EE_105_021_23W Lab 1

Max Curren

TOTAL POINTS

100 / 100

QUESTION 1

1 Point 3 - Matrices and Arrays **15 / 15**

✓ **- 0 pts** *Correct*

QUESTION 2

2 Point 5 - Scripts **20 / 20**

✓ **- 0 pts** *Correct*

QUESTION 3

Point 6 - More Advanced Scripts 25 pts

3.1 First script **5 / 5**

✓ **- 0 pts** *Correct*

3.2 Second script **5 / 5**

✓ **- 0 pts** *Correct*

✓ **- 0 pts** *Did not implement as a function file*

3.3 Plot **15 / 15**

✓ **- 0 pts** *Correct*

QUESTION 4

Point 7 - Integral Approximation 40 pts

4.1 part a) **5 / 5**

✓ **- 0 pts** *Correct*

4.2 part b) **15 / 15**

✓ **- 0 pts** *Correct part i.*

✓ **- 0 pts** *Correct part ii.*

✓ **- 0 pts** *Correct part iii.*

1 This can be done without a for loop

`p = x - (dx/2);`

Subtraction is already an elementwise operation

4.3 part c) **10 / 10**

✓ **- 0 pts** *Correct*

4.4 part d) **10 / 10**

✓ **- 0 pts** *Correct*

3. This code defines column matrices A and B and solves for C = A ^{T}B:

Command Window

```
>> A = [pi; sqrt(2); exp(1)];
>> B = [1; 5; 7];
>> C = A'*B

C =

   29.2406
```

5. This code performs the same function as the previous code with a for loop:
The for loop's function is to increment D by (A at index i) * (B at index i) from i=1:3.

Editor - C:\Users\maxcu\Documents\MATLAB\EE105 ...

solveforc.m*

```
%clear memory
clear
%Define matrices A and B
A = [pi;sqrt(2);exp(1)];
B = [1;5;7];
%For loop which calculates D=A'*B
D = 0;
for i=1:3
    D = D + (A(i)*B(i));
end
```

Workspace

Name	Value
A	[3.1416;1.4142;2....
B	[1;5;7]
D	29.2406
i	3

6i. First .m function file where y is the output and x is the input:

mfile1.m | mfile2.m

```
function y = mfile1(x)
%Define y = f(x) = sin(x)/(1+exp(2x))
y = sin(x)./(1+exp(2.*x));
end
```

1 Point 3 - Matrices and Arrays **15 / 15**

✓ **- 0 pts** *Correct*

3. This code defines column matrices A and B and solves for $C = A^{T}B$:

Command Window

```
>> A = [pi; sqrt(2); exp(1)];
>> B = [1; 5; 7];
>> C = A'*B

C =

   29.2406
```

5. This code performs the same function as the previous code with a for loop:
The for loop's function is to increment D by (A at index i) * (B at index i) from i=1:3.

Editor - C:\Users\maxcu\Documents\MATLAB\EE105 ...

solveforc.m*

```
%clear memory
clear
%Define matrices A and B
A = [pi;sqrt(2);exp(1)];
B = [1;5;7];
%For loop which calculates D=A'*B
D = 0;
for i=1:3
    D = D + (A(i)*B(i));
end
```

Workspace

Name	Value
A	[3.1416;1.4142;2....
B	[1;5;7]
D	29.2406
i	3

6i. First .m function file where y is the output and x is the input:

mfile1.m | mfile2.m

```
function y = mfile1(x)
%Define y = f(x) = sin(x)/(1+exp(2x))
y = sin(x)./(1+exp(2.*x));
end
```

2 Point 5 - Scripts **20 / 20**

✓ **- 0 pts** *Correct*

3. This code defines column matrices A and B and solves for $C = A^TB$:

Command Window

```
>> A = [pi; sqrt(2); exp(1)];
>> B = [1; 5; 7];
>> C = A'*B

C =

   29.2406
```

5. This code performs the same function as the previous code with a for loop:
The for loop's function is to increment D by (A at index i) * (B at index i) from i=1:3.

Editor - C:\Users\maxcu\Documents\MATLAB\EE105 ...

solveforc.m*

```
%clear memory
clear
%Define matrices A and B
A = [pi;sqrt(2);exp(1)];
B = [1;5;7];
%For loop which calculates D=A'*B
D = 0;
for i=1:3
    D = D + (A(i)*B(i));
end
```

Workspace

Name	Value
A	[3.1416;1.4142;2....
B	[1;5;7]
D	29.2406
i	3

6i. First .m function file where y is the output and x is the input:

mfile1.m | mfile2.m

```
function y = mfile1(x)
%Define y = f(x) = sin(x)/(1+exp(2x))
y = sin(x)./(1+exp(2.*x));
end
```

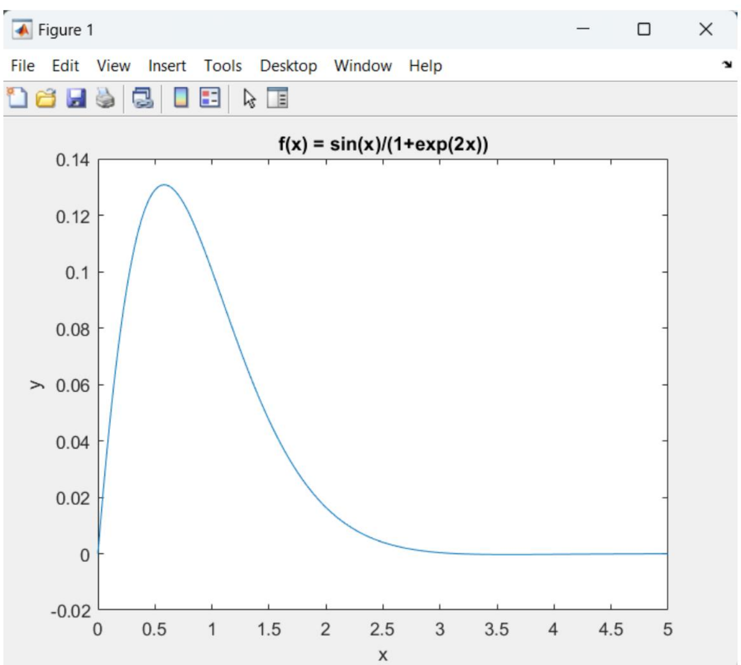
3.1 First script **5 / 5**

✓ **- 0 pts** *Correct*

6ii. 2nd .m file:

```
clear
%Define integer input N
N = 200;
%Defines x so that it contains (N+1) equal spaced points
x = 0:5/N:5;
%Call mfile1 which calculates y=f(x)
y = mfile1(x);
%Plots y as a function of x
plot(x,y)
xlabel('x')
ylabel('y')
title('f(x) = sin(x)/(1+exp(2x))')
```

Graph of f produced with N = 200:



The value of N increases the accuracy of the function by increasing the size of the x vector.

3.2 Second script **5 / 5**

✓ **- 0 pts** *Correct*

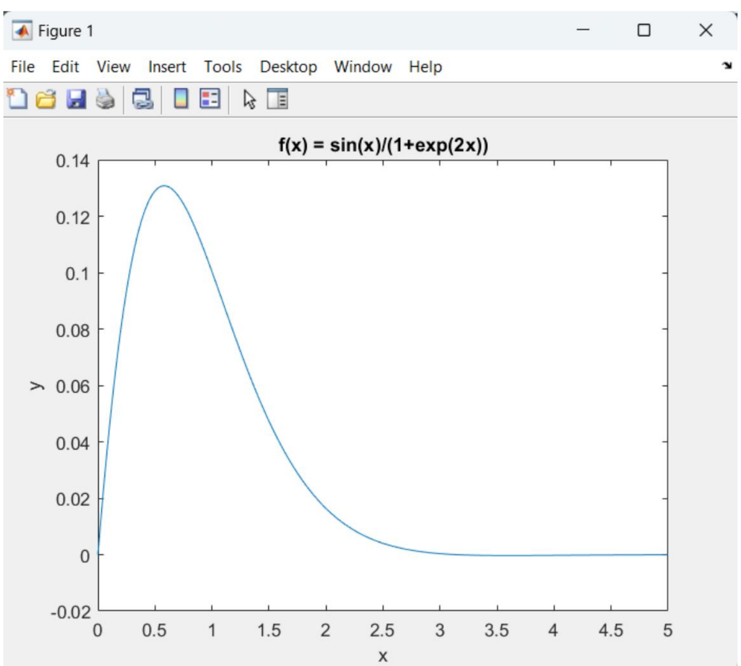
✓ **- 0 pts** *Did not implement as a function file*

6ii. 2nd .m file:

mfile1.m mfile2.m

```
clear
%Define integer input N
N = 200;
%Defines x so that it contains (N+1) equal spaced points
x = 0:5/N:5;
%Call mfile1 which calculates y=f(x)
y = mfile1(x);
%Plots y as a function of x
plot(x,y)
xlabel('x')
ylabel('y')
title('f(x) = sin(x)/(1+exp(2x))')
```

Graph of f produced with N = 200:



The value of N increases the accuracy of the function by increasing the size of the x vector.

3.3 Plot **15 / 15**

✓ **- 0 pts** *Correct*

7a. Implementation of integral of f(x) from 0 to 5

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

Value of Q from workspace after running code: Q 0.1587

7bi. Implementation of right rectangle approximation for Q2

mfile1.m | mfile2.m* | sixc.m | solveforc.m

```
%PART 6&7
clear
%Define integer input N and dx
N = 50;
dx = 5/N;

%Defines x so that it contains (N+1) equal spaced points
x = 0:dx:5;
%Call mfile1 which calculates y=f(x)
y = mfile1(x);
%------------------------------------------------------------------------
%PART 7b
%Left shift for right rectangular approx.
for u = 2:N
    %p(u) = the x value from midpoint - half of dx; This shifts the value
    %for p to the left by dx/2
    p(u) = x(u) - dx/2;
end
%append 5 to p
p = [p 5];

bar(p,y,2)
xlabel('x')
ylabel('y')
title('Q2: Right Rectangle Approx. with f(x) overlay')
hold on
plot(x,y,'LineWidth',3)
hold off
```

The bar function in MatLab places the center of each bar on the line from the characteristic equation. To correct this, we shift a dummy variable p(x) to the left by dx/2 and use this in the bar function. This correctly places the top right corner of each bar on the line.

4.1 part a) **5 / 5**

✓ **- 0 pts** *Correct*

7a. Implementation of integral of f(x) from 0 to 5

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

Value of Q from workspace after running code: Q 0.1587

7bi. Implementation of right rectangle approximation for Q2

mfile1.m | mfile2.m* | sixc.m | solveforc.m

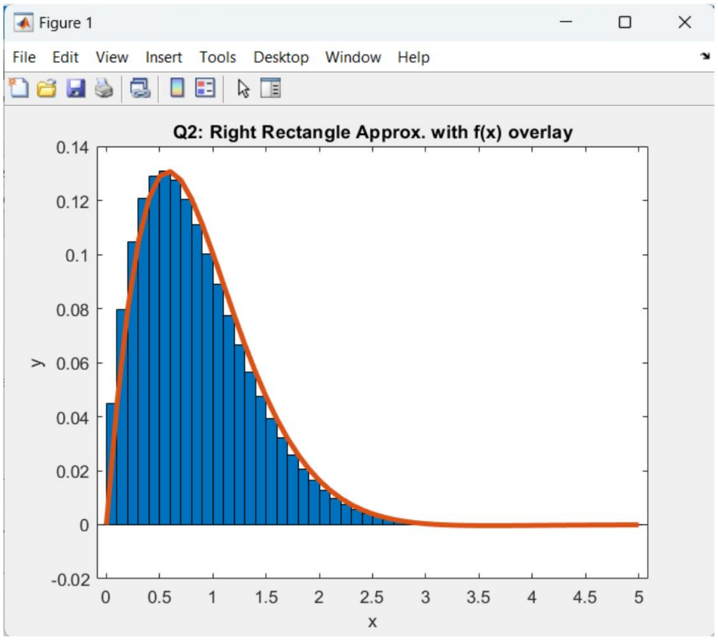
```
%PART 6&7
clear
%Define integer input N and dx
N = 50;
dx = 5/N;

%Defines x so that it contains (N+1) equal spaced points
x = 0:dx:5;
%Call mfile1 which calculates y=f(x)
y = mfile1(x);
%-----------------------------------------------------------------------
%PART 7b
%Left shift for right rectangular approx.
for u = 2:N
    %p(u) = the x value from midpoint - half of dx; This shifts the value
    %for p to the left by dx/2
    p(u) = x(u) - dx/2;
end
%append 5 to p
p = [p 5];

bar(p,y,2)
xlabel('x')
ylabel('y')
title('Q2: Right Rectangle Approx. with f(x) overlay')
hold on
plot(x,y,'LineWidth',3)
hold off
```

The bar function in MatLab places the center of each bar on the line from the characteristic equation. To correct this, we shift a dummy variable p(x) to the left by dx/2 and use this in the bar function. This correctly places the top right corner of each bar on the line.

Graph of Q2



Q) Explain the equation for the computation of Q2

Q2 is a summation of $f(x_i)dx_{i-1}$ from i=2 to i=N. $f(x_i)$ is multiplied by the width of the rectangle at i-1: dx_{i-1}, and summed with every $f(x_i)dx_{i-1}$ until i = N. i starts at 2 and ends at N to allow for N+1 equal rectangles.

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

The formula is only valid for small dx since increasing dx causes the number of subdivisions in the approximation to decrease. By decreasing dx, we increase the number of subdivisions in the approximation. This should increase the accuracy of the approximation.

7bii) What are the tradeoffs related to decreasing dx?

The variable x may become very large and take up a lot of memory, and computing each element of a large dx may increase computation time.

7biii A) Derive the above equation for Q3: $f(x_0)dx_0 + 0.5(f(x_1) - f(x_0))dx_0.$

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

1st segment $= f(x_0)dx_0 + \frac{1}{2}(f(x_1) - f(x_0))dx_0$

Area of rectangle + Area of triangle =
$= \text{base}\cdot\text{height}_r + \frac{1}{2}\text{base}\cdot\text{height}$
$= f(x_0)dx_0 + \frac{1}{2}(f(x_1) - f(x_0))dx$
= 1st segment

base $= dx_0$
$\text{height}_r = f(x_0) - 0 = f(x_0)$
$\text{height}_t = f(x_1) - f(x_0)$

7biii B) Show that for equally spaced x_i (i.e., dxi = dxi−1): Q3 = (Q1 + Q2)/2.

$$Q_3 = \frac{Q_1 + Q_2}{2} = \frac{1}{2}\left(\sum_{i=1}^{N-1} f(x_i)\,dx_i + \sum_{i=2}^{N} f(x_i)\,dx_{i-1} \right)$$

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

$$= \frac{1}{2}\left[\left(f(x_1)\,dx_1 + f(x_N)\,dx_{N-1} \right) \cdot 2\sum_{i=2}^{N-1} f(x_i)\,dx_i \right]$$

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

$$= Q_3$$

7c. Code for part 7c

sevenc.m | sevend.m | mfile2.m

```
%PART 7c
clear
%Computes Q1 from N = 2:200. This turns Q1 into a vector
%which shows how Q1 changes with N.
for N = 2:200
    dx = 5/(N-1);
    x = 0:dx:5;
    y = mfile1(x);
    A = [ones(N-1,1); 0];
    Q1(N) = y*A*dx;
end

subplot(2,1,1)
hold on
%plots Q1 by N
plot(1:200,Q1,'LineWidth',2)
xlabel("N")
ylabel("Accuracy")
title("Q1 by N with integral of y overlay")

%Computes integral of y
y = @(x) sin(x)./(1+exp(2.*x));
%plots value of integral onto graph
yline(integral(y,0,200),'g');
hold off
%calculate and plot the error between Q1(N) and the
%constant value of Q determined above
subplot(2,1,2)
for N = 2:200
```

4.2 part b) 15 / 15

✓ **- 0 pts** *Correct part i.*

✓ **- 0 pts** *Correct part ii.*

✓ **- 0 pts** *Correct part iii.*

1 This can be done without a for loop

`p = x - (dx/2);`

Subtraction is already an elementwise operation

7biii B) Show that for equally spaced x_i (i.e., dxi = dxi−1): Q3 = (Q1 + Q2)/2.

$$Q_3 = \frac{Q_1 + Q_2}{2} = \frac{1}{2}\left(\sum_{i=1}^{N-1} f(x_i)dx_i + \sum_{i=2}^{N} f(x_i)\, dx_{i-1} \right)$$

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

$$= \frac{1}{2}\left[\left(f(x_1)dx_1 + f(x_N)dx_{N-1}\right) \cdot 2\sum_{i=2}^{N-1} f(x_i)dx_i \right]$$

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

$$= Q_3$$

7c. Code for part 7c

sevenc.m | sevend.m | mfile2.m | +

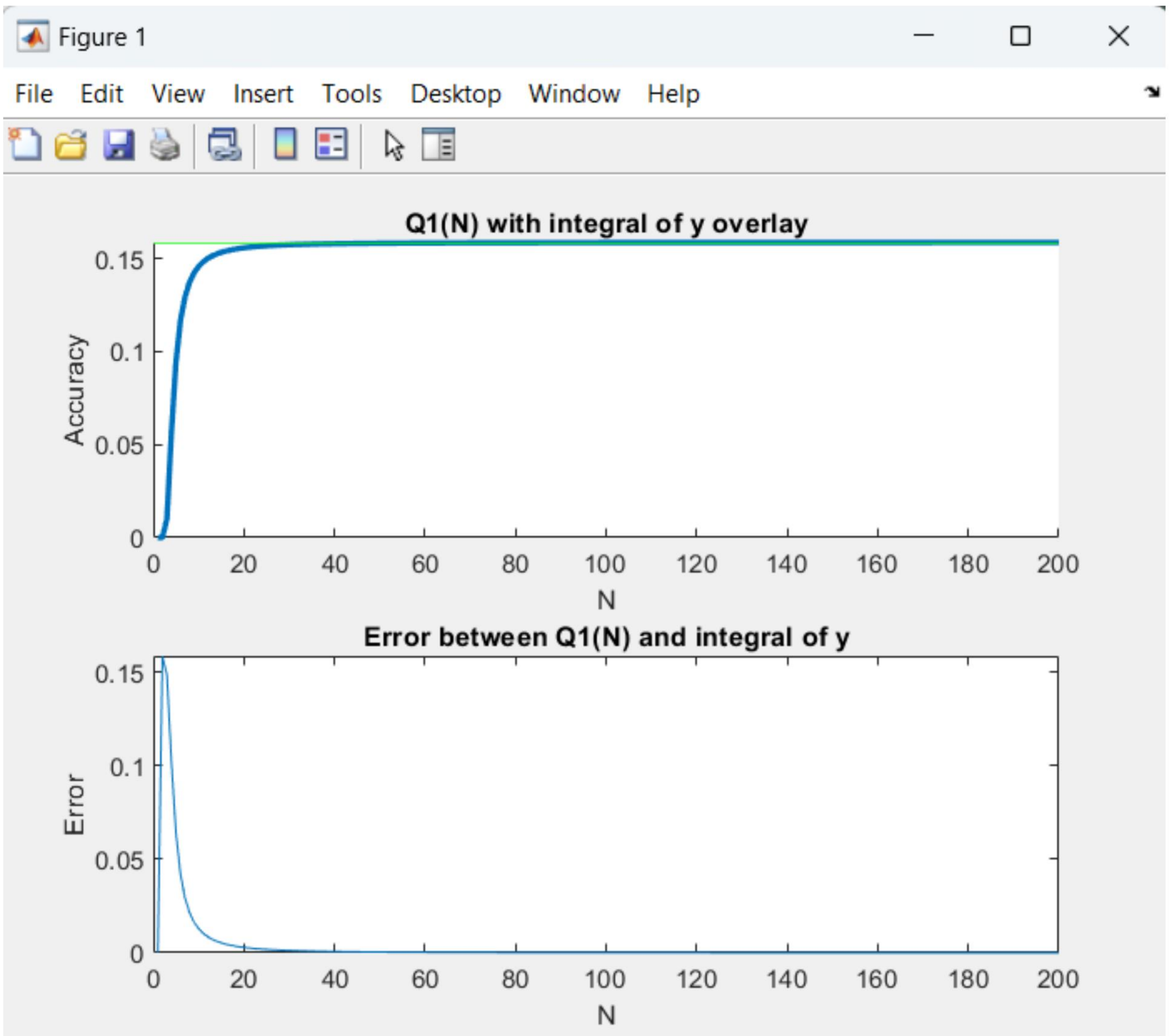
```
%PART 7c
clear
%Computes Q1 from N = 2:200. This turns Q1 into a vector
%which shows how Q1 changes with N.
for N = 2:200
    dx = 5/(N-1);
    x = 0:dx:5;
    y = mfile1(x);
    A = [ones(N-1,1); 0];
    Q1(N) = y*A*dx;
end

subplot(2,1,1)
hold on
%plots Q1 by N
plot(1:200,Q1,'LineWidth',2)
xlabel("N")
ylabel("Accuracy")
title("Q1 by N with integral of y overlay")

%Computes integral of y
y = @(x) sin(x)./(1+exp(2.*x));
%plots value of integral onto graph
yline(integral(y,0,200),'g');
hold off
%calculate and plot the error between Q1(N) and the
%constant value of Q determined above
subplot(2,1,2)
for N = 2:200
```

```
    error(N) = integral(y,0,200)- Q1(N);
end
plot(1:200,error)
xlabel("N")
ylabel("Error")
title("Error between Q1(N) and integral of y")
```

Plots for 7c.



Q) Compare $Q_1(N)$ versus N with the constant value of Q from integral function

As N increases, the accuracy of Q1 increases and at about N=30, the error is basically zero.

4.3 part c) **10 / 10**

✓ **- 0 pts** *Correct*

7d) Code for part 7d.

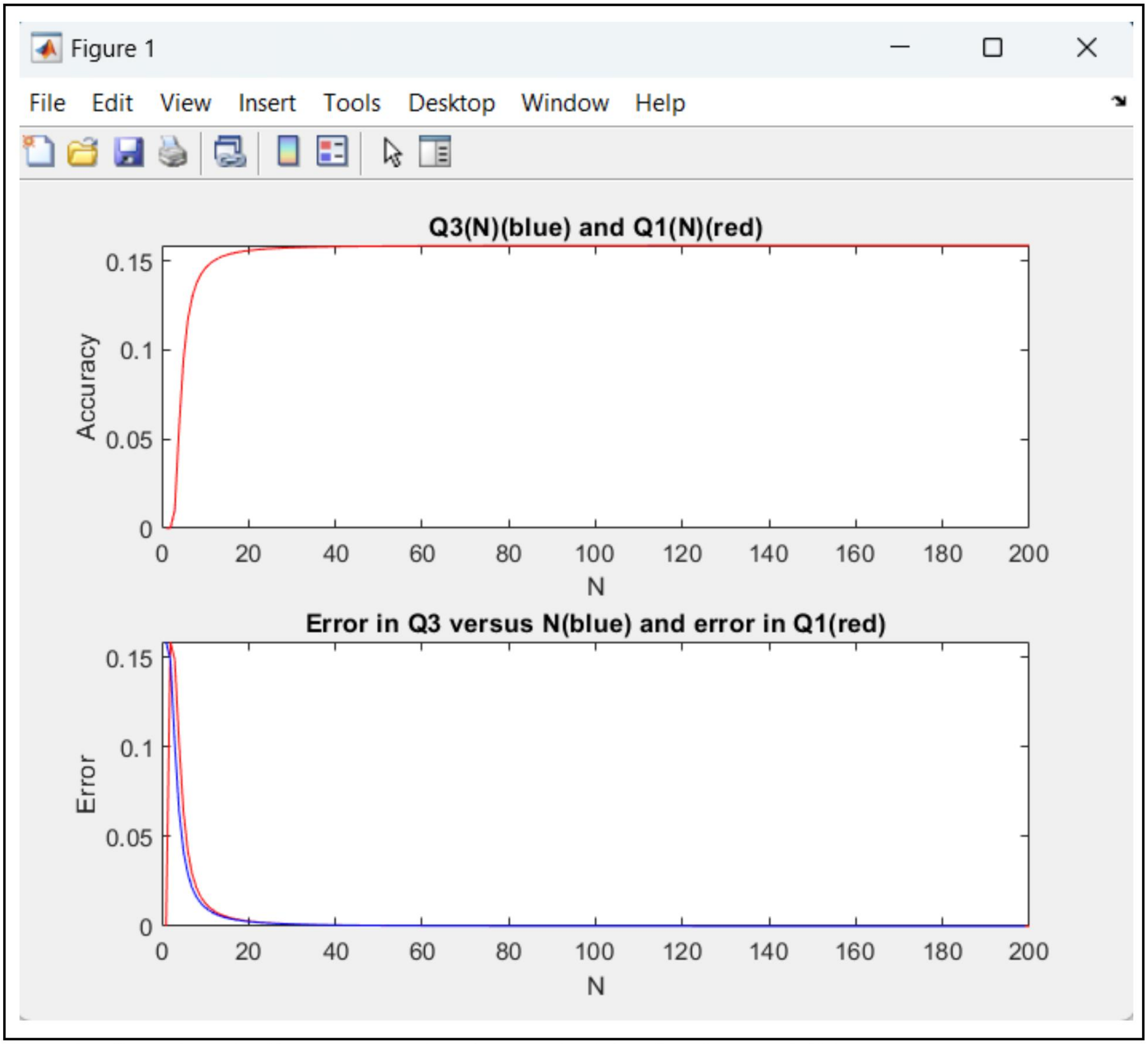
sevenc.m | sevend.m

```
%Part 7D
for N = 2:200
    dx = 5/(N-1);
    x = 0:dx:5;
    y = mfile1(x);
    %1/2 is coefficient for 1st term in Q3 and the
    %summation has N-2 terms. 1/2 is the coefficient
    %for the last term
    A = [1/2;ones(N-2,1);1/2];
    Q3(N-1) = y*A*dx;
end

subplot(2,1,1)
%plots Q3(N) and Q1(N) on same graph
plot(2:200,Q3,'b')
hold on
plot(1:200,Q1,'r')
hold off
xlabel("N")
ylabel("Accuracy")
title("Q3(N)(blue) and Q1(N)(red)")

%error in Q3 versus N on the same graph as the error in Q1
y = @(x) sin(x)./(1+exp(2.*x));
for N = 1:199
    errorQ3(N) = integral(y,0,200)- Q3(N);
end
subplot(2,1,2)
plot(1:200,errorQ1,'r')
hold on
plot(1:199,errorQ3,'b')
hold off
xlabel("N")
ylabel("Error")
title("Error in Q3 versus N(blue) and error in Q1(red)")
```

Plots for 7d.



*For the first plot, the lines are overlapping so the blue line is not visible

Q) Is Q1 or Q3 a better algorithm?

From the error plot for Q1 and Q3, we can see that the curve for Q3 is slightly lower than the curve for Q1. This means that the error for Q3 is slightly less than for Q1, making Q3 a better algorithm for getting a higher accuracy.

8. Code implemented for part 8

4.4 part d) **10 / 10**

✓ **- 0 pts** *Correct*

Maxwell Curren; SID: 862210814

Lab 1: MATLAB as an Engineer’s Problem Solving Tool

Lab Section 21

TA: Kathryn Hammar

eight.m

```
%Part 8
%1st code segment
clear x
tic
for i = 1:1000000
    x(i) = 1;
end
toc

%Define dimensions of vector x before for loop
clear x
x = zeros(1,1000000);
tic
for i = 1:1000000
    x(i) = 1;
end
toc

%Indexing
clear x
tic
x(1:1000000) = 1;
toc

%Ones
clear x
tic
x = ones(1,1000000);
toc
```

Command Window output for Part 8

Command Window

```
>> eight
Elapsed time is 0.064921 seconds.
Elapsed time is 0.001749 seconds.
Elapsed time is 0.001350 seconds.
Elapsed time is 0.000947 seconds.
fx >>
```

Q) Compare results

For the fastest method for creating a 1x1000000 matrix full of ones is to use the built-in MatLab function ones. The slowest method would be to use a for loop and set x(i) = 1 for i = 1:1000000 without predefining the dimensions of x.