ECSE 343: Numerical Methods in Engineering

Assignment 4

Due Date: 6th December 2021

Student Name:Lawi Mwirigi

Student ID: 260831614

Please type your answers and write you code in this .mlx script. If you choose to provide the handwritten answers, please scan your answers and include those in SINGLE pdf file.

Please submit this **.mlx** file along with the **PDF** copy of this file.

Question 1:

Your task is to find the solution for the following system of differential equations

$$\frac{d}{dt}(\boldsymbol{x}(t)) = \boldsymbol{A}\,\boldsymbol{x}(t) \qquad (1)$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$ (i.e. $\mathbf{A}$ is a $N \times N$ matrix) and $\boldsymbol{x} \in \mathbb{R}^{N\times N}$ (i.e. $\boldsymbol{x}$ is a $N \times 1$ vector)

We are also given the initial conditions,

$$\boldsymbol{x}(0) = \boldsymbol{X}_0 \qquad (2)$$

If the matrix A, has unique and real eigen values then the solution for the above equations can be written as,

$$\boldsymbol{x}(t) = c_1 e^{-\lambda_1 t}\boldsymbol{v}_1 + c_2 e^{-\lambda_2 t}\boldsymbol{v}_2 + \ \ldots \ + c_n e^{-\lambda_n t}\boldsymbol{v}_n \qquad (3)$$

The $\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n$ are the eigen vectors of matrix ***A*** and λ_1, λ_2, ..., λ_n are the eigen vectors. The eigen values and the eigen vectors of the matrix ***A*** can be computed using the eigen value decomposition given below,

$$\boldsymbol{A} = \boldsymbol{V}\,\Lambda \boldsymbol{V}^{\boldsymbol{T}} \qquad (4)$$

where the matrix $\boldsymbol{V}$ contains the eigen vectors as the columns, the matrix Λ is a diagonal matrix containing the eigen values. The equation (3) can be written as

$$A = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \dots & \boldsymbol{v}_{n-1} & \boldsymbol{v}_n \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 & 0 \\ 0 & \lambda_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \lambda_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & \lambda_n \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \dots & \boldsymbol{v}_{n-1} & \boldsymbol{v}_n \end{bmatrix}^T$$

a) Use the approach described in (3) to find the solution of the system of Ordinary Differential Equations described in (1).

Use this approach in order to find the solution and plot the first entry of the x vector as a function of time from 0 to 6seonds.

Use at least 500 time points in order to get smooth plot. You may use the matlab function eig in order to compute the eigenvalues and eigenvectors.

```
clear all

clf
load ODE_Data %load the matrix A and initial condition X0
A;
X0;

% Your code to compute x goes here.
[V,D] = eig(A);
V;
D
```

```
D = 5×5
10^4 ×
   -5.0000         0         0         0         0
         0   -0.0100         0         0         0
         0         0   -0.0010         0         0
         0         0         0   -0.0001         0
         0         0         0         0   -0.0005
```

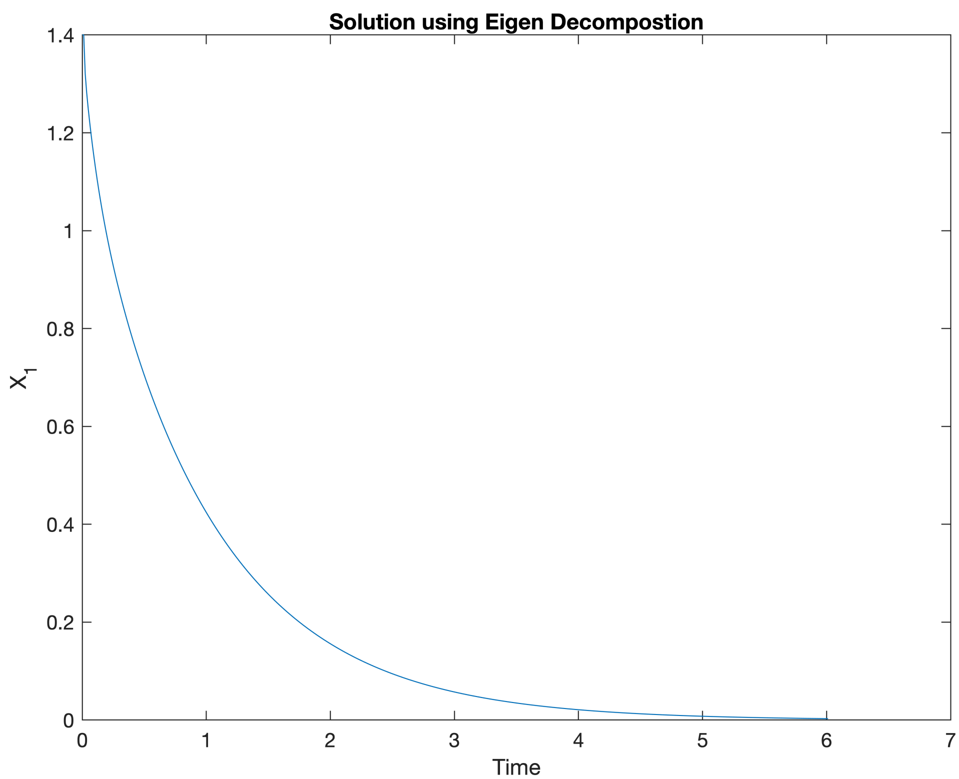
```
ans = D* [1 0 0 0 0;1 0 0 0 0; 1 0 0 0 0;1 0 0 0 0;1 0 0 0 0]
```

```
ans = 5×5
10^4 ×
   -5.0000         0         0         0         0
   -0.0100         0         0         0         0
   -0.0010         0         0         0         0
   -0.0001         0         0         0         0
   -0.0005         0         0         0         0
```

```

%how to obtain the constants
C=V^-1*X0;
i=1;
Xeigen(:,i)=X0
```

```
Xeigen = 5×1
    1.4000
   -3.5000
   -1.6000
   -1.0000
    1.5000
```

```
t=0;

% Name your soltuion vector as Xeigen
while t<6
    tpoints(i+1)=i*0.012;
    Xeigen(:,i+1)=(C(1)*(exp(ans(1)*t))*V(:,1))+(C(2)*(exp(ans(2)*t))*V(:,2))+(C(3)*(ex

    i=i+1;
    t=t+0.012;
end

% Plot the first entry of the x vector
plot(tpoints,Xeigen(1,:))
xlabel('Time')
ylabel('X_1')
title('Solution using Eigen Decompostion')
```



b) Another approach for numerically finding the solution is to use the Backward Eurler (BE) integration formula.

Using the Backward Euler aprroximation, the derivative at n^{th} time step can be written as,

$$\frac{d}{dt}(\boldsymbol{x}_n) = \frac{\boldsymbol{x}_n - \boldsymbol{x}_{n-1}}{\Delta t}$$

Using the above approximation and write the resulting system of algabaraic equations here.

% Write your solution here

$\frac{d}{dt}(\boldsymbol{x}_n)*\ \Delta t = \boldsymbol{x}_n - \boldsymbol{x}_{n-1}$

remember $\frac{d}{dt}(\boldsymbol{x}(t)) = \boldsymbol{A}\,\boldsymbol{x}(t)$ from 1

and $\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \frac{d}{dt}(\boldsymbol{x}_n)*\Delta t$ therefore replace $\frac{d}{dt}(\boldsymbol{x}(t))$ **by** $\boldsymbol{A}\,\boldsymbol{x}(t)$ to obtain

$\boldsymbol{x}_n = \Delta t\,\boldsymbol{A}\,\boldsymbol{x}(n) + \boldsymbol{x}_{n-1}$ to obtain $\boldsymbol{x}_n - \Delta t\,\boldsymbol{A}\,\boldsymbol{x}(n) = \boldsymbol{x}_{n-1}$ factor out $\boldsymbol{x}_n$(I-$\Delta t\,\boldsymbol{A}$) = $\boldsymbol{x}_{n-1}$ therefore multiply by (I - $\Delta t\,\boldsymbol{A}$)^-1 to obtain

$\boldsymbol{x}_n = (I - \Delta t\,A)^{-1} * \boldsymbol{x}_{n-1}$

c) Using the equation found in part (b) using Backward Euler formula. Use atleast 500 timepoints (i.e. choose a suitable time step, Δt , such that you have atleast 500 time points) to find the solution between 0 and 6 seconds.

Note: The analytical solution shown in (1) is for the matrices which have unique and real eigen values. However, the Backward Euler or Forwad Euler method can be used for solving any general system of ODE's.

```
% clear, clc
% % Model Parameters
% T = 5;
% a = -1/T;
% % Simulation Parameters
% Ts = 0.1; %s
% Tstop = 30; %s
% x(1) = 1;
% % Simulation
% for k=1:(Tstop/Ts)
% x(k+1) = (x(k)-(x(k)*Ts)/T);
%ble(:,i)-
% end
% % Plot the Simulation Results
```

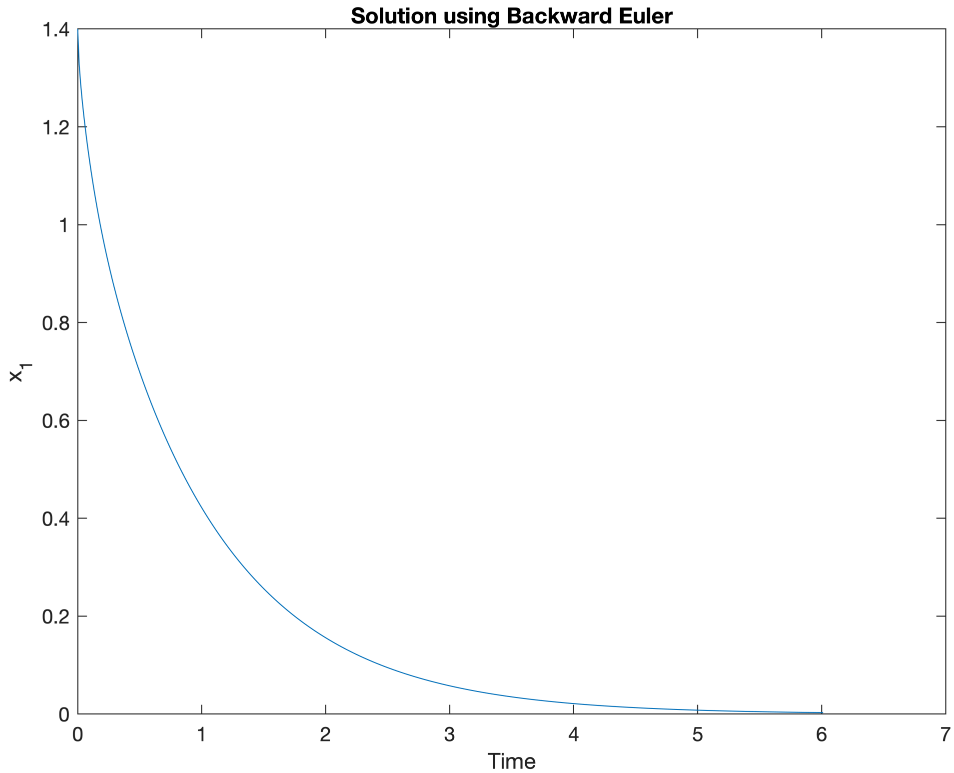
```
% t=0:Ts:Tstop;
% plot(t,x)
% grid on

clear all

load ODE_Data %load the matrix A and initial condition X0
A
```

```
A = 5×5
10^4  ×
   -0.2863   -0.4402   -0.0966    1.0275   -0.2488
   -0.4402   -0.6837   -0.1524    1.6050   -0.3868
   -0.0966   -0.1524   -0.0358    0.3625   -0.0860
    1.0275    1.6050    0.3625   -3.7865    0.9075
   -0.2488   -0.3868   -0.0860    0.9075   -0.2192
```

```
X0
```

```
X0 = 5×1
    1.4000
   -3.5000
   -1.6000
   -1.0000
    1.5000
```

```
% Name your time points as t_be  and  Name your solution vector as Xbe.
i=1;
ble(:,i)=X0;
tube(1)=0;
t=0;
while t<6

    tube(i+1)=i*0.012;
    ble(:,i+1)=(eye(5)-(0.012*A))^-1*ble(:,i);
    i=i+1;
     t=t+0.012;
end



% Plot the first entry of the x vector.
figure()
plot(tube,ble(1,:))
xlabel('Time')
ylabel('x_1')
title('Solution using Backward Euler');
```



d) In this part use the Forward Euler approximation to compute the solution of (1). The Forward Euler approximation for the derivative at n^{th} time step can be written as,

$$\frac{d}{dt}(\boldsymbol{x}_n) = \frac{\boldsymbol{x}_{n+1} - \boldsymbol{x}_n}{\Delta t}$$

Using the above approximation and write the resulting system of algabaraic equations here.

% Type your solution here.

$$\frac{d}{dt}(\boldsymbol{x}_n) * \Delta t = \boldsymbol{x}_{n+1} - \boldsymbol{x}_n$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \frac{d}{dt}(\boldsymbol{x}_n) * \Delta t$$

$$\boldsymbol{x}_{n+1} = \Delta t \, A(\boldsymbol{x}_n) + \boldsymbol{x}_n$$

e) Find the solution vector $\boldsymbol{x}(t)$, using the algebraic equation obtained using Forward Euler formula. Use at least 500 time points to find the solution between 0 and 6 seconds.

```
clear all

clf

load ODE_Data %load the matrix A and initial condition X0
A
```

```
A = 5×5
10^4 ×
   -0.2863   -0.4402   -0.0966    1.0275   -0.2488
   -0.4402   -0.6837   -0.1524    1.6050   -0.3868
   -0.0966   -0.1524   -0.0358    0.3625   -0.0860
    1.0275    1.6050    0.3625   -3.7865    0.9075
   -0.2488   -0.3868   -0.0860    0.9075   -0.2192
```

```
X0
```

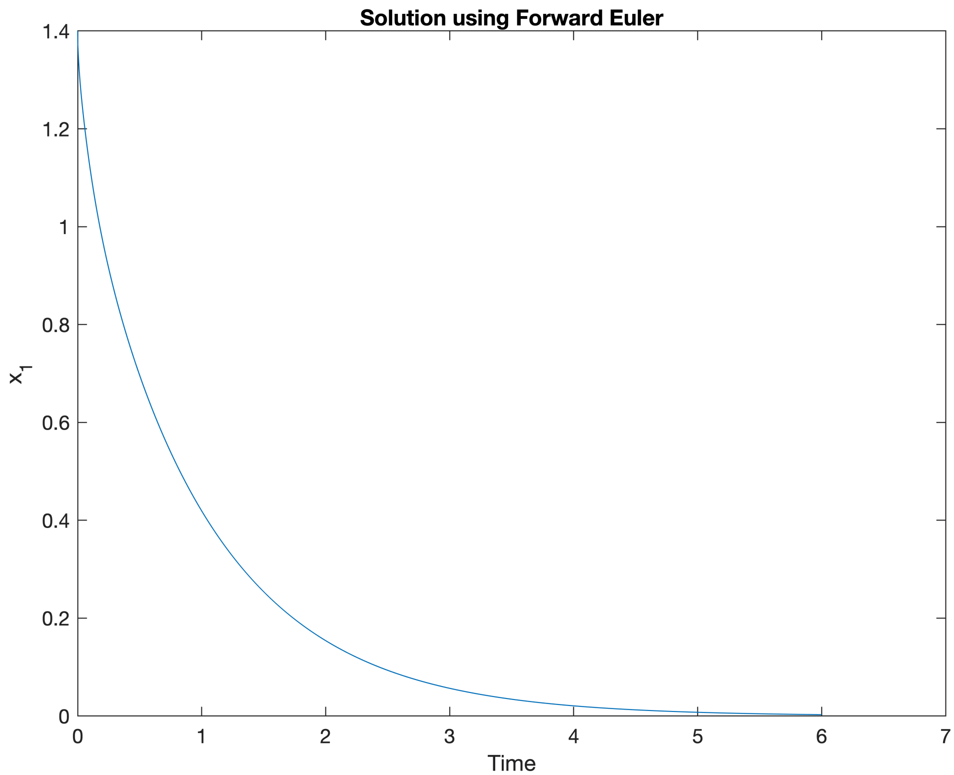
```
X0 = 5×1
    1.4000
   -3.5000
   -1.6000
   -1.0000
    1.5000
```

```
 %Name your time points as t_fe  and  Name your solution vector as Xfe.


Xfe(:,1)=X0
```

```
Xfe = 5×1
    1.4000
   -3.5000
   -1.6000
   -1.0000
    1.5000
```

```

tpointsfe(1)=0;
   i=1;
   t=0;
 while t<6

     tpointsfe(i+1)= i * 0.000005;

     Xfe(:,i+1)= Xfe(:,i) + 0.000005 * A * Xfe(:,i);

     i=i+1;
     t=t+0.000005;
end

% Plot the first entry of the x vector
figure()
plot(tpointsfe,Xfe(1,:))
```

```
xlabel('Time')
ylabel('x_1')
title('Solution using Forward Euler')
```



e) Comment on the results of B.E and F.E in parts 2 and 3 above. Which one would you use for this example and why?

They both obtain the same results. However Backward Eurler (BE) in B has a step size of 0.012 therefore just looping 600 times.For the Forward Euler approximation i started with a step size of 0.012 which gave me a wrong answer (the graph was a inverted staright line axis sort of) , then i kept on decresing the step size and untill my step size got to 0.000005 implying it is much slower.

Therefore I would use backward Eurler method.