Hans Guthrie
5/1/2014
ECE 540
Dr. Day

Project IV: Systems Modeling and Analysis

Purpose

For this project, we evaluated how a systems response changes as certain parameters are changed.

a) Here is the transfer equation for the full system.

$$H(s) = \frac{V_o}{Vi} = \frac{(K_p s + K_i)}{(\tau_g \tau_s s^3 + (\tau_g + \tau_s)s^2 + (1+K_p)s + K_i)}$$

Now we are going to write it as a differential equation and convert it to system of state variables.

$$Vi(K_p S + K_i) = V_o(\tau_g \tau_s s^3 + (\tau_g + \tau_s)s^2 + (1 + K_p)s + K_i)$$

$$K_p \frac{dV_i}{dt} + K_i = \tau_g \tau_s \frac{d^3 V_o}{dt^3} + \ (\tau_g + \tau_s)\frac{d^2 V_o}{dt^2} + (1 + K_p)\frac{dV_o}{dt} + K_i$$

$$\frac{d^3 V_o}{dt^3} = -\left(\frac{\tau_g + \tau_s}{\tau_g \tau_s}\right)\frac{d^2 V_o}{dt^2} - \left(\frac{1+K_p}{\tau_g \tau_s}\right)\frac{dV_o}{dt} + \left(\frac{K_p}{\tau_g \tau_s}\right)\frac{dV_i}{dt} - \left(\frac{V_o K_i}{\tau_g \tau_s}\right) + \left(\frac{V_i K_i}{\tau_g \tau_s}\right)$$

$$\begin{Bmatrix} dV_o \\ d^2 V_o \\ d^3 V_o \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\frac{K_i}{\tau_g \tau_s} & -\frac{(1+K_p)}{\tau_g \tau_s} & -\frac{(\tau_g + \tau_s)}{\tau_g \tau_s} \end{bmatrix} \begin{bmatrix} V_o \\ dV_o \\ d^2 V_o \end{bmatrix}$$

$$\frac{d}{dx}\overline{x} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\frac{K_i}{\tau_g \tau_s} & -\frac{(1+K_p)}{\tau_g \tau_s} & -\frac{(\tau_g + \tau_s)}{\tau_g \tau_s} \end{bmatrix} \overline{x} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{K_i}{\tau_g \tau_s} & \frac{K_p}{\tau_g \tau_s} \end{bmatrix} \begin{bmatrix} V_i \\ \frac{dV_i}{dt} \end{bmatrix}$$

b) Using the state variable model we simulated the step response of the system for $\tau_g = 3.0\ s, \tau_s = 0.5\ s,$ and $\Delta t = 1\ ms$ simulated over 6 seconds. We varied the controller gains according to the following table. To do this, we used the `Exp_A()` function given to us in `Exp_A.hpp`.

$\boldsymbol{K_p}$	$\boldsymbol{K_i}$
4	0
8	0
12	0
4	2
8	2
12	2
4	4
8	4
12	4

Table 1. Set of Gains to be Simulated

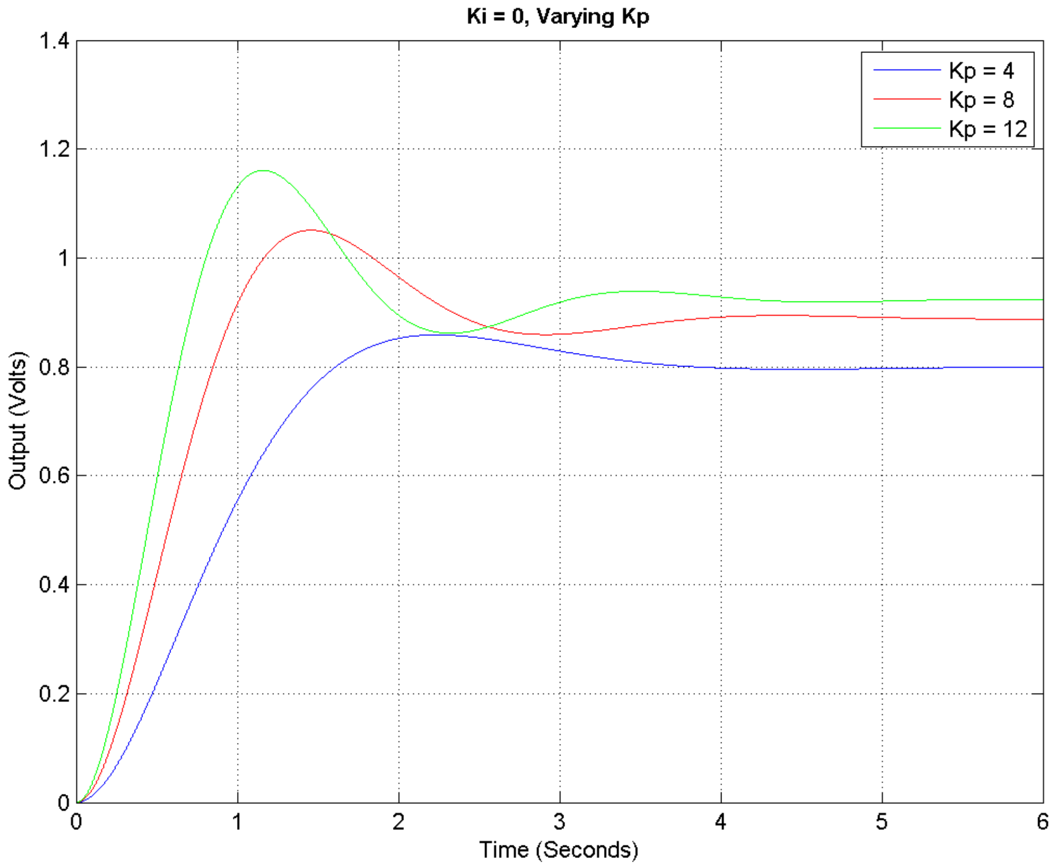
c) Our code generated 6 csv files as output. Each file held one of the parameters constant, and varied the other parameters through their values. The plots are included in Appendix 1.

d) As Ki increases, the graph tends to settle closer to 1. Also, as Kp increases, the initial slope of the lines increases. But what about when Ki = 0?

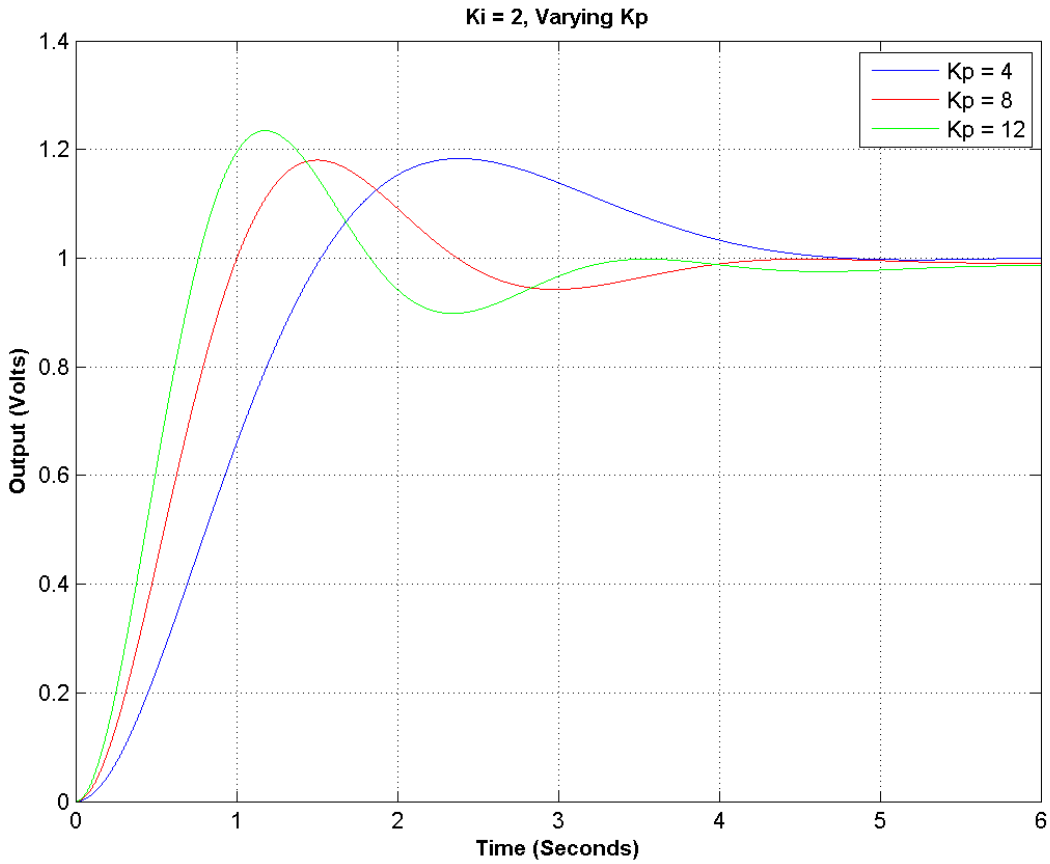
e) Plot 5 seems to be the best with $K_i = 2$ and $K_p = 8$ since it has a steep slope without rising very far above 1.

Appendix 1: Plots

Plot 1

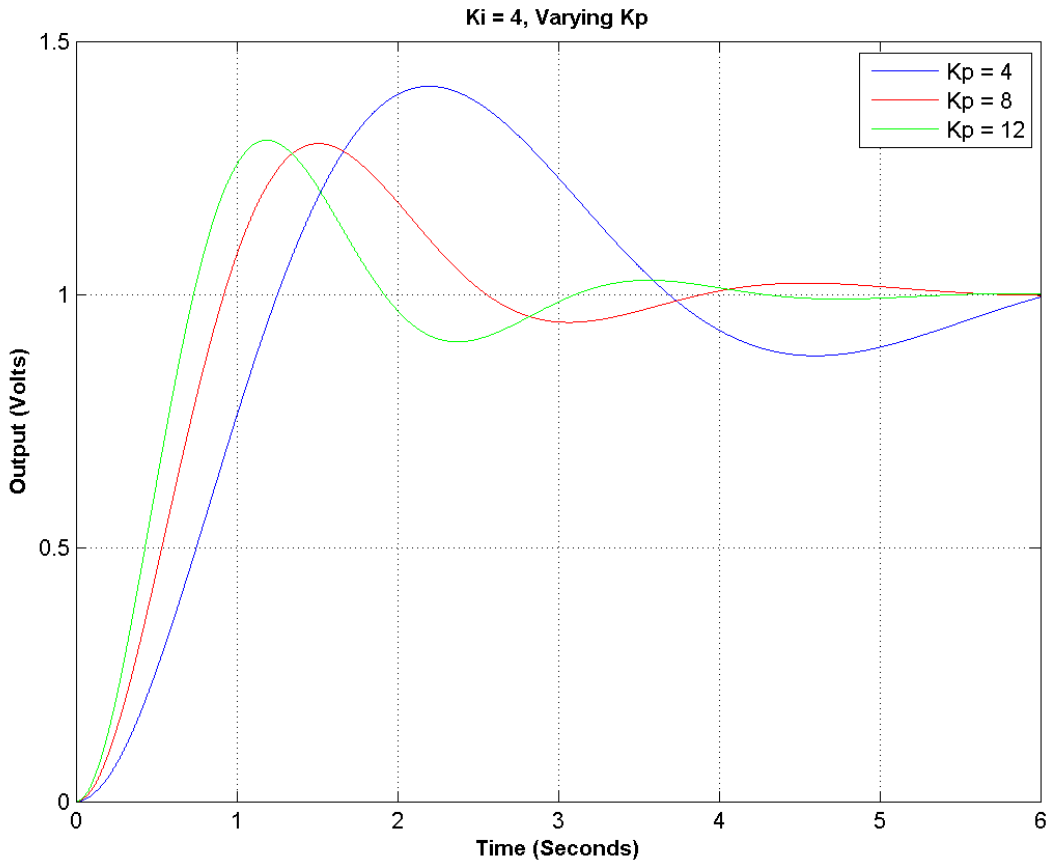


Plot 2



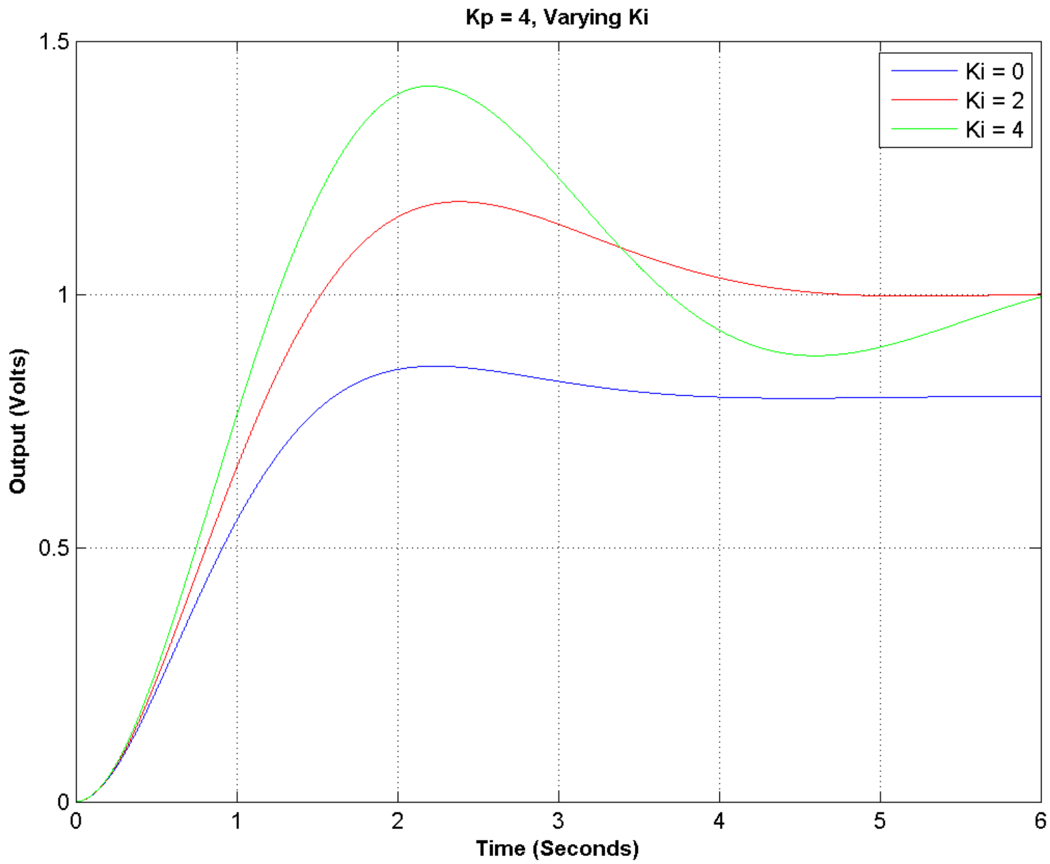
Ki = 2, Varying Kp
Kp = 4
Kp = 8
Kp = 12
Output (Volts)
Time (Seconds)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1
2
3
4
5
6

Plot 3



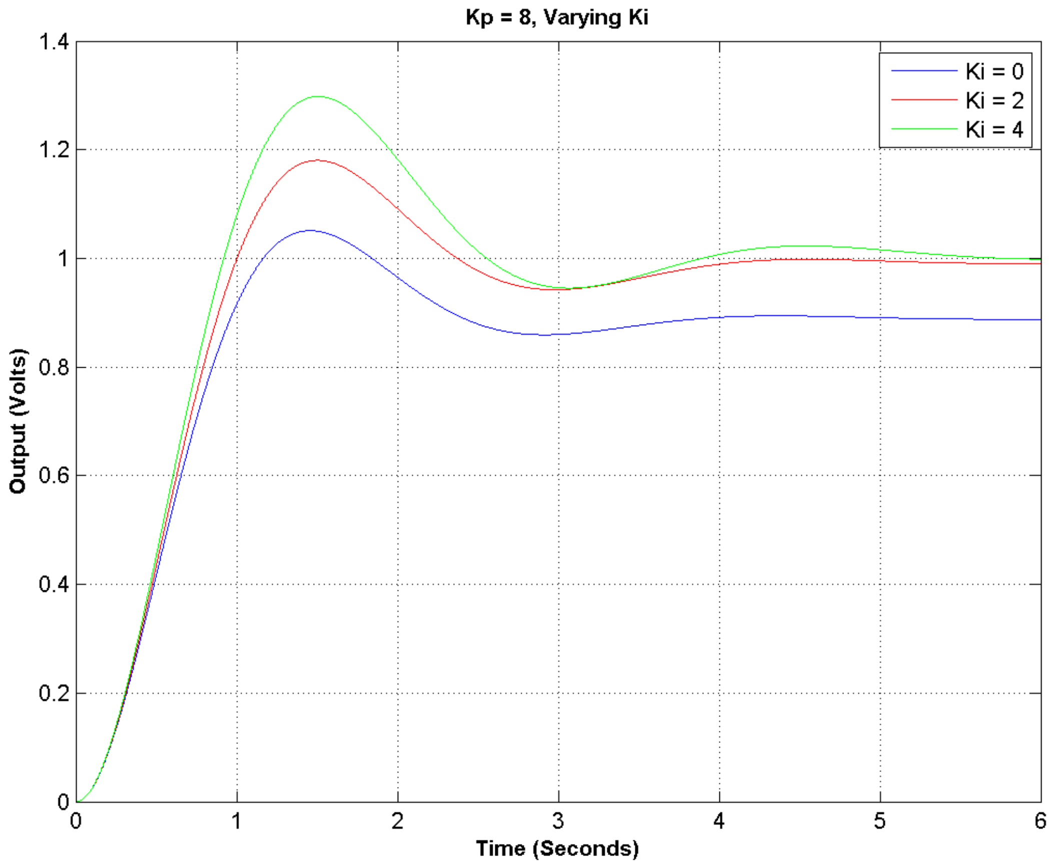
Ki = 4, Varying Kp
Kp = 4
Kp = 8
Kp = 12
Output (Volts)
Time (Seconds)
0
0.5
1
1.5
0
1
2
3
4
5
6

Plot 4

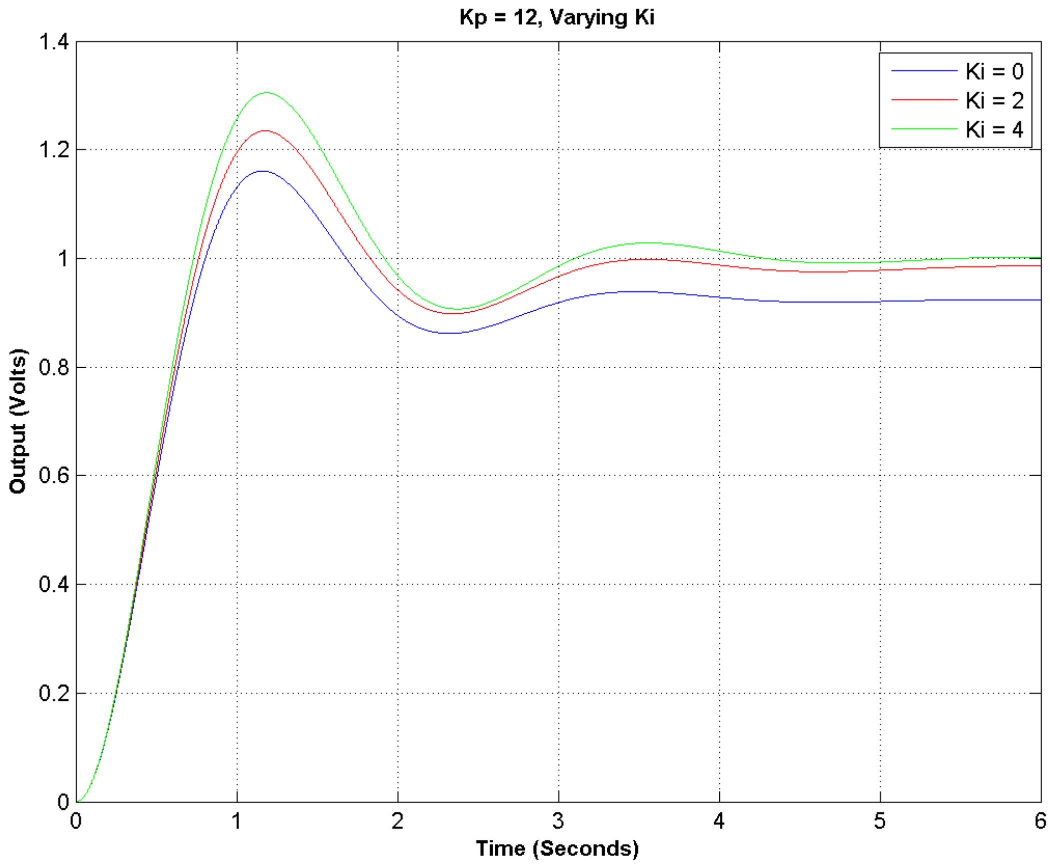


Kp = 4, Varying Ki
Ki = 0
Ki = 2
Ki = 4
1.5
1
0.5
0
Output (Volts)
0
1
2
3
4
5
6
Time (Seconds)

Plot 5


Kp = 8, Varying Ki
Ki = 0
Ki = 2
Ki = 4
Output (Volts)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
Time (Seconds)

Plot 6


Kp = 12, Varying Ki
Ki = 0
Ki = 2
Ki = 4
Output (Volts)
Time (Seconds)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1
2
3
4
5
6

Appendix 2: Code

```
#include <stdio.h>
#include <math.h>
#include <complex>
//local include files
#include "MatrixOutputs.hpp"
#include "Exp_A.hpp"

#ifdef _MSC_VER
#define _CRT_SECURE_NO_WARNINGS

using namespace std;

#define eps 2.22e-16

void main( void )
{
    int k;
    //char array for file name
    char Name[ 64 ];
    FILE *fout;
    matrix A, exp_Adt, x( 3 ), b( 3 );
    double dt, // Time Step
           TimeInterval = 6, // Time interval for simulationl
           gamma = 2.0e-2,
           epsilon = 2.0e-2,
           Tg = 3.0,
           Ts = 0.5,
           kp,
           ki,
           alpha = Tg*Ts,
           beta = Tg + Ts;

    // Allocate and set up system matrix.
    A = matrix( 3, 3 );
    // Check for valid allocation of A
    if ( A.isInvalid( ) )
    {
           printf( "Error allocating A\n" );
           getchar( );
           return;
    } // end of check for valid allocation of A

    // Load state variable matrix.
    for ( ki = 0; ki <= 4; ki += 2 )
    {
           for ( kp = 4; kp <= 12; kp += 4 )
           {
                  gamma = 1.0 + kp;
                  A( 0, 0 ) = 0.0;
                  A( 0, 1 ) = 1.0;
                  A( 0, 2 ) = 0.0;
                  A( 1, 0 ) = 0.0;
                  A( 1, 1 ) = 0.0;
                  A( 1, 2 ) = 1.0;
```

```
                A( 2, 0 ) = -ki / alpha;
                A( 2, 1 ) = -gamma / alpha;
                A( 2, 2 ) = -beta / alpha;
                // We will use the "exact" solution, exp( A * dt );
                // Note that the time step is pretty large.
                dt = 1.0e-3;
                exp_Adt = Exp_A( A * dt );

                // Check for A valid.
                if ( exp_Adt.isInvalid( ) )
                {
                       printf( "Error creating Exp_A for exact Solution\n" );
                       getchar( );
                       return;
                }
                printf( "Print out of matrix exp( A * dt )\n\n" );
                PrintMatrix( exp_Adt );

                // Having computed exp( A * dt ),
                // We now use it to simulate system.
                // Open file for output data, check for validity.
                //creates files with different names based on the ki and kp values
                sprintf( Name, "ki_%i_and_kp_%i.csv", (int)ki, (int)kp );
                fout = fopen( Name, "w" );

                if ( fout )
                {
                       x( 0 ) = 0.0; // initial values are zero.
                       x( 1 ) = 0.0;
                       x( 2 ) = 0.0;
                       // Input vector
                       b( 0 ) = 0.0;
                       b( 1 ) = 0.0;
                       b( 2 ) = ( kp + ki*Ts*dt ) / alpha; // This is the value of
the inputs
                       // at the start of the step.
                       // Loop through time interval
                       for ( k = 0.0; k < (int)( TimeInterval / dt ); k++ )
                       {
                              fprintf( fout, "%18.16lg, %18.16lg, %18.16lg,
%18.16lg\n",
                                     k*dt, x( 0 ), x( 1 ), x( 2 ) );
                              // Take step by A*x
                              x = exp_Adt * x + b;
                              if ( !k )
                                     b( 2 ) = ki*dt / alpha;
                       } // end of time step loop.
                       fclose( fout );
                } // end of valid file test for Exact Solution.
         }
    }
    printf( "\n Simulation complete:" );
    getchar( );
    return;
} // end of main
#endif
```

```
Project 4
   a) Written Derivation ...........  4/4
        Description of each step ...  3/3
      Correct Final ................  3/3

   b) Simulation ...................  5/5
       Using exp_A() ...............  5/5
       Looping through Kp,Ki's .....  5/5

   c) Plotting
       Labels, Titles, Legend ......  5/5
       Axis correct (Time) .........  5/5
       Proper Sequence .............  5/5

   d) Artifact .....................  2/5

   e) Optimum Arguement ............  3/3
       Case run ....................  2/2

Total ..............................  47/50
```