Hshmat Sahak, 1005903710
Juho Kim, 1005955085

ECE356 Lab 1

Part 3: Linear Algebra

Question 1a) Consult the MATLAB documentation to understand what each line means. Then run “inv matrix” in MATLAB and note the outputs.

Answer: The code in inv_matrix.m is pasted below:

```
n = 500;
Q = orth(randn(n,n));
d = logspace(0,-10,n);
A = Q*diag(d)*Q';
x = randn(n,1);
b = A*x;
tic, y = inv(A)*b; toc
err = norm(y-x)
res = norm(A*y-b)
pause
tic, z = A\b; toc
err = norm(z-x)
res = norm(A*z-b)
```

We will now describe what each line means:
Line 1: Set matrix dimension, an integer n, to 500
Line 2: Set Q to the orthogonalized matrix of size n*n by orthogonalizing an n*n matrix of random values
Line 3: Set d to an array of n logarithmically equally-spaced values from 10^0 to 10^-10
Line 4: Set A=Q*diag(d)*Q’, explicitly written in eigendecomposition form
Line 5: Set x to a random n-dimensional column vector. Will be used as correct/true value of inv(A)*b, as we will discover in future steps
Line 6: Set b = A*x. By setting a problem of form A*? = A*x, we see that the correct answer is obviously x.
Line 7: This line sets y = inv(A)*b, which should be x but the whole point is to determine numerical accuracy. Tic and toc are keywords in Matlab that will find elapsed time for computation and print this value to the command window
Line 8: This line computes the error, defined as the euclidean norm of y-x. Intuitively, “how far is y from x.”
Line 9: We want A*y = b, since we defined y as y = inv(A)*b. To compute how well Matlab did at solving the equation, determine the euclidean norm of A*y-b
Line 10: pause will halt the program until user input (I pressed <Enter> to continue program execution)
Line 11: Do the same thing as in Line 7, but now to determine the compute time of mldivide, not inv() function.
Line 12: Compute error of mldivide, same way as line 8

Hshmat Sahak, 1005903710
Juho Kim, 1005955085

Line 13: Compute residual of mldivide, same way as line 9

After understanding the code, we run "inv_matrix.m" on the command line and note the outputs. They are pasted below:

```
>> inv_matrix
Elapsed time is 0.020302 seconds.

err =

  7.0076e-06


res =

  5.0095e-07

Elapsed time is 0.010127 seconds.

err =

  6.1134e-06


res =

  4.1336e-15
```

Question 1b) Similarly, there is an operation called mrdivide or matrix right divide, A/B, which has the effect of AB^{-1} when B is invertible. Let A = randn(4,4). Compute the inverse of A using inv, mldivide, and mrdivide. Show your results to your TA. Explain how you would compare the accuracy of the three methods. For square matrices with low dimensions, all three calculations should give comparable accuracy.

Answer:
The inverse of A using inv can be found using: A_inv = inv(A)
The inverse of A using mldivide can be found using A_inv = A \ I
The inverse of A using mrdivide can be found using A_inv = A / I
where I is the 4*4 identity matrix in mldivide and mrdivide.

Since we do not know the correct/true value of A^{-1}, we can't directly compute an error between Matlab's computation of A_inv and the true A^{-1}. However, we can use the relation $A*A^{-1} = I$ to determine the accuracy of A_inv by computing the Frobenius norm of the difference $I - A*A^{-1}$ and $I - A^{-1}*A$.

Hshmat Sahak, 1005903710
Juho Kim, 1005955085

The output of our script for this question is shown below:

```
Part B
Method 1: Using inv

err1 =

   2.5962e-15


err2 =

   7.1855e-16

Method 2: Using mldivide

err1 =

   1.8216e-15


err2 =

   1.9493e-15

Method 3: Using mrdivide

err1 =

   2.5962e-15


err2 =

   7.1855e-16
```

Note: If we wanted to use a matrix for which we know the inverse, we could construct Q = orth(rand(n,n)). The inverse of an orthogonal matrix (orthonormal columns) is its transpose, so we could compute error (Q^T-Q_inv) too, rather than just residual (I - Q*Q^T / I - Q^T*Q).

Question 2a)

$$A = \begin{bmatrix} 7 & 2 & -3 \\ 4 & 6 & -4 \\ 5 & 2 & -1 \end{bmatrix}$$

Hshmat Sahak, 1005903710
Juho Kim, 1005955085

.
Use the MATLAB command [V, D] = eig(A) to determine the eigenvalues and eigenvectors of A.

Answer: The outputs of our script is shown below:

```
Part A
Eigenvectors and Eigenvavlues of A:
V =

    0.2673    0.5774    0.7071
    0.5345    0.5774   -0.0000
    0.8018    0.5774    0.7071


D =

    2.0000         0         0
         0    6.0000         0
         0         0    4.0000
```

Question 2b) For the matrix in II(a), you can use the command eig(A,'nobalance') to produce more "readily recognizable" eigenvectors. Recall that eigenvectors are determined up to a scalar multiple. From the results of eig(A,'nobalance'), determine 3 eigenvectors of A whose entries are all integers.

Answer:
First, we will paste the output of the line [V_nb, D_nb] = eig(A,'nobalance'); this is shown below:

```
V_nb =

    0.2673    0.5774    0.7071
    0.5345    0.5774   -0.0000
    0.8018    0.5774    0.7071


D_nb =

    2.0000         0         0
         0    6.0000         0
         0         0    4.0000
```

Now, to determine 3 eigenvectors of A whose entries are all integers, we scale the columns of V_nb by $\sqrt{14},\ \sqrt{3}\ and\ \sqrt{2}$. This yields:

Hshmat Sahak, 1005903710
Juho Kim, 1005955085

```
Integer eigenvector associated with eigenvalue 2.000000:
v1 =

     1
     2
     3

Integer eigenvector associated with eigenvalue 6.000000:
v2 =

     1
     1
     1

Integer eigenvector associated with eigenvalue 4.000000:
v3 =

     1
     0
     1
```

Question 2c) Determine the characteristic polynomial of A using the command poly(A), and determine the eigenvalues by applying the command roots to the resulting polynomial. Compare the answer with those from (a).

Answer: The output of our script for this section is:

```
Part C
Characteristic polynomial of A:
p =

    1.0000  -12.0000   44.0000  -48.0000

Solutions (i.e. eigenvalues of A):
r =

    6.0000
    4.0000
    2.0000
```

The characteristic polynomial, as realized by observing output of poly(A) is $\lambda^3 - 12\lambda^2 + 44\lambda - 48$.The roots of this equation, determined using the roots command in Matlab, are 6, 4, and 2. These are the same roots obtained in part a, which contained the roots as elements of the diagonal matrix D.

Hshmat Sahak, 1005903710
Juho Kim, 1005955085

Question 2d) From the results of IV(a), determine norm(AV-VD). Show more significant digits in MATLAB using the command format long. From this determine the exact values of the eigenvalues and eigenvectors (up to a scalar multiple). Now verify that AV − V D = 0.

Answer:
The value of norm(AV-VD) is 2.453379040363149e-15. Since this is very close to 0, the exact values of eigenvectors are the vectors of integer components found in part b. Furthermore, the eigenvalues matrix is the diagonal matrix with entries 6, 4, and 2 along the diagonal. The output of our script for this section is shown below. We have verified that AV-VD = 0, when V and D have integer entries.

```
Part D
Difference norm:
n =

   2.453379040363149e-15

Exact values of eigenvectors and eigenvalues:
V_e =

   1    1    1
   2    1    0
   3    1    1


D_e =

   2    0    0
   0    6    0
   0    0    4

Difference norm:
n_e =

   0
```

Question 2e) Check that the matrix in II(a) can be diagonalized.

Answer: The output of V \ A * V, which is used to compute the diagonal matrix is shown below:

```
Part E
Diagonalization:
D_diag =
```

Hshmat Sahak, 1005903710
Juho Kim, 1005955085

```
   1.999999999999999   0.000000000000001   0.000000000000002
  -0.000000000000002   5.999999999999999  -0.000000000000001
  -0.000000000000001  -0.000000000000002   4.000000000000000
```

Clearly, this rounds to the diagonal matrix with entries 2, 6, 4 along its diagonal. These are the same as eigenvalues of the matrix, so the matrix is diagonalizable.

Question 2f) Determine by hand calculation the eigenvector corresponding to the eigenvalue 1, and verify there is only one independent eigenvector. Try using eig on this A. Does the resulting [V,D] satisfy AV = V D? Is V invertible?

Answer:
Hand calculation to show just one eigenvector with eigenvalue 1 is shown here:
Let a,b be any real numbers. We want to find particular a,b that satisfy:

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\implies \begin{bmatrix} a+b \\ b \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}$$
$$\implies b = 0$$

So, eigenvectors are of form
$a\begin{bmatrix} 1 \\ 0 \end{bmatrix}$, ie scalar multiples of $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

The output of our script is shown below:

```
Part F
Eigenvectors and Eigenvalues of A:
V =

   1.000000000000000  -1.000000000000000
                   0   0.000000000000000


D =

     1     0
     0     1

Difference norm:
n =
     2.220446049250313e-16
```

Hshmat Sahak, 1005903710
Juho Kim, 1005955085

As we can see, AV - VD = 0 is satisfied, since norm(AV - VD) is nearly 0, and any difference is because of the underlying MATLAB software. Note that we could get norm(AV-VD)=0 exactly if we rounded the matrices to integer-valued matrices, as we did in part d. However, V is not invertible- it has a 0 row, and its determinant is 0.

Question 2g)

$$\text{For the matrix A} = \begin{bmatrix} 0 & 4 & 3 \\ 0 & 20 & 16 \\ 0 & -25 & -20 \end{bmatrix}, \text{ find its Jordan form}$$

Answer: This was done using the jordan command in Matlab. The output of the Matlab script is shown below:

```
Part G
Jordan Form:
J =

   0   1   0
   0   0   1
   0   0   0
```

Hshmat Sahak, 1005903710
Juho Kim, 1005955085

Part 4: Ordinary Differential Equations and Transfer Functions

Question 4a) Create the sys object corresponding to the second-order system described by (4)-(5).

Answer: From equations (4) and (5), we can obtain the values of matrices A, B, C, and D. We then create the sys object using the command sys(A, B, C, D). The output is shown below.

```
Part A
sys =

  A =
      x1  x2
   x1   0   1
   x2  -4  -2

  B =
      u1
   x1   0
   x2   4

  C =
      x1  x2
   y1   1   0

  D =
      u1
   y1   0

Continuous-time state-space model.
```

Hshmat Sahak, 1005903710
Juho Kim, 1005955085

Question 4b) Here, determine the step response using the command [Y,T,X]=step(sys), where sys is the object you created using the ss command. Use plot(T,X) to plot the trajectories of both states.

Answer: The plot of trajectories is shown below. The blue plot shows the value of x_1 = y(t). The red plot shows the value of y'(t). The response of the system to the step input is that y(t) will start at 0, rise and overshoot above steady state value, then decay to some steady state value.

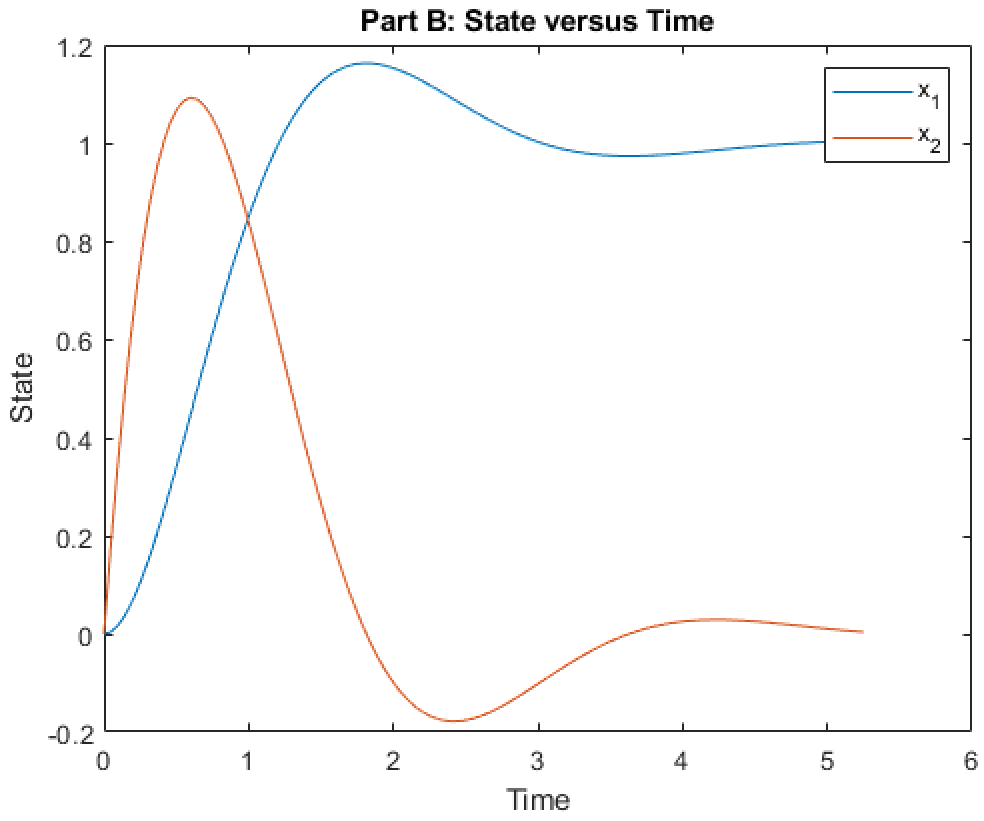

Figure 1: State-Space Diagram for Question 4b

Hshmat Sahak, 1005903710
Juho Kim, 1005955085

Question 4c) Use the command [Y,T,X]=initial(sys init,x0) to determine the response to initial conditions when:

$$x0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Again plot the state responses.

Answer: The plots of the state variables with the given initial conditions is shown below. It is easy to verify visually that both initial conditions are satisfied; i.e., $x_1(0) = y(0) = 0$ and $x_2 = y'(0) = 1$. Furthermore, we see that the initial conditions "die away", i.e. the system reaches a steady state value of 0 after some initial rise and fall.

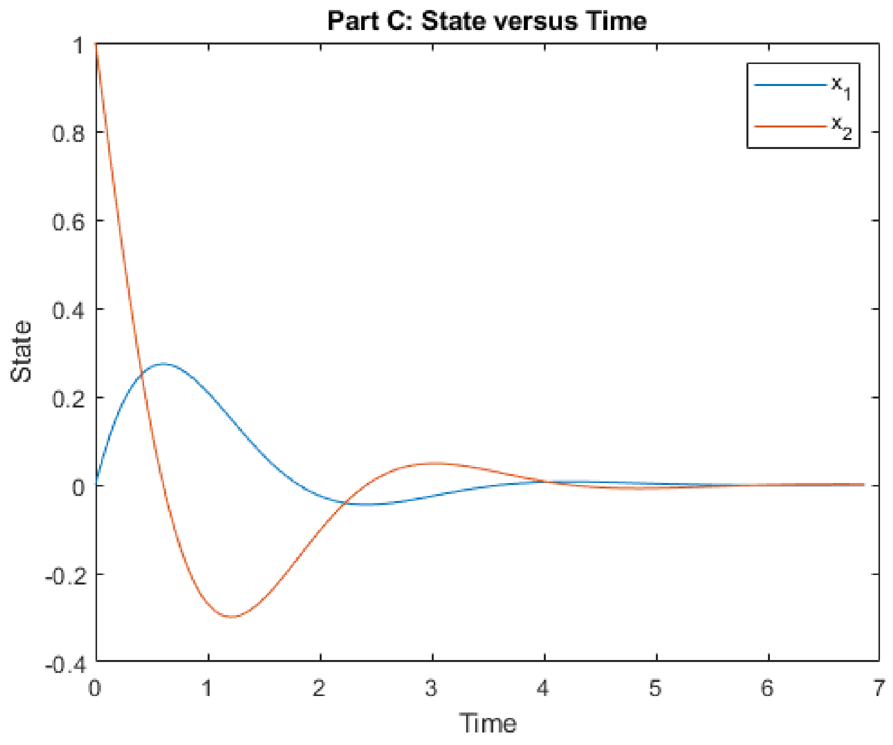

Figure 2: State-Space Diagram for Question 4c

Hshmat Sahak, 1005903710
Juho Kim, 1005955085

Question 4d) Use [Y,t,X]=lsim(sys,u,t,x0) to produce the response of the second-order system due to the initial condition and sinusoidal input defined above. Plot the state trajectories.

Answer: The plot of state trajectories given initial conditions is shown below. It is easy to see that initial conditions are met; i.e., $x_1(0) = y(0) = 0$ and $x_2(0) = y'(0) = 1$. Furthermore, as time passes, the effects of initial conditions "die away" again and the output y(t) matches the input. This is expected behavior.

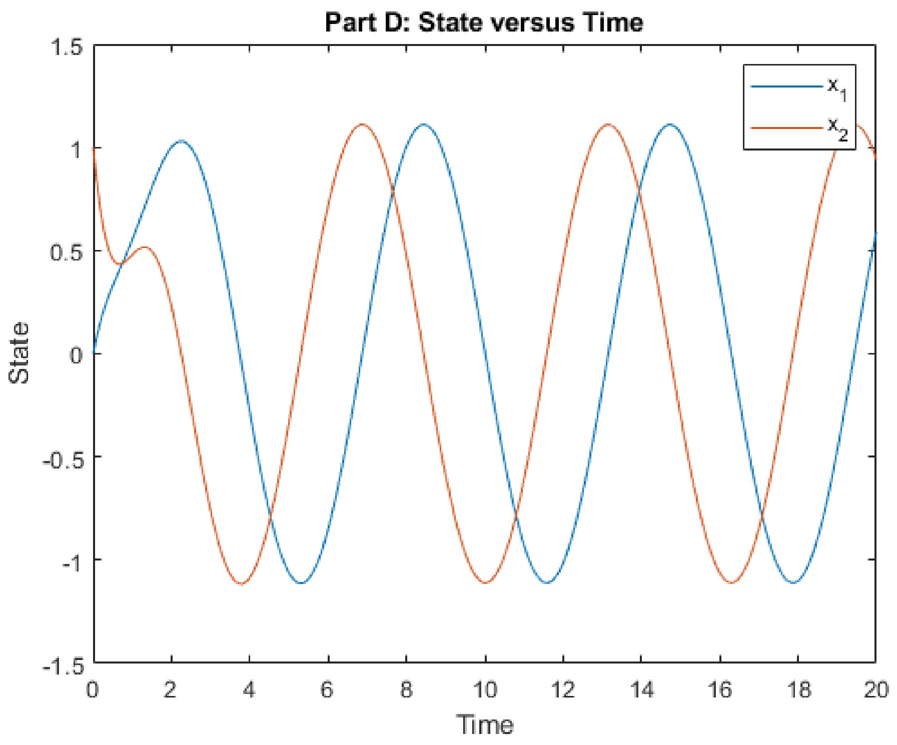

Figure 3: State-Space Diagram for Question 4d

Question 4e) Use the ss2tf command, followed by the tf command to produce the transfer function for the systems described by (4)-(5). Verify that it is the same as (3).

Answer: Using ss2tf and tf functions as described, we obtain the following output from our MATLAB script:

```
Part E
Transfer function:
f =

      4
  -------------
  s^2 + 2 s + 4

Continuous-time transfer function.
```

This is the same as (3), as expected and desired.

Hshmat Sahak, 1005903710
Juho Kim, 1005955085

Part 5: Introduction to Simulink

For each of the parts below Simulink models was constructed. The corresponding files are uploaded alongside this document. The plots of the outputs (and states if applicable) of each of the parts are shown below:

Question 5a) Bring in the state space block and use the same (A, B, C, D) matrices as in (4)-(5). Connect to the input port a unit step. Change the start time of the step to 0. Connect the output port to a scope. Click the “simulate” button and examine the output on the scope.

Answer:
An image of the state-space model, constructed using Simulink, is shown below:

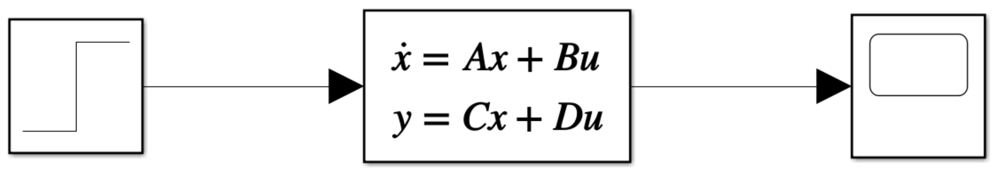

Figure 4: Simulink Model for Question 5a

The output of the scope is shown below:

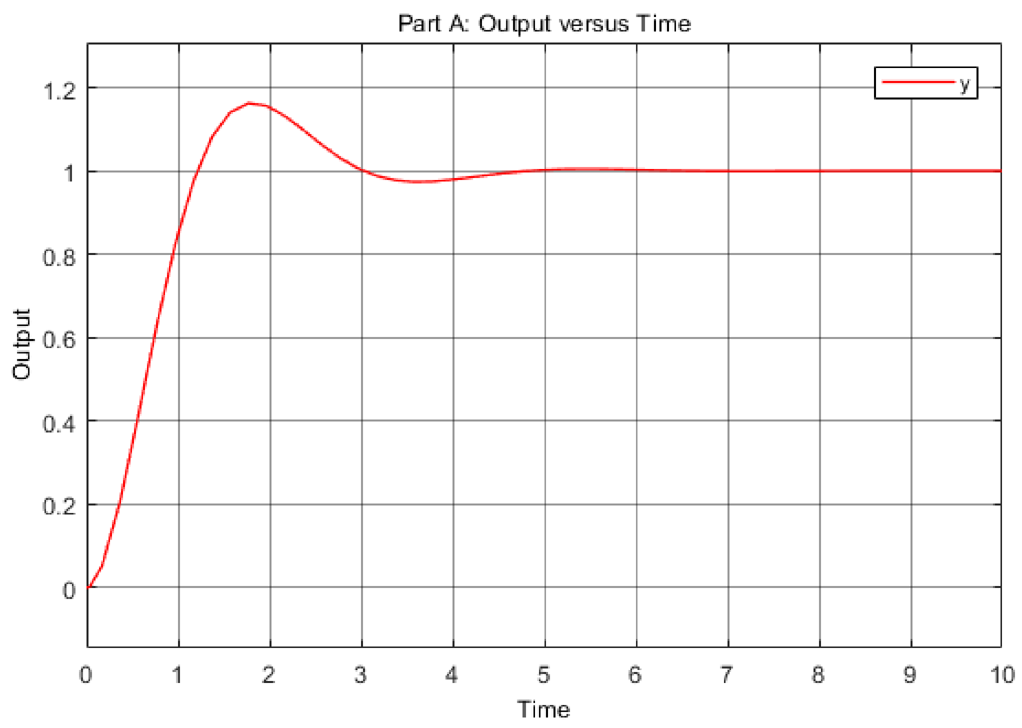

Figure 5: Output Diagram for Question 5a

Hshmat Sahak, 1005903710
Juho Kim, 1005955085

Question 5b) One trick to overcoming this limitation is to change the C matrix in the state space block to the identity so that the outputs are the states. Now connect the states to a gain matrix C. Note that by default, the gain matrix multiplies the input element-wise. Finally, connect the output to the scope and check that the simulation gives the same output as in (a). Connect the states from the output of the state space block to another scope, and check that one of the 2 signals displayed is the same as the output y.

Answer:
An image of the state-space model, constructed using Simulink, is shown below:

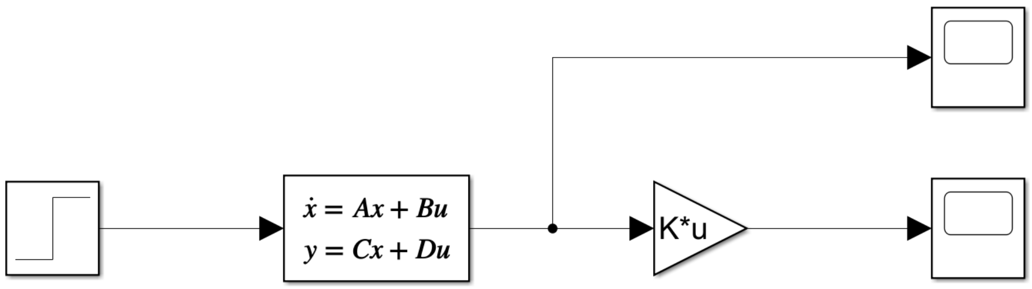

Figure 6: Simulink Model for Question 5b

Plots for output and state components with time are shown below:

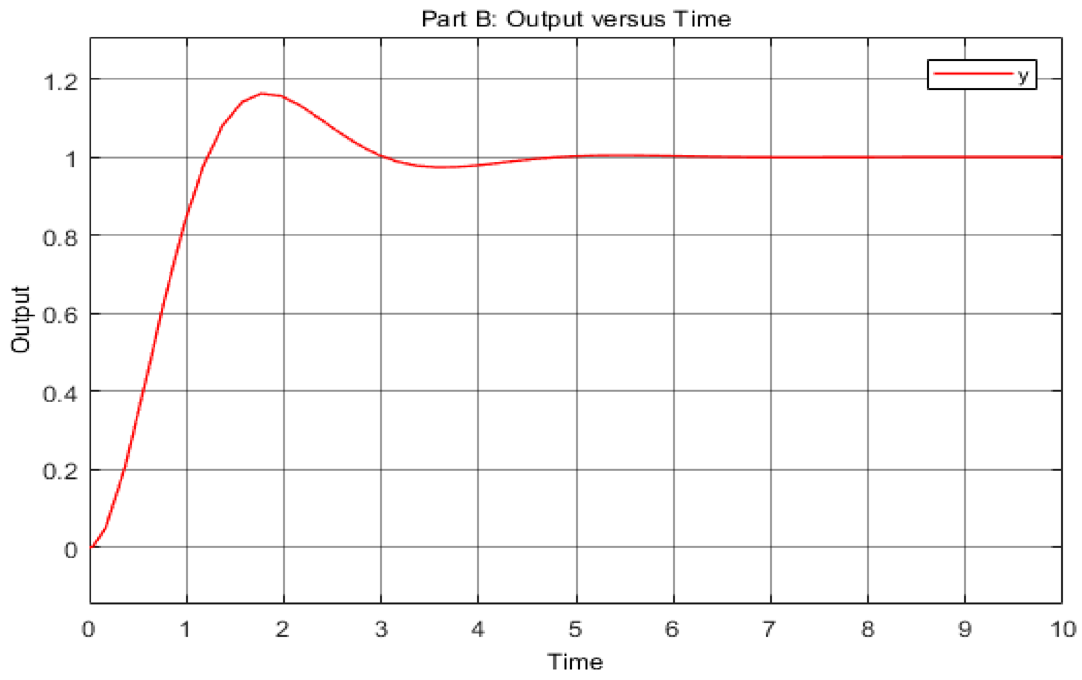

Figure 7: Output Diagram for Question 5b

We can verify visually that the output, y(t), is the same as in part a.

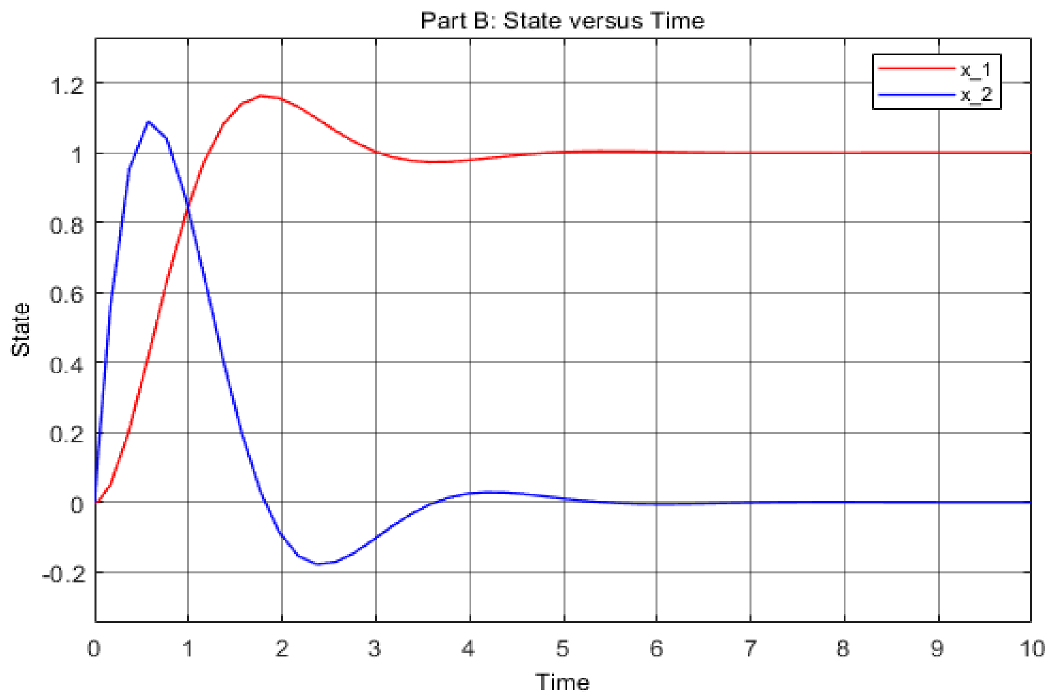

Figure 8: State-Space Diagram for Question 5b

The red signal, $x_1(t)$, is the same as the plot in Figure 5. So we have verified that one of the 2 signals here is the same as the output y(t).

Question 5c) To represent the signals in a state-space system in terms of more basic components, start a new model and bring in the integrator block. The input to the integrator block is $\dot{x} = Ax + Bu$ and the output is x. The right-hand side for $\dot{x}$ requires you to put in some gain matrices and a sum block so that the output of the sum block, $\dot{x}$, satisfies the state equation. The output of the integrator block, x, can now be connected to the gain matrix C, whose output goes into a scope. With the input of a unit step, check that the output on the scope is again the same as those in (a).

Answer:
An image of the state-space model, constructed using Simulink, is shown below:

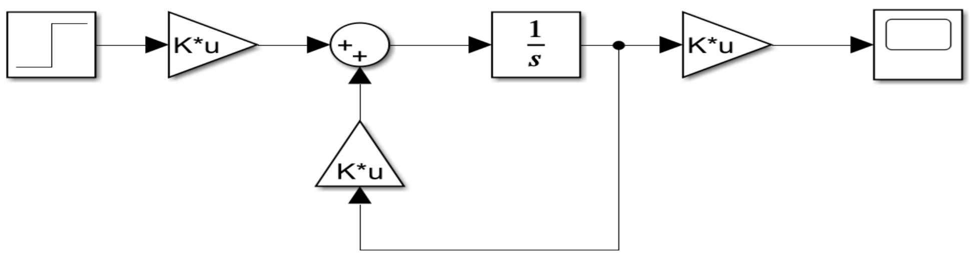

Figure 9: Simulink Model for Question 5c

A plot of the output y(t) with time is shown below. The output of the scope is again the same as part a.

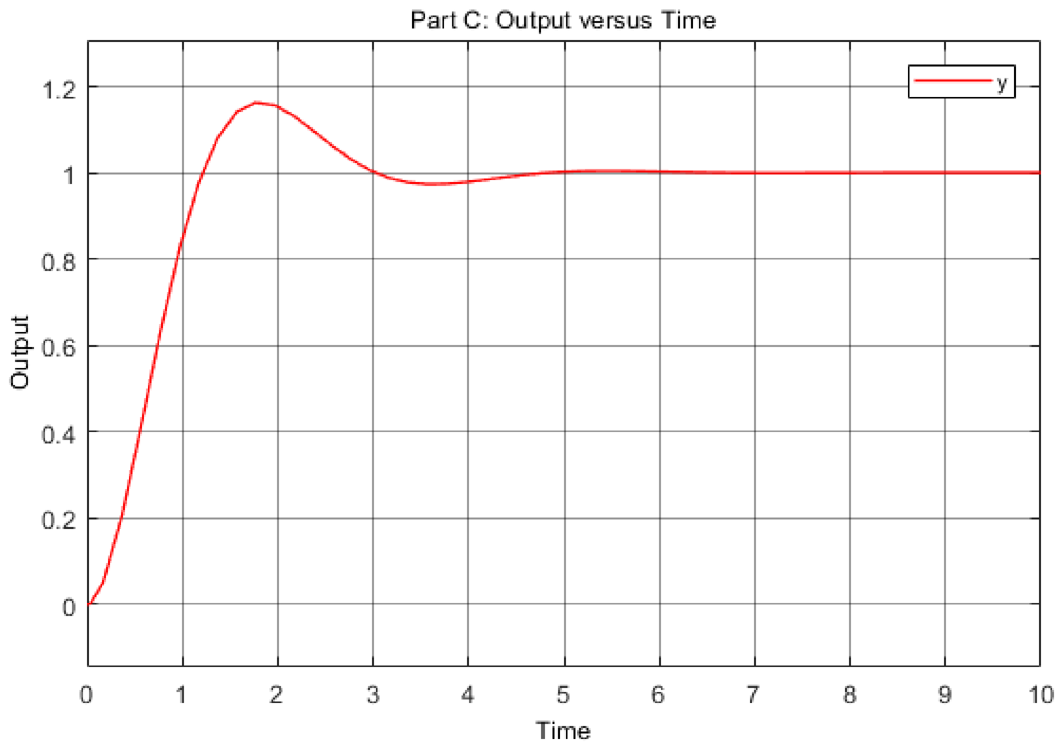

Figure 10: Output Diagram for Question 5c

Question 5d) To understand how this is done, in a new Simulink model, bring in 2 integrator blocks. For the system described by (4)-(5), the input to the first integrator block should be x˙ 2. Its output is x2 = ˙x1, which is the input to the second integrator block. Using scalar gains, complete the Simulink model so that the step response shown in the scope is again the same as that of (a). Show all your Simulink models and output scope displays from (a) to (d) to your TA.

Answer:
An image of the state-space model, constructed using Simulink, is shown below:

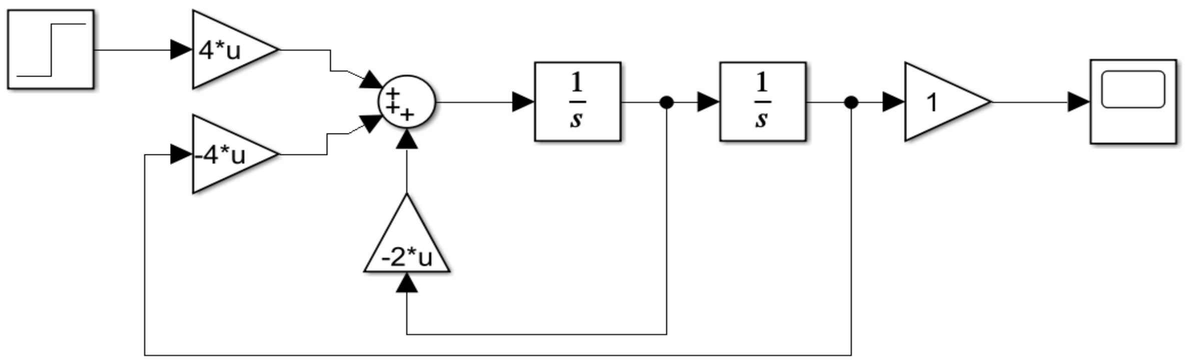

Figure 11: Simulink Model for Question 5d

Hshmat Sahak, 1005903710
Juho Kim, 1005955085

A plot of the output y(t) with time is shown below. The output of the scope is again the same as part a.

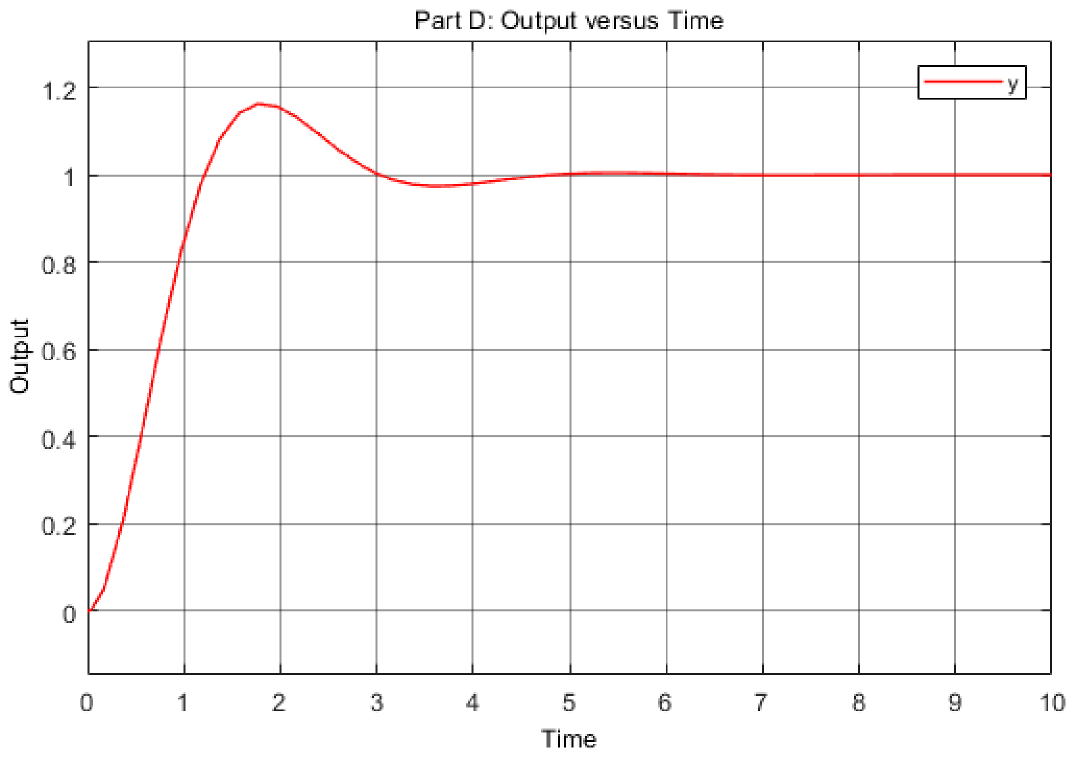

Figure 12: Output Diagram for Question 5d