A Study of Dynamic Control of the Inverted Pendulum System

Koon T. Ang

University of Central Florida

Part of the Computer Engineering Commons

Find similar works at: https://stars.library.ucf.edu/rtd

University of Central Florida Libraries http://library.ucf.edu

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

https://stars.library.ucf.edu/rtd/4899
A STUDY OF THE DYNAMIC CONTROL OF
THE INVERTED PENDULUM SYSTEM

BY

KOON TOCK ANG
B.S.E., University of Central Florida, 1983

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

Fall Term
1986
ABSTRACT

This report describes the simulation of an inverted pendulum control system. The purpose is to provide an interesting learning process through high resolution color graphics animations in the control of dynamic systems. The software uses the graphic capabilities extensively to make it very user-friendly and highly interactive. A numerical analysis method is used to solve the systems of equations. The animation driven by the results is then displayed on the video terminal. Facilities range from selection of controllers, changing of system parameters, plotting graphs, and hardcopy outputs.
ACKNOWLEDGMENTS

First, I would like to give a special acknowledgment to those who have influenced me to strive for a higher education: to my Mom, Dad, and sister without whose love and support, I would not have made it. My love and heartiest appreciation to my fiancee for her love, understanding, and patience. Special thanks are also extended to Dr. Harold I. Klee, who served as my advisor, teacher, and chairman of my committee. And to Lisa, who proofread for me.
TABLE OF CONTENTS

LIST OF TABLES ........................................ vi
LIST OF FIGURES ....................................... vii
INTRODUCTION ........................................... 1

Chapter

I. SYSTEM DESCRIPTION ................................. 3
   System Limitations ................................ 5

II. SYSTEM MODEL ....................................... 6
   Mathematical Model ................................ 6
   Numerical Analysis Method ....................... 11
   Animation ........................................... 14

III. CONTROLLERS ....................................... 15
   Proportional Control (P) ........................ 15
   Prop. plus Integral Control (PI) ............... 18
   Prop. plus Derivative Control (PD) ............ 18
   Prop. plus Integral plus Derivative (PID) .... 21
   Lead-Lag Control (Lead-Lag) .................... 21

IV. COMPUTER GRAPHICS ................................. 22
   Software Design ................................... 22
   Inverted Pendulum Animation .................... 24
   User's Manual ..................................... 28
   1. Simulate System ................................ 28
   2. Change Controller ............................... 30
   3. Change Parameters .............................. 30
   4. Plot Graphs .................................... 31
   5. Numerical Outputs ............................... 32

V. SOFTWARE OPTIMIZATION ............................ 45

VI. RESULTS ............................................ 48

VII. SUMMARY AND CONCLUSIONS ....................... 59
Appendices

1. PASCAL LISTINGS OF CONTROLLERS
   AND ACSL MODEL LISTINGS . . . . . . . . . . . . . . . . . 61
2. SOFTWARE LISTING . . . . . . . . . . . . . . . . . . . . . . . 81

REFERENCES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 107
<table>
<thead>
<tr>
<th>Table Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Computer Printout from the Numerical Outputs Procedure</td>
<td>44</td>
</tr>
<tr>
<td>2</td>
<td>Runge-Kutta Versus Exact Solution for the Proportional Controller</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>Runge-Kutta Versus Exact Solution for the Proportional Plus Derivative Controller</td>
<td>54</td>
</tr>
<tr>
<td>4</td>
<td>Runge-Kutta Output of the Lead-Lag Controller</td>
<td>57</td>
</tr>
<tr>
<td>5</td>
<td>ACSL Output of the Lead-Lag Controller</td>
<td>58</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

1. Inverted Pendulum System ........................................ 4
2. Free-Body Diagrams ................................................. 8
3. Closed-Loop Block Diagram of the Inverted Pendulum System ................................................. 10
4. Response of the Proportional Control for Different Gain Settings ................................................. 17
5a. Response of the Proportional Plus Derivative Control ................................................. 19
5b. An Overdamped Response of the Proportional Plus Derivative Control ................................................. 20
6. Definition of the Points that Describe the Pendulum and the Cart ................................................. 27
7. Main Menu .......................................................... 35
8a. Simulate System Menu (Integration Step Size) .......................................................... 36
8b. Simulate System Menu (Bypass Animation) .......................................................... 37
8c. Animation of the Pendulum and Cart Assembly .......................................................... 38
9. Change Controller Menu ................................................ 39
10. Change Parameters Menu ................................................ 40
11a. Plot of $\theta$ Versus Time for the Lead-Lag Controller Example Run ................................................. 41
11b. Plot of $\theta$ Versus $\theta$ for the Lead-Lag Controller Example Run ................................................. 42
11c. Plot of Distance Versus $\theta$ for the Lead-Lag Controller Example Run ................................................. 43
12a. Proportional Controller Response for Gain $K_mK_t=5.0$ (i.e., $<K_{cr}$) ................................................. 51
12b. Proportional Controller Response for Gain $K_mK_t=11.772$ (i.e., $=K_{cr}$) ................................................. 52
12c. Proportional Controller Response for Gain $K_m \cdot K_t = 30.0$ (i.e., $> K_{cr}$) ....... 53

13a. Proportional Plus Derivative Controller Response for Gains, $K_m = 10.0$ and $K_t = 10.0$ ....... 55

13b. Proportional Plus Derivative Controller Response for Gains, $K_m = 25.0$ and $K_t = 1.0$ (High Gain) ....... 56
INTRODUCTION

One of the areas of interest to the Computer Engineering Department is development of software programs for use as teaching tools. These programs would help the undergraduate engineering students to understand the subject of dynamic systems and control theory. The focus of control studies is to determine whether a particular system will achieve steady state.

This research report describes the control of an unstable mechanical system: the inverted pendulum, also known as the "broom" or "stick balancer." It also includes one method of using desktop computers to simulate and analyze that system. The software is user-friendly and utilizes the graphic capabilities of the computer to animate the mechanical system, making it possible to investigate the effects of different initial conditions, system parameters, physical components, and choices of controllers. Within the given parameters, a solution to the system of equations can be found. A pictorial representation of the effects is shown through animation. Other options include plots of curve and hardcopy outputs of any of the calculated variables.
Several similar projects have been completed. Projects on controlling level in a fluid system and modelling an automobile suspension system, are examples of the software package. The projects not only function as a teaching tool but also demonstrate the potential of portable computers. The software programs should be easily modifiable by other students and portable to other computers of similar or greater capabilities. A suitable higher level programming language must satisfy this criterion. Because the PASCAL programming language is widely used by college students, the TURBO PASCAL package is the obvious choice; and its accompanying TURBO TOOLBOX GRAPHIX package enhances the animation.

The system descriptions and its limitations are discussed in Chapter 1. Chapter 2 follows through the method of using the Runge-Kutta technique to solve the set of first-order linearized differential equations. A thorough investigation of the various controllers is described in Chapter 3. Chapter 4 reviews the software design, the graphic animation, and the user's manual. Finally, Chapter 5 discusses the software optimization. Appendices 1 and 2 contain listings of the simulated software programs.
CHAPTER I

SYSTEM DESCRIPTION

The inverted pendulum system shown in Figure 1 combines a mixture of mechanical and electrical components. A broom or stick is mounted with a ball bearing pivot on a motor driven carriage. The broom is free to fall only in the plane of the paper. The main objective is to control or keep the broom in a vertical position or at some angle from the vertical axis. The tilt angle is measured as an electrical signal and input to a controller that drives the motor which moves the cart in one direction or the other, thereby balancing the pendulum. This system is similar to trying to balance a broom on the palm of the hand; to keep the broom vertical, one must execute rapid horizontal movements at the lower end.

The system represents a controlled laboratory idealization of unstable mechanical systems that are encountered from time-to-time. One might notice the similarities of this system with controlling a space rocket booster on takeoff; although the difference is that the booster can fall in any direction while the inverted pendulum system can only fall in the plane of the paper.
Figure 1. Inverted Pendulum System
System Limitations

As with all mechanical systems, there are several assumptions or limitations involved in modelling of the inverted pendulum system. Assumptions are made that (1) the pivot between the broom and the cart is completely frictionless, (2) the dynamic response of the motor is sufficiently fast, (3) the wheels of the cart do not slip, and (4) there is sufficient room for the cart to move around. Owing to the physical limitations of the components of the system, only small angular deviations should be attempted. Neglecting this could result in inaccurate data.
CHAPTER II

SYSTEM MODEL

Simulating the single broom problem involves three steps. The first step is to apply some fundamental principles of physics to describe the system and obtain the system of differential equations. Second, the numerical integration can then be used to solve this system of equations. The final step involves applying the calculated output and animating the inverted pendulum system on the computer screen.

Mathematical Model

Before proceeding any further, the following system variables should be defined since they are used throughout this research paper:

\( \Theta \) : Angular Deviations of the Pendulum from the Vertical Axis (degrees)

\( \dot{\Theta} \) : Angular Velocity of the Pendulum (degrees/second)

\( X \) : Horizontal Displacements of the Cart (meters)

\( \dot{X} \) : Velocity of the Cart and Pendulum Assembly (meters/second)

\( L \) : Half Length of the Pendulum (meters)

\( M \) : Mass of the Pendulum (kilograms)
Mc: Mass of the Cart (kilograms)

g: Gravitational Acceleration = 9.81
(meters/second.second)

f: Control Force of the Inverted Pendulum System
(NEWTONS)

\( \Theta(0) \): initial conditions for \( \Theta \) (degrees)

\( X(0) \): initial conditions for \( X \) (degrees)

\( \Theta_{sp} \): set point value (degrees)

Kt: transmitter gain (volts/degree)

Km: the controller gain (NEWTONS/volt)

h: integration step size (seconds)

tf\(_{final}\): period of time for which the simulation is
executed (seconds)

t: time variable of the system (seconds)

Because the cart moves in horizontal directions and the
broom's behavior affects this movement, two separate free
body diagrams were needed (Figure 2). By taking moments
around the center of gravity of the broom and summing up all
forces acting on the cart and broom in vertical and horizon-
tal directions, the following system of equations is ob-
tained:
Figure 2. Free-Body Diagrams
\[ \begin{align*}
Mc^{\dddot{X}} + M^{\dddot{X}} + M^{\dddot{L}*} \Theta \cdot \cos(\Theta) - M^{\dddot{L}*} (\Theta) \cdot (\Theta) \\
\cdot \sin(\Theta) &= f \quad \text{equation 1}
\end{align*} \]

\[ \begin{align*}
(M^{\dddot{L}*}L*)/3 + M^{\dddot{L}*[X \cdot \cos(\Theta) + L* \Theta]} - \\
M^{\cdot \cdot \cdot} g^{\cdot \cdot \cdot} L^* \cdot \sin(\Theta) &= 0 \quad \text{equation 2}
\end{align*} \]

The two second-order differential equations above are non-linear, and thus do not have elementary solutions. Because this study is confined to small angular deviations, certain terms can be dropped. After linearizing, the two second-order differential equations that describe the inverted pendulum system are defined as follows:

\[ \begin{align*}
(Mc + M)^{\dddot{X}} + M^{\dddot{L}*} \Theta &= f \quad \text{equation 3}
\end{align*} \]

\[ \begin{align*}
M^{\dddot{L}*}X + [(M^{\dddot{L}*}L)/3 + M^{\dddot{L}*}L^]* \Theta - \\
M^{\cdot \cdot \cdot} g^{\cdot \cdot \cdot} L^* \Theta &= 0 \quad \text{equation 4}
\end{align*} \]

By rearranging the terms, the equations now become:

\[ \begin{align*}
\Theta &= 3*(Mc+M)^*g^* \Theta - \frac{3f}{L^*(4*Mc+M)} \quad \text{equation 5}
\end{align*} \]

\[ \begin{align*}
X &= - \frac{3*M^*g^* \Theta}{(4*Mc+M)} + \frac{4f}{(4*Mc+M)} \quad \text{equation 6}
\end{align*} \]
Even though the number of independent variables match the number of unknowns, the equations are only a function of the tilt angle, \( \Theta \), and the control force, \( f \). That is, \( \Theta \) is independent of \( X \) but the opposite is not true. As such, the system of equations remains unsolvable because the control force, \( f \), is as yet unspecified. Figure 3 depicts the closed-loop diagram of the total control system.

![Figure 3. Closed-Loop Block Diagram of the Inverted Pendulum System](image)

From the block diagram, the control force, \( f \), is shown as a function of the deviation angle, the set point angle, and the selection of the controllers. The user will have a choice of five different controllers. Chapter 3 focuses on an in-depth study of these controllers and their effects on this control system. Please refer to this chapter for detailed explanations of \( K_m \) and \( K_t \).
Numerical Analysis Method

After selection of the physical parameters of the inverted pendulum system, a numerical analysis method is used to solve the two second-order differential equations. Together they define a fourth-order system. Further reduction can be made by introducing four new variables as follows:

\[ Z[1] = \Theta \text{ or angular deviations} \]
\[ Z[2] = \dot{\Theta} \text{ or angular velocity} \]
\[ Z[3] = X \text{ or horizontal displacement} \]
\[ Z[4] = \dot{X} \text{ or velocity} \]

Using these new state variables, equation 5 and equation 6 can be rewritten as a set of four first-order differential equations as follows:

\[ Z[2] = \frac{3 \cdot (Mc + M) \cdot g \cdot Z[1]}{L \cdot (4Mc + M)} - \frac{3f}{L \cdot (4Mc + M)} \]
\[ Z[4] = -\frac{3 \cdot M \cdot g \cdot Z[1]}{(4Mc + M)} + \frac{4f}{(4Mc + M)} \]
The system of differential equations is now in its normal form. The fourth-order Runge-Kutta numerical integration technique was chosen because it is highly accurate and easily adapted to solve any systems of equations. It is given by:

\[ Y[n+1] = Y[n] + \left(\frac{h}{6}\right) \left(K_{n1} + 2K_{n2} + 2K_{n3} + K_{n4}\right) \]

where

- \( K_{n1} = g(X_n, Y_n) \)
- \( K_{n2} = g(X_n + 0.5h, Y_n + 0.5hK_{n1}) \)
- \( K_{n3} = g(X_n + 0.5h, Y_n + 0.5hK_{n2}) \)
- \( K_{n4} = g(X_n + h, Y_n + hK_{n3}) \)

\( h \) is the integration step size

\( g(X_n, Y_n) \) is the function to be evaluated and \( X_n \) and \( Y_n \) are the independent variables.

i.e., \( Y' = g(X_n, Y_n) \)

The sum \( \left(K_{n1} + 2K_{n2} + 2K_{n3} + K_{n4}\right)/6 \) can be interpreted as an average slope. Note that \( K_{n1} \) is the slope at the left-hand end of the interval; \( K_{n2} \) is the slope at the midpoint using the Euler formula to go from \( X_n \) to \( X_n + h/2 \); \( K_{n3} \) is a second approximation to the slope at the midpoint; and finally, \( K_{n4} \) is the slope at \( X_n + h \) using the Euler formula and the slope \( K_{n3} \) to go from \( X_n \) to \( X_n + h \). The smaller integration step size, \( h \), gives accurate results.
Applying the Runge-Kutta algorithm to the inverted pendulum system for the $Z[1]$ state variable resulted in:


Applying the same algorithm to state variable $Z[2]$ generated a different set of slopes:

$$K_1[2] = 3*(M_c+M)*g*(Z[1]) - \frac{3f}{L*(4*M_c+M)}$$
$$K_2[2] = 3*(M_c+M)*g*(Z[1]+0.5h*K_1[1]) - \frac{3f}{L*(4*M_c+M)}$$
$$K_3[2] = 3*(M_c+M)*g*(Z[1]+0.5h*K_2[1]) - \frac{3f}{L*(4*M_c+M)}$$
$$K_4[2] = 3*(M_c+M)*g*(Z[1]+h*K_3[1]) - \frac{3f}{L*(4*M_c+M)}$$


The technique is repeated to establish the Runge-Kutta representations for state variables $Z[3]$ and $Z[4]$. The Runge-Kutta algorithm is then tested for accuracy for all the controllers. The testings and the results are left for discussion in the next chapter. The different PASCAL programs are listed in appendices 1 and 2.
Animation

The calculated values from the numerical analysis method -- namely the angular deviations, $\Theta$, and the translational dynamics in $X$ -- are used to drive the animation section of the software program. The whole cart and pendulum assembly can be seen to move across the computer screen. This movement is made as realistic as possible within the limitations of the computer hardware. Positive increasing $\Theta$ is shown by the clockwise movements of the pendulum with respect to the pivot point and vice-versa; whereas increasing horizontal displacements of the cart and pendulum assembly are indicated by movements toward the right of the screen. Since the size of the video display terminal is limited to eighty (80) columns, when the animated assembly reaches the edge of the screen, the movements are wrapped around to the opposite edge.
CHAPTER III

CONTROLLERS

There is a choice of five different controllers in the software program. They are Proportional Control (P), Proportional plus Integral Control (PI), Proportional plus Derivative Control (PD), Proportional plus Integral plus Derivative Control (PID), and Lead-Lag Control (Lead-Lag). Once selected, the controller will be utilized to generate the control force, \( f \).

**Proportional Control (P)**

Selection of the proportional control will cause the motor to produce a force proportional to the angular deviation; that is,

\[
f = -Km*Kt*(\Phi \text{ sp} - \Phi)
\]

Plugging this into equations 3 and 4 and double integrating both equations, the analytical solutions become

\[
\Phi = \Phi(0)*\cos(w*t)
\]

\[
X = \Phi(0)*\left(a + b*w^2\right)*\left[1 - \cos(w*t)\right]
\]

where the initial conditions are given as

\[
X(0) = 0.0
\]

\[
X(0) = 0.0
\]

\[
\Phi(0) = 0.0
\]
and where

\[ w^2 = -\frac{3(Mc+M)g + \frac{3KmKt}{L(4Mc+M)}}{L*4Mc+M} \]

\[ a = -\frac{(KmKt)}{(M + Mc)} \]

\[ b = \frac{M*L}{(M + Mc)} \]

After an exhaustive investigation into these solutions, the following observations apply. There is a "critical" gain, Kcr, for this particular controller and it is simply defined as \( Kcr = g(M + Mc) \).

If \( KmKt \) is greater than \( Kcr \), then

\[ \Theta = \Theta(0)\cos(w*t) \]

\[ X = \Theta(0)*(a+b*w^2)*[1-\cos(w*t)] \]

\[ (w^2) \]

If \( KmKt \) is less than \( Kcr \), then

\[ \Theta = \Theta(0)\cosh(|w|*t) \]

\[ X = \Theta(0)*(a-b*|w|^2)*[\cosh(|w|*t)-1] \]

\[ (|w|^2) \]

and if \( KmKt \) is equal to \( Kcr \), then

\[ \Theta = \Theta(0) \]

\[ X = \Theta(0)*a*t^2 \]

\[ 2.0 \]

The graphs in Figure 4 show the results for the different gain settings. It is concluded that the inverted pendulum system cannot be controlled using Proportional control; i.e., the system is unstable. This controller is included to educate the students that not all controllers
have a stable response.

Figure 4. Response of the Proportional Control for Different Gain Settings
Proportional Plus Integral Control (PI)

With the inclusion of the integral signal to the inverted pendulum system, memory is introduced into the system. The control force, \( f \), now becomes:

\[
f = - K_m [K_t (\dot{\Theta} - sp - \Theta) - \int_0^t (\dot{\Theta} - sp - \Theta) dt]
\]

Proportional Plus Derivative Control (PD)

Usage of the proportional plus derivative control causes the motor to take action to counteract the movement of the pendulum when the deviations are about to occur. The control force, \( f \), is then defined as:

\[
f = - K_m K_t (\dot{\Theta} - sp - \Theta) + K_m (\Theta)
\]

Applying this control force into equation 3 and solving that equation resulted in the exact solution of:

\[
\Theta = (\Theta(0) W \exp(-\alpha t)) \times \\
\sin\left((t \sqrt{(w^2 - \alpha^2) + \arctan\left(\frac{\sqrt{(w^2 - \alpha^2)}}{\alpha}\right)})\right) \\
\sqrt{(w^2 - \alpha^2)}
\]

and where

\[
w^2 = - \frac{3(M_c + M)g}{L*(4*M_c + M)} + \frac{3*K_m K_t}{L*(4*M_c + M)}
\]

\[
\alpha = \frac{3*K_m}{2*L*(4*M_c + M)}
\]

A graphical response of this controller is shown on the next page in Figure 5a. The above result is only true if \( w \) and \( \alpha \) values render the radical of the square root real. Otherwise the following overdamped system response is true (Figure 5b):
\[
\vartheta = \frac{(0.5 \cdot \vartheta(0) + \sqrt{\alpha^2 - w^2})}{\sqrt{\alpha^2 - w^2}} \cdot \left[ (\alpha + \sqrt{\alpha^2 - w^2}) \cdot \exp\left(\alpha - \sqrt{\alpha^2 - w^2}\right) \cdot t \right] - (\alpha - \sqrt{\alpha^2 - w^2}) \cdot \exp\left(\alpha - \sqrt{\alpha^2 - w^2}\right) \cdot t
\]

Figure 5a. Response of the Proportional Plus Derivative Control
Figure 5b. An Overdamped Response of the Proportional Plus Derivative Control
**Proportional Plus Integral Plus Derivative Control (PID)**

Here, both the properties of the integral and derivative controls are integrated into the proportional control. The control force, $f$, now becomes:

$$f = -K_m[-\Phi + K_t(\Theta sp - \Theta) - \int_0^t (\Theta sp - \Theta) dt]$$

**Lead-Lag Control (Lead-Lag)**

A different type of control is introduced. The control force, $f$, is found to be:

$$f = [(K_m K_t \sigma_A )/\sigma_B] \{ (1 - \sigma_B/\sigma_A )^1 Z[5] + \sigma_B (\sigma_A /\sigma_B ) (\Theta sp - \Theta) \}$$

where $\sigma_A$ and $\sigma_B$ are design parameters; and as a reasonable design choice, $\sigma_A$ is equal to 1.0 and $\sigma_B$ is set to 4.0. The variable $Z[5]$ is the result from another Runge-Kutta that evaluates the Lead-Lag controller. $Z[5]$ is defined as:

$$Z[5] = -\sigma_B Z5 + \sigma_B (\Theta sp - \Theta)$$

where $Z5$ is some dummy variable.
CHAPTER IV

COMPUTER GRAPHICS

Credit for this research report is due in part to the results and feasibility studies of earlier research works compiled for the Computer Aided Instruction package. Earlier works have indicated that the best choice of programming language is to use TURBO PASCAL in conjunction with the TURBO TOOLBOX GRAPHIX package. The TOOLBOX is a powerful graphic tool containing facilities for window management, drawing tools, plotting curves, and the ability to create animation. With the TOOLBOX, animation of the inverted pendulum is simplified.

Software Design

One of the strong points of this research report is its user-friendly program -- thus the TOOLBOX is used extensively. In the main menu, the closed-loop block diagram of the inverted pendulum system is displayed on the screen using several standard procedures of the TOOLBOX; for example, DRAWLINE and DRAWSQUARE. By entering the "up" or the "down" arrow key from the keyboard, followed by a carriage return, selections are made. Passage to different functions of the software; for example, to change the controller, to change the system parameters, or to plot graphs, and obtain hard
copies, is also executed very easily. All changes or actions are prompted directly on the screen. Any incorrect entry by the user will be detected by the software program and the appropriate actions taken. Thus, the software program can only be exited through normal channels.

A menu of all the available controllers is displayed on the lower right-hand corner of the screen when the change of controller selection is made. On the other hand, if the change of parameters is required, control of the software is transferred directly to the figure of the closed-loop block diagram. The values of the parameters can be seen to blink. Movement from one parameter to another is achieved by the "up" and "down" arrow keys. When the desired position of the parameter is found, the user must first enter the "space bar," and then make the new changes, followed by the "carriage return." After all changes have been made, move the cursor to the blinking message 'changes COMPLETED' and hit "carriage return," which returns the control to the main menu.

The simulation of the system can now proceed. The user will be prompted for the integration step size, \( h \), and the length of which to simulate the system, \( t_{\text{final}} \). On simulating the system, all the calculated variables such as time, \( \theta, \dot{\theta}, X, X, \) and force, \( f \), are stored in arrays. The storage capacity is seven hundred (700) points, and the maximum and minimum values of these points are also computed by the
software. The plotting procedures use these maximum and minimum values as the scaling factor for the axis. Here the procedures DRAWAXIS and DRAWPOLYGON of the TOOLBOX were implemented since they facilitate the drawing of curves. Any of the calculated variables can be plotted on the screen. The user will be prompted for the Y and X axis variables; even the same variable can be plotted. Of course, this would give a straight line passing through the origin at an angle of 45 degrees from the axis. The plotted curves will also be labelled with the appropriate headings.

Other options available to the user include viewing the animation of the inverted pendulum system on the screen or obtaining a hard copy result of the simulation. One to six calculated variables can be output to the printer.

Because this is a rather large software program, normal compilation cannot be executed; consequently, overlays were used in the software program. The usage of overlays added a slight overhead on the runtime execution.

A listing of the software program is contained in Appendix 2.

Inverted Pendulum Animation

The animation of the inverted pendulum poses quite a challenge. The graphics should be displayed as precisely and as quickly as possible. In order to give the effects of a moving cart and pendulum assembly, the pendulum is drawn separately from the cart. (Refer to Figure 6 to observe the
points that describe the cart and pendulum. They are found in the overlay procedure INVERTEDPENDULUM of the main program in Appendix 2.) The pendulum is drawn with four pairs of points and the pivot is represented by the point C. To accurately show the exact deviated angle, all new angles are calculated by procedures CALCULATE and DEVIATION found in the overlay procedure INVERTEDPENDULUM. These procedures perform the adding and/or subtracting of the appropriate sine and cosine of the new angle. The new deviation is then drawn onto WINDOW 2 with the support of the pendulum. (See the points F, G, H, and I of pendulum in Figure 6.) Although, the deviated angle can span from -360 to 360 degrees, the physical design of the inverted pendulum system can only accommodate angles from approximately -93 to 93 degrees. Otherwise, the pendulum will hit the surface. The animation section of the software program reflects this limitation.

Next, the cart, which is described by eight different points, is drawn in another separate window. Drawing involves TOOLBOX functions DRAWLINE, DRAWSQUARE, and DRAWCIRCLESEGMENT. Once the drawing of the cart is accomplished, it is stored in WINDOW 3 and never redrawn.

The animation or the movement of the cart and pendulum assembly is achieved by defining WINDOW 11, which is slightly larger than both the windows containing the drawings of the pendulum and the cart (WINDOWS 2 and 3). WINDOW 11 is
then moved across the screen by the amount specified by the integer variable MOVESIZE. To make the animation even more realistic, the road is defined by WINDOW 4, and its motion across the screen is shown by the movements of hatches.

Each time a new set of values of $\Theta$ and $X$ is encountered, a new position of the pendulum and the road is drawn on a "hidden" screen or in Random Access Memory. The new drawing is then copied into the video display area. This process is executed as many times as there are $\Theta$ and $X$ points; thus, giving the impression of one continuous movement.

The values of $\Theta$, $X$, the variables time and $\Theta$ sp are also shown simultaneously during the display of the animation. While in this mode, the user, by the use of function keys, has the options of freezing the screen (F2), of changing or altering any of the design parameters (F1), and/or to exit the software program directly to DOS (F3).
Figure 6. Definition of the Points that Describe the Pendulum and the Cart
User's Manual

To run this software program, type "PENDULUM" in response to the DOS prompt. This action will load and run the program. The main menu (Figure 7) will be displayed on the screen. Note that, on all start up, the Lead-Lag controller was chosen for the design system. This is simply the design of the software program. The user now has a choice of one of the following selections:

```
Selections:

- simulate system
- change controller
- change parameters
- plot graphs
- numerical outputs
- exit simulation
```

A blinking arrow will be shown on the screen. To move this blinking arrow, enter the "up" or "down" arrow key from the keyboard. To make the selection, enter "carriage return." Selecting the option 'exit simulation' will exit the program to DOS.

SECTION 1 - Simulate System

If this selection is made by the user, the main menu will be replaced by a different menu (Figure 8a). The user will be asked to enter the integration step size, h, and the
length of simulation, \( t_{\text{final}} \). Entries should be of type real positive number greater than zero. Of course, the value of \( t_{\text{final}} \) should be greater than \( h \). However, the smaller integration step size, \( h \), generate better results.

Once everything is entered correctly, the simulation can proceed. A message 'Please WAIT!' with the word 'WAIT' blinking will be displayed on the screen. When completed, the message 'Bypass the animation ?' will replace the earlier message (Figure 8b). Answer with either "Y" or "y" for a yes response or "N" or "n" for no. A yes answer will not show the animation of the broom and cart assembly (Figure 8c). During the animation, the user can interrupt the computer by entering one of three function keys from the keyboard:

- **F1** will cause the animation to stop and pass the control to the change parameter procedure.
- **F2** will freeze or pause the animation. This will help to fully observe the animation of the cart and pendulum. Entering F2 again will continue with the animation.
- **F3** will terminate the software program and exit to DOS.

When the animation is completed, the main menu (Figure 7) will return to the screen.
SECTION 2 - Change Controller

To change the system's controller, the following menu will replace the main menu:

Controller Selections:
--- proportional
     prop. integral
     prop. derivative
     prop. integral derivative
     lead-lag

The selection of controllers can also be accomplished by moving the blinking arrow to the desired controller and hitting "carriage return." The new selected controller will appear in the controller block of the closed-loop block diagram (Figure 9). Control then passes back to the main menu.

SECTION 3 - Change Parameters

Changing the parameters is executed slightly differently than the above two options. The following message replaces the main menu and the $\phi(0)$ value is shown blinking.

first hit the SPACE BAR,
then enter the new value followed by CARRIAGE RETURN.

Again the "up" and "down" arrow keys will move the cursor. Once the position is located, enter the "space bar," which clears the old value for a new value. Enter the new parame-
ter and follow with a "carriage return." The new value will appear blinking. As many parameters as desired can be changed. To exit this selection, make the selection of 'changes COMPLETED' at the bottom of the screen (Figure 10). Control then passes to the main menu.

SECTION 4 - Plot Graphs

To plot graphs of any of the calculated variables, select the plot graphs routine from the main menu (Figure 7). The following menu will be shown.

```
Select variables:
---)time
   theta
   thetaDot
   distance
   velocity
   force
Y-axis =
```

Select a variable for the Y-axis. Do not forget the "carriage return." The selected variable will be shown as follows:
Select variables:

--- time
theta
tetaDot
distance
velocity
force

Y-axis = theta & X-axis =

Now pick a variable for the X-axis. The graphs of the selected variables will be drawn automatically. Examples of some of these graphs are given in figures 11a, 11b, and 11c. To make hard copies of the graphs, simultaneously enter the "shift" key and the "PrtSc," (print screen key). Return to the main menu by hitting "space bar." Selection to plotting procedures can be entered as many times as desired, but no graphs can be plotted without first simulating the system.

SECTION 5 - Numerical Outputs

Making hardcopy outputs is also achieved with similar ease. First, make the selection 'numerical outputs' from the main menu (Figure 7). The menu below will be shown:
Select variables:

- time
- theta
- thetaDot
- distance
- velocity
- force

Select a variable. This variable will be indicated on the right hand side of the menu under 'var. #1 =,' and the message 'Another variable?' prompted.

Select variables:

- time var. #1 = time
- theta
- thetaDot
- distance
- velocity
- force

Another variable? y

Answer either yes or no. A total of six variables can be output to the printer in any order. See the menu shown below:

Select variables:

- time var. #1 = time
- theta var. #2 = theta
- thetaDot var. #3 = distance
- distance var. #4 = force
- velocity var. #5 = thetaDot
- force

Another variable? y
If the answer is no, or after all six variables are selected, a different message will appear:

**Select variables:**

- time \( \text{var. #1 = } \text{time} \)
- \( \theta \) \( \text{var. #2 = } \theta \)
- \( \frac{d\theta}{dt} \) \( \text{var. #3 = } \text{distance} \)
- distance \( \text{var. #4 = } \text{force} \)
- velocity \( \text{var. #5 = } \frac{d\theta}{dt} \)
- force \( \text{var. #6 = } \text{velocity} \)

**Output every 25____ data points.**

The cursor waits for a positive integer greater than zero to be input on the drawn line. An entry of "1" will print every point of the selected variables on the printer. Table 1 shows an example of a computer printout with an entry of twenty-five (25), for which every twenty-five data points will be printed.

From the printer, the control again transmits back to the main menu. Like the 'plot graphs' routine, no outputs can be printed without first simulating the system.
Inverted Pendulum System

Selections:
- simulate system
- change controller
- change parameters
- plot graphs
- numerical outputs
- exit simulation

Figure 7. Main Menu
Inverted Pendulum System

Figure 8a. Simulate System Menu
(Integration Step Size)
Inverted Pendulum System

Enter the followings:
integration step, h => 0.01
length of simulation => 5
Bypass the animation? n

Figure 8b. Simulate System Menu (Bypass Animation)
Figure 8c. Animation of the Pendulum and Cart Assembly
Inverted Pendulum System

Selections:
- simulate system
- change controller
- change parameters
- plot graphs
- numerical outputs
- exit simulation

Figure 9. Change Controller Menu
Inverted Pendulum System

$\dot{\theta}_{sp} = 0.0$

Kt

$1.0$

Es

controller

(PD)

Ec

$30.0$

process

$\dot{\theta}_o = 10.0$

Xo

$X = 0$

Changes COMPLETED

L = 1.5

M = 0.2

Mc = 1.0

First hit the SPACE BAR,
then enter the new value
followed by CARRIAGE RETURN.

Figure 10. Change Parameters Menu
Figure 11a. Plot of $\theta$ Versus Time For the Lead-Lag Controller Example Run
Figure 11b. Plot of $\dot{\theta}$ Versus $\theta$ For the Lead-Lag Controller Example Run
Figure 11c. Plot of Distance Versus $\theta$ For the Lead-Lag Controller Example Run
### TABLE 1

**COMPUTER PRINTOUT FROM THE NUMERICAL OUTPUTS PROCEDURE**

**Inverted Pendulum System.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>set point angle</strong> (θSP)</td>
<td>0.000 degrees</td>
</tr>
<tr>
<td><strong>initial angle</strong> (θinit)</td>
<td>10.000 degrees</td>
</tr>
<tr>
<td><strong>initial distance</strong> (distinit)</td>
<td>0.000 meters</td>
</tr>
<tr>
<td><strong>transmitter gain</strong>, Kt</td>
<td>1.000 volts/degree</td>
</tr>
<tr>
<td><strong>controller gain</strong>, Km</td>
<td>-154.000 Newtons/degree</td>
</tr>
<tr>
<td><strong>half length of pendulum</strong>, L</td>
<td>1.500 meters</td>
</tr>
<tr>
<td><strong>mass of pendulum</strong>, M</td>
<td>0.200 kilograms</td>
</tr>
<tr>
<td><strong>mass of cart</strong>, Mc</td>
<td>1.000 kilograms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>θ</th>
<th>Distance</th>
<th>Force</th>
<th>θDot</th>
<th>Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.250</td>
<td>-1.985</td>
<td>26.207</td>
<td>-602.938</td>
<td>-52.940</td>
<td>118.415</td>
</tr>
<tr>
<td>0.500</td>
<td>-4.309</td>
<td>32.577</td>
<td>-399.877</td>
<td>26.767</td>
<td>-57.179</td>
</tr>
<tr>
<td>0.750</td>
<td>2.440</td>
<td>18.659</td>
<td>475.473</td>
<td>8.990</td>
<td>-18.755</td>
</tr>
<tr>
<td>1.000</td>
<td>0.030</td>
<td>23.975</td>
<td>-76.248</td>
<td>-17.160</td>
<td>37.957</td>
</tr>
<tr>
<td>1.250</td>
<td>-1.603</td>
<td>27.859</td>
<td>-175.149</td>
<td>5.466</td>
<td>-10.473</td>
</tr>
<tr>
<td>1.500</td>
<td>0.429</td>
<td>23.605</td>
<td>125.811</td>
<td>5.093</td>
<td>-11.119</td>
</tr>
<tr>
<td>1.750</td>
<td>0.154</td>
<td>24.094</td>
<td>9.331</td>
<td>-4.883</td>
<td>10.083</td>
</tr>
<tr>
<td>2.000</td>
<td>-0.549</td>
<td>25.519</td>
<td>-63.053</td>
<td>0.527</td>
<td>-1.526</td>
</tr>
<tr>
<td>2.250</td>
<td>-0.005</td>
<td>24.186</td>
<td>27.006</td>
<td>2.081</td>
<td>-5.390</td>
</tr>
<tr>
<td>2.500</td>
<td>0.061</td>
<td>23.777</td>
<td>12.017</td>
<td>-1.193</td>
<td>1.407</td>
</tr>
<tr>
<td>2.750</td>
<td>-0.184</td>
<td>24.010</td>
<td>-19.976</td>
<td>-0.178</td>
<td>0.0834</td>
</tr>
<tr>
<td>3.000</td>
<td>-0.056</td>
<td>23.406</td>
<td>3.962</td>
<td>0.727</td>
<td>-2.985</td>
</tr>
<tr>
<td>3.250</td>
<td>0.008</td>
<td>22.891</td>
<td>5.977</td>
<td>-0.225</td>
<td>-1.089</td>
</tr>
<tr>
<td>3.500</td>
<td>-0.065</td>
<td>22.646</td>
<td>-5.696</td>
<td>-0.139</td>
<td>-1.336</td>
</tr>
<tr>
<td>3.750</td>
<td>-0.040</td>
<td>22.173</td>
<td>-0.272</td>
<td>0.229</td>
<td>-2.220</td>
</tr>
<tr>
<td>4.000</td>
<td>-0.008</td>
<td>21.661</td>
<td>2.115</td>
<td>-0.014</td>
<td>-1.779</td>
</tr>
<tr>
<td>4.250</td>
<td>-0.026</td>
<td>21.241</td>
<td>-1.474</td>
<td>-0.058</td>
<td>-1.751</td>
</tr>
<tr>
<td>4.500</td>
<td>-0.022</td>
<td>20.764</td>
<td>-0.554</td>
<td>0.068</td>
<td>-2.046</td>
</tr>
<tr>
<td>4.750</td>
<td>-0.009</td>
<td>20.255</td>
<td>0.632</td>
<td>0.015</td>
<td>-1.977</td>
</tr>
<tr>
<td>5.000</td>
<td>-0.012</td>
<td>19.771</td>
<td>-0.351</td>
<td>-0.018</td>
<td>-1.936</td>
</tr>
</tbody>
</table>
CHAPTER V

SOFTWARE OPTIMIZATION

The main criterion of the design is to generate the graphic pictures as quickly as possible. The animation should closely represent a real time simulation of the actual inverted pendulum system. Hence, considerable effort was committed to optimize the graphics section of the software program.

The LENIPEN (Duncan-Atwell Computerized Technologies, Inc., Hillside, N.J.) graphic package was the first consideration. LENIPEN is a collection of different programs that facilitates the drawings on the graphic screen; the function, LENIMATION, produces animation on the computer by taking the already drawn screens and storing them in memory. During animation, these screens are passed to the video screen one-by-one, thus giving the effects of movement. This is analogous to the process of showing movies. The LENIPEN package proved to be ineffective because it permits a limited amount of screen storage, determined by the computer memory. The simulated section of the design program would have to generate a maximum of 700 screens, which corresponds to the space allocated for the arrays. One final note, the LENIPEN package, can only interface with BASIC programming language and not to PASCAL, which is the design
language for the software program of the inverted pendulum system.

A different attempt was to use the graphics commands of the TURBO PASCAL, which uses a simpler draw and plot routine. Movements are indicated by the drawing and redrawing of the figures. For the design system, horizontal movements are also desired. A lot of points will have to be drawn on the terminal screen to generate even a single image; thus, the speed will be reduced considerably. This attempt was dropped in favor of the TURBO GRAPHIX TOOLBOX. One advantage of using the TURBO PASCAL commands is that images can be drawn in different colors.

After the first image is drawn, not all the points are needed to create the images of either the cart or the pendulum. This observation leads to an alternative method. Since the points that described the cart remain the same, they do not have to be redrawn. This is not true for the case of the drawing of the pendulum. The points have to produce new images of the pendulum at various angles. First, both images of the cart and pendulum are drawn onto a window; then, only the altered image of the pendulum is copied onto the window. Movements are accomplished by moving this window. Two screens are used by the software to maximize speed. While displaying one screen, images are being drawn on the other. Swapping of these screens gives the motion of the animation.
The TURBO compiler as well as the TURBO-87 compiler underwent a thorough investigation. TURBO-87 uses the 8087 math processor to compute mathematical calculations and to increase the speed of execution. The results, however, did not indicate any apparent changes in the speed of execution. This is obviously the case, since the mathematical compilation used in the software is rather short and simple.

A different usage of the mathematical algorithm was investigated. The first method was to use the calculated data to directly simulate the graphic images. The second method stored the calculated points in some data files and then drew the pictures. One can see the advantages and the disadvantages of both methods. Obviously, the second method would require a tremendous amount of memory space — this is negotiable. The disadvantage of the first method is that, the same calculated data points are used by the plotting procedures. The second alternative was chosen for this software program.
CHAPTER VI

RESULTS

The comparison of the chosen Runge-Kutta numerical method with the exact solution is a special point of interest. To analyze this comparison different and smaller programs in PASCAL and ACSL (Advanced Continuous Simulation Language; Mitchell and Gauthier Associates, Concord, Ma.) were written. ACSL is used for the purpose of modelling systems described by time-dependent, linear or non-linear differential equations, and/or transfer functions. Listings of these program are found in Appendix 1.

Table 2 compares the Runge-Kutta numerical method output with the exact solution given in Chapter III for the Proportional controller. The parameters used to simulate the system also appear in Table 2. By comparing the theta and thetaE columns, it can be concluded that the Runge-Kutta method is extremely accurate. Figures 12a, 12b, and 12c are the plots obtained from the software program of the inverted pendulum system. These graphs show the response of the Proportional controller for different controller gain settings (i.e., Km*Kt), at values of 30.0, 11.772, and 5.0 Newtons/volt, respectively. The initial value of \( \Theta \), \( \Theta(0) \) was set at 10.0 degrees, and the set point angle, \( \Theta_{sp} \), at 0.0 degree. The results obtained in figures 12a, 12b, and
12c compliment those presented for the response of the Proportional controller (Figure 4). These are the responses of the controller for the three different cases with reference to the critical gain, $K_{cr}$, which for the given design parameters, is calculated to be 11.772 Newtons/volt.

A similar analysis was also conducted for the Proportional plus Derivative controller (PD). Again the results are favorable as indicated in Table 3 -- observe the outputs of theta and thetaE. Figures 13a and 13b are plots from the software program using the Proportional plus Derivative controller. These plots are also graphs of different gain settings. Figure 13b uses a relatively large controller gain. Figures 13a and 13b support the curves shown in Figures 5a and 5b.

One other test was conducted, this time, for the Lead-Lag controller. Since no exact analytical solution could be found, the Runge-Kutta result was compared with the ACSL model. For comparison purposes, both programs were run using the same parameters and initial conditions. Tables 4 and 5 are the results of the Runge-Kutta method and the ACSL program, respectively. Observe the column of theta (Table 4) and the column THE (Table 5). The outcome promotes the Runge-Kutta technique.
TABLE 2
RUNGE-KUTTA VERSUS EXACT SOLUTION
FOR THE PROPORTIONAL CONTROLLER

Are the following correct ???

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value (θ)</th>
<th>Value (θE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial θ value</td>
<td>θinit</td>
<td>10.00</td>
</tr>
<tr>
<td>theta set point value</td>
<td>θSp</td>
<td>0.00</td>
</tr>
<tr>
<td>initial dist. x value</td>
<td>distXinit</td>
<td>0.00</td>
</tr>
<tr>
<td>integration step size</td>
<td>h</td>
<td>0.01</td>
</tr>
<tr>
<td>final time value</td>
<td>tfinal</td>
<td>0.30</td>
</tr>
<tr>
<td>gain of controller</td>
<td>Km</td>
<td>10.00</td>
</tr>
<tr>
<td>gain</td>
<td>Kt</td>
<td>10.00</td>
</tr>
<tr>
<td>mass of cart</td>
<td>Mc</td>
<td>1.00</td>
</tr>
<tr>
<td>mass of pendulum</td>
<td>M</td>
<td>0.20</td>
</tr>
<tr>
<td>length of pendulum</td>
<td>L</td>
<td>1.50</td>
</tr>
</tbody>
</table>

......enter Y<es or N<o =>y

<table>
<thead>
<tr>
<th>t</th>
<th>θ</th>
<th>θE</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>0.00</td>
<td>10.0000</td>
</tr>
<tr>
<td>t</td>
<td>0.01</td>
<td>9.9790</td>
</tr>
<tr>
<td>t</td>
<td>0.02</td>
<td>9.9160</td>
</tr>
<tr>
<td>t</td>
<td>0.03</td>
<td>9.8115</td>
</tr>
<tr>
<td>t</td>
<td>0.04</td>
<td>9.6658</td>
</tr>
<tr>
<td>t</td>
<td>0.05</td>
<td>9.4794</td>
</tr>
<tr>
<td>t</td>
<td>0.06</td>
<td>9.2532</td>
</tr>
<tr>
<td>t</td>
<td>0.07</td>
<td>8.9882</td>
</tr>
<tr>
<td>t</td>
<td>0.08</td>
<td>8.6854</td>
</tr>
<tr>
<td>t</td>
<td>0.09</td>
<td>8.3461</td>
</tr>
<tr>
<td>t</td>
<td>0.10</td>
<td>7.9719</td>
</tr>
<tr>
<td>t</td>
<td>0.11</td>
<td>7.5640</td>
</tr>
<tr>
<td>t</td>
<td>0.12</td>
<td>7.1245</td>
</tr>
<tr>
<td>t</td>
<td>0.13</td>
<td>6.6550</td>
</tr>
<tr>
<td>t</td>
<td>0.14</td>
<td>6.1576</td>
</tr>
<tr>
<td>t</td>
<td>0.15</td>
<td>5.6343</td>
</tr>
<tr>
<td>t</td>
<td>0.16</td>
<td>5.0873</td>
</tr>
<tr>
<td>t</td>
<td>0.17</td>
<td>4.5190</td>
</tr>
<tr>
<td>t</td>
<td>0.18</td>
<td>3.9317</td>
</tr>
<tr>
<td>t</td>
<td>0.19</td>
<td>3.3290</td>
</tr>
<tr>
<td>t</td>
<td>0.20</td>
<td>2.7101</td>
</tr>
<tr>
<td>t</td>
<td>0.21</td>
<td>2.0809</td>
</tr>
<tr>
<td>t</td>
<td>0.22</td>
<td>1.4430</td>
</tr>
<tr>
<td>t</td>
<td>0.23</td>
<td>0.7990</td>
</tr>
<tr>
<td>t</td>
<td>0.24</td>
<td>0.1517</td>
</tr>
<tr>
<td>t</td>
<td>0.25</td>
<td>-0.4963</td>
</tr>
<tr>
<td>t</td>
<td>0.26</td>
<td>-1.1421</td>
</tr>
<tr>
<td>t</td>
<td>0.27</td>
<td>-1.7832</td>
</tr>
</tbody>
</table>
Figure 12a. Proportional Controller Response
For Gain $K_m*K_t=5.0$ (i.e., $< K_{cr}$)
Figure 12b. Proportional Controller Response
For Gain $K_mK_t=11.772$ (i.e., $= K_cr$)
Figure 12c. Proportional Controller Response For Gain \( \text{Km*Kt}=30.0 \) (i.e., > Kcr)
### TABLE 3

**RUNGE-KUTTA VERSUS EXACT SOLUTION FOR THE PROPORTIONAL PLUS DERIVATIVE CONTROLLER**

Are the following correct? 

- initial theta value, \( \theta_{\text{init}} = 10.00 \)
- theta set point value, \( \theta_{\text{Sp}} = 0.00 \)
- initial dist. x value, \( \text{distX}_{\text{init}} = 0.00 \)
- integration step size, \( h = 0.01 \)
- final time value, \( t_{\text{final}} = 0.22 \)
- gain of controller, \( K_m = 10.00 \)
- gain, \( K_t = 10.00 \)
- mass of cart, \( M_c = 1.00 \)
- mass of pendulum, \( M = 0.20 \)
- length of pendulum, \( L = 1.50 \)

......enter Y<es or N<o

\[
\begin{align*}
  t & = 0.00 \quad \theta = 10.00000 \quad \theta_{E} = 10.00000 \\
  t & = 0.01 \quad \theta = 9.97933 \quad \theta_{E} = 9.97933 \\
  t & = 0.02 \quad \theta = 9.91869 \quad \theta_{E} = 9.91869 \\
  t & = 0.03 \quad \theta = 9.82019 \quad \theta_{E} = 9.82019 \\
  t & = 0.04 \quad \theta = 9.68600 \quad \theta_{E} = 9.68600 \\
  t & = 0.05 \quad \theta = 9.51832 \quad \theta_{E} = 9.51832 \\
  t & = 0.06 \quad \theta = 9.31939 \quad \theta_{E} = 9.31939 \\
  t & = 0.07 \quad \theta = 9.09150 \quad \theta_{E} = 9.09150 \\
  t & = 0.08 \quad \theta = 8.83691 \quad \theta_{E} = 8.83691 \\
  t & = 0.09 \quad \theta = 8.55792 \quad \theta_{E} = 8.55792 \\
  t & = 0.10 \quad \theta = 8.25681 \quad \theta_{E} = 8.25681 \\
  t & = 0.11 \quad \theta = 7.93583 \quad \theta_{E} = 7.93583 \\
  t & = 0.12 \quad \theta = 7.59723 \quad \theta_{E} = 7.59723 \\
  t & = 0.13 \quad \theta = 7.24321 \quad \theta_{E} = 7.24321 \\
  t & = 0.14 \quad \theta = 6.87596 \quad \theta_{E} = 6.87596 \\
  t & = 0.15 \quad \theta = 6.49757 \quad \theta_{E} = 6.49757 \\
  t & = 0.16 \quad \theta = 6.11014 \quad \theta_{E} = 6.11014 \\
  t & = 0.17 \quad \theta = 5.71566 \quad \theta_{E} = 5.71566 \\
  t & = 0.18 \quad \theta = 5.31609 \quad \theta_{E} = 5.31609 \\
  t & = 0.19 \quad \theta = 4.91329 \quad \theta_{E} = 4.91329 \\
  t & = 0.20 \quad \theta = 4.50907 \quad \theta_{E} = 4.50907 \\
  t & = 0.21 \quad \theta = 4.10515 \quad \theta_{E} = 4.10515
\]
Figure 13a. Proportional Plus Derivative Controller Response For Gains, $K_m=10.0$ and $K_t=10.0$
Figure 13b. Proportional Plus Derivative Controller Response For Gains, $K_m=25.0$ and $K_t=1.0$ (High Gain)
TABLE 4

RUNGE-KUTTA OUTPUT OF THE LEAD-LAG CONTROLLER

Are the following correct ???
initial theta value, \( \text{thetainit} = 10.00 \)
theta set point value, \( \text{thetaSp} = 0.00 \)
initial dist. x value, \( \text{distXinit} = 0.00 \)
integration step size, \( h = 0.00 \)
final time value, \( t_{\text{final}} = 0.20 \)
gain of controller, \( K_m = -154.00 \)
gain, \( K_t = 1.00 \)
mass of cart, \( M_c = 1.00 \)
mass of pendulum, \( M = 0.20 \)
length of pendulum, \( L = 1.50 \)
parameter of leadlag, \( \text{SigA} = 1.00 \)
parameter of leadlag, \( \text{SigB} = 4.00 \)

......enter Y<es or N<o => y

t = 0.00 theta = 10.00000
 t = 0.00 theta = 9.99154
 t = 0.01 theta = 9.96629
 t = 0.01 theta = 9.92458
 t = 0.02 theta = 9.86673
 t = 0.02 theta = 9.79311
 t = 0.03 theta = 9.70408
 t = 0.03 theta = 9.60003
 t = 0.04 theta = 9.48138
 t = 0.04 theta = 9.34856
 t = 0.05 theta = 9.20201
 t = 0.05 theta = 9.04219
 t = 0.06 theta = 8.86957
 t = 0.06 theta = 8.68463
 t = 0.07 theta = 8.48789
 t = 0.07 theta = 8.27984
 t = 0.08 theta = 8.06100
 t = 0.08 theta = 7.83191
 t = 0.09 theta = 7.59311
 t = 0.09 theta = 7.34512
 t = 0.10 theta = 7.08852
 t = 0.10 theta = 6.82385
 t = 0.11 theta = 6.55166
 t = 0.11 theta = 6.27254
 t = 0.12 theta = 5.98703
TABLE 5
ACSL OUTPUT OF THE LEAD–LAG CONTROLLER

```
type broom1.out
IACSL RUN-TIME EXEC PC VERSION LEVEL BF7 86/12/04 15:47:51 PAGE 1

PREF 'ALL'
OUTPUT T,THE,F
S TMX=0.12,KM=-154,CINT=0.005
D THEIC,MAX,TMX,KM,KT,CINT
THEIC 10.0000000 MAX 0.000000000000
KM-154.0000000 KT 1.000000000000 CINT 0.005000000000

START
T O.00000000
T 0.00500000 THE 9.99158000 F 1540.000000
T 0.01000000 THE 9.96652000 F 1515.840000
T 0.01500000 THE 9.92512000 F 1469.610000
T 0.02000000 THE 9.86772000 F 1431.230000
T 0.02500000 THE 9.79466000 F 1399.240000
T 0.03000000 THE 9.70632000 F 1365.500000
T 0.03500000 THE 9.60397000 F 1330.080000
T 0.04000000 THE 9.48537000 F 1293.060000
T 0.04500000 THE 9.35588000 F 1254.540000
T 0.05000000 THE 9.20818000 F 1214.600000
T 0.05500000 THE 9.04460000 F 1173.330000
T 0.06000000 THE 8.87833000 F 1130.820000
T 0.06500000 THE 8.69484000 F 1087.140000
T 0.07000000 THE 8.49962000 F 1042.400000
T 0.07500000 THE 8.29319000 F 996.686000
T 0.08000000 THE 8.07606000 F 950.082000
T 0.08500000 THE 7.84878000 F 902.679000
T 0.09000000 THE 7.61179000 F 854.567000
T 0.09500000 THE 7.36573000 F 805.835000
T 0.10000000 THE 7.11111000 F 756.572000
T 0.10500000 THE 6.84847000 F 706.867000
T 0.11000000 THE 6.57837000 F 656.805000
T 0.11500000 THE 6.30140000 F 606.476000
T 0.12000000 THE 6.01808000 F 555.963000
T 0.12500000 THE 5.72898000 F 505.352000

STOP
C:\MODELS>
```
CHAPTER VII

SUMMARY AND CONCLUSIONS

The system, which has been the topic of this research report, has demonstrated some of the graphic capabilities of common desktop computers to produce animations. The TURBO PASCAL and TURBO GRAPHIX TOOLBOX features were used extensively to generate animations for this system. This technique is just one way of creating moving figures. The speed with which drawings are made is a determining factor in animation; by using assembly language, speed can be increased. Besides speed, the software package should also direct the user through the various tasks. User-friendliness is another important criterion.

The software program described in this report is highly interactive. All menus are displayed directly on the video terminal. Every action is accomplished by a single key entry from the keyboard followed by a "carriage return."

The inverted pendulum system discussed in this paper serves as an excellent teaching aid in the demonstration of linear control theory. This study is by no means complete. Even though five controllers were included, other types of controllers should be explored. Different algorithms to drive the inverted pendulum system should also be attempted. One might experiment with non-linear controllers. Instead
of using a constant set point, some time varying function can be used.
APPENDIX 1

PASCAL LISTINGS OF CONTROLLERS

AND ACSL MODEL LISTINGS
program GetThetaX;  {Proportional}

const
  g = 9.81;
var
  force,h,t,Mc,M,L : real;
  thetainit,distXinit : real;
  thetaExact,tfinal,Kt,thetaSP,Km : real;
  Z,K1,K2,K3,K4 : array [1..4] of real;
  mul1,mul2,mul3,mul4,mul5,mul6 : real;

procedure initvalue; forward;

procedure display;
var  Ch : char;
begin
  repeat
   ClrScr;
    writeln('Are the following correct ???
    writeln(' initial theta value , thetainit = ' ,thetainit:8:2);
    writeln(' theta set point value, thetaSP = ' ,thetaSP:8:2);
    writeln(' initial dist. x value, distXinit = ' ,distXinit:8:2);
    writeln(' integration step size, h = ' ,h:8:2);
    writeln(' final time value , tfinal = ' ,tfinal:8:2);
    writeln(' gain of controller , Km = ' ,Km:8:2);
    writeln(' gain , Kt = ' ,Kt:8:2);
    writeln(' mass of cart , Mc = ' ,Mc:8:2);
    writeln(' mass of pendulum , M = ' ,M:8:2);
    writeln(' length of pendulum , L = ' ,L:8:2);
    writeln;
    writeln;
    write ('......enter Y<es or N<o ' , ==>');
    readln(Ch);
    until(Ch=#110)or(Ch=#78)or(Ch=#121)or(Ch=#90);
    if (Ch=#110) or (Ch=#78) then initvalue;
  end;  {display}
procedure initvalue;
begin
  writeln('Enter the following :');
  write(' initial theta value, thetainit ==>');
  readln(thetainit);
  write(' theta set point value, thetaSp ==>');
  readln(thetaSP);
  write(' initial dist. x value, distXinit ==>');
  readln(distXinit);
  write(' integration step size, h ==>');
  readln(h);
  write(' final time value, tfinal ==>');
  readln(tfinal);
  write(' gain of controller, Km ==>');
  readln(Km);
  write(' gain, Kt ==>');
  readln(Kt);
  write(' mass of cart, Mc ==>');
  readln(Mc);
  write(' mass of pendulum, M ==>');
  readln(M);
  write(' length of pendulum, L ==>');
  readln(L);
  Mc := 1.0;
  M := 0.2;
  L := 1.5;
  t := 0.0;
  mul3 := (3.0*(Mc+M)*g)/(L*(4.0*Mc+M));
  mul4 := -3.0/(L*(4.0*Mc+M));
  mul5 := -(3.0*M*g)/(4.0*Mc+M);
  mul6 := 4.0/(4.0*Mc+M);
  Z[1] := thetainit;
  Z[2] := 0.0;
  Z[3] := distXinit;
  Z[4] := 0.0;
  theta Exact := Z[1];
  display;
  delay(1000);
  ClrScr;
end; (initvalue)

procedure nextZ;
const  mul1 = 2.0;
       mul2 = 2.0;
var  n : integer;
begin
  for n := 1 to 4 do
  begin
      *K3[n]+K4[n]);
  end;
end; (nextZ)
procedure RungaKutta;
var  suml, sum2, sum3, sum4 : real;
begin {RungaKutta}
  force := -Km*Kt*(thetaSP-Z[1]);
  K1[1] := Z[2];
  K1[3] := Z[4];
  suml := Z[1] + 0.5*h*K1[1];
  sum2 := Z[2] + 0.5*h*K1[2];
  sum3 := Z[3] + 0.5*h*K1[3];
  sum4 := Z[4] + 0.5*h*K1[4];
  force := -Km*Kt*(thetaSP-suml);
  K2[1] := sum2;
  K2[4] := mul5*suml + mul6*force;
  sum1 := Z[1] + 0.5*h*K2[1];
  sum2 := Z[2] + 0.5*h*K2[2];
  sum3 := Z[3] + 0.5*h*K2[3];
  sum4 := Z[4] + 0.5*h*K2[4];
  force := -Km*Kt*(thetaSP-sum1);
  K3[1] := sum2;
  K3[2] := mul3*sum1 + mul4*force;
  K3[4] := mul5*sum1 + mul6*force;
  sum1 := Z[1] + h*K3[1];
  sum2 := Z[2] + h*K3[2];
  sum3 := Z[3] + h*K3[3];
  sum4 := Z[4] + h*K3[4];
  force := -Km*Kt*(thetaSP-sum1);
  K4[1] := sum2;
  K4[2] := mul3*sum1 + mul4*force;
  K4[4] := mul5*sum1 + mul6*force;
  nextZ;
end; {RungaKutta}

procedure exactvalue;
var  omegasqr, omega : real;
begin {exactvalue}
  omegasqr := -Km*Kt*mul4 - mul3;
  omega := sqrt(omegasqr);
  thetaExact := thetainit*cos(omega*t);
end; {exactvalue}
begin (GetThetaX)
    initvalue;
    repeat
        exactvalue;
        writeln(Lst,'t = ',t:6:2,' theta = ',Z[1]:8:5
              ', thetaE = ',thetaExact:8:5);
        RungaKutta;
        t := t + h;
        until t>=tfinal;
end.  (GetThetaX)
program GetThetaX; {proportional integral}
const
g = 9.81;

h,t,Mc,M,L : real;
thetaSP,Kt,thetaSP,Km : real;
Z,K1,K2,K3,K4 : array [1..4] of real;
mul3,mul4,mul5,mul6 : real;
area,force : real;

var

thetainit,distXinit : real;
tfinal,tfinal,thetaSP,Km : real;

procedure initvalue; forward;

procedure display;
var Ch : char;
begin
repeat
ClrScr;
writeln('Are the following correct ???');
writeln(' initial theta value , thetainit = ' ,thetainit:8:2);
writeln(' theta set point value, thetaSP = ' ,thetaSP:8:2);
writeln(' initial dist. x value, distXinit = ' ,distXinit:8:2);
writeln(' integration step size, h = ' ,h:8:2);
writeln(' final time value , tfinal = ' ,tfinal:8:2);
writeln(' gain of controller , Km = ' ,Km:8:2);
writeln(' gain , Kt = ' ,Kt:8:2);
writeln(' mass of cart , Mc = ' ,Mc:8:2);
writeln(' mass of pendulum , M = ' ,M:8:2);
writeln(' length of pendulum , L = ' ,L:8:2);
writeln;
write ('......enter Y<es or N<o ' ,etc:8:2);
readln(Ch);
until (Ch=#110)or(Ch=#78)or(Ch=#121)or(Ch=#90);
if (Ch=#110) or (Ch=#78) then initvalue;
end; {display}
procedure initvalue;
begin
  writeln('Enter the following :');
  write(' initial theta value, thetainit ==>');
  readln(thetainit);
  write(' theta set point value, thetaSP ==>');
  readln(thetaSP);
  write(' initial dist. x value, distXinit ==>');
  readln(distXinit);
  write(' integration step size, h ==>');
  readln(h);
  write(' final time value, tfinal ==>');
  readln(tfinal);
  write(' gain of controller, Km ==>');
  readln(Km);
  write(' gain, Kt ==>');
  readln(Kt);
  { write(' mass of cart, Mc ==>');
    readln(Mc);
    write(' mass of pendulum, M ==>');
    readln(M);
    write(' length of pendulum, L ==>');
    readln(L);
  }
  Mc := 1.0;
  M := 0.2;
  L := 1.5;
  t := 0.0;
  mul3 := (3.0*(Mc+M)*g)/(L*(4.0*Mc+M));
  mul4 := -3.0/(L*(4.0*Mc+M));
  mul5 := -(3.0*M*g)/(4.0*Mc+M);
  mul6 := 4.0/(4.0*Mc+M);
  Z[1] := thetainit;
  Z[2] := 0.0;
  Z[3] := distXinit;
  Z[4] := 0.0;
  area := 0.0;
  display;
  delay(1000);
  ClrScr;
end; {initvalue}

procedure nextZ;
const   mul1 = 2.0;
        mul2 = 2.0;
var    n : integer;
begin
  for n := 1 to 4 do
    begin
                               *K3[n] + K4[n]);
    end;
end; {nextZ}
procedure RungaKutta;
var sum1, sum2, sum3, sum4 : real;
begin {RungaKutta}
area := area + h*(Z[1]);
force := -Km*(Kt*(thetaSP-Z[1]) - area);
K1[1] := Z[2];
K1[3] := Z[4];
sum1 := Z[1] + 0.5*h*K1[1];
sum2 := Z[2] + 0.5*h*K1[2];
sum3 := Z[3] + 0.5*h*K1[3];
sum4 := Z[4] + 0.5*h*K1[4];
force := -Km*(Kt*(thetaSP-sum1) - area);
K2[1] := sum2;
K2[2] := mul3*sum1 + mul4*force;
K2[4] := mul5*sum1 + mul6*force;
sum1 := Z[1] + 0.5*h*K2[1];
sum2 := Z[2] + 0.5*h*K2[2];
sum3 := Z[3] + 0.5*h*K2[3];
sum4 := Z[4] + 0.5*h*K2[4];
force := -Km*(Kt*(thetaSP-sum1) - area);
K3[1] := sum2;
K3[2] := mul3*sum1 + mul4*force;
K3[4] := mul5*sum1 + mul6*force;
sum1 := Z[1] + h*K3[1];
sum2 := Z[2] + h*K3[2];
sum3 := Z[3] + h*K3[3];
sum4 := Z[4] + h*K3[4];
force := -Km*(Kt*(thetaSP-sum1) - area);
K4[1] := sum2;
K4[2] := mul3*sum1 + mul4*force;
K4[4] := mul5*sum1 + mul6*force;
nextZ;
end; {RungaKutta}
begin {GetThetaX}
initvalue;
repeat
writeln('t = ',t:6:2,' theta = ',Z[1]:8:5
',,' area = ',area:10:4);
RungaKutta;
t := t + h;
until t>=tfinal;
end. {GetThetaX}
program GetThetaX; {Proportional Derivative}

const 
g = 9.81;
var force,h,t,Mc,M,L : real;
thetaInit,dist INIT : real;
thetaExact,tfinal,Kt,thetaSP,Km : real;
Z,K1,K2,K3,K4 : array [1..4] of real;
mul1,mul2,mul3,mul4,mul5,mul6 : real;

procedure initvalue; forward;

procedure display;
var Ch : char;
begin
repeat
ClrScr;
write('Are the following correct ???');
writeln(' initial theta value , thetainit = ',thetainit:8:2);
writeln(' theta set point value, thetaSP = ',thetaSP:8:2);
writeln(' initial dist. x value, distXinit = ',distXinit:8:2);
writeln(' integration step size, h = ',h:8:2);
writeln(' final time value , tfinal = ',tfinal:8:2);
writeln(' gain of controller , Km = ',Km:8:2);
writeln(' gain , Kt = ',Kt:8:2);
writeln(' mass of cart , Mc = ',Mc:8:2);
writeln(' mass of pendulum , M = ',M:8:2);
writeln(' length of pendulum , L = ',L:8:2);
write; writeln;
write ('......enter Y<es or N<o ==>' );
readln(Ch);
until(Ch=#110)or(Ch=#78)or(Ch=#121)or(Ch=#90);
if (Ch=#110) then initvalue;
end; {display}
procedure initvalue;
begin
    writeln('Enter the following :');
    write(' initial theta value, thetainit  ==>');
    readln(thetainit);
    write(' theta set point value, thetaSp   ==>');
    readln(thetaSp);
    write(' initial dist. x value, distXinit ==>');
    readln(distXinit);
    write(' integration step size, h      ==>');
    readln(h);
    write(' final time value, tfinal    ==>');
    readln(tfinal);
    write(' gain of controller, Km     ==>');
    readln(Km);
    write(' gain, Kt                ==>');
    readln(Kt);
    write(' mass of cart, Mc           ==>');
    readln(Mc);
    write(' mass of pendulum, M        ==>');
    readln(M);
    write(' length of pendulum, L      ==>');
    readln(L);
    Mc := 1.0;
    M := 0.2;
    L := 1.5;
    t := 0.0;
    mul3 := (3.0*(Mc+M)*g)/(L*(4.0*Mc+M));
    mul4 := -3.0/(L*(4.0*Mc+M));
    mul5 := -(3.0*M*g)/(4.0*Mc+M);
    mul6 := 4.0/(4.0*Mc+M);
    Z[1] := thetainit;
    Z[2] := 0.0;
    Z[3] := distXinit;
    Z[4] := 0.0;
    thetaExact := Z[1];
    display;
    delay(1000);
    ClrScr;
end; {initvalue}

procedure nextZ;
const  mull = 2.0;
       mul2 = 2.0;
var      n : integer;
begin  for n := 1 to 4 do begin
    Z[n] := Z[n+(h/6.0)*(K1[n]+mul1*K2[n]+mul2
                 *K3[n]+K4[n])];
end; {nextZ}
procedure RungaKutta;
var sum1, sum2, sum3, sum4 : real;
begin {RungaKutta}
  force := Km*(Z[2] - Kt*(thetaSP-Z[1]));
  K1[1] := Z[2];
  K1[3] := Z[4];
  sum1 := Z[1] + 0.5*h*K1[1];
  sum2 := Z[2] + 0.5*h*K1[2];
  sum3 := Z[3] + 0.5*h*K1[3];
  sum4 := Z[4] + 0.5*h*K1[4];
  force := Km*(sum2 - Kt*(thetaSP-sum1));
  K2[1] := sum2;
  k2[2] := mul3*sum1 + mul4*force;
  K2[4] := mul5*sum1 + mul6*force;
  sum1 := Z[1] + 0.5*h*K2[1];
  sum2 := Z[2] + 0.5*h*K2[2];
  sum3 := Z[3] + 0.5*h*K2[3];
  sum4 := Z[4] + 0.5*h*K2[4];
  force := Km*(sum2 - Kt*(thetaSP-sum1));
  K3[1] := sum2;
  k3[2] := mul3*sum1 + mul4*force;
  K3[4] := mul5*sum1 + mul6*force;
  sum1 := Z[1] + h*K3[1];
  sum2 := Z[2] + h*K3[2];
  sum3 := Z[3] + h*K3[3];
  sum4 := Z[4] + h*K3[4];
  force := Km*(sum2 - Kt*(thetaSP-sum1));
  K4[1] := sum2;
  k4[2] := mul3*sum1 + mul4*force;
  K4[4] := mul5*sum1 + mul6*force;
nextZ;
end; {RungaKutta}

procedure exactvalue;
var alpha, omegasqr, omega, square, temsin : real;
begin {exactvalue}
  alpha := -0.5*mul4*Km;
  omegasqr := -Km*Kt*mul4 - mul3;
  omega := sqrt(omegasqr);
  square := sqrt(omegasqr - alpha*alpha);
  temsin := sin(square*t + arctan(square/alpha));
  thetaExact := (temsin*thetainit*omega*Exp(-alpha*t)) /square;
end; {exactvalue}
begin {GetThetaX}
  initvalue;
  repeat
    exactvalue;
    writeln(Lst,'t = ',t:6:2,' theta = ',Z[1]:8:5,' thetaE = ',thetaExact:8:5);
    RungaKutta;
    t := t + h;
    until t>=tfinal;
  end. {GetThetaX}
program GetThetaX;
{proportional integral derivative}
g = 9.81;
h,t,Mc,M,L : real;
thetainit,distXinit : real;
tfinal,Kt,thetaSP,Km : real;
Z,K1,K2,K3,K4 : array [1..4] of real;
mul3,mul4,mul5,mul6 : real;
area,force : real;

procedure initvalue; forward;

procedure display;
var Ch : char;
begin
repeat
  ClrScr;
  writeln('Are the following correct ???
');
  writeln(' initial theta value , thetainit = ' ,thetainit:8:2);
  writeln(' theta set point value, thetaSp = ' ,thetaSP:8:2);
  writeln(' initial dist. x value, distXinit = ' ,distXinit:8:2);
  writeln(' integration step size, h = ' ,h:8:2);
  writeln(' final time value , tfinal = ' ,tfinal:8:2);
  writeln(' gain of controller , Km = ' ,Km:8:2);
  writeln(' gain , Kt = ' ,Kt:8:2);
  writeln(' mass of cart , Mc = ' ,Mc:8:2);
  writeln(' mass of pendulum , M = ' ,M:8:2);
  writeln(' length of pendulum , L = ' ,L:8:2);
  writeln;
  writeln;
  write ('......enter Y<es or N<o
');
  readln(Ch);
  until(Ch=#110)or(Ch=#78)or(Ch=#121)or(Ch=#90);
  if (Ch=#110) or (Ch=#78) then initvalue;
end; {display}
procedure initvalue;
begin
  writeln('Enter the following :');
  write(' initial theta value, thetainit  ==>');
  readln(thetainit);
  write(' theta set point value, thetaSp  ==>');
  readln(thetaSP);
  write(' initial dist. x value, distXinit ==>');
  readln(distXinit);
  write(' integration step size, h ==>');
  readln(h);
  write(' final time value, tfinal ==>');
  readln(tfinal);
  write(' gain of controller, Km ==>');
  readln(Km);
  write(' gain, Kt ==>');
  readln(Kt);
  { write(' mass of cart, Mc ==>');
    readln(Mc);
    write(' mass of pendulum, M ==>');
    readln(M);
    write(' length of pendulum, L ==>');
    readln(L); }
  Mc := 1.0;
  M := 0.2;
  L := 1.5;
  t := 0.0;
  mul3 := (3.0*(Mc+M)*g)/(L*(4.0*Mc+M));
  mul4 := -3.0/(L*(4.0*Mc+M));
  mul5 := -(3.0*M*g)/(4.0*Mc+M);
  mul6 := 4.0/(4.0*Mc+M);
  Z[1] := thetainit;
  Z[2] := 0.0;
  Z[3] := distXinit;
  Z[4] := 0.0;
  area := 0.0;
  display;
  delay(1000);
  ClrScr;
end; {initvalue}

procedure nextZ;
const  mul1 = 2.0;
       mul2 = 2.0;
var    n : integer;
begin
  for n := 1 to 4 do
  begin
    Z[n]:= Z[n]+(h/6.0)*(K1[n]+mul1*K2[n]+mul2
    *K3[n]+K4[n]);
  end;
end; {nextZ}
procedure RungaKutta;
var sum1, sum2, sum3, sum4 : real;
begin {RungaKutta}
area := area + h*(Z[1]);
force := -Km*(-Z[2] + Kt*(thetaSP-Z[1]) - area);
K1[1] := Z[2];
K1[3] := Z[4];
sum1 := Z[1] + 0.5*h*K1[1];
sum2 := Z[2] + 0.5*h*K1[2];
sum3 := Z[3] + 0.5*h*K1[3];
sum4 := Z[4] + 0.5*h*K1[4];
force := -Km*(-sum2 + Kt*(thetaSP-sum1) - area);
K2[1] := sum2;
K2[2] := mul3*sum1 + mul4*force;
K2[4] := mul5*sum1 + mul6*force;
sum1 := Z[1] + 0.5*h*K2[1];
sum2 := Z[2] + 0.5*h*K2[2];
sum3 := Z[3] + 0.5*h*K2[3];
sum4 := Z[4] + 0.5*h*K2[4];
force := -Km*(-sum2 + Kt*(thetaSP-sum1) - area);
K3[1] := sum2;
K3[2] := mul3*sum1 + mul4*force;
K3[4] := mul5*sum1 + mul6*force;
sum1 := Z[1] + h*K3[1];
sum2 := Z[2] + h*K3[2];
sum3 := Z[3] + h*K3[3];
sum4 := Z[4] + h*K3[4];
force := -Km*(-sum2 + Kt*(thetaSP-sum1) - area);
K4[1] := sum2;
K4[2] := mul3*sum1 + mul4*force;
K4[4] := mul5*sum1 + mul6*force;
extZ;
end; {RungaKutta}

begin {GetThetaX}
initvalue;
repeat
writeln('t = ',t:6:2,' theta = ',Z[1]:8:5
area = ',area:10:4);
RungaKutta;
t := t + h;
until t>=tfinal;
end. {GetThetaX}
program GetThetaX; {leadlag}
const
g  = 9.81;
var
h, t, Mc, M, L, SigA, SigB : real;
thetaInit, distXInit : real;
tFinal, Kt, thetaSP, Km : real;
Z, K1, K2, K3, K4 : array [1..5] of real;
mul3, mul4, mul5, mul6 : real;
force : real;

procedure initvalue; forward;
procedure display;
var Ch : char;
begin
repeat
  ClrScr;
  writeln('Are the following correct ???');
  writeln(' initial theta value, thetaInit = ');
  writeln(' initial dist. x value, distXInit = ');
  writeln(' integration step size, h = ');
  writeln(' final time value, tFinal = ');
  writeln(' gain of controller, Km = ');
  writeln(' gain, Kt = ');
  writeln(' mass of cart, Mc = ');
  writeln(' mass of pendulum, M = ');
  writeln(' length of pendulum, L = ');
  writeln(' parameter of ledlag, SigA = ');
  writeln(' parameter of ledlag, SigB = ');
  writeln;
  writeln;
  write ('...... enter Y<es or N<o ==>');
  readln(Ch);
  until (Ch='Y') or (Ch='N') or (Ch='y') or (Ch='n');
  if (Ch='Y') or (Ch='y') then initvalue;
until (Ch='Y') or (Ch='y') or (Ch='N') or (Ch='n');
end; {display}
procedure initvalue;
begint
  writeln('Enter the following :');
  write(' initial theta value, thetainit ==>');
  readln(thetainit);
  write(' theta set point value, thetaSP ==>');
  readln(thetaSP);
  write(' initial dist. x value, distXinit ==>');
  readln(distXinit);
  write(' integration step size, h ==>');
  readln(h);
  write(' final time value, tfinal ==>');
  readln(tfinal);
  write(' gain of controller, Km ==>');
  readln(Km);
  write(' gain, Kt ==>');
  readln(Kt);
  { write(' mass of cart, Mc ==>');
    readln(Mc);
    write(' mass of pendulum, M ==>');
    readln(M);
    write(' length of pendulum, L ==>');
    readln(L); }
  Mc := 1.0;
  M := 0.2;
  L := 1.5;
  SigA := 1.0;
  SigB := 4.0;
  t := 0.0;
  mul3 := (3.0*(Mc+M)*g)/(L*(4.0*Mc+M));
  mul4 := -3.0/(L*(4.0*Mc+M));
  mul5 := -(3.0*M*g)/(4.0*Mc+M);
  mul6 := 4.0/(4.0*Mc+M);
  Z[1] := thetainit;
  Z[2] := 0.0;
  Z[3] := distXinit;
  Z[4] := 0.0;
  Z[5] := 0.0;
  display;
  delay(1000);
  ClrScr;
end; {initvalue}
procedure nextZ;
const  mul1 = 2.0;
       mul2 = 2.0;
var    n : integer;
begin
  for n := 1 to 4 do
  begin
                       *K3[n]+K4[n]);
  end;
end; {nextZ}

procedure RKforce(var ZS : real; Zl : real);
var KlS, K2S, K3S, K4S, sums : real;
begin {RKforce}
  KlS := -SigB*ZS + SigB*(thetaSp-Zl);
  sums := zs + 0.5*h*KlS;
  K2S := -SigB*sums + SigB*(thetaSp-Zl);
  sums := zs + 0.5*h*K2S;
  K3S := -SigB*sums + SigB*(thetaSp-Zl);
  sums := zs + h*K3S;
  K4S := -sigB*sums + SigB*(thetasp-Zl);
  ZS := ZS+ (h/6.0)*(K1S+2.0*K2S+2.0
                 *K3S+K4S);
end; {RKforce}

procedure RungaKutta;
var suml, sum2, sum3, sum4 : real;
begin {RungaKutta}
  if (t>O) then RKforce(Z[S],Z[l]);
  force := ((Km*Kt*SigA)/SigB)*((l.O-SigB/SigA)*Z[S]
             + (SigB/SigA)*(thetaSp-suml));
  K1[1] := Z[2];
  K1[3] := Z[4];
  suml := Z[1] + 0.5*h*K1[1];
  sum2 := Z[2] + 0.5*h*K1[2];
  sum3 := Z[3] + 0.5*h*K1[3];
  sum4 := Z[4] + 0.5*h*K1[4];
  force := ((Km*Kt*SigA)/SigB)*((1.0-SigB/SigA)*Z[S]
             + (SigB/SigA)*(thetaSp-suml));
  K2[1] := sum2;
  K2[2] := mul3*sum1 + mul4*force;
  K2[4] := mul5*sum1 + mul6*force;
  sum1 := Z[1] + 0.5*h*K2[1];
  sum2 := Z[2] + 0.5*h*K2[2];
  sum3 := Z[3] + 0.5*h*K2[3];
  sum4 := Z[4] + 0.5*h*K2[4];
  force := ((Km*Kt*SigA)/SigB)*((1.0-SigB/SigA)*Z[S]
             + (SigB/SigA)*(thetaSp-sum1));
K3[1] := sum2;
K3[2] := mul3*sum1 + mul4*force;
K3[4] := mul5*sum1 + mul6*force;

sum1 := Z[1] + h*K3[1];
sum2 := Z[2] + h*K3[2];
sum3 := Z[3] + h*K3[3];
sum4 := Z[4] + h*K3[4];

force := ((Km*Kt*SigA)/SigB)*((1.0-SigB/SigA)*Z[5] + (SigB/SigA)*(thetaSp-sum1));

K4[1] := sum2;
K4[2] := mul3*sum1 + mul4*force;
K4[4] := mul5*sum1 + mul6*force;

nextZ;
end; (RungaKutta)

begin {GetThetaX}
initvalue;
repeat
    writeln(Lst,'t = ',t:6:2,' theta = ',Z[1]:8:5);
    RungaKutta;
    t := t + h;
until t>=tfinal;
end. (GetThetaX)
PROGRAM INVERTED PENDULUM

INITIAL
CONSTANT
  XIC = 0.0 ,  THEIC = 10.0 ...
  XDIC = 0.0 ,  THEDIC = 0.0 ...
  MC = 1.0 ,  M = 0.2 ,  L = 1.5
  G = 9.81 ...
  MAX = 0.0 ,  TZ = 1.0 ,  TMX = 20.0
  KM = 1.0 ...
  KT = 1.0 ,  SIGA = 1.0 ,  SIGB = 4.0

CINTERVAL  CINT = 0.1
NSTEPS  NSTP = 10
ALGORITHM  IALG = 5

  MUL3 = (3.0*(MC+M)*G)/(L*(4.0*MC+M))
  MUL4 = -3.0/(L*(4.0*MC+M))
  T1 = 1.0/SIGA
  T2 = 1.0/SIGB

END "$ OF INITIAL"

DYNAMIC

DERIVATIVE
  THE = INTEG ( THED,THEIC )
  THED = INTEG ( THEDD,THEDIC )
  THEDD = MUL4 * F + MUL3 * THE
  THESP = MAX * STEP( TZ )
  VT = KT * THE
  ES = KT * THESP
  EV = ES - VT
  V = ( SIGA/SIGB ) * EV
  EC = LEDLAG (T1,T2,V,0.0)
  F = KM * EC

END "$ OF DERIVATIVE"
TERMT ( T.GE.TMX )
END "$ OF DYNAMIC"

END "$ OF PROGRAM"
APPENDIX 2

SOFTWARE LISTING
program main; {Research Report By Koon Tock Ang. Fall 1986}

type
  coordinates = array[1..10,1..2] of integer;
  sevenhundred = array[1..1000] of real;

const
  point : coordinates = ((71,3),(74,4),(11,6),(13,8)
  , (53,8),(53,12),(7,71),(7,72)
  , (7,73),(1,74));
  MaxWorldX : real = 1000.0;
  MaxWorldY : real = 1000.0;
  g : real = 9.81;

var
  Arrow, countNum, contPointer : integer;
  quit, start, controllerflag : boolean;
  IntFlag, RealFlag : boolean;
  M, L, SigA, SigB : real;
  thetaSP, thetainit, distXinit : real;
  t, h, tfinal, Kt, Km : real;
  calNum1, calNum2, calNum3 : sevenhundred;
  calNum4, calNum5, calNum6 : sevenhundred;
  ch : char;

{$I typedef.sys}
{$I graphix.sys}
{$I kernel.sys}
{$I windows.sys}
{$I circsegm.hgh}
{$I axis.hgh}
{$I polygon.hgh}
{I broom.fun}

procedure define;
begin {define}
  DefineWorld(3,0,0,MaxWorldX,MaxWorldY);
  DefineWindow(5,0,0,79,199);
  DefineWindow(15,39,119,71,180);
end; {define}
procedure initial;
begin {initial}
ch := ' '; start := false; quit := false; controllerflag := false; contPointer := 73;
thetainit := 10.0; thetaSp := 0.0; distXinit := 0.0; h := 0.1; t := 0.0; tfinal := 0.0;
Km := -154.0; Kt := 1.0; Mc := 1.0; M := 0.2; L := 1.5;
SigA := 1.0; SigB := 4.0;
end; {initial}

procedure DrawSystem;

procedure Labeling;
begin {Labeling}
GotoXY(13,8); write(Kt:6:1);
GotoXY(53,12); write(Kt:6:1);
GotoXY(53,8); write(Km:6:1);
GotoXY(11,6); write(thetaSP:6:1);
GotoXY(71,3); write(thetainit:6:1);
GotoXY(74,4); write(distXinit:6:1);
GotoXY(7,71); write(L:6:1);
GotoXY(7,72); write(M:6:1);
GotoXY(7,73); write(Mc:6:1);
case contPointer of
  69 : write('( P )');
  70 : write('( PI )');
  71 : write('( PD )');
  72 : write('( PID )');
  73 : write('(lead-lag)');
end; {case}
GotoXY(3,6); write('O sp = ');
GotoXY(64,3); DrawLine(23,220,43,220); write('O o = ');
GotoXY(68,4); DrawLine(785,100,805,100); write('Xo = ');
GotoXY(1,71); write('L = ');
GotoXY(1,72); write('M = ');
GotoXY(1,73); write('Mc = ');
GotoXY(20,7); write('Kt');
GotoXY(60,7); write('Km');
GotoXY(60,11); write('Kt');
GotoXY(34,7); write('controller');
GotoXY(66,8); write('process');
GotoXY(22,9); write('Es');
GotoXY(30,9); write('Ev');
GotoXY(46,9); write('Ec');
GotoXY(62,9); write('f');
GotoXY(30,11); write('Vt');
GotoXY(26,10); write('-');
GotoXY(78,6); write('x');
GotoXY(78,10); write('O');
DrawLine(960,380,980,380);
end; 

begin 

{Labeling}

GotoXY(28,2);
write('Inverted Pendulum System');
DrawLine(340,81,640,81);
DrawLine(60,250,60,300);
DrawLine(49,285,60,300);
DrawLine(71,285,60,300);
DrawLine(22,300,117,300);
DrawLine(104,288,117,300);
DrawLine(104,312,117,300);
DrawLine(244,300,313,300);
DrawLine(300,288,313,300);
DrawLine(300,312,313,300);
SetAspect(1);
DrawCircle(335,300,0.15);

{arrow #0}
DrawLine(357,300,400,300);
DrawLine(387,288,400,300);
DrawLine(387,312,400,300);
DrawSquare(400,241,548,359,false);

{arrow #1}
DrawSquare(800,241,909,359,false);
DrawSquare(117,270,244,330,false);

{arrow #2}
DrawSquare(617,430,743,490,false);
DrawSquare(617,270,743,330,false);

{arrow #3}
DrawLine(548,300,617,300);
DrawLine(604,288,617,300);
DrawLine(604,312,617,300);
DrawLine(743,300,800,300);
DrawLine(787,288,800,300);

{arrow #4}
DrawLine(909,270,983,270);
DrawLine(970,258,983,300);
DrawLine(970,282,983,300);
DrawLine(909,330,983,330);

{arrow #5}
DrawLine(970,282,983,300);
DrawLine(970,318,983,300);
DrawLine(909,270,983,270);

{arrow #6}
DrawLine(970,258,983,300);
DrawLine(970,282,983,300);
DrawLine(909,270,983,270);

{arrow #7}
DrawLine(970,318,983,300);
DrawLine(970,342,983,300);
DrawLine(820,117,820,241);

{arrow #8}
DrawLine(809,226,820,241);
DrawLine(831,226,820,241);

{arrow #9}
DrawLine(858,173,858,241);
DrawLine(847,226,858,241);
DrawLine(869,226,858,241);

{arrow #10}
DrawLine(942,330,942,460);

{arrow #11}
DrawLine(942,460,743,460);
DrawLine(756,448,743,460);
procedure ArrowPointer;
var k : integer;
    chArrow : char;
begin {ArrowPointer}
    k := 69; chArrow := ' '; repeat
        GotoXY(25,k); write('--->');
        GotoXY(25,k); write(' ');
        if KeyPressed then begin
            read(Kbd,chArrow);
            quit := (chArrow = ^C);
            if (chArrow = #27) and KeyPressed then begin
                read(Kbd,chArrow);
                case ord(chArrow) of
                72 : k := k - 1; { Up arrow }
                80 : k := k + 1; { Down arrow }
                end; {case}
            end;
        end;
        if quit then begin
            leavegraphic;
            halt;
        end;
    if (k = 68) and not(controllerflag) then k := 74;
    if (k = 75) and not(controllerflag) then k := 69;
    if (k = 68) and (controllerflag) then k := 73;
    if (k = 74) and (controllerflag) then k := 69;
    if (ord(chArrow) = 13) then Arrow := k;
    until (ord(chArrow) = 13);
end; {ArrowPointer}
procedure RealVal(var tempvalue : real);
var testvalue : string[7];
    error : integer;
begin {RealVal}
    RealFlag := true;
    readln(testvalue);
    Val(testvalue,tempvalue,error);
    if (error <> 0) then RealFlag := false;
end; {RealVal}

procedure IntVal(var tempvalue : integer);
var testvalue : string[7];
    error : integer;
begin {IntVal}
    IntFlag := true;
    readln(testvalue);
    Val(testvalue,tempvalue,error);
    if (error <> 0) then IntFlag := false;
end; {IntVal}

procedure submenu1;
begin {submenu1}
    GotoXY(25,68);
    write('Selections: ');
    GotoXY(29,69);
    write('simulate system ');
    GotoXY(29,70);
    write('change controller ');
    GotoXY(29,71);
    write('change parameters ');
    GotoXY(29,72);
    write('plot graphs ');
    GotoXY(29,73);
    write('numerical outputs ');
    GotoXY(29,74);
    write('exit simulation ');
end; {submenu1}

procedure submenu2;
begin {submenu2}
    GotoXY(25,68);
    write('Controller Selections: ');
    GotoXY(29,69);
    write('proportional ');
    GotoXY(29,70);
    write('prop. integral ');
    GotoXY(29,71);
    write('prop. derivative ');
    GotoXY(29,72);
    write('prop. integral derivative ');
    GotoXY(29,73);
    write('lead-lag ');

procedure submenu3;
begin {submenu3}
GotoXY(25,68);
write('Enter the followings: ');
repeat
  GotoXY(29,69);
  write('integration step, h ==>');
  RealVal(h);
  if (h <= 0.0) then RealFlag := false;
  if not(RealFlag) then begin
    GotoXY(29,69);
    write('integration step, h ==> ');
  end;
until RealFlag;
repeat
  GotoXY(29,70);
  write('length of simulation ==>');
  RealVal(tfinal);
  if (tfinal <= 0.0) or (tfinal <= h) then RealFlag := false;
  if not(RealFlag) then begin
    GotoXY(29,70);
    write('length of simulation ==> ');
  end;
until RealFlag;
if (tfinal/h > 700.0) then begin
  GotoXY(29,72);
  write('ERROR !!. > 700 pts. calculated');
  GotoXY(53,69);
  write(' ');
  GotoXY(53,70);
  write(' ');
end else begin
  GotoXY(29,72);
  write('Please WAIT ! ');
end;
end; {submenu3}

procedure submenu45;
begin {submenu45}
GotoXY(25,68);
write('Select variables: ');
GotoXY(29,69); write('time');
GotoXY(29,70); write('theta');
GotoXY(29,71); write('thetaDot');
GotoXY(29,72); write('distance');
GotoXY(29,73); write('velocity');
GotoXY(29,74); write('force');
end; {submenu45}
procedure menuCaseArrow;
begin {menuCaseArrow}
  case Arrow of
    69: write('time ');
    70: write('theta');
    71: write('thetaDot');
    72: write('distance');
    73: write('velocity');
    74: write('force');
  end; {case}
end; {menuCaseArrow}

procedure clearMenu;
begin {clearMenu}
  SelectWindow(15);
  SetBackGround(0);
  SelectWindow(5);
end; {clearMenu}

{---------------------------------------------}

procedure InvertedPendulum(time,theta,distX : sevenhundred);
const
  m : real = 0.017453292;
  FlagPause : boolean = True;
  c : array [1..8] of integer =
    (15,30,60,120,240,225,195,135);
var
  Ax, Ay, Bx, By, Dx, Dy, Ex, Ey : real;
  oldX, newX : real;
  cntNum, MoveStep, MoveSize : integer;

procedure definePendulum;
begin
  DefineWorld(1,0,0,MaxWorldX,MaxWorldY);
  DefineWorld(2,0,0,2070,955);
  DefineWindow(2,4,18,76,130);
  DefineWindow(3,34,130,46,155);
  DefineWindow(4,4,155,76,170);
  DefineWindow(11,4,18,5,19);
  MoveStep := 0;
  MoveSize := 0;
  cntNum := 1;
end; {definePendulum}

procedure ClearEOL(i,j : integer);
var k : integer;
begin {ClearEOL}
  GotoXY(i,j);
  for k:= 1 to 70 do write(' ');
end; {ClearEOL}
procedure delay(n : real); forward;

procedure Pause;
var i : integer;
begin
repeat
  FlagPause := False;
  ClearEOL(7,23);
  GotoXY(7,23);
  write('Pause');
  i := 1;
  repeat
    i := i + 1;
    write('.');
    delay(400);
    until (FlagPause) or (i=65);
  until FlagPause;
  ClearEOL(7,23);
until FlagPause;
end; {Pause}

procedure delay;
var i : real;
j : integer;
quit : boolean;
begin
  i := 0;
  ch := ' ';
  repeat
    i := i+1;
    quit := false;
    if KeyPressed
then begin
  read(Kbd,ch);
  quit := (ch='C);
  if (ch = #27) and KeyPressed then begin
    read(Kbd,ch);
    case ord(ch) of
      59 : begin
        i := n;
        cntNum := countNum;
        end;
      60 : if FlagPause then pause
      else FlagPause := True;
      61 : quit := true;
      end; {case}
  end;
  end;
  if quit
then begin
leavegraphic;
halt;
procedure BorderBlock (x, y, z : real);
const  
  CharHeight : real = 40.0;
  CharWidth  : real = 12.5;
begin  
  DrawSquare(((x-1)*CharWidth)-6,((y-1)*CharHeight)-6,
  ((x+z)*CharWidth)+6,(y)*CharHeight)+6,false);
end;  
{BorderBlock}

procedure text;
begin 
  SelectWorld(1);
  SelectWindow(1);
  BorderBlock(3,2,14);
  GotoXY(3,2); write('time = ',time[cntNum]:8:3);
  BorderBlock(21,2,16);
  GotoXY(21,2); write('theta = ',theta[cntNum]:9:2);
  BorderBlock(41,2,17);
  GotoXY(41,2); write('dist X = ',distX[cntNum]:9:2);
  BorderBlock(62,2,16);
  GotoXY(62,2); write('theta SP = ',thetaSP:6:1);
end;  
{text}

procedure drawcart;
var   
  alx, aly, angle : real;
begin
  SelectWorld(1);
  SelectWindow(3);
  DrawSquare(0,0,1000,455,true);  
  SetAspect(4);
  alx := 114.0; aly := 659.0; angle := -275;
  DrawCircleSegment(200,773,alx,aly,0.0,0.0,angle,0,'0',0,0);
  alx := 886.0; aly := 659.0; angle := 275;
  DrawCircleSegment(800,773,alx,aly,0.0,0.0,angle,0,'0',0,0);
  DrawPoint(200,773);  
  DrawPoint(800,773);  
  DrawLine(32,455,200,773);  
  DrawLine(406,455,200,773);  
  DrawLine(594,455,800,773);  
  DrawLine(968,455,800,773);
  StoreWindow(3);
end;  
{drawcart}
procedure DrawPendulum;

  procedure deviation( thetatt : real );
  var
    clx, cly, c2x, c2y : real;
    si, co, il, j1 : real;

  procedure calculate;
  begin
    si := sin(thetatt*m);
    co := cos(thetatt*m);
    clx := (800.0*si)+1035.0;
    cly := (800.0-800.0*co)+70.0;
    c2x := 1035.0-(20.0*si);
    c2y := 870.0+(20.0*co);
    il := 20.0*co;
    j1 := 20.0*si;
  end; {calculate}

  begin {deviation}
    calculate;
    Ax := clx-il;
    Ay := cly-j1;
    Dx := clx+il;
    Dy := cly+j1;
    Bx := c2x-il;
    By := c2y-j1;
    Ex := c2x+il;
    Ey := c2y+j1;
  end; {deviation}

  begin {DrawPendulum}
    SelectWorld(2);
    SelectWindow(2);
    SetBackGround(0);
    deviation(theta[cntNum]);
    DrawPoint(1035,870);
    DrawLine(995,845,995,955);
    DrawLine(1075,845,1075,955);
    DrawLine(995,845,1075,845);
    DrawLine(Ax,Ay,Bx,By);
    DrawLine(Dx,Dy,Ex,Ey);
    DrawLine(Ax,Ay,Dx,Dy);
    DrawLine(Bx,By,Ex,Ey);
  end; {DrawPendulum}
procedure DrawDistance;
begin
  newX := distX[cntNum];
  if newX > oldX
    then begin
      MoveSize := MoveSize + 1;
      MoveStep := MoveStep + 3;
      if MoveStep > 8 then MoveStep := MoveStep - 8;
    end;
  if newX < oldX
    then begin
      MoveSize := MoveSize - 1;
      MoveStep := MoveStep - 3;
      if MoveStep < 1 then MoveStep := MoveStep + 8;
    end;
  SelectWorld(2);
  SelectWindow(4);
  { MoveStep := 4; }
  SetBackGround(c[MoveStep]);
  DrawLine(0,0,2070,0);
  oldX := newX;
end; {DrawDistance}

procedure MoveCart;
var
  x1,y1,x2,y2 : real;
  XminMove, YminMove, XmaxMove, YmaxMove : integer;
function max(a, b : real) : integer;
begin
  if a > b then max := trunc(a)
  else max := trunc(b);
end; {max}

function min(a, b : real) : integer;
begin
  if a < b then min := trunc(a)
  else min := trunc(b);
end; {min}

begin
  XminMove := min(Ax*72/2070+3,34);
  XmaxMove := max(Dx*72/2070+6,46);
  YminMove := min(Ay*112/995+11,119);
  YmaxMove := max(Dy*112/995+18,155);
  if (((XmaxMove*640.0/79.0)+(8.0*MoveSize)) > 635) then
    begin
      MoveSize := 1 - trunc((XminMove*640.0)/(79.0*8.0));
    end;
  if (((XminMove*640.0/79.0)+(8.0*MoveSize)) < 5) then
    begin
      MoveSize := -1 + trunc((640.0 - (XmaxMove*640.0)/79.0)/8.0);
    end;
  RedefineWindow(11,XminMove,YminMove,XmaxMove,YmaxMove);
  SelectWindow(11);
procedure Draw;
begin
  text;
  DrawPendulum;
  DrawDistance;
  DrawCart;
end; {Draw}

begin (InvertedPendulum)
  definePendulum;
  clearScreen;
  SelectWorld(1);
  SelectWindow(1);
  DefineHeader(1,'F1-change parameter  
  s F2-pause F3-quit to DOS');
  SetHeaderOn;
  DrawBorder;
  StoreWindow(1);
  Draw;
  delay(2000);
  CopyScreen;
  repeat
    SelectScreen(2);
    RestoreWindow(1,0,0);
    text;
    SelectWindow(11);
    SetBackGround(0);
    DrawPendulum;
    RestoreWindow(3,0,0);
    MoveCart;
    DrawDistance;
    CopyScreen;
    SelectScreen(1);
    cntNum := cntNum + 1;
    if (theta[cntNum] > 93.0) then theta[cntNum]:= 93.0;
    if (theta[cntNum] < -93.0) then theta[cntNum]:= -93.0;
    delay(7);
    until (ord(ch)=59) or (cntNum=countNum) ;
end; {InvertedPendulum}

{------------------------------------------}
procedure StartSimulation;
var  area, force, SigA, SigB : real;
    Z, K1, K2, K3, K4 : array [1..5] of real;
    mul3, mul4, mul5, mul6 : real;
    min1, max1, min2, max2, min3, max3 : real;
    min4, max4, min5, max5, min6, max6 : real;
    chArrow : char;

procedure Setvalue;
begin {Setvalue}
  mul3 := (3.0*(Mc+M)*g)/(L*(4.0*Mc+M));
  mul4 := -3.0/(L*(4.0*Mc+M));
  mul5 := -(3.0*M*g)/(4.0*Mc+M);
  mul6 := 4.0/(4.0*Mc+M);
  min1 := t ; max1 := t;
  min2 := thetainit; max2 := thetainit;
  min3 := 0.0 ; max3 := 0.0;
  min4 := distXinit; max4 := distXinit;
  min5 := 0.0 ; max5 := 0.0;
  min6 := 0.0 ; max6 := 0.0;
  Z[1] := thetainit;
  Z[2] := 0.0;
  Z[3] := distXinit;
  Z[4] := 0.0;
  Z[5] := 0.0;
  SigA := 1.0 ; SigB := 4.0;
  force := 0.0;
  area := 0.0;
  t := 0.0;
end;  {Setvalue}

procedure StoreValue;
begin {StoreValue}
  calNum1[countNum] := t;
  if (t < min1) then min1 := t;
  if (t > max1) then max1 := t;
  calNum2[countNum] := Z[1];
  if (Z[1] < min2) then min2 := Z[1];
  if (Z[1] > max2) then max2 := Z[1];
  calNum3[countNum] := Z[2];
  if (Z[2] < min3) then min3 := Z[2];
  if (Z[2] > max3) then max3 := Z[2];
  calNum4[countNum] := Z[3];
  if (Z[3] < min4) then min4 := Z[3];
  if (Z[3] > max4) then max4 := Z[3];
  calNum5[countNum] := Z[4];
  if (Z[4] < min5) then min5 := Z[4];
  if (Z[4] > max5) then max5 := Z[4];
  calNum6[countNum] := force;
  if (force < min6) then min6 := force;
  if (force > max6) then max6 := force;
end;  {StoreValue}
procedure StoreMinMax;
begin {StoreMinMax}
calNum1[countNum] := min1; {min. val. time}
calNum1[countNum+1] := max1; {max. val. time}
calNum2[countNum] := min2; {min. theta}
calNum2[countNum+1] := max2; {max. theta}
calNum3[countNum] := min3; {min. thetadot}
calNum3[countNum+1] := max3; {max. thetadot}
calNum4[countNum] := min4; {min. distX}
calNum4[countNum+1] := max4; {max. distX}
calNum5[countNum] := min5; {min. distXdot}
calNum5[countNum+1] := max5; {max. distXdot}
calNum6[countNum] := min6; {min. force}
calNum6[countNum+1] := max6; {max. force}
end; {StoreMinMax}

procedure RKforce(var Z5 : real; Z1 : real);
var K15, K25, K35, K45, sum5 : real;
begin {RKforce}
K15 := -SigB*Z5 + SigB*(thetaSp-Z1);
sum5 := Z5 + 0.5*h*K15;
K25 := -SigB*sum5 + SigB*(thetaSp-Z1);
sum5 := Z5 + 0.5*h*K25;
K35 := -SigB*sum5 + SigB*(thetaSp-Z1);
sum5 := Z5 + h*K35;
K45 := -SigB*sum5 + SigB*(thetaSp-Z1);
Z5 := Z5 + (h/6.0)*(K15+2.0*K25+2.0*K35+K45);
end; {RKforce}

procedure RungaKutta;
var sum1, sum2, sum3, sum4 : real;
procedure nextZ;
var n : integer;
begin {nextZ}
for n := 1 to 4 do
begin
Z[n] := Z[n] + (h/6.0)*(K1[n]+2.0*K2[n]+2.0*K3[n]+K4[n]);
end;
end; {nextZ}

begin {RungaKutta}
case contPointer of
69 : force := -Km*Kt*(thetaSP-Z[1]); {P controller}
70 : begin
area := area + h*Z[1];
force := -Km*(Kt*(thetaSP-Z[1]) - area);
end;
end; {RungaKutta}
72: begin
  area := area + h*Z[1];
  force := -Km*(-Z[2]+Kt*(thetaSP-Z[1])-area);
end;

73: begin
  if (t>0) then RKforce(Z[5],Z[1]);
  force := ((Km*Kt*SigA)/SigB)*((1.0-SigB/SigA)
     *(Z[5])+((SigB/SigA)*(thetaSP-Z[1])));
end;
end; {case}
K1[1] := Z[2];
K1[3] := Z[4];

suml := Z[1] + 0.5*h*K1[1];
sum2 := Z[2] + 0.5*h*K1[2];
sum3 := Z[3] + 0.5*h*K1[3];
sum4 := Z[4] + 0.5*h*K1[4];

case contPointer of
  69: force := - Km*Kt*(thetaSP-suml);
  70: force := - Km*(Kt*(thetaSP-suml) - area);
  71: force := Km*(sum2 - Kt*(thetaSP-suml));
  72: force := - Km*(-sum2 + Kt*(thetaSP-suml)
    - area);  
  73: force := ((Km*Kt*SigA)/SigB)*((1.0-SigB/SigA)
    *(Z[5]) + (SigB/SigA)*(thetaSP-suml));
end; {case}
K2[1] := sum2;
K2[2] := mul3*sum1 + mul4*force;
K2[4] := mul5*sum1 + mul6*force;

suml := Z[1] + 0.5*h*K2[1];
sum2 := Z[2] + 0.5*h*K2[2];
sum3 := Z[3] + 0.5*h*K2[3];
sum4 := Z[4] + 0.5*h*K2[4];

case contPointer of
  69: force := - Km*Kt*(thetaSP-suml);
  70: force := - Km*(Kt*(thetaSP-suml) - area);
  71: force := Km*(sum2 - Kt*(thetaSP-suml));
  72: force := - Km*(-sum2 + Kt*(thetaSP-suml)
    - area);  
  73: force := ((Km*Kt*SigA)/SigB)*((1.0-SigB/SigA)
    *(Z[5]) + (SigB/SigA)*(thetaSP-suml));
end; {case}
K3[1] := sum2;
K3[2] := mul3*sum1 + mul4*force;
K3[4] := mul5*sum1 + mul6*force;

suml := Z[1] + h*K3[1];
sum2 := Z[2] + h*K3[2];
sum3 := Z[3] + h*K3[3];
sum4 := Z[4] + h*K3[4];
case contPointer of
69 : force := - Km*Kt*(thetaSP-sum1);
70 : force := - Km*(Kt*(thetaSP-sum1) - area);
71 : force := Km*(sum2 - Kt*(thetaSP-sum1));
72 : force := - Km*(-sum2 + Kt*(thetaSP-sum1) - area);
73 : force := ((Km*Kt*SigA)/SigB)*((l.O-SigB/SigA)*Z[5] + (SigB/SigA)*(thetaSp-sum1));
end; {case}
K4[1] := sum2;
K4[2] := mul3*sum1 + mul4*force;
K4[4] := mul5*sum1 + mul6*force;
extZ;
end; {RungaKutta}

begin {StartSimulation}
start := true;
countNum := 1;
clearMenu;
repeat submenu3 until (tfinal/h <= 700.0);
Setvalue;
repeat
  GotoXY(36,72); write('WAIT !');
  GotoXY(36,72); write(' !');
  StoreValue;
  RungaKutta;
  t := t + h;
  countNum := countNum + 1;
until t>tfinal;
StoreMinMax;
repeat
  GotoXY(29,72);
  write('');
  GotoXY(29,72);
  write('Bypass the animation ? ');
  readln(chArrow);
  until (chArrow in ['Y','y','N','n']);
if (chArrow in ['N','n']) then
  InvertedPendulum(calNum1,calNum2,calNum4);
ClearScreen;
SelectWorld(3);
SelectWindow(5);
SetHeaderOff;
end; {StartSimulation}
Overlay procedure ChangeController;
begin (ChangeController)
    start := false;
    controllerflag := true;
    clearMenu;
    submenu2;
    ArrowPointer;
    GotoXY(34,8);
    case Arrow of
        69 : write('(' P ')');
        70 : write('(' PI ')');
        71 : write('(' PD ')');
        72 : write('(' PID ')');
        73 : write('(' lead-lag ')');
    end; (case)
    contPointer := Arrow;
    controllerflag := false;
end; (ChangeController)

Overlay procedure PlotGraphs;
var
    a : PlotArray;
    k : integer;
    chArrow : char;
procedure GenerateFunction;
var cnt1 : integer;
begin (GenerateFunction)
    for cnt1 := 1 to countNum+1 do
        begin
            case Arrow of
                69 : a[cnt1,1] := calNum1[cnt1];
                70 : a[cnt1,1] := calNum2[cnt1];
                71 : a[cnt1,1] := calNum3[cnt1];
                72 : a[cnt1,1] := calNum4[cnt1];
                73 : a[cnt1,1] := calNum5[cnt1];
                74 : a[cnt1,1] := calNum6[cnt1];
            end; (case)
            case k of
                69 : a[cnt1,2] := calNum1[cnt1];
                70 : a[cnt1,2] := calNum2[cnt1];
                71 : a[cnt1,2] := calNum3[cnt1];
                72 : a[cnt1,2] := calNum4[cnt1];
                73 : a[cnt1,2] := calNum5[cnt1];
                74 : a[cnt1,2] := calNum6[cnt1];
            end; (case)
        end;
    end; (GenerateFunction)
begin (PlotGraphs)
  controllerFlag := false;
  clearMenu;
  submenu45;
  GotoXY(25,75); write('Y-axis = ');
  ArrowPointer;
  GotoXY(34,75); menuCaseArrow;
  k := Arrow;
  GotoXY(45,75); write('& X-axis = ');
  ArrowPointer;
  DefineWindow(9,0,0,XMaxGlb,YMaxGlb);
  GotoXY(56,75);
  case Arrow of
    69 : begin
      case k of
        69 : DefineHeader(9,'time versus time');
        70 : DefineHeader(9,'theta versus time');
        71 : DefineHeader(9,'thetaDot versus time');
        72 : DefineHeader(9,'distance versus time');
        73 : DefineHeader(9,'velocity versus time');
        74 : DefineHeader(9,'force versus time');
      end; {case}
    end;
    70 : begin
      case k of
        69 : DefineHeader(9,'time versus theta');
        70 : DefineHeader(9,'theta versus theta');
        71 : DefineHeader(9,'thetaDot versus theta');
        72 : DefineHeader(9,'distance versus theta');
        73 : DefineHeader(9,'velocity versus theta');
        74 : DefineHeader(9,'force versus theta');
      end; {case}
    end;
    71 : begin
      case k of
        69 : DefineHeader(9,'time versus thetaDot');
        70 : DefineHeader(9,'theta versus thetaDot');
        71 : DefineHeader(9,'thetaDot versus thetaDot');
        72 : DefineHeader(9,'distance versus thetaDot');
        73 : DefineHeader(9,'velocity versus thetaDot');
        74 : DefineHeader(9,'force versus thetaDot');
      end; {case}
    end;
    72 : begin
      case k of
        69 : DefineHeader(9,'time versus distance');
        70 : DefineHeader(9,'theta versus distance');
        71 : DefineHeader(9,'thetaDot versus distance');
        72 : DefineHeader(9,'distance versus distance');
        73 : DefineHeader(9,'velocity versus distance');
        74 : DefineHeader(9,'force versus distance');
      end; {case}
end;
73 : begin
  case k of
    69 : DefineHeader(9,'time versus velocity');
    70 : DefineHeader(9,'theta versus velocity');
    71 : DefineHeader(9,'thetaDot versus velocity');
    72 : DefineHeader(9,'distance versus velocity');
    73 : DefineHeader(9,'velocity versus velocity');
    74 : DefineHeader(9,'force versus velocity');
  end; {case}
end;
end; {case}
GenerateFunction;
ClearScreen;
DefineWorld(4,a[countNum,1]-0.5,a[countNum,2]-0.5,
           a[countNum+1,1]+0.5,a[countNum+1,2]+0.5);
SelectWorld(4);
SelectWindow(9);
SetBackground(0);
SetForegroundColor(11);
SetHeaderOn;
DrawBorder;
DrawAxis(9,-9,0,0,0,0,0,0,true);
DrawPolygon(a,1,countNum-1,0,0,0);
repeat until Keypressed;
SetForegroundColor(14);
ClearScreen;
SelectWorld(3);
SelectWindow(5);
SetHeaderOff;
end; {PlotGraphs}

{-----------------------------------~----------------------}
Overlay procedure NumericalOutputs;
var n,n1,n2,n3,n4,n5,n6,k,i,temp : integer;
    chArrow : char;

begin (NumericalOutputs)
    controllerflag := false;
    n := 1; n1 := 0; n2 := 0; n3 := 0;
    n4 := 0; n5 := 0; n6 := 0;
clearMenu;
submenu45;
repeat
    ArrowPointer;
    if (n = 1) then begin
        n1 := Arrow;
        GotoXY(41,69);
        write('var. #1 =');
    end;
    if (n = 2) then begin
        n2 := Arrow;
        GotoXY(41,70);
        write('var. #2 =');
    end;
    if (n = 3) then begin
        n3 := Arrow;
        GotoXY(41,71);
        write('var. #3 =');
    end;
    if (n = 4) then begin
        n4 := Arrow;
        GotoXY(41,72);
        write('var. #4 =');
    end;
    if (n = 5) then begin
        n5 := Arrow;
        GotoXY(41,73);
        write('var. #5 =');
    end;
    if (n = 6) then begin
        n6 := Arrow;
        GotoXY(41,74);
        write('var. #6 =');
    end;
    menuCaseArrow;
n := n + 1;
if (n < 7) then begin
    repeat
        GotoXY(41,75);
        write('Another variable? ');
        GotoXY(25,75);
        write('Another variable? ');
        readln(chArrow);
    until (chArrow in ['Y','y','N','n']);
    GotoXY(25,75); write('Another variable? ');
end;
until (chArrow in ['N','n']) or (n = 7);
repeat
  GotoXY(25,75);
  write('Output every data points.');
  GotoXY(38,75); IntVal(k);
  if (k <= 0) then IntFlag := false;
until IntFlag;
n := 1;
writeln(Lst, 'Inverted Pendulum System.');
------------------------
write(Lst,'set point angle(thetaSP) = ',thetaSP:13:3,' degrees.');
write(Lst,'initial angle(thetainit) = ',thetainit:13:3,' degrees.');
write(Lst,'initial distance(distXinit) = ',distXinit:13:3,' meters.');
write(Lst,'transmitter gain,Kt = ',Kt:13:3,' volts/degree.');
write(Lst,'controller gain,Km = ',Km:13:3,' Newtons/degree.');
write(Lst,'half length of pendulum,L = ',L:13:3,' meters.');
write(Lst,'mass of pendulum,M = ',M:13:3,' kilograms.'陌
write(Lst,'mass of cart,Mc = ',Mc:13:3,' kilograms.');
for i := 1 to 5 do begin
  temp := 0;
  if (i=1) and (n1<>0) then temp := n1;
  if (i=2) and (n2<>0) then temp := n2;
  if (i=3) and (n3<>0) then temp := n3;
  if (i=4) and (n4<>0) then temp := n4;
  if (i=5) and (n5<>0) then temp := n5;
  case temp of
    69 : write(Lst, 'time');
    70 : write(Lst, 'theta');
    71 : write(Lst, 'thetaDot');
    72 : write(Lst, 'distance');
    73 : write(Lst, 'velocity');
    74 : write(Lst, 'force');
  end; {case}
end;
if (n6 <> 0) then begin
  case n6 of
    69 : writeln(Lst, 'time');
    70 : writeln(Lst, 'theta');
    71 : writeln(Lst, 'thetaDot');
    72 : writeln(Lst, 'distance');
    73 : writeln(Lst, 'velocity');
    74 : writeln(Lst, 'force');
  end; {case}
end; {case}
repeat
  for i := 1 to 5 do begin
    temp := 0;
    if (i=1) and (n1<>0) then temp := n1;
    if (i=2) and (n2<>0) then temp := n2;
    if (i=3) and (n3<>0) then temp := n3;
    if (i=4) and (n4<>0) then temp := n4;
    if (i=5) and (n5<>0) then temp := n5;
    case temp of
      69 : write(Lst, ',calNum1[n]:8:3);
      70 : write(Lst, ',calNum2[n]:11:3);
      71 : write(Lst, ',calNum3[n]:11:3);
      72 : write(Lst, ',calNum4[n]:11:3);
      73 : write(Lst, ',calNum5[n]:13:3);
      74 : write(Lst, ',calNum6[n]:13:3);
    end; {case}
  end;
end; {case}
if (n6 <> 0) then begin
  case n6 of
    69 : writeln(Lst, ',calNum1[n]:8:3);
    70 : writeln(Lst, ',calNum2[n]:11:3);
    71 : writeln(Lst, ',calNum3[n]:11:3);
    72 : writeln(Lst, ',calNum4[n]:11:3);
    73 : writeln(Lst, ',calNum5[n]:13:3);
    74 : writeln(Lst, ',calNum6[n]:13:3);
  end; {case}
end else writeln(Lst, ' ');  
n := n + k;  
until (n > countNum-1);  
GotoXY(25,75);  
write(' ');  
end; {NumericalOutputs}

 Overlay procedure ChangeParameters;  
 var n : integer;  
    chArrow : char;  
    temp : real;  
 begin {ChangeParameters}  
    start := false;  
    n := 1;  
    chArrow := ' ';  
    clearMenu;  
    GotoXY(29,71);  
    write('first hit the SPACE BAR,');  
    GotoXY(29,72);  
    write('then enter the new value');  
    GotoXY(29,73);  
    write('followed by CARRIAGE RETURN.');  
    GotoXY(1,74);  
    write('changes COMPLETED');  
    repeat  
        GotoXY(point[n,1],point[n,2]);  
        case n of  
            1 : write(thetainit:6:1);  
            2 : write(distXinit:6:1);  
            3 : write(thetaSP:6:1);  
            4 : write(Kt:6:1);  
            5 : write(Km:6:1);  
            6 : write(Kt:6:1);  
            7 : write(L:6:1);  
            8 : write(M:6:1);  
            9 : write(Mc:6:1);  
            10 : write('changes COMPLETED');  
        end; {case}  
        GotoXY(point[n,1],point[n,2]);  
        if n = 10 then write('')  
            else write('');  
        if KeyPressed then begin  
            read(Kbd,chArrow);  
        end;  
    end; {repeat}  
end.
if (chArrow = #32) then begin
  repeat
    GotoXY(point[n,1],point[n,2]);
    RealVal(temp);
    if not(RealFlag) then begin
      GotoXY(point[n,1],point[n,2]);
      write(' ');
    end;
  until RealFlag;
  case n of
    1 : thetainit := temp;
    2 : distXinit := temp;
    3 : thetaSP := temp;
    4 : Kt := temp;
    5 : Km := temp;
    6 : Kt := temp;
    7 : L := temp;
    8 : M := temp;
    9 : Mc := temp;
    end; {case}
  end;
  GotoXY(point[n,1],point[n,2]);
  case n of
    1 : write(thetainit:6:1);
    2 : write(distXinit:6:1);
    3 : write(thetaSP:6:1);
    4 : begin
      write(Kt:6:1);
      GotoXY(point[n+2,1],point[n+2,2]);
      write(Kt:6:1);
    end;
    5 : write(Km:6:1);
    6 : begin
      write(Kt:6:1);
      GotoXY(point[n-2,1],point[n-2,2]);
      write(Kt:6:1);
    end;
    7 : write(L:6:1);
    8 : write(M:6:1);
    9 : write(Mc:6:1);
    10 : write('changes COMPLETED');
    end; {case}
  if (chArrow = #27) and KeyPressed then begin
    read(Kbd,chArrow);
    case ord(chArrow) of
      72 : n := n - 1; { Up arrow }
      80 : n := n + 1; { Down arrow }
    end; {case}
  end;
end;
if (n = 11) then n := 1;
if (n = 0) then n := 10;
until (n = 10) and (ord(chArrow) = 13);
GotoXY(1,74);
write(' ');
end; {ChangeParameters}

{----------------------------------------------------------}
begin {main-editor}
InitGraphic;
SetBreakOn;
SetMessageOn;
SetHeaderToBottom;
SetForegroundColor(14);
define;
initial;
SelectWorld(3);
SelectWindow(5);
ClearScreen;
DrawSystem;
repeat
submenul;
if (ord(ch) = 59) then begin
  Arrow := 71;
  ch := ' ';
end
else ArrowPointer;
case Arrow of
  69 : StartSimulation;
  70 : ChangeController;
  71 : ChangeParameters;
  72 : if start then begin
       PlotGraphs;
       Arrow := 72;
     end;
  73 : if start then NumericalOutputs;
  74 : quit := true;
end; {case}
if (Arrow = 72) or (Arrow = 69) then Drawsyste;m;
until quit;
LeaveGraphic;
end. {main-editor}
REFERENCES

ACSL-Advanced Continuous Simulation Language.

Boyce, William E., and DiPrima, Richard C.
Elementary Differential Equations and Boundary

Cannon, Robert H., Jr. Dynamics of Physical Systems.

Cooper, Doug., and Clancy, Michael. Oh! Pascal!

Elgerd, Olle I. Control Systems Theory.

LENIPEN The Intelligent Color Graphics System.

Melsa, James L., and Schultz, Donald G.

Rainville, Earl D., and Bedient, Phillip E.


TURBO GRAPHIX TOOLBOX Manual.