Laboratory 1

Helen Du

SID: 862081856

TA: Wang Hu

January 12, 2021

MODELING AND SIMULATION OF DYNAMIC SYSTEMS EE 105

SECTION 023

Objective:

The purpose of this lab is a familiarization with the usage of MATLAB and the usage of MATLAB *help* facilities for students to teach themselves how to learn about MATLAB.

MATLAB Tutorial:

Matrices and Arrays

Use matrix operations to find the matrix C where the superscript T denotes a matrix transpose.

```
% A = [pi; sqrt(2); exp(1)] (1)
% B = [1; 5; 7]             (2)
% C = A^(T)B                (3)
```

```
A = [pi; sqrt(2); exp(1)]
```

```
A = 3×1
    3.1416
    1.4142
    2.7183
```

```
B = [1; 5; 7]
```

```
B = 3×1
     1
     5
     7
```

```
C= A.'*B
```

```
C = 29.2406
```

Scripts

Write a script that clears the memory, defines the matrices A and B given above, and implements a 'for loop' to compute $D = \sum_{i=1}^{3} a_i b_i$.

```
clear
A = [pi; sqrt(2); exp(1)];
B = [1; 5; 7];
D = 0;
for i = 1:3
    D = D + (A(i)*B(i));
end
D
```

```
D = 29.2406
```

More Advanced Scripts

Part A

Create a m-file that will have an input column vector x and an output column vector y where the i-th element of the output vector is $y_i = f(x_i)$ and $f(x_i) = \dfrac{\sin(x_i)}{1+\exp(2x_i)}$.

```
% function y = myfunc(x)
%     y = sin(x)./(1+exp(2.*x));
% end
```

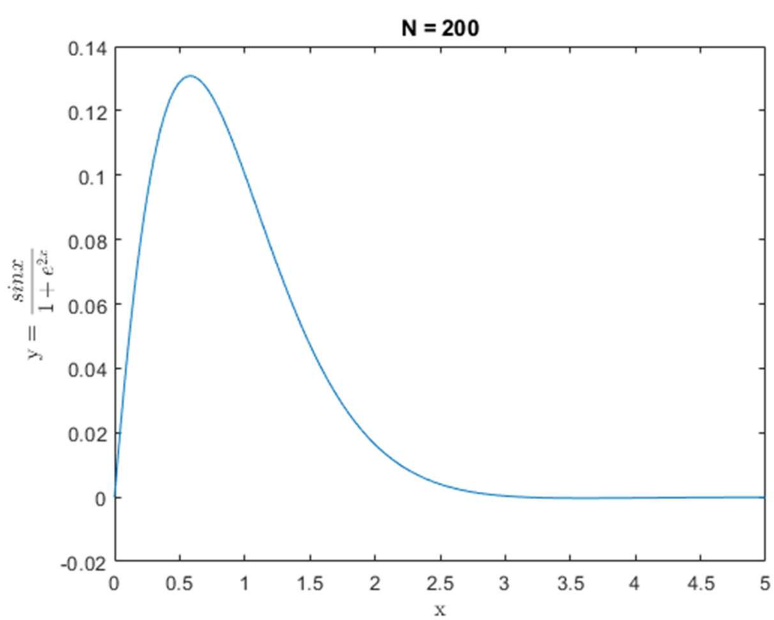
Part B

Create a second m-file will have an integer input N and no outputs. The m-file should define the vector x so that it contains (N + 1) equally spaced points on the interval [0, 5], call the previous m-file to calculate the vector $y = f(x)$, and plot y as a function of x. The spacing between the elements of the vector x is $\mathrm{dx} = \dfrac{5}{N}$.

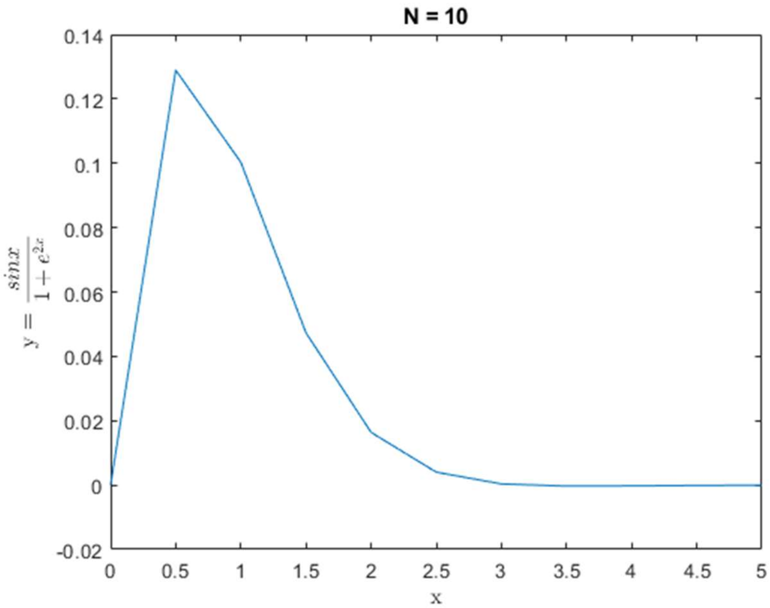
```
% function plotting(N)
%     dx = 5/N;
%     x = 0:dx:5;
%     y = myfunc(x);
%     figure
%     plot(x, y)
%     xlabel('x', 'Interpreter', 'latex');
%     ylabel('y = $$\frac{sin{x}}{1+e^{2x}}$$','Interpreter', 'latex')
%     title(['N = ',num2str(N)])
% end
```

Starting with a large value of N = 200, run the function and repeat this process with smaller values of N such as 5 or 10.

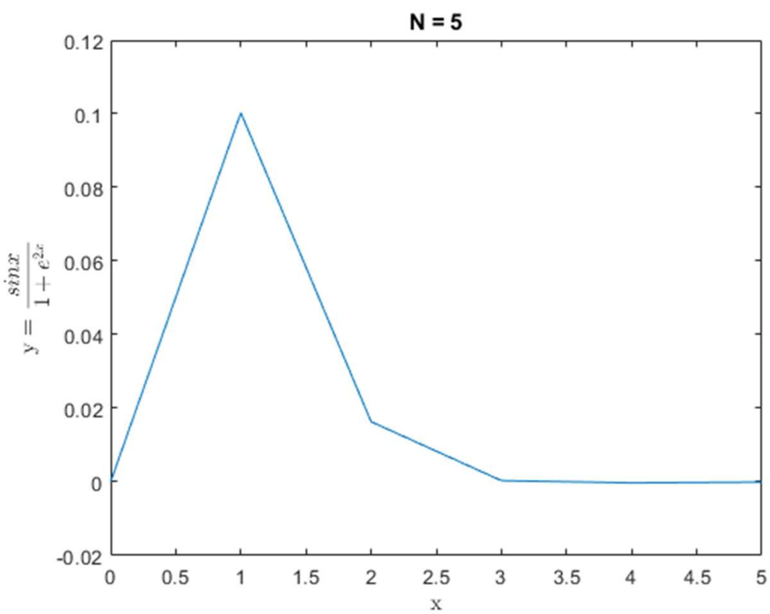
```
plotting(200)
```



```
plotting(10)
```




```
plotting(5)
```



Calculating Area Under the Curve

Part A

Use the MATLAB 'integral' function to approximate $Q = \int_0^5 f(x)\mathrm{dx}$.

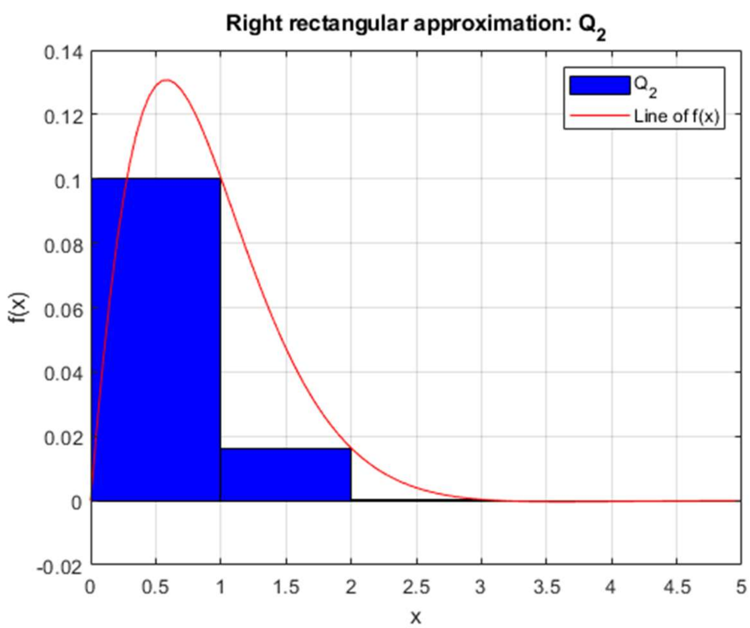
```
myfunc = @(x) sin(x)./(1+exp(2.*x));
Q = integral(myfunc, 0, 5)
```

```
Q = 0.1587
```

Part B

Draw a picture of $Q_2 = \sum_{i=2}^{N} f(x_i) dx_{i-1}$ using the 'bar' function and explain the equation of the computation of Q_2.

```
N = 5;
dx = 5/N;
x_bar = 1:dx:5;
y_bar = myfunc(x_bar);
x = 0:5/100:5;
y = myfunc(x);

figure
bar(x_bar-dx/2, y_bar, 1, 'blue')
xlim([0, 5])
hold on
plot(x, y, 'red')
grid on
title('Right rectangular approximation: Q_2')
xlabel('x')
ylabel('f(x)')
legend('Q_2', 'Line of f(x)')
```



The right rectangular approximation can be computed using the formula $Q_2 = \sum_{i=2}^{N} f(x_i) \mathrm{dx}_{i-1}$. We can derive this formula by first calculating the height of each rectangle at i using $f(x_i)$. Next, we multiple the height of the rectangle $f(x_i)$ by the width of each rectangle dx_i to find the area of each rectangle. Finally, we sum all the rectangles together by setting the lower bound to 2 and upper bound to N. The lower bound is set as 2 in order to calculate the right-hand side of the function as the height. In doing this, we also have to set dx_i to dx_{i-1} so that the array which contains the width of the rectangles starts at the first index and not the second.

Explain why the formula is valid only for small dx_i and why the accuracy is enhanced by decreasing dx.

The formula is valid only for a small dx_i because a small dx_i uses many rectangles to approximate the area underneath the curve. If large dx_i such as 1 is used, we can see in the above graph that two rectangles cannot provide an accurate approximation of the area. By decreasing dx, there will be more rectangles used for measuring the area, allowing for greater accuracy.

What are the tradeoffs related to decreasing dx?

By decreasing dx, the accuracy of the approximation increases. However, the time that it takes to compute the approximation also increases.

Derive the equation for Q_3 which uses a trapezoidal approximation on each interval.

We start off with a summation of N trapezoids which is equal to the area of the rectangle $f(x_i)\mathrm{dx}_i$ plus the area of the triangle $\frac{1}{2}(f(x_{i+1}) - f(x_i))\mathrm{dx}_i$.

$$Q_3 = \sum_{i=1}^{N-1} \left(f(x_i) + \frac{1}{2}(f(x_{i+1}) - f(x_i)) \right) \mathrm{dx}_i$$

$$Q_3 = \sum_{i=1}^{N-1} \left(f(x_i) + \frac{1}{2} f(x_{i+1}) - \frac{1}{2} f(x_i) \right) \mathrm{dx}_i$$

$$Q_3 = \sum_{i=1}^{N-1} \left(\frac{1}{2} f(x_i) + \frac{1}{2} f(x_{i+1}) \right) \mathrm{dx}_i$$

$$Q_3 = \frac{1}{2} \sum_{i=1}^{N-1} f(x_i)\mathrm{dx}_i + \frac{1}{2} \sum_{i=1}^{N-1} f(x_{i+1})\mathrm{dx}_i$$

$$Q_3 = \frac{1}{2} \sum_{i=1}^{N-1} f(x_i)\mathrm{dx}_i + \frac{1}{2} \sum_{i=2}^{N} f(x_i)\mathrm{dx}_{i-1}$$

$$Q_3 = \frac{1}{2} \sum_{i=2}^{N-1} f(x_i)\mathrm{dx}_i + \frac{1}{2} f(x_1)\mathrm{dx}_1 + \frac{1}{2} \sum_{i=2}^{N-1} f(x_i)\mathrm{dx}_{i-1} + \frac{1}{2} f(x_N)\mathrm{dx}_{N-1}$$

$$Q_3 = \frac{1}{2} f(x_1) dx_1 + \sum_{i=2}^{N-1} f(x_i) dx_i + \frac{1}{2} f(x_N) dx_{N-1}$$

Our final result matches the given formula in the lab manual.

Show that for equally spaced x_i (i.e., $\mathrm{dx}_i = \mathrm{dx}_{i-1}$): $Q_3 = \frac{(Q_1 + Q_2)}{2}$.

As shown in the earlier calculation, $Q_3 = \frac{1}{2}\sum_{i=1}^{N-1} f(x_i)\mathrm{dx}_i + \frac{1}{2}\sum_{i=2}^{N} f(x_i)\mathrm{dx}_{i-1}$. The formulas given in the lab manual state that $Q_1 = \frac{1}{2}\sum_{i=1}^{N-1} f(x_i)\mathrm{dx}_i$ and $\frac{1}{2}\sum_{i=2}^{N} f(x_i)\mathrm{dx}_{i-1}$. From this we can see that $Q_3 = \frac{(Q_1 + Q_2)}{2}$ meaning that Q_3 is actually the average of both Q_1 and Q_2.

Part C

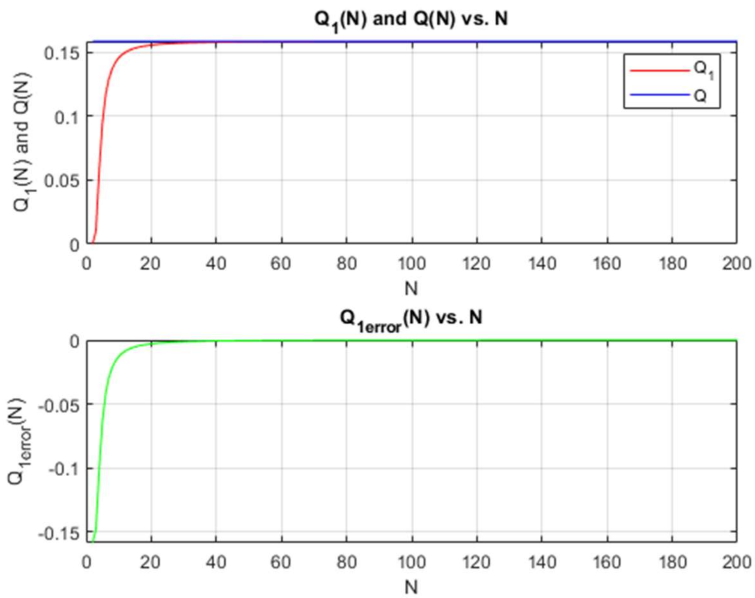
The summation in the calculation of Q_1 can be written as the product of two vectors and calculated using vector arithmetic using MATLAB code which is implemented here.

```
% function Q_1 = Q_1func(N)
%     dx = (5-0)/(N-1);
%     x = 0:dx:5;
%     y = myfunc(x);
%     A = [ones(N-1, 1); 0];
%     Q_1 = y*A*dx;
% end
```

Implement another function which computes and plots Q_1 for all integer values of N from 2 to 200. Plot Q_1(N) vs. N. Plot the constant value of Q determined by the MATLAB 'integral' function. Compare the results. Plot the error between Q_1(N) and the constant value of Q(N) determined above.

```
N = [2:1:200];
Q_1N = zeros(1, length(N));
for i=2:length(N)+1
    Q_1N(i-1) = Q_1func(i);
end

Q_N = ones(1, length(N));
Q_N = Q_N*Q;

Q_1error = zeros(1, length(N));
Q_1error = Q_1N - Q_N;

figure
subplot(2, 1, 1)
plot(N, Q_1N, 'red')
grid on
hold on
plot(N, Q_N, 'blue')
title('Q_1(N) and Q(N) vs. N')
xlabel('N')
ylabel('Q_1(N) and Q(N)')
legend('Q_1', 'Q')

subplot(2, 1, 2)
plot(N, Q_1error, 'green')
grid on
```

```
title('Q_{1error}(N) vs. N')
xlabel('N')
ylabel('Q_{1error}(N)')
```



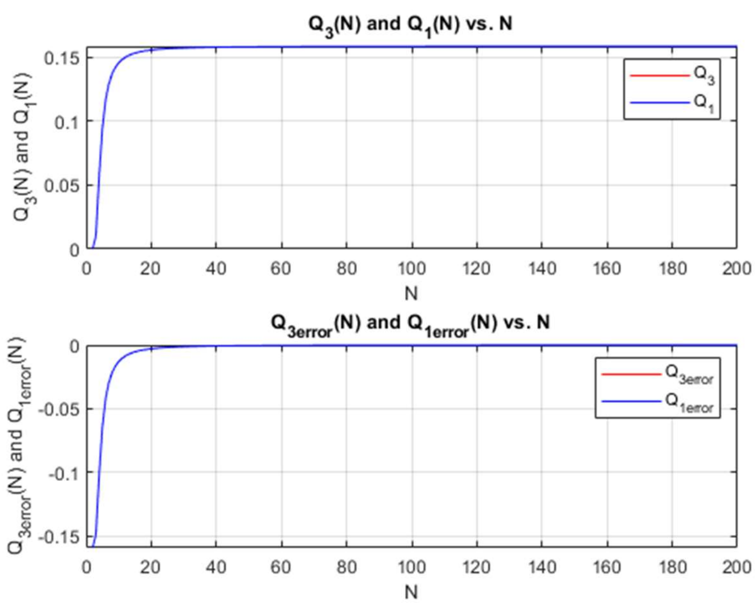
We can see that was N increases, Q_1 approaches Q. This is because as N increases, so does the accuracy of the area approximation causing the approximation Q_1 to approach the actual area Q.

Part D

Write you own code similar to the above to compute Q_3 directly and clearly state your definition of the matrix A. Compute Q_3 for all integer values of N from 2 to 200. Plot Q_3 (N) versus N on the same graph as Q_1 versus N. Also plot the error in Q_3 versus N on the same graph as the error in Q_1.

```
% function Q_3 = Q_3func(N)
%     dx = (5-0)/(N-1);
%     x = 0:dx:5;
%     y = myfunc(x);
%     A = [0.5; ones(N-2, 1); 0.5];
%     Q_3 = y*A*dx;
% end

N = [2:1:200];
Q_3N = zeros(1, length(N));
for i=2:length(N)+1
    Q_3N(i-1) = Q_3func(i);
end

Q_3error = zeros(1, length(N));
Q_3error = Q_3N - Q_N;

figure
subplot(2, 1, 1)
plot(N, Q_3N, 'red')
grid on
```

```
hold on
plot(N, Q_1N, 'blue')
title('Q_3(N) and Q_1(N) vs. N')
xlabel('N')
ylabel('Q_3(N) and Q_1(N)')
legend('Q_3', 'Q_1')

subplot(2, 1, 2)
plot(N, Q_3error, 'red')
grid on
hold on
plot(N, Q_1error, 'blue')
title('Q_{3error}(N) and Q_{1error}(N) vs. N')
xlabel('N')
ylabel('Q_{3error}(N) and Q_{1error}(N)')
legend('Q_{3error}', 'Q_{1error}')
```



Compare the rates of convergence for Q_1 and Q_3. Which is the better algorithm?

```
rate_of_convergence_Q1 = (Q_1N(length(Q_1N))-Q)/(Q_1N(length(Q_1N)-1)-Q)
```

```
rate_of_convergence_Q1 = 0.9899
```

```
rate_of_convergence_Q3 = (Q_3N(length(Q_3N))-Q)/(Q_3N(length(Q_3N)-1)-Q)
```

```
rate_of_convergence_Q3 = 0.9900
```

As we can see, the rate of convergence for Q_3 is very slightly higher than Q_1. This means although not by much, Q_3 is the better algorithm.

Conclusion:

Overall, the objective of this lab was accomplished. We are now a lot more familiar with the usage of MATLAB and know how to use MATLAB *help* facilities to learn about MATLAB on our own.