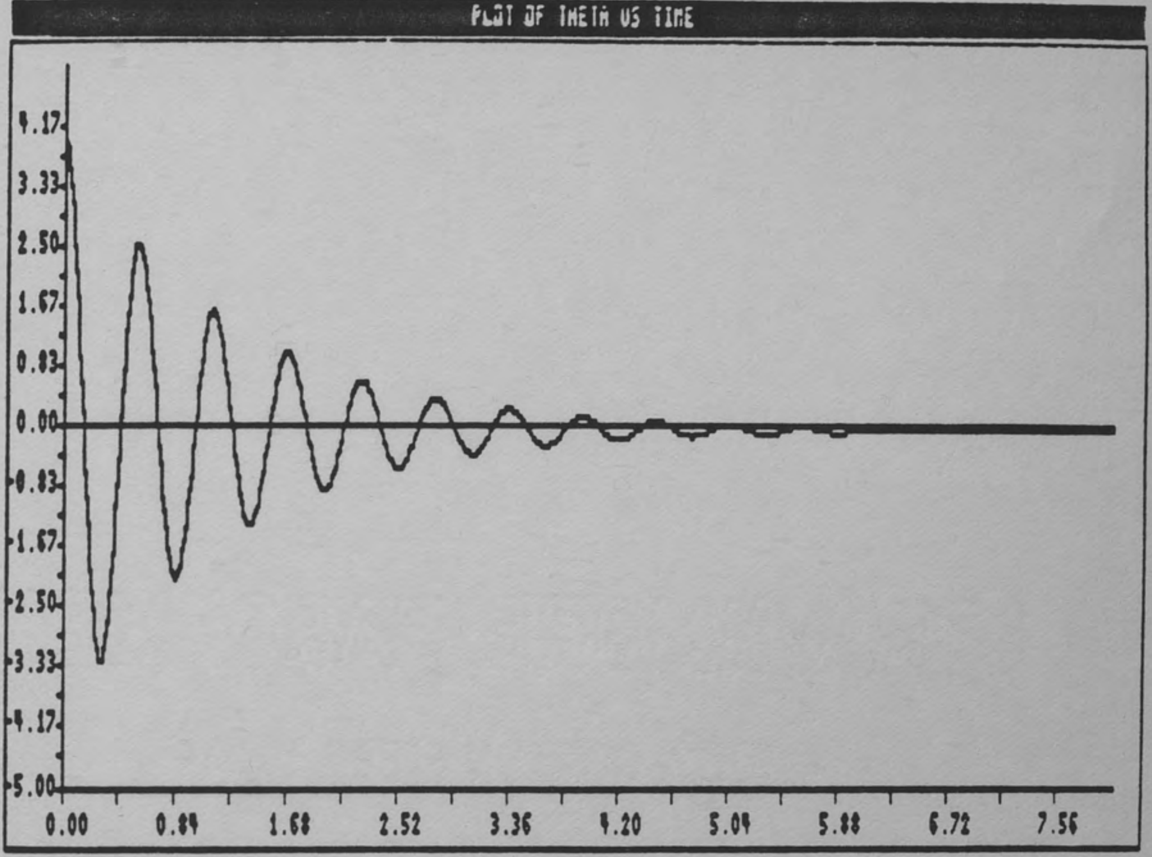


Figure 9. Case 1 : Plot of TH versus TIME

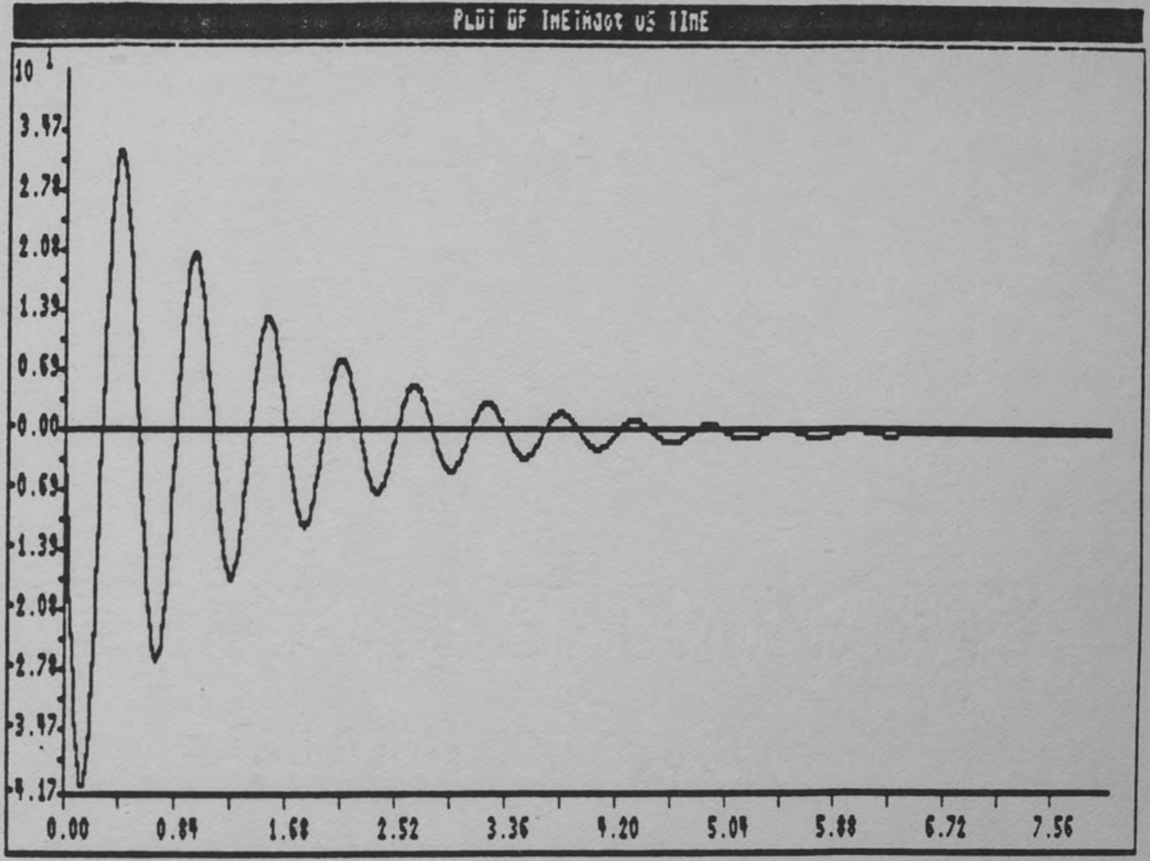


Figure 10. Case 1 : Plot of THdot versus TIME

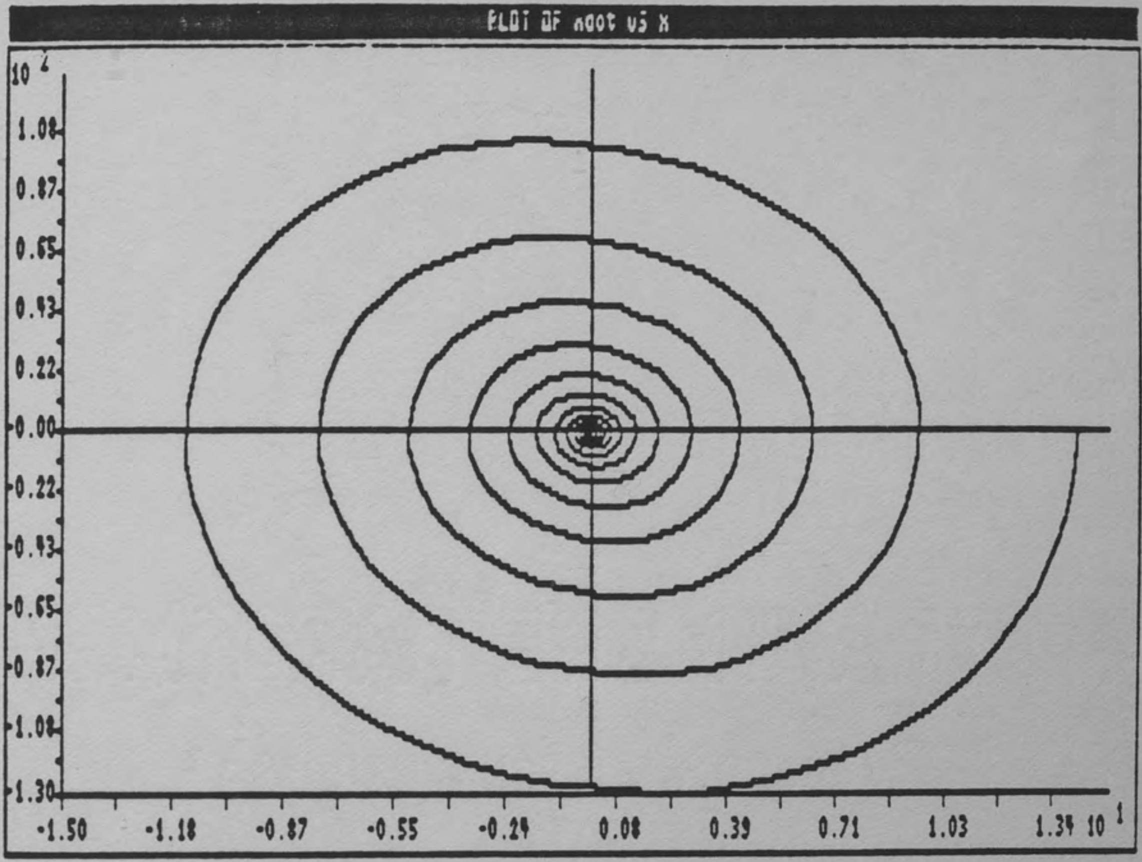


Figure 11. Case 1 : Plot of Xdot versus X

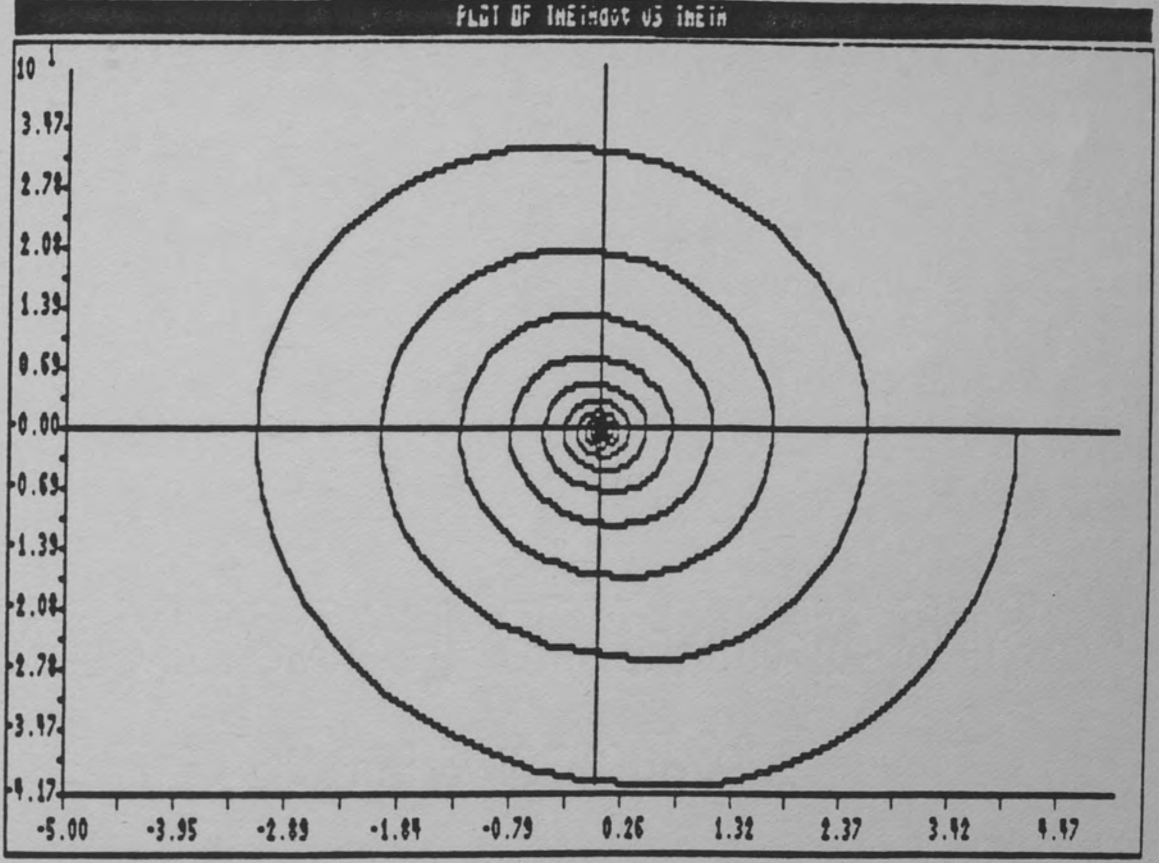


Figure 12. Case 1 : Plot of THdot versus TH

TABLE 6

CASE 1 : NUMERIC OUTPUT

SIMULATION HARDCOPY OUTPUT

```
Fx1 = 0.0      Fx2 = 0.0  B1 =  5.0        B2 =  5.0
K1 = 400.0       K2 = 400.0  L1 = 50.0        L2 = 50.0
```

H	Z1	Z2	Z3	Z4
0.00	14.00	0.00	4.00	0.00
0.10	7.66	-113.19	1.81	-37.88
0.20	-4.70	-112.45	-1.98	-29.89
0.30	-11.40	-12.19	-3.11	8.69
0.40	-7.20	86.77	-0.77	32.32
0.50	2.88	96.91	2.04	18.10
0.60	9.19	19.99	2.26	-13.25
0.70	6.60	-65.36	0.05	-25.88
0.80	-1.52	-82.50	-1.89	-9.13
0.90	-7.31	-24.49	-1.53	14.84
1.00	-5.93	48.21	0.40	19.50
1.10	0.54	69.43	1.62	2.74
1.20	5.75	26.59	0.94	-14.45
1.30	5.24	-34.64	-0.64	-13.73
1.40	0.14	-57.78	-1.31	1.47
1.50	-4.46	-26.95	-0.48	12.86
1.60	-4.56	24.03	0.73	8.90
1.70	-0.61	47.57	0.99	-3.94
1.80	3.40	26.13	0.16	-10.69
1.90	3.92	-15.87	-0.72	-5.08
2.00	0.90	-38.74	-0.70	5.10
2.10	-2.55	-24.53	0.06	8.35
2.20	-3.32	9.69	0.64	2.24
2.30	-1.06	31.20	0.46	-5.34
2.40	1.87	22.47	-0.19	-6.13
2.50	2.79	-5.11	-0.54	-0.28
2.60	1.13	-24.83	-0.27	5.00
2.70	-1.33	-20.18	0.25	4.19
2.80	-2.32	1.79	0.42	-0.96
2.90	-1.13	19.52	0.12	-4.32
3.00	0.92	17.82	-0.26	-2.60
3.10	1.90	0.54	-0.31	1.63
3.20	1.08	-15.13	-0.02	3.50
3.30	-0.60	-15.51	0.25	1.37
3.40	-1.54	-2.10	0.21	-1.90
3.50	-1.01	11.55	-0.04	-2.67

CHAPTER VI

SUMMARY AND CONCLUSIONS

This research report consists of a software program in a Computer Aided Instruction package designed for the Dynamics Systems and Control Theory fields for undergraduate students. The system being described is an automobile with no horizontal motion. This paper defines the system equations, the numerical analysis method chosen to solve the equations and the method used to animate the output on the video display terminal.

The system is designed so that students can change any system parameter defined in the system equations and also change any initial condition on the differential equations that describe the X and TH values. The solutions to the system equations are used to drive the animation on the video display terminal. A student is able to visualize the automobile's position as the system equations are solved.

Additional features can be incorporated in this software project. An additional feature would be allowing for a horizontal motion on the automobile. This would require a new set of system equations that account for the road profile and the velocity of the automobile. The animation would be changed to show the automobile moving along a road. Another enhancement could be further optimization of the graphics software. Currently the graphics is performed mainly by TURBO TOOLBOX commands. In some instances these commands could be optimized by using assembly language routines instead of the TURBO TOOLBOX commands. This would increase the animation speed.

APPENDIX 1

ALGORITHM VERIFICATION

```
program classical;
var
  fz1,fz2,m,b1,b2,b,k1,k2,k,bl1,bl2,bl,kl1,kl2,kl : real;
  j,l1,l2,fxl1,fxl2,blsq1,blsq2,blsq,klsq1,klsq2 : real;
  k1z1,k2z1,k3z1,k4z1,k1z2,k2z2,k3z2,k4z2,klsq : real;
  k1z3,k2z3,k3z3,k4z3,k1z4,k2z4,k3z4,k4z4 : real;
  h,t1,t2,t3,t4 : real;
  i : integer;
  z1 : array[0..1000] of real;
  z2 : array[0..1000] of real;
  z3 : array[0..1000] of real;
  z4 : array[0..1000]of real;
begin
   m := 7.772;
   j := 15000.0;
   fz1 := 0.0;
   fz2 := 0.0;
   b1 := 0.0;
   b2 := 0.0;
   k1 := 400.0;
   k2 := 400.0;
   l1 := 50.0;
   l2 := 50.0;
   z1[0] := 5;
   z2[0] := 0;
   z3[0] := 0;
   z4[0] := 0;
   b := b1 + b2;
   k := k1 + k2;
   bl1 := b1 * l1;
   bl2 := b2 * l2;
   bl := bl2 - bl1;
   kl1 := k1 * l1;
   kl2 := k2 * l2;
   kl := kl2 - kl1;
   fxl1 := fz1 * l1;
   fxl2 := fz2 * l2;
   blsq1 := b1 * sqr(l1);
   blsq2 := b2 * sqr(l2);
   blsq := blsq1 + blsq2;
   klsq1 := k1 * sqr(l1);
   klsq2 := k2 * sqr(l2);
   klsq := klsq1 + klsq2;
   h := 0.01;
   i := 0;
   repeat
     k1z1 := z2[i];
     k1z2 := (1/m)*(fz1+fz2 - (b*z2[i]) - (k*z1[i]) -
                       (bl*z4[i]) - (kl*z3[i]));
     k1z3 := z4[i];
```

```
      k1z4 := (1/j)*(fxl2-fxl1-(blsq*z4[i])-(klsq*z3[i])-
                      (bl*z2[i])-(kl*z1[i]));
      t1 := z1[i] + 0.5*h*k1z1;
      t2 := z2[i] + 0.5*h*k1z2;
      t3 := z3[i] + 0.5*h*k1z3;
      t4 := z4[i] + 0.5*h*k1z4;
      k2z1 := z2[i] + 0.5*h*k1z2;
      k2z2 := (1/m)*(fz1 + fz2 - (b*t2) - (k*t1) -
                      (bl*t4) - (kl*t3));
      k2z3 := z4[i]+0.5*h*k1z4;
      k2z4 := (1/j)*(fxl2 - fxl1 - (blsq*t4) -(klsq*t3) -
                      (bl*t2) - (kl*t1));
      t1 := z1[i] + 0.5*h*k2z1;
      t2 := z2[i] + 0.5*h*k2z2;
      t3 := z3[i] + 0.5*h*k2z3;
      t4 := z4[i] + 0.5*h*k2z4;
      k3z1 := z2[i] + 0.5*h*k2z2;
      k3z2 := (1/m)*(fz1 + fz2 - (b*t2) - (k*t1) -
                      (bl*t4) - (kl*t3));
      k3z3 := z4[i] + 0.5*h*k2z4;
      k3z4 := (1/j)*(fxl2 - fxl1 - (blsq*t4) - (klsq*t3) -
                      (bl*t2) - (kl*t1));
      t1 := z1[i] + h*k3z1;
      t2 := z2[i] + h*k3z2;
      t3 := z3[i] + h*k3z3;
      t4 := z4[i] + h*k3z4;
      k4z1 := z2[i] + h*k3z2;
      k4z2 := (1/m)*(fz1 + fz2 - (b*t2) - (k*t1) -
                      (bl*t4) - (kl*t3));
      k4z3 := z4[i] + h*k3z4;
      k4z4 := (1/j)*(fxl2 - fxl1 - (blsq*t4) - (klsq*t3) -
                      (bl*t2) - (kl*t1));
      z1[i+1] := z1[i] + (h/6)*(k1z1 + 2*k2z1 +
                                   2*k3z1 + k4z1);
      z2[i+1] := z2[i] + (h/6)*(k1z2 + 2*k2z2 +
                                   2*k3z2 + k4z2);
      z3[i+1] := z3[i] + (h/6)*(k1z3 + 2*k2z3 +
                                   2*k3z3 + k4z3);
      z4[i+1] := z4[i] + (h/6)*(k1z4 + 2*k2z4 +
                                   2*k3z4 + k4z4);
   until  keypressed;
end.
```

```
program gills;
var
   fz1,fz2,m,b1,b2,b,k1,k2,k,bl1,bl2,bl,kl1,kl2,kl : real;
   j,l1,l2,fxl1,fxl2,blsq1,blsq2,blsq,klsq1,klsq2 : real;
   k1z1,k2z1,k3z1,k4z1,k1z2,k2z2,k3z2,k4z2,klsq : real;
   k1z3,k2z3,k3z3,k4z3,k1z4,k2z4,k3z4,k4z4 : real;
   h,t1,t2,t3,t4,c1,c2,c3,c4 : real;
   i : integer;
   z1 : array[0..2200] of real;
   z2 : array[0..2200] of real;
   z3 : array[0..2200] of real;
   z4 : array[0..2200] of real;

begin
      m := 7.772;
      j := 15000.0;
      fz1 := 0.0;
      fz2 := 0.0;
      b1 := 0.0;
      b2 := 0.0;
      k1 := 400.0;
      k2 := 400.0;
      l1 := 50.0;
      l2 := 50.0;
      z1[0] := 5;
      z2[0] := 0;
      z3[0] := 0;
      z4[0] := 0;
      c1 := 1/sqrt(2);
      c2 := 1 - c1;
      c3 := 1 + c1;
      c4 := -0.5 + c1;
      b := b1 + b2;
      k := k1 + k2;
      bl1 := b1 * l1;
      bl2 := b2 * l2;
      bl := bl2 - bl1;
      kl1 := k1 * l1;
      kl2 := k2 * l2;
      kl := kl2 - kl1;
      fxl1 := fz1 * l1;
      fxl2 := fz2 * l2;
      blsq1 := b1 * sqr(l1);
      blsq2 := b2 * sqr(l2);
      blsq := blsq1 + blsq2;
      klsq1 := k1 * sqr(l1);
      klsq2 := k2 * sqr(l2);
      klsq := klsq1 + klsq2;
      h := 0.01;
      i := 0;
```

```
     repeat
       k1z1 := z2[i];
       k1z2 := (1/m)*(fz1+fz2 - (b*z2[i]) - (k*z1[i]) -
                    (b1*z4[i]) - (k1*z3[i]));
       k1z3 := z4[i];
       k1z4 := (1/j)*(fx12-fx11-(b1sq*z4[i])-(k1sq*z3[i])-
                    (b1*z2[i])-(k1*z1[i]));
       t1 := z1[i] + 0.5*h*k1z1;
       t2 := z2[i] + 0.5*h*k1z2;
       t3 := z3[i] + 0.5*h*k1z3;
       t4 := z4[i] + 0.5*h*k1z4;
       k2z1 := z2[i] + 0.5*h*k1z2;
       k2z2 := (1/m)*(fz1 + fz2 - (b*t2) - (k*t1) -
                    (b1*t4) - (k1*t3));
       k2z3 := z4[i]+0.5*h*k1z4;
       k2z4 := (1/j)*(fx12 - fx11 - (b1sq*t4) - (k1sq*t3) -
                    (b1*t2) - (k1*t1));
       t1 := z1[i] + c4*h*k1z1 + c2*h*k2z1;
       t2 := z2[i] + c4*h*k1z2 + c2*h*k2z2;
       t3 := z3[i] + c4*h*k1z3 + c2*h*k2z3;
       t4 := z4[i] + c4*h*k1z4 + c2*h*k2z4;
       k3z1 := z2[i] + c4*h*k1z2 + c2*h*k2z2;
       k3z2 := (1/m)*(fz1 + fz2 - (b*t2) - (k*t1) -
                    (b1*t4) - (k1*t3));
       k3z3 := z4[i] + c4*h*k1z4 + c2*h*k2z4;
       k3z4 := (1/j)*(fx12 - fx11 - (b1sq*t4) - (k1sq*t3) -
                    (b1*t2) - (k1*t1));
       t1 := z1[i] - c1*h*k2z1 + c3*h*k3z1;
       t2 := z2[i] - c1*h*k2z2 + c3*h*k3z2;
       t3 := z3[i] - c1*h*k2z3 + c3*h*k3z3;
       t4 := z4[i] - c1*h*k2z4 + c3*h*k3z4;
       k4z1 := z2[i] - c1*h*k2z2 + c3*h*k3z2;
       k4z2 := (1/m)*(fz1 + fz2 - (b*t2) - (k*t1) -
                    (b1*t4) - (k1*t3));
       k4z3 := z4[i] - c1*h*k2z4 + c3*h*k3z4;
       k4z4 := (1/j)*(fx12 - fx11 - (b1sq*t4) - (k1sq*t3) -
                    (b1*t2) - (k1*t1));
       z1[i+1] := z1[i] + (h/6)*(k1z1 + 2*c2*k2z1 +
                                        2*c3*k3z1 + k4z1);
       z2[i+1] := z2[i] + (h/6)*(k1z2 + 2*c2*k2z2 +
                                        2*c3*k3z2 + k4z2);
       z3[i+1] := z3[i] + (h/6)*(k1z3 + 2*c2*k2z3 +
                                        2*c3*k3z3 + k4z3);
       z4[i+1] := z4[i] + (h/6)*(k1z4 + 2*c2*k2z4 +
                                        2*c3*k3z4 + k4z4);
    i := i + 1;
    until keypressed;
end.
```

APPENDIX 2

SOFTWARE LISTING

```
program madrsm;


{ Research Report For Bruce M. Skeldon }

{  Dynamic Vehicle Suspension System }

{ The Following modules are included from the graphics }
{toolbox kit. Refer to manuals for specifics }

{$I typedef.sys}
{$I graphix.sys}
{$I kernel.sys}
{$I axis.hgh}
{$i polygon.hgh}


{ The following arrays define the points to draw the wheels
and hubcaps. The WHLXX/WHLYY arrays define the wheels. The
HUBXX/HUBYY arrays define the hubcaps. The arrays were used
to speed up drawing the wheels during the simulation. }


type
   whlxx = array [1..109] of integer;
   whlyy = array [1..109] of integer;
   hubxx = array [1..52] of integer;
   hubyy = array [1..52] of integer;

var

{ The next set of variables are used in the runga math to }
{define the variables in the equation. }

   fz1,fz2,m,b1,b2,b,k1,k2,k,bl1,bl2,bl,kl1,kl2,kl : real;
   j,l1,l2,fxl1,fxl2,blsq1,blsq2,blsq,klsq1,klsq2 : real;
   klsq,k1z1,k2z1,k3z1,k4z1,k1z2,k2z2,k3z2,k4z2 : real;
   k1z3,k2z3,k3z3,k4z3,k1z4,k2z4,k3z4,k4z4 : real;
   h,hh,t1,t2,t3,t4 : real;

{ Variables used for keyboard inputs , counters }

   dif,adminvalue,newz1 : real;
   i,admininput : integer;
   selection : char;
   admininputst : string[2];
   adminvaluest : string[6];
   noexit,adminexit,exitrunga,drawit : boolean;
   increment,count,ptr,ercode : integer;
```

```
{The following arrays are used to store the math output for
plotting/numeric output. Z1 = X , Z2 = Xdot , Z3 = THETA,
Z2 = THETA dot , TM = TIME }

   z1 : array[0..800] of real;
   z2 : array[0..800] of real;
   z3 : array[0..800] of real;
   z4 : array[0..800] of real;
   tm : array[0..800] of real;

{ This defines the PLOTARRAY needed by DRAWPOLYGON }

   a  : PlotArray;

{ These variables store the maximum value of the associated
arrays.  They are used during DRAWPOLYGON to label axis }

   z1max,z2max,z3max,z4max,tmmax : real;

{Arrays of two(2) numbers used during the math calculation
 ZZx[0] = ZZx[i] , ZZx[1] = ZZx[i + 1] }

   zz1 : array[0..1] of real;
   zz2 : array[0..1] of real;
   zz3 : array[0..1] of real;
   zz4 : array[0..1] of real;

{These values are used to speed the recalculation of the
x/y points after the math finds new value.

i.e s5 = sin * 5 }

   s5,c5,s10,c10,s15,c15,s30,c30,s241,c241,c,s : real;


{ These points describe the outline of the car as follows:

{                5 ________________________ 6          }
{                 /                        \           }
{                /                          \          }
{  3____________/                            \________8 }
{2 /            4        centerx             7        \ }
{ |                      centery                      | 9}
{1 |    17_______16                  13_______12      | }
{  \_____/         \__________________/        \______| }
{  19   18                           14        11     10 }
{______________________________________________________}
```

```
   px1,px2,px3,px4,px5,px6,px7,px8,px9,px10 : integer;
   px11,px12,px13,px14,px15,px16,px17,px18,px19 :integer;
   py1,py2,py3,py4,py5,py6,py7,py8,py9,py10 :integer;
   py11,py12,py13,py14,py15,py16,py17,py18,py19 : integer;
   centerx,centery : integer;

{ Used to store the previous py10/py19 values so that you }
{know when to redisplay the car.}

   oldpy10,oldpy19 : integer;

const

{ This defines the x points used to draw the wheels. The }
{ values are added to 160 for the front wheel and 425 for }
{the rear wheel. }

  whlx:whlxx=(1,1,2,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
              18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,
              33,34,35,36,
              37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,
              52,53,54,55,
              56,56,57,57,57,56,56,55,54,53,52,51,50,49,48,
              47,46,45,44,
              46,45,44,43,42,41,40,39,38,37,36,35,23,22,21,
              20,19,18,17,
              16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,2,1);

{ This defines the y points used to draw the wheels. The }
{road is at y = 138 }

  whly:whlyy=(126,125,124,123,122,121,120,120,119,119,118,
              117,117,117,
              116,116,116,115,115,115,115,115,115,115,114,
              114,114,114,
              114,114,114,114,114,114,114,114,114,115,115,
              115,115,115,
              115,115,116,116,116,117,117,117,118,119,119,
              120,120,121,
              122,123,124,125,126,127,128,129,130,131,132,
              132,133,133,
              134,135,135,135,136,136,136,137,137,137,137,
              137,137,137,
              137,137,137,137,137,137,137,137,137,136,136,
              136,135,135,
              135,134,133,133,132,132,131,130,129,128,127);

{ This defines the x points used to draw the hubcaps. The }
{ values are added to 177 for the front  and 443 for the  }
{ rear. }
```

```
  hubx:hubxx=(0,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,
             17,18,19,20,
             21,22,23,24,24,24,23,22,21,20,19,18,17,16,
             15,14,13,12,
             11,10,9,8,7,6,5,4,3,2,1,0);

{ This defines the y points used to draw the hubcaps }

  huby:hubyy=(126,125,124,123,122,122,121,121,121,121,
             120,120,120,120,
             120,120,120,121,121,121,121,122,122,123,
             124,125,126,127,
             128,129,130,130,131,131,131,131,132,132,
             132,132,132,132,
             132,131,131,131,131,130,130,129,128,127);

{This procedure gets a user input from the keyboard and }
{verifies that the entry was a real number (ercode = 0) }

procedure getnewvalue;
  begin
    adminvalue := 0.0;
    write('                    ENTER NEW VALUE > ');
    readln(adminvaluest);
    val(adminvaluest,adminvalue,ercode);
    if ercode > 0 then
    begin
      admininput := 0
    end;
  end;

{ Must calculat the moment of inertia (j) when L or M is }
{changed }

procedure calculatej;
  begin
    j := (m/12)*(sqr(l1+l2+50) + sqr(40));
  end;

{Only allow external forces in the range -1500 to 1500 LBS
Set invalid inputs to a default of 0 LBS }

procedure checkrangef;
  begin
    getnewvalue;
    if (adminvalue < -1500) or (adminvalue > 1500) then
       adminvalue := 0;
    if admininput = 1 then
      fz1 := adminvalue
    else
```

```
      fz2 := adminvalue;
  end;

{ Only allow damping constants in the range 0 to 500 LB-
SEC/IN Set invalid inputs to a default of 0 LB-SEC/IN }

procedure checkrangeb;
  begin
    getnewvalue;
    if (adminvalue < 0) or (adminvalue > 500) then
       adminvalue := 0;
    if admininput = 3 then
      b1 := adminvalue
    else
      b2 := adminvalue;
  end;

{ Only allow spring constant values in the range 10 to
1000 LB/IN Set invalid inputs to a default of 5 LB/IN }

procedure checkrangek;
  begin
    getnewvalue;
    if (adminvalue < 10) or (adminvalue > 1000) then
       adminvalue := 10;
    if admininput = 5 then
      k1 := adminvalue
    else
      k2 := adminvalue;
  end;

{ Only allow lengths from the center of the car to the }
{wheels in the range 10 to 75 IN.  Set invalid inputs to a}
{default of 10 IN. }

procedure checkrangel;
  begin
    getnewvalue;
    if (adminvalue < 10) or (adminvalue > 75) then
       adminvalue := 10;
    if admininput = 7 then
      l1 := adminvalue
    else
      l2 := adminvalue;
    calculatej;
  end;

{ Only allow car weight in the range 1500 to 4000 LBS. Set
invalid inputs to a default of 2500 LBS. Store mass (m)
in slugs (m/386).}

procedure checkrangem;
  begin
    getnewvalue;
    if (adminvalue < 1500) or (adminvalue > 4000) then
       adminvalue := 2500;
    m := adminvalue/386;
    calculatej;
  end;

{Only allow Z1[0](x initial condition) in the range -25 to
15 IN. Set invalid inputs to a default of 0 IN. The }
positive x direction is down }

procedure checkrange1;
  begin
    getnewvalue;
    if (adminvalue < -25) or (adminvalue > 15) then
       adminvalue := 0;
    z1[0] := adminvalue;
  end;

procedure checkrange2;
  begin
    getnewvalue;
    if (adminvalue < -25) or (adminvalue > 15) then
       adminvalue := 0;
    z2[0] := adminvalue;
  end;

{ Only allow Z3[0](theta initial condition) in the range
- 4 to 4 degrees. Set invalid inputs to a default of 0
degrees.   Convert degrees to radians }

procedure checkrange3;
  begin
    getnewvalue;
    if (adminvalue < -4) or (adminvalue > 4) then
       adminvalue := 0;
    z3[0] := adminvalue/57.3;
  end;

procedure checkrange4;
  begin
    getnewvalue;
    if (adminvalue < -4) or (adminvalue > 4) then
       adminvalue := 0;
    z4[0] := adminvalue;
  end;
```

```
{ Allow a single entry to clear all initial conditions }

procedure clearinitcon;
  begin
    z1[0] := 0;
    z2[0] := 0;
    z3[0] := 0;
    z4[0] := 0;
  end;

{Main routine called to change any parameters. Loop asking}
{for new parameter until a 0 or invalid entry is made. }

procedure admin;
   begin
     ptr := 0;
     adminexit := true;
     while adminexit do
     begin
      clrscr;
      writeln;
      write('  This section allows you to change ');
      writeln(' certain variables used');
      write('  in the automobile simulation.');
      writeln(' The following variables and');
      write('  thier values are changable');
      writeln(' [units are in () ] :');
      writeln;
      writeln('    0: NO CHANGES REQUIRED' );
      write('    1: Fx1 (force on rear (lbs) [-1500 to');
      writeln(' 1500])=  ',fz1:8:2);
      write('    2: Fx2 (force on front(lbs) [-1500 to');
      writeln(' 1500])=  ',fz2:8:2);
      write('    3: B1(dmping cnst rear(lb-sec/in)[0 to');
      writeln(' 500])=  ',b1:8:2);
      write('    4: B2(dmping cnst frnt(lb-sec/in)[0 to ');
      writeln('500])=  ',b2:8:2);
      write('    5: K1(spring const rear (lb/in)[10 to ');
      writeln('1000])=  ',k1:8:2);
      write('    6: K2(spring const frnt (lb/in)[10 to ');
      writeln('1000])=  ',k2:8:2);
      write('    7: L1  (center to rear    (in)   [10 to');
      writeln(' 75])=  ',l1:8:2);
      write('    8: L2  (center to front   (in)   [10 to');
      writeln(' 75])=  ',l2:8:2);
      write('    9: M   (auto mass  (lbs)      [1500 to');
      writeln(' 4000])=  ',m*386:8:2);
      write('   10: X(0) (x init cond   (in)      [-25 to');
      writeln('15])=  ',z1[0]:8:2);
      write('   11: XDOT(0) (x dot init cond (in/sec) ');
      writeln('        =  ',z2[0]:8:2);
      write('   12: T(0) (theta init cond  (deg) [-4 to');
      writeln(' 4])=  ',z3[0]*57.3:8:2);
      write('   13: THEDOT(0) (theta dot init con) ');
      writeln('          =  ',z4[0]:8:2);
      writeln('    14: CLEAR ALL INITIAL CONDITIONS ');
      writeln;
      write('      J   (moment of inertia)        ');
      writeln('         =  ',j:8:2);
      writeln;
      write('                 ENTER SELECTION AND RETURN > ');
      readln(admininputst);

 { Get entry from the keyboard and check for noentry (''),
 { 0 entry or invalid input (ercode). Exit loop on any of }
 { these conditions. }

      val(admininputst,admininput,ercode);
      if admininputst = '' then
         adminexit := false;
      if admininput = 0 then
        adminexit := false;
      if ercode > 0 then
        adminexit := false;

{If an integer was entered use CASE to call the proper
routine }

      if adminexit then
      begin
        case admininput of
           1 :  checkrangef;
           2 :  checkrangef;
           3 :  checkrangeb;
           4 :  checkrangeb;
           5 :  checkrangek;
           6 :  checkrangek;
           7 :  checkrangel;
           8 :  checkrangel;
           9 :  checkrangem;
          10 :  checkrangel;
          11 :  checkrange2;
          12 :  checkrange3;
          13 :  checkrange4;
          14 :  clearinitcon;
        else
          adminexit := false;
        end;
      end;
     end;
```

```
   end;

{ Need to determine the maximum stored value for all arrays
{to have proper dimensions on the plots. }

procedure calculatemax;
  begin
    i := 0;
    z1max := abs(z1[i]);
    z2max := abs(z2[i]);
    z3max := abs(z3[i]);
    z4max := abs(z4[i]);
    tmmax := abs(tm[i]);

{Loop from 1 until the number of points saved during the }
{simulation run (ptr max = 800). Store the largest value }

    i := 1;
    while i < ptr do
    begin
      if abs(z1[i]) > z1max then
        z1max := abs(z1[i]);
      if abs(z2[i]) > z2max then
        z2max := abs(z2[i]);
      if abs(z3[i]) > z3max then
        z3max := abs(z3[i]);
      if abs(z4[i]) > z4max then
        z4max := abs(z4[i]);
      if abs(tm[i]) > tmmax then
        tmmax := abs(tm[i]);
      i := i + 1;
    end;

{Make Z1 and Z2 maximum equal to one more than maximum. }
{Convert Z3 and Z4 to radians and store maximum + 1. }

    z1max := z1max + 1;
    z2max := z2max + 1;
    z3max := (z3max)*57.3 + 1;
    z4max := (z4max)*57.3 + 1;
  end;

{ Calculate PLOTARRAY for X vs TIME plot }

procedure pltxt;
  begin
    while i <= ptr do
    begin
      a[i+1,1] := tm[i];
      a[i+1,2] := z1[i];
      i := i + 1;
    end;
    defineheader(1,'PLOT OF X VS TIME');
    defineworld(1,0,z1max,tmmax,-z1max);
  end;

{ Calculate PLOTARRAY for Xdot vs TIME plot. }

procedure pltxdt;
  begin
    while i <= ptr do
    begin
      a[i+1,1] := tm[i];
      a[i+1,2] := z2[i];
      i := i + 1;
    end;
    defineheader(1,'PLOT OF Xdot VS TIME');
    defineworld(1,0,z2max,tmmax,-z2max);
  end;

{ Calculate PLOTARRAY for THETA vs TIME plot. }

procedure pltthetat;
  begin
    while i <= ptr do
    begin
      a[i+1,1] := tm[i];
      a[i+1,2] := z3[i]*57.3;
      i := i + 1;
    end;
    defineheader(1,'PLOT OF THETA VS TIME');
    defineworld(1,0,z3max,tmmax,-z3max);
  end;

{ Calculate PLOTARRAY for THETAdot vs TIME plot. }

procedure pltthetadt;
  begin
    while i <= ptr do
    begin
      a[i+1,1] := tm[i];
      a[i+1,2] := z4[i]*57.3;
      i := i + 1;
    end;
    defineheader(1,'PLOT OF THETAdot VS TIME');
    defineworld(1,0,z4max,tmmax,-z4max);
  end;

{ Calculate PLOTARRAY for Xdot vs X plot. }
```

```
procedure pltxdx;
  begin
    while i <= ptr do
    begin
      a[i+1,1] := z1[i];
      a[i+1,2] := z2[i];
      i := i + 1;
    end;
    defineheader(1,'PLOT OF Xdot VS X');
    defineworld(1,-z1max,z2max,z1max,-z2max);
  end;

{ Calculate PLOTARRAY for THETAdot vs THETA plot. }

procedure pltthetadtheta;
  begin
    while i <= ptr do
    begin
      a[i+1,1] := z3[i]*57.3;
      a[i+1,2] := z4[i]*57.3;
      i := i + 1;
    end;
    defineheader(1,'PLOT OF THETAdot VS THETA');
    defineworld(1,-z3max,z4max,z3max,-z4max);
  end;

{ This is the main routine used to draw the various plots }
{ Must first calculate the maximum values stored in the
{arrays for dimensioning. }

procedure pltroutine;
  begin
    calculatemax;
    adminexit := true;

  { The simulation must have been run and stored three }
  { data points before any plotting will take place }

    if ptr < 3 then
      adminexit := false;

{Loop asking for plot type until 0 entry or invalid entry }

  while adminexit do
  begin
    clrscr;
    writeln;
    writeln;
    writeln;
    writeln;
    writeln('        The following plots are available : ');
    writeln;
    writeln('                 0 :  NO PLOT DESIRED ');
    writeln('                 1 :  X           vs    TIME ');
    writeln('                 2 :  Xdot        vs    TIME ');
    writeln('                 3 :  THETA       vs    TIME ');
    writeln('                 4 :  THETAdot    vs    TIME ');
    writeln('                 5 :  Xdot        vs    X ');
    writeln('                 6 :  THETAdot    vs    THETA ');
    writeln;
    write('                 ENTER SELECTION AND RETURN > ');
    readln(admininputst);

{Get entry from the keyboard and check for noentry (''), }
{or entry outside the range 1 to 6. Exit loop on any of }
{ these conditions. }

      val(admininputst,admininput,ercode);
      if admininputst = '' then
         adminexit := false;
      if (admininput < 1) or (admininput > 6) then
        adminexit := false;
      if adminexit then
      begin

{If no errors so far set up graphics and define window }
{plotting. }

        entergraphic;
        definewindow(1,0,0,xmaxglb,ymaxglb);

 { Use CASE statement to create PLOTARRAY for plots }

        i := 0;
        case admininput of
           1 :  pltxt;
           2 :  pltxdt;
           3 :  pltthetat;
           4 :  pltthetadt;
           5 :  pltxdx;
           6 :  pltthetadtheta;
        else
          adminexit := false;
        end;

{Select world/window, draw header, drawborder, draw axis, }
{ and draw the plot using a = PLOTARRAY }

        selectscreen(1);
        selectworld(1);
```

```
       selectwindow(1);
       setheaderon;
       drawborder;
       drawaxis(8,-8,0,0,0,0,0,0,false);
       drawpolygon(a,1,ptr,0,0,0);

 { Loop until keyboard entry causes exit from graphics }

       repeat until keypressed;
       removeheader(1);
       leavegraphic;
     end;
   end;
 end;

{This routine is used to print the variables on the
 printer }

procedure hrdcpy;
  begin
    if ptr > 3 then
    begin
      adminexit := true;
      writeln;
      writeln;
      write('                    INCREMENT NUMBER > ');
      readln(admininputst);
      val(admininputst,increment,ercode);
      if ercode > 0 then
        adminexit := false;
      if admininputst = '' then
        adminexit := false;
      if adminexit then
      begin
        dif := ptr;
        i := 0;
        write(lst,'          THIS IS A HARDCOPY OUTPUT');
        writeln(lst,' OF THE SIMULATION ');
        write(lst,'              Fx1 = ',fz1:3:1,'   ');
        write(lst,' Fx2 = ',fz2:3:1);
        writeln(lst,'  B1 = ',b1:4:1,'      B2 = ',b2:4:1);
        write(lst,'              K1 = ',k1:3:1,'     ');
        write(lst,K2 = ',k2:3:1);
        writeln(lst,'  L1 = ',l1:4:1,'      L2 = ',l2:4:1);
        count := 3;
        repeat
          write(lst,'   h= ',tm[i]:3:2,'  z1= ');
          write(lst,z1[i]:10:4,'  z2= ',z2[i]:10:4);
          write(lst,'    z3 = ',z3[i]*57.3:7:4);
          writeln(lst,'   z4 = ',z4[i]*57.3:7:4);
```

```
            i := i + increment;
            count := count + 1;
            if count = 63 then
            begin
              writeln(lst);
              writeln(lst);
              writeln(lst);
              writeln(lst);
              count := 2;
            end;
            dif := dif - increment;
          until keypressed or (dif < 0);
        end;
      end;
    end;

  {This routine calculates the differential equations using
  a fourth order RUNGA KUTTA algoithm. The four equations
  are :}

  { z1 = z2                                                  }
  { z2 = (1/m)[fx1+fx2-(b1+b2)z2-(k1+k2)z1-(b2l2-b1l1)z4-    }
  {           (k2l2-k1l1)z3]                                 }
  {  z3 = z4                                                 }
  {  z4 = (1/j)[fx2l2-fx1l1-(b2l2**2-b1l1**2)z4-(k2l2**2-    }
  {            k1l1**2)z3}                                   }
  {           -(b2l2-b1l1)z2-(k2l2-k1l1)z1]                  }

  { Where z1 = X, z2 = Xdot, z3 = THETA and z4 = THETAdot }
  { The algoithm used is as  follows: }

  {   k1 = z1[0]                                             }
  {   k2 = z1[0] + .5*h*k1                                   }
  {   k3 = z1[0] + .5*h*k2                                   }
  {   k4 = z1[0] + k3*h                                      }
  {z1[1] = z1[0] + (h/6)[k1 + .5*k2 + .5*k3 + k4]            }
  {z1[0] = z1[1]                                             }

  procedure gills;
     begin
         k1z1 := zz2[0];
         k1z2 := (1/m)*(fz1+fz2-(b*zz2[0])-(k*zz1[0])-
                     (b1*zz4[0])-(k1*zz3[0]));
         k1z3 := zz4[0];
         k1z4:=(1/j)*(fx12-fx11-(b1sq*zz4[0])-(k1sq*zz3[0])-
                     (b1*zz2[0])-(k1*zz1[0]));
         t1 := zz1[0] + 0.5*h*k1z1;
         t2 := zz2[0] + 0.5*h*k1z2;
         t3 := zz3[0] + 0.5*h*k1z3;
         t4 := zz4[0] + 0.5*h*k1z4;
        k2z1 := zz2[0] + 0.5*h*k1z2;
        k2z2 := (1/m)*(fz1+fz2-(b*t2)-(k*t1)-(b1*t4)
                 -(k1*t3));
        k2z3 := zz4[0] + 0.5*h*k1z4;
        k2z4 := (1/j)*(fx12 - fx11 - (b1sq*t4) - (k1sq*t3) -
                (b1*t2) - (k1*t1));
        t1 := zz1[0] + 0.5*h*k2z1;
        t2 := zz2[0] + 0.5*h*k2z2;
        t3 := zz3[0] + 0.5*h*k2z3;
        t4 := zz4[0] + 0.5*h*k2z4;
        k3z1 := zz2[0] + 0.5*h*k2z2;
        k3z2 := (1/m)*(fz1 + fz2 - (b*t2) - (k*t1) -
                (b1*t4) - (k1*t3));
        k3z3 := zz4[0] + 0.5*h*k2z4;
        k3z4 := (1/j)*(fx12 - fx11 - (b1sq*t4) - (k1sq*t3) -
                 (b1*t2) - (k1*t1));
        t1 := zz1[0] + h*k3z1;
        t2 := zz2[0] + h*k3z2;
        t3 := zz3[0] + h*k3z3;
        t4 := zz4[0] + h*k3z4;
        k4z1 := zz2[0] + h*k3z2;
        k4z2 := (1/m)*(fz1 + fz2 - (b*t2) - (k*t1) -
                 (b1*t4) - (k1*t3));
        k4z3 := zz4[0] + h*k3z4;
        k4z4 := (1/j)*(fx12 - fx11 - (b1sq*t4) - (k1sq*t3) -
                (b1*t2) - (k1*t1));
        zz1[1] := zz1[0] + (h/6)*(k1z1+2*k2z1+2*k3z1+k4z1);
        zz2[1] := zz2[0] + (h/6)*(k1z2+2*k2z2+2*k3z2+k4z2);
        zz3[1] := zz3[0] + (h/6)*(k1z3+2*k2z3+2*k3z3+k4z3);
        zz4[1] := zz4[0] + (h/6)*(k1z4+2*k2z4+2*k3z4+k4z4);
        hh := hh + h;
        zz1[0] := zz1[1];
        zz2[0] := zz2[1];
        zz3[0] := zz3[1];
        zz4[0] := zz4[1];
      end;

{Display an error message when display is either below the}
road or to high. DRAWIT flag is used to draw/no draw new}
position }

procedure displayerror;
  begin
    gotoxy(2,7);
    write('  DISPLAY LIMIT EXCEEDED ');
    drawit := false;
  end;
```

```
{ Check the range of py6, py10, py19 and py3 to ensure
that the car is within drawing window. If not display
error. }

procedure rangecheck;
  begin
    if (py6<=50)or(py10>=137)or(py19>=137)or(py3<=50) then
      displayerror;
    gotoxy(16,22);
    write('TIME = ',hh:5:2,'    X = ',zz1[1]:6:2);
    write('    THETA = ',zz3[1]*57.3:6:2);
  end;

{ This procedure is a modified version of the toolbox}
DRAWLINEDIRECT It was modified to add speed to the}
redrawing }

procedure drawlinedirectt(x1,y1,x2,y2:integer);
   var x,y,deltax,deltay,xstep,ystep,direction:integer;
  begin
   x := x1;
   y := y1;
   xstep :=1;
   ystep :=1;
   if x1>x2 then xstep :=-1;
   if y1>y2 then ystep :=-1;
   deltax:=abs(x2-x1);
   deltay:=abs(y2-y1);
   if deltax=0 then direction:=-1
   else direction:=0;
   while not ((x=x2) and (y=y2)) do
   begin
     dp(x,y);
     if direction<0 then
      begin
       y:=y+ystep;
       direction:=direction+deltax;
      end
     else
      begin
       x:=x+xstep;
       direction:=direction-deltay;
      end;
   end;
  end;
```

```
{ This procedure draws both the front and rear wheels using
the wheel arrays. The front wheel is drawn when the y
points are greater than py17+2. The rear wheel is drawn
when the y points are greater than py12+2. This gives
impression that wheel is entering the wheel well }

procedure drawwheel;
  begin
    i := 1;
    while i < 110 do
    begin
      if whly[i] > py17+2 then
        dp(whlx[i]+160,whly[i]);
      if whly[i] > py12+2 then
        dp(whlx[i]+425,whly[i]);
      i := i + 1;
    end;
  end;

{ This procedure draws both the front and rear hubcaps
inner wheel The front hub is drawn when the y points are
greater than py17+2. The rear hub is drawn when the y
points are greater than py12+2. }

procedure drawhub;
  begin
    i := 1;
    while i < 53 do
    begin
      if huby[i] > py17+2 then
        dp(hubx[i]+177,huby[i]);
      if huby[i] > py12+2 then
        dp(hubx[i]+443,huby[i]);
      i := i + 1;
    end;
  end;

procedure drawnewcar;
   begin
       if drawit then
       begin
         GrafBase := seg(ScreenGlb^);   { selectscreen(2)}
         fillchar(mem[GrafBase:$0750],3645,0);
         fillchar(mem[GrafBase:$2750],3645,0);
         oldpy10 := py10;
         oldpy19 := py19;
         s5 := s*5;
         c5 := c*5;
         px1 := trunc(centerx-c241-s10);
         py1 := trunc(newz1-s241+c10);
```

```
px2 := trunc(centerx-c241+s5);
py2 := trunc(newzl-s241-c5);
px4 := trunc(centerx+c*(-76)+s10);
py4 := trunc(newzl+s*(-76)-c10);
px5 := trunc(centerx+c*(-36)+s30);
py5 := trunc(newzl+s*(-36)-c30);
px7 := trunc(centerx+c*(139)+s10);
py7 := trunc(newzl+s*(139)-c10);
px8 := trunc(centerx+c*(234)+s10);
py8 := trunc(newzl+s*(234)-c10);
px9 := trunc(centerx+c241+s5);
py9 := trunc(newzl+s241-c5);
px11 := trunc(centerx+c*(179)-s15);
py11 := trunc(newzl+s*(179)+c15);
px12 := trunc(centerx+c*(159)-s5);
py12 := trunc(newzl+s*(159)+c5);
px13 := trunc(centerx+c*(119)-s5);
py13 := trunc(newzl+s*(119)+c5);
px14 := trunc(centerx+c*(99)-s15);
py14 := trunc(newzl+s*(99)+c15);
px15 := trunc(centerx+c*(-86)-s15);
py15 := trunc(newzl+s*(-86)+c15);
px16 := trunc(centerx+c*(-106)-s5);
py16 := trunc(newzl+s*(-106)+c5);
px17 := trunc(centerx+c*(-146)-s5);
py17 := trunc(newzl+s*(-146)+c5);
px18 := trunc(centerx+c*(-166)-s15);
py18 := trunc(newzl+s*(-166)+c15);
drawwheel;
drawhub;
drawlinedirectt(px15,py15,px14,py14);{low mid seg}
drawlinedirectt(px19,py19,px18,py18);{low frt seg}
drawlinedirectt(px19,py19,px1,py1);
drawlinedirectt(px1,py1,px2,py2);     {  grill  }
drawlinedirectt(px2,py2,px3,py3);
drawlinedirectt(px3,py3,px4,py4);     {  hood  }
drawlinedirectt(px4,py4,px5,py5); { windshield  }
drawlinedirectt(px5,py5,px6,py6);     {  roof  }
drawlinedirectt(px6,py6,px7,py7);  {rear window }
drawlinedirectt(px7,py7,px8,py8);     {   trunk  }
drawlinedirectt(px8,py8,px9,py9);
drawlinedirectt(px9,py9,px10,py10);{back grill}
drawlinedirectt(px10,py10,px11,py11);{low rer seg}
drawlinedirectt(px14,py14,px13,py13);{rer whl wel}
drawlinedirectt(px13,py13,px12,py12);
drawlinedirectt(px12,py12,px11,py11);
drawlinedirectt(px18,py18,px17,py17);{frt whl wel}
drawlinedirectt(px17,py17,px16,py16);
drawlinedirectt(px16,py16,px15,py15);
drawlinedirectt(0,138,0,25); {redraw left border }
```

```
           drawlinedirectt(639,138,639,25);{redrw rigt bordr}
           move(mem[Grafbase:$0750],mem[$B800:$0750],3645);
           move(mem[GrafBase:$2750],mem[$B800:$2750],3645);
         end;
    end;


procedure exitloop;
  begin
     if abs(zz1[1]) < 0.01 then
        zz1[1] := 0.0;
     if abs(zz3[1]) < 0.0001 then
        zz3[1] := 0.0;
     newz1 := centery + zz1[1];
     c:=cos(-zz3[1]);
     s:=sin(-zz3[1]);
     s10 := s*10;
     c10 := c*10;
     s30 := s*30;
     s15 := s*15;
     c30 := c*30;
     c15 := c*15;
     s241 := s*241;
     c241 := c*241;
     px3 := trunc(centerx+c*(-231)+s10);
     py3 := trunc(newz1+s*(-231)-c10);
     px6 := trunc(centerx+c*(104)+s30);
     py6 := trunc(newz1+s*(104)-c30);
     px10 := trunc(centerx+c241-s15);
     py10 := trunc(newz1+s241+c15);
     px19 := trunc(centerx+c*(-231)-s15);
     py19 := trunc(newz1+s*(-231)+c15);
     rangecheck;
     if ptr < 799 then
     begin
         ptr := ptr + 1;
         z1[ptr] := zz1[1];
         z2[ptr] := zz2[1];
         z3[ptr] := zz3[1];
         z4[ptr] := zz4[1];
         tm[ptr] := hh;
     end;
     if (abs(oldpy10-py10)>=1)or(abs(oldpy19-py19)>=1) then
       drawnewcar;
   end;

 procedure car;
    begin
      zz1[0] := z1[0];
      zz2[0] := z2[0];
```

```
zz3[0] := z3[0];
zz4[0] := z4[0];
zz1[1] := zz1[0];
zz3[1] := zz3[0];
ptr := 0;
b := b1 + b2;
k := k1 + k2;
bl1 := b1 * l1;
bl2 := b2 * l2;
bl := bl2 - bl1;
kl1 := k1 * l1;
kl2 := k2 * l2;
kl := kl2 - kl1;
fxl1 := fz1 * l1;
fxl2 := fz2 * l2;
blsq1 := b1 * sqr(l1);
blsq2 := b2 * sqr(l2);
blsq := blsq1 + blsq2;
klsq1 := k1 * sqr(l1);
klsq2 := k2 * sqr(l2);
klsq := klsq1 + klsq2;
h := 0.01;
hh := 0.0;
drawit := true;
c := cos(-z3[0]);
s := sin(-z3[0]);
newz1 := centery + z1[0];
px1 := trunc(centerx+c*(-241)-s*(10));
py1 := trunc(newz1+s*(-241)+c*(10));
px2 := trunc(centerx+c*(-241)-s*(-5));
py2 := trunc(newz1+s*(-241)+c*(-5));
px3 := trunc(centerx+c*(-231)-s*(-10));
py3 := trunc(newz1+s*(-231)+c*(-10));
px4 := trunc(centerx+c*(-76)-s*(-10));
py4 := trunc(newz1+s*(-76)+c*(-10));
px5 := trunc(centerx+c*(-36)-s*(-30));
py5 := trunc(newz1+s*(-36)+c*(-30));
px6 := trunc(centerx+c*(104)-s*(-30));
py6 := trunc(newz1+s*(104)+c*(-30));
px7 := trunc(centerx+c*(139)-s*(-10));
py7 := trunc(newz1+s*(139)+c*(-10));
px8 := trunc(centerx+c*(234)-s*(-10));
py8 := trunc(newz1+s*(234)+c*(-10));
px9 := trunc(centerx+c*(241)-s*(-5));
py9 := trunc(newz1+s*(241)+c*(-5));
px10 := trunc(centerx+c*(241)-s*(15));
py10 := trunc(newz1+s*(241)+c*(15));
px11 := trunc(centerx+c*(179)-s*(15));
py11 := trunc(newz1+s*(179)+c*(15));
px12 := trunc(centerx+c*(159)-s*(5));
```

```
pyl2 := trunc(newzl+s*(159)+c*(5));
pxl3 := trunc(centerx+c*(119)-s*(5));
pyl3 := trunc(newzl+s*(119)+c*(5));
pxl4 := trunc(centerx+c*(99)-s*(15));
pyl4 := trunc(newzl+s*(99)+c*(15));
pxl5 := trunc(centerx+c*(-86)-s*(15));
pyl5 := trunc(newzl+s*(-86)+c*(15));
pxl6 := trunc(centerx+c*(-106)-s*(5));
pyl6 := trunc(newzl+s*(-106)+c*(5));
pxl7 := trunc(centerx+c*(-146)-s*(5));
pyl7 := trunc(newzl+s*(-146)+c*(5));
pxl8 := trunc(centerx+c*(-166)-s*(15));
pyl8 := trunc(newzl+s*(-166)+c*(15));
pxl9 := trunc(centerx+c*(-231)-s*(15));
pyl9 := trunc(newzl+s*(-231)+c*(15));
oldpyl0 := pyl0;
oldpyl9 := pyl9;
entergraphic;
setwindowmodeoff;
setaspect(1);
setforegroundcolor(14);
selectscreen(2);
clearscreen;
drawborder;
drawlinedirectt(1,138,640,138);    { road  }
rangecheck;
if drawit then
begin
  drawwheel;
  drawhub;
  drawlinedirectt(pxl5,pyl5,pxl4,pyl4);{low midle seg}
  drawlinedirectt(pxl9,pyl9,pxl8,pyl8);{low frnt seg }
  drawlinedirectt(pxl9,pyl9,pxl,pyl);
  drawlinedirectt(pxl,pyl,px2,py2);        {  grill  }
  drawlinedirectt(px2,py2,px3,py3);
  drawlinedirectt(px3,py3,px4,py4);     {  hood  }
  drawlinedirectt(px4,py4,px5,py5);   {  windshield  }
  drawlinedirectt(px5,py5,px6,py6);    {  roof  }
  drawlinedirectt(px6,py6,px7,py7);   { rear window  }
  drawlinedirectt(px7,py7,px8,py8);    {   trunk  }
  drawlinedirectt(px8,py8,px9,py9);
  drawlinedirectt(px9,py9,px10,py10);  {back grill  }
  drawlinedirectt(px10,py10,px11,py11);{low rear seg }
  drawlinedirectt(px14,py14,px13,py13);{rear whl well}
  drawlinedirectt(px13,py13,px12,py12);
  drawlinedirectt(px12,py12,px11,py11);
  drawlinedirectt(px18,py18,px17,py17);{frnt whl well}
  drawlinedirectt(px17,py17,px16,py16);
  drawlinedirectt(px16,py16,px15,py15);
end;
```

```
     gotoxy(14,3);
     write('Dynamic Systems Analysis Of Vehicle');
     write(' Suspension System');
     copyscreen;
     repeat
       drawit := true;
       GrafBase := HardwareGrafBase;    { selectscreen(1) }
       gills;
       exitloop;
     until keypressed;
     leavegraphic;
end;

{ This is the main routine in the simulation system. It
displays the main menu and calls the various
subsystems.An entry of 'E' will exit back to DOS }

begin
   centerx := 316;
   centery := 105;
   tm[0] := 0;
   ptr := 0;
   m := 3000/386;
   j := 15608.81;
   fz1 := 0.0;
   fz2 := 0.0;
   b1 := 40.0;
   b2 := 40.0;
   k1 := 400.0;
   k2 := 400.0;
   l1 := 50.0;
   l2 := 50.0;
   z1[0] := 0;
   z2[0] := 0;
   z3[0] := 0;
   z4[0] := 0;
   initgraphic;
   leavegraphic;
   noexit := true;
   while noexit do
   begin
   clrscr;
   writeln;
   writeln;
   writeln;
   writeln;
   write('                 WELCOME TO THE AUTOMOBILE ');
   writeln('SIMULATION SYSTEM');
   writeln;
   writeln('                        C =  CHANGE PARAMETERS');
   writeln('                              R =  RUN SIMULATION');
   writeln('                              P =  PLOT OUTPUT');
   writeln('                              N =  NUMERIC OUTPUT');
   writeln('                              E =  EXIT TO DOS ');
   writeln;
   write('                      ENTER SELECTION AND RETURN > ');
   readln(selection);
   case selection of
         'c' : admin;
         'C' : admin;
         'r' : car;
         'R' : car;
         'p' : pltroutine;
         'P' : pltroutine;
         'n' : hrdcpy;
         'N' : hrdcpy;
         'e' : noexit := false;
         'E' : noexit := false;
      end;
    end;
end.
```

APPENDIX 3

USER'S MANUAL AND CASE STUDY

USER'S MANUAL AND CASE STUDY

Getting Started

This software package is installed on the two IBM AT computers in the systems lab. The program is entered by typing AUTO in response to the DOS prompt after the initial power up. This will initilize the graphics printing, change to the AUTO directory and then enter the program. The Main Menu contained in Figure 13 is displayed. Enter the letter of the function that is desired. The functions are described in detail below. An E entry will exit from the program.

Change Parameters

This section is used to modify the system parameters. A sample of the Parameter Selection Menu is contained in Figure 15. To change a parameter value first enter the selection number 1-14 and then enter the new value. Any entry other than a 1-14 will exit back to the Main Menu. A parameter value that is outside the indicated parameter value range will result in a default value being assigned to the parameter. The error default values are:

Fx1 = Fx2 = 0	M = 2500
B1 = B2 = 0	X(0) = Xdot(0) = 0
K1 = K2 = 10	TH(0) = THdot(0) = 0
L1 = L2 = 10	

Limits were placed on the parameters for two reasons. The first reason was a real world system was being simulated therefore, real world values should be used. The second reason was to keep the automobile in the display area as much as possible.

The moment of inertia J is displayed but not changeable because it is dependent on the M, L1 and L2 entries according to the following equation:

J = (M/12) * (D**2 + H**2)

where M is the mass in slugs, D is the total length of the automobile, D = L1 + L2 + 50, and H is the height of the automobile, H = 40.

A selection of 0 will exit back to the Main Menu while a selection of 14 will clear all initial conditions, X(0) = Xdot(0) = TH(0) = THdot(0) = 0.

Run Simulation

This command will start the animation display. The automobile is first displayed using the initial conditions on X and TH. The animation is continued until any key is entered at the keyboard. The system returns to the Main Menu after a key is pressed. If plots are desired it is best to allow the animation to run as long as possible. This is because the TIME plots are from TIME = 0 to the time the simulation was stopped. Figure 16 is a sample animation display.

Plot Output

This section allows TIME and phase plots of the four state variables. Figure 14 is the Plot Selection Menu and shows what plots are available. To create a plot just enter the number of the desired plot, 1-6, and return. Any entry other than a 1-6 will exit back to the Main Menu. The animation must run and collect at least 3 data points before any plots can be generated. Figures 17-22 are sample plot outputs. These plots were generated using the parameter values displayed in Figure 15. Note that when B1 = B2 = 0 the system has no damping and will oscillate forever. This is shown by the four TIME plots in Figures 17-20. The values of X and TH oscillate from the initial condition to the negative of the initial condition. The two phase plots are circles because the system oscillates and therefore the X and TH values do not reach a steady state value.

Numeric Output

This section allows for a hardcopy output of the simulation. It prompts for an increment number. This number determines how many data points to skip over before printing the next value. The output contains the system parameters, the time (H) and the four state variables Z1-Z4. A sample numeric output is contained in Table 7. The state variables and their meanings are:

Z1	=	X
Z2	=	Xdot
Z3	=	TH
Z4	=	THdot

```
WELCOME TO THE AUTOMOBILE SIMULATION SYSTEM

        C = CHANGE PARAMETERS
        R = RUN SIMULATION
        P = PLOT OUTPUT
        N = NUMERIC OUTPUT
        E = EXIT TO DOS

ENTER SELECTION AND RETURN >
```

Figure 13. Main Menu

```
The following plots are available :

      0 :   NO PLOT DESIRED
      1 :   X           vs   TIME
      2 :   Xdot        vs   TIME
      3 :   THETA       vs   TIME
      4 :   THETAdot    vs   TIME
      5 :   Xdot        vs   X
      6 :   THETAdot    vs   THETA

  ENTER SELECTION AND RETURN >
```

Figure 14. Plot Selection Menu

```
This section allows you to change certain variables used
in the automobile simulation. The following variables and
their values are changeable [units are in () ] :

   0: NO CHANGES REQUIRED
   1: Fx1 (force on rear (lbs) [-1500 to 1500])=        0.00
   2: Fx2 (force on front(lbs) [-1500 to 1500])=        0.00
   3: B1(dmping cnst rear(lb-sec/in)[0 to 500])=        0.00
   4: B2(dmping cnst frnt(lb-sec/in)[0 to 500])=        0.00
   5: K1(spring const rear (lb/in)[10 t0 1000])=      400.00
   6: K2(spring const frnt (lb/in)[10 to 1000])=      400.00
   7: L1  (center to rear    (in)   [10 to 75])=       50.00
   8: L2  (center to front   (in)   [10 to 75])=       50.00
   9: M   (auto mass  (lbs)     [1500 to 4000])=     3000.00
  10: X(0) (x init cond   (in)      [-25 to 15])=      14.00
  11: XDOT(0) (x dot init cond (in/sec)        =        0.00
  12: T(0) (theta init cond  (deg)   [-4 to 4])=        4.00
  13: THEDOT(0) (theta dot init con)           =        0.00
  14: CLEAR ALL INITIAL CONDITIONS

      J   (moment of inertia)                  =    15608.81

              ENTER SELECTION AND RETURN >
```

Figure 15. Case 2 : Parameter Selection Menu

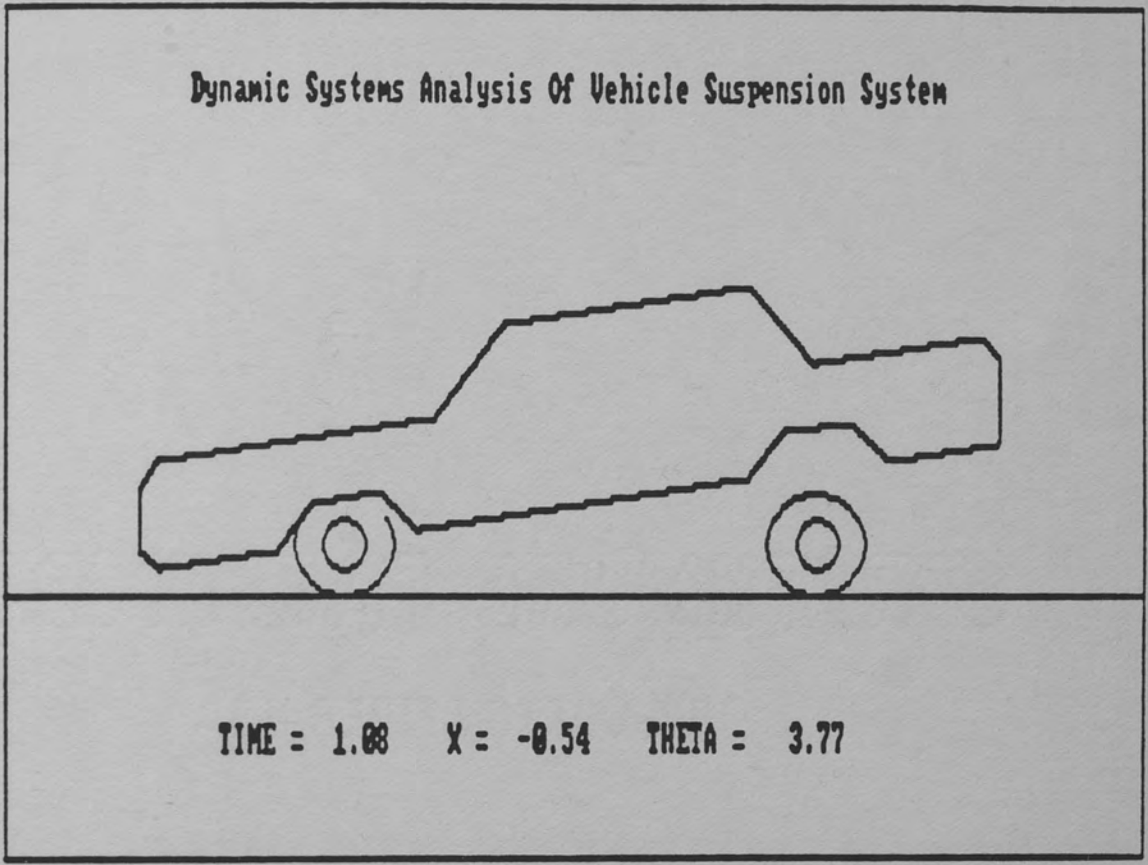

Dynamic Systems Analysis Of Vehicle Suspension System
TIME = 1.00 X = -0.54 THETA = 3.77

Figure 16. Case 2 : Automobile

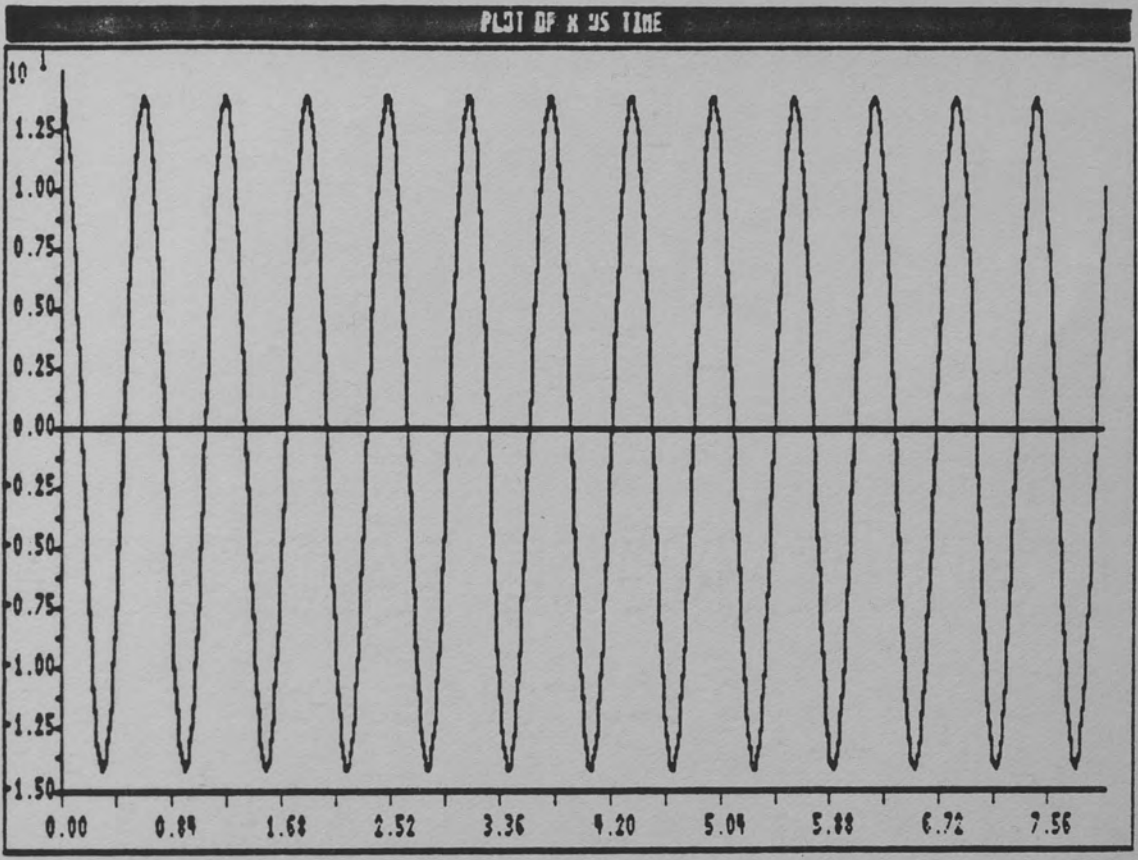


Figure 17. Case 2 : Plot of X versus TIME

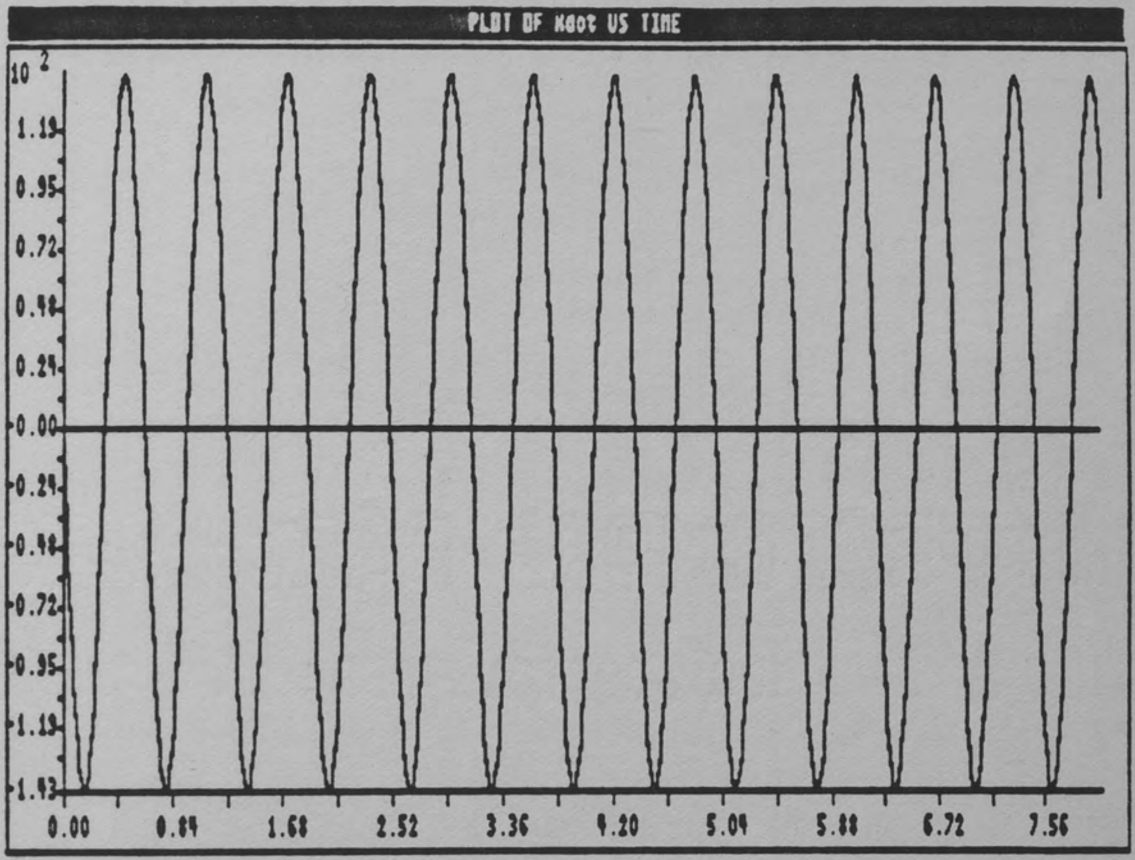


Figure 18. Case 2 : Plot of Xdot versus TIME

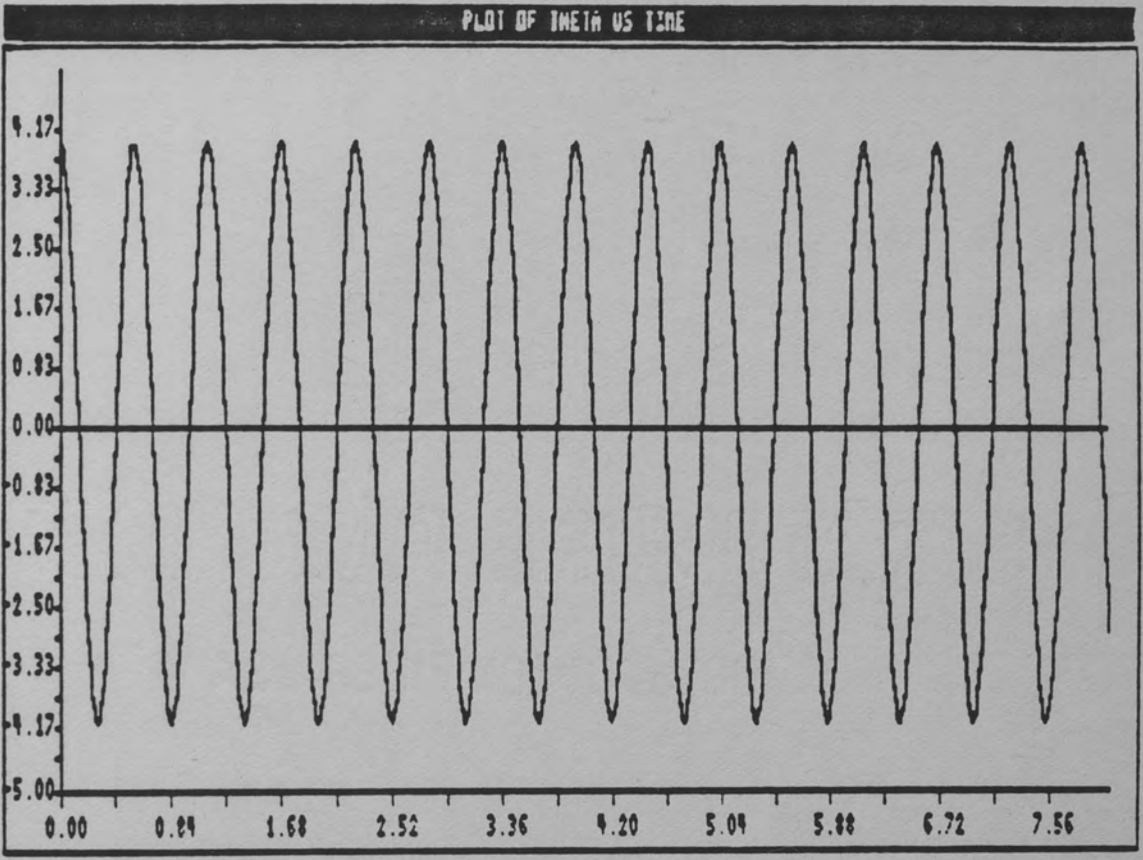


Figure 19. Case 2 : Plot of TH versus TIME

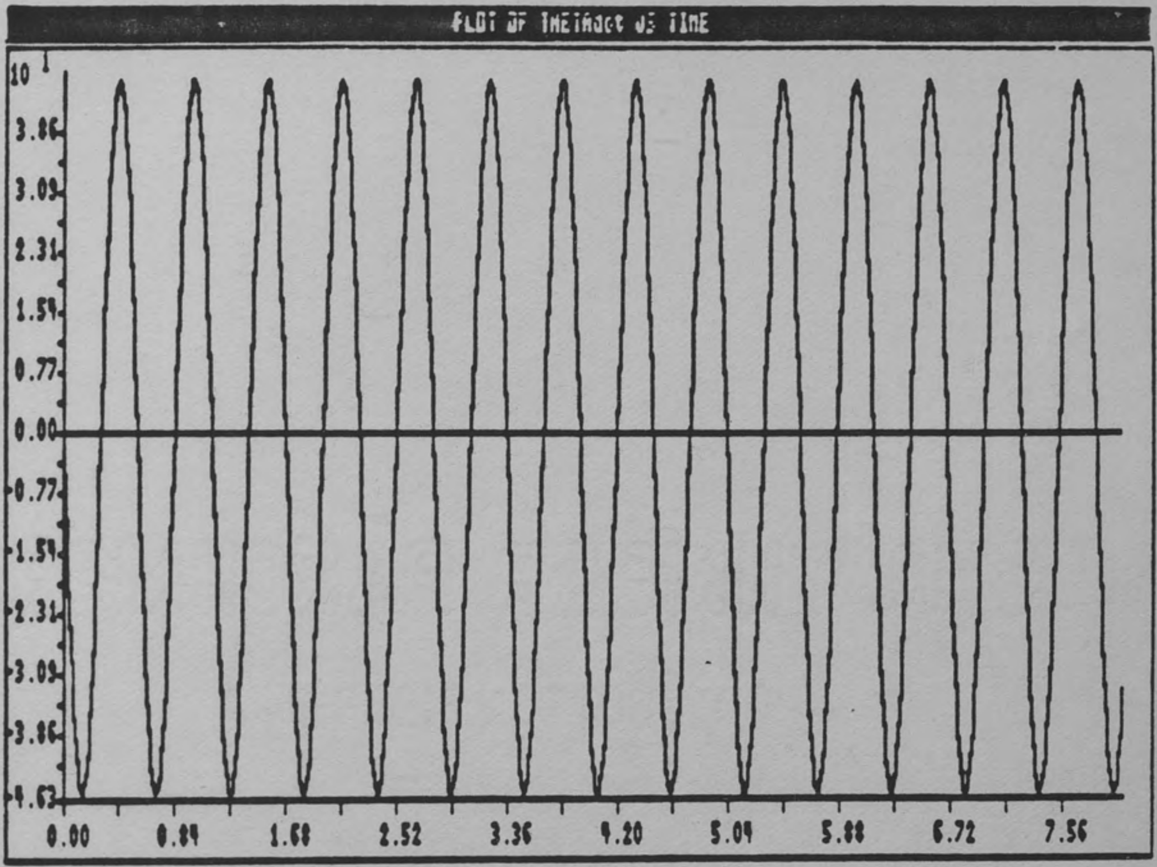


Figure 20. Case 2 : Plot of THdot versus TIME

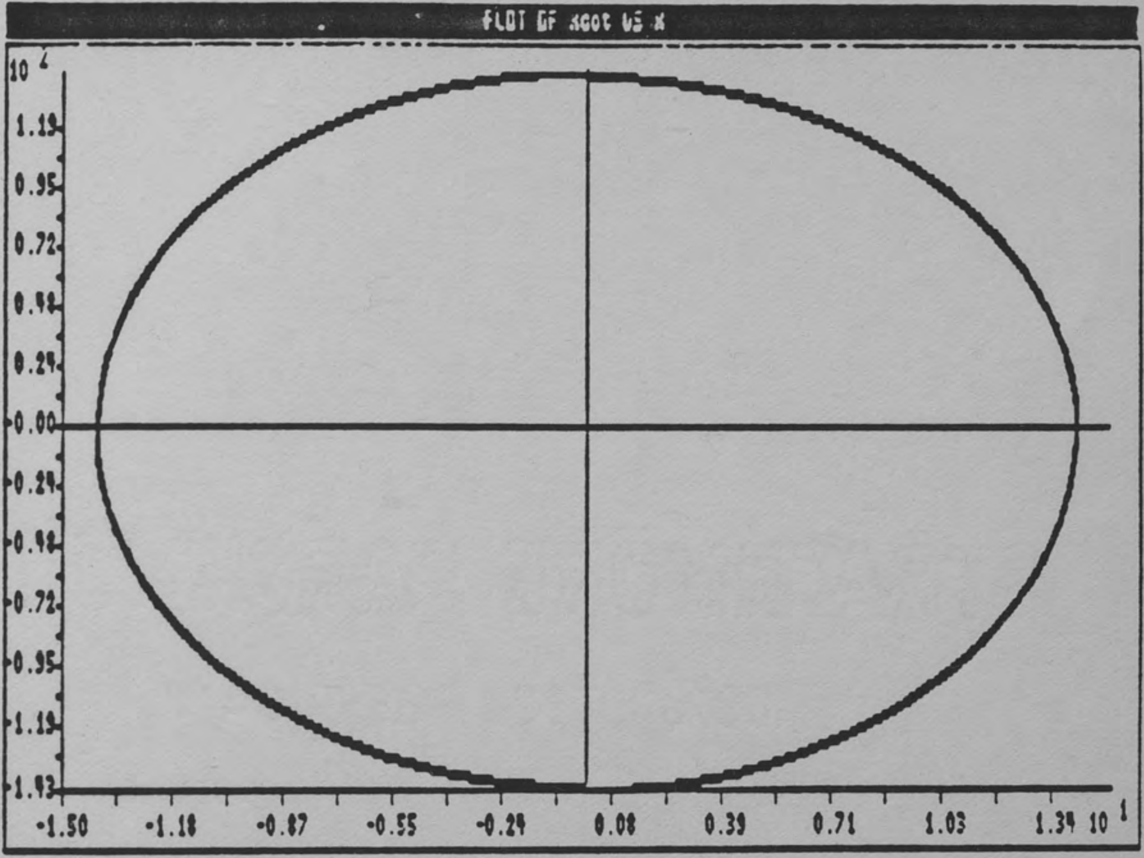


Figure 21. Case 2 : Plot of Xdot versus X

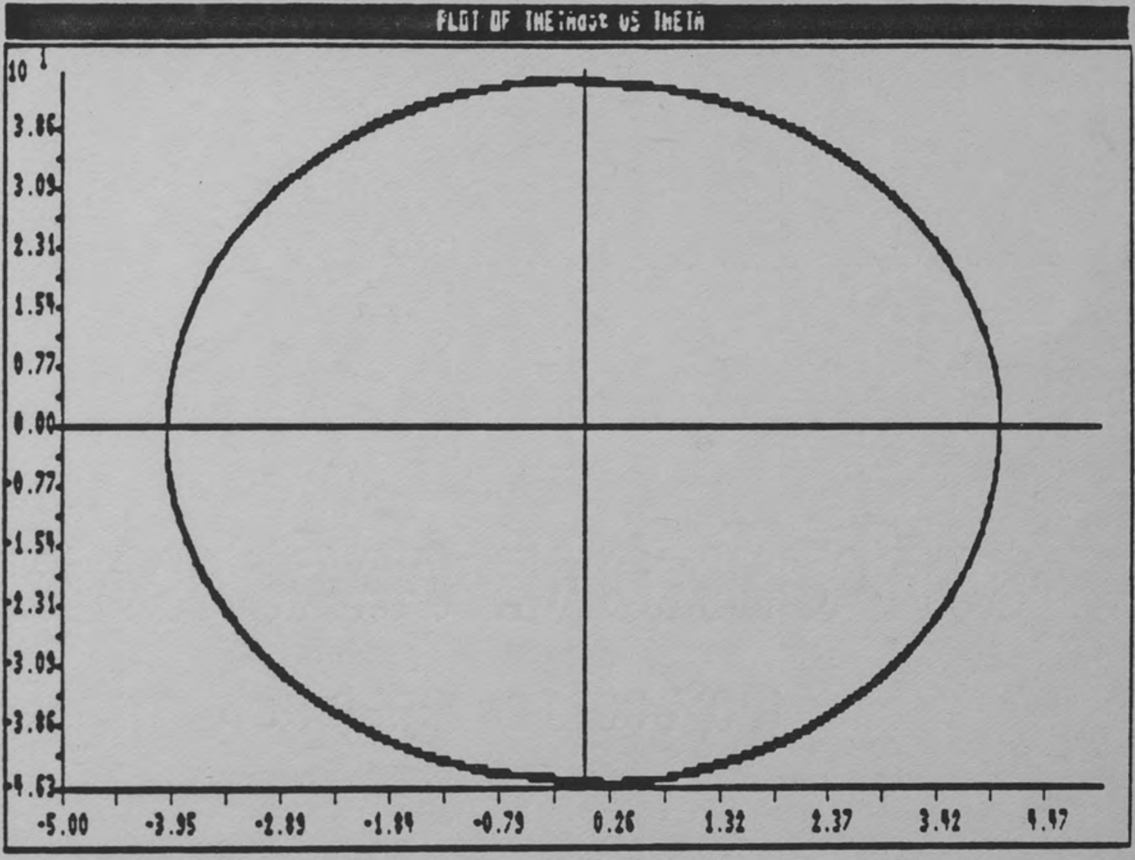


Figure 22. Case 2 : Plot of THdot versus TH

TABLE 7

CASE 2 : NUMERIC OUTPUT

SIMULATION HARDCOPY OUTPUT

```
Fx1 = 0.0    Fx2 = 0.0  B1 =  0.0     B2 =  0.0
K1 = 400.0    K2 = 400.0  L1 = 50.0     L2 = 50.0
```

H	Z1	Z2	Z3	Z4
0.00	14.00	0.00	4.00	0.00
0.10	7.39	-120.63	1.70	-40.99
0.20	-6.19	-127.38	-2.56	-34.83
0.30	-13.93	-13.89	-3.87	11.39
0.40	-8.52	112.72	-0.73	44.51
0.50	4.94	132.91	3.25	26.43
0.60	13.73	27.64	3.49	-22.05
0.70	9.56	-103.73	-0.28	-45.17
0.80	-3.63	-137.17	-3.73	-16.34
0.90	-13.40	-41.12	-2.89	31.29
1.00	-10.52	93.75	1.27	42.92
1.10	2.29	140.12	3.97	5.19
1.20	12.94	54.22	2.10	-38.51
1.30	11.37	-82.86	-2.19	-37.92
1.40	-0.93	-141.72	-3.96	6.29
1.50	-12.36	-66.79	-1.18	43.26
1.60	-12.11	71.19	2.96	30.47
1.70	-0.44	141.97	3.69	-17.37
1.80	11.65	78.73	0.18	-45.23
1.90	12.74	-58.83	-3.54	-21.07
2.00	1.80	-140.86	-3.19	27.33
2.10	-10.84	-89.91	0.83	44.29
2.20	-13.25	45.91	3.89	10.31
2.30	-3.15	138.39	2.48	-35.53
2.40	9.92	100.23	-1.79	-40.50
2.50	13.63	-32.55	-4.00	1.11
2.60	4.47	-134.60	-1.61	41.45
2.70	-8.91	-109.58	2.63	34.11
2.80	-13.88	18.88	3.85	-12.46
2.90	-5.75	129.53	0.64	-44.70
3.00	7.81	117.89	-3.30	-25.52
3.10	13.99	-5.03	-3.44	23.01
3.20	6.97	-123.21	0.38	45.08
3.30	-6.64	-125.07	3.76	15.29
3.40	-13.97	-8.87	2.82	-32.08
3.50	-8.12	115.71	-1.37	-42.56

REFERENCES

Basic by MicroSoft Corp. Boca Raton, Fla.: IBM Personal Computer Software Library, 1982.

Bronson, Richard. Matrix Methods An Introduction. New York : Academic Press, Inc., 1970.

Foley, J.D. and Van Dam, A. Fundamentals of Interactive Computer Graphics. New York : Academic Press, Inc., 1985.

Held, Gilbert. IBM PC Users Reference Manual. Hasbrouck Heights, N.J.: Hayden Book Company, 1984.

LENIPEN The Intellegent Color Graphics System. Hillside, N.J.: Duncan-Atwell Computerized Technologies, Inc., 1984.

TURBO PASCAL Reference Manual Version 3.0. Scotts Valley, Ca.: Borland International, 1984.

TURBO GRAPHIX TOOLBOX Graphics Tools for TURBO PASCAL. Scotts Valley, Ca.: Borland International, 1985.

TUTSIM. Palo Alto, Ca.: APPLIED i, 1985.