ESC 103F Engineering Mathematics and Computation

MATLAB Test 1

Name (last name, first name):

Student #:

Tutorial Section Number:

Test Instructions

(Note to Class of 2T6: these are the test instructions used last year. These instructions are currently being reviewed by ECF staff and will be revised for 2022 as necessary.)

1. Sign into one of the ECF lab computers using the username and password given here (**,**).
2. Right click on the Start Menu found in the lower left-hand corner of the screen and select File Explorer. Then select This PC. Make sure that you can access both the W:\ drive (for working) and the S:\ drive (for submission). If you do not see both drives, notify a TA.
3. Return to the Start Menu and open MATLAB R2021a.
4. Include your **name and student number** in a comment at the top of your live script file.
5. Make sure to save your code periodically on the W:\ drive as you work through the test.
6. Save your live script file using the naming convention **TUT01##_lastname_studentnumber.mlx** where TUT01## is your tutorial section number, TUT0101-TUT0112.
7. Once you have finished the test, **copy and paste** your final live script file to:

   S:\courses\ESC103\Submit\username

   Note: you **cannot** save your file directly to the S:\ drive.

8. After submission to the S:\drive, if you want to submit a better version, do **not** try to edit file in the S:\drive. Instead, copy and paste the better version using the following naming convention **TUT01##_lastname_studentnumber_v2.mlx**. Your better (i.e. v2) version is the one that will be marked.
9. Right click on the Start Menu and select sign out.
10. Complete and hand in this test paper to the TA and have the TA check to be sure they can see your submitted file before you leave the computer lab.

Given information

The least squares solution to the system of equations corresponding to $A\vec{x} = \vec{b}$ is given by:

$$A^T A\vec{x}_{LS} = A^T\vec{b}$$

(Note to Class of 2T6: students are expected to solve these equations by first defining $\boldsymbol{A^* = A^T A}$ and $\boldsymbol{\vec{b}^* = A^T\vec{b}}$ and then solving $\boldsymbol{A^*\vec{x}_{LS} = \vec{b}^*}$ using the '\' command.)

and the associated error vector is given by:

$$\vec{e} = \vec{b} - A\vec{x}_{LS}$$

Q1: It is known that the tensile strength of a plastic increases as a function of the time if it is heat treated. The following data has been collected (units not provided):

Time (T)	10	15	20	25	40	50	55	60	75
Tensile strength (TS)	5	20	18	40	33	54	70	60	78

a) Develop a MATLAB live script that fits three different curves to the above data based on the least squares solution:

   i. $TS = a_1 + b_1 T$ (curve 1)
   ii. $TS = a_2 + b_2 T^2$ (curve 2)
   iii. $TS = a_3 + b_3\sqrt{T}$ (curve 3)

b) Within your live script, produce a single plot that shows the 9 data points and the three fitted curves with time on the x-axis and tensile strength on the y-axis. Be sure to use a small time step $\Delta T = 0.1$ when plotting the three curves so that the curves appear smooth. Also, be sure to include labels for the x and y axes and a legend.

c) Based on your three curves, calculate within your live script the squared magnitude of the error vector ($||\vec{e}||^2$) corresponding to each fit. Record your results here:

   i. _______.
   ii. _______.
   iii. _______.

d) Based on your best fit of the data, i.e., the curve that gives the smallest squared magnitude of the error vector in part (c), estimate within your live script the tensile strength of the plastic if it is heat treated for 32 time units and record your estimate here: _______.

ESC103 2021F MATLAB Test

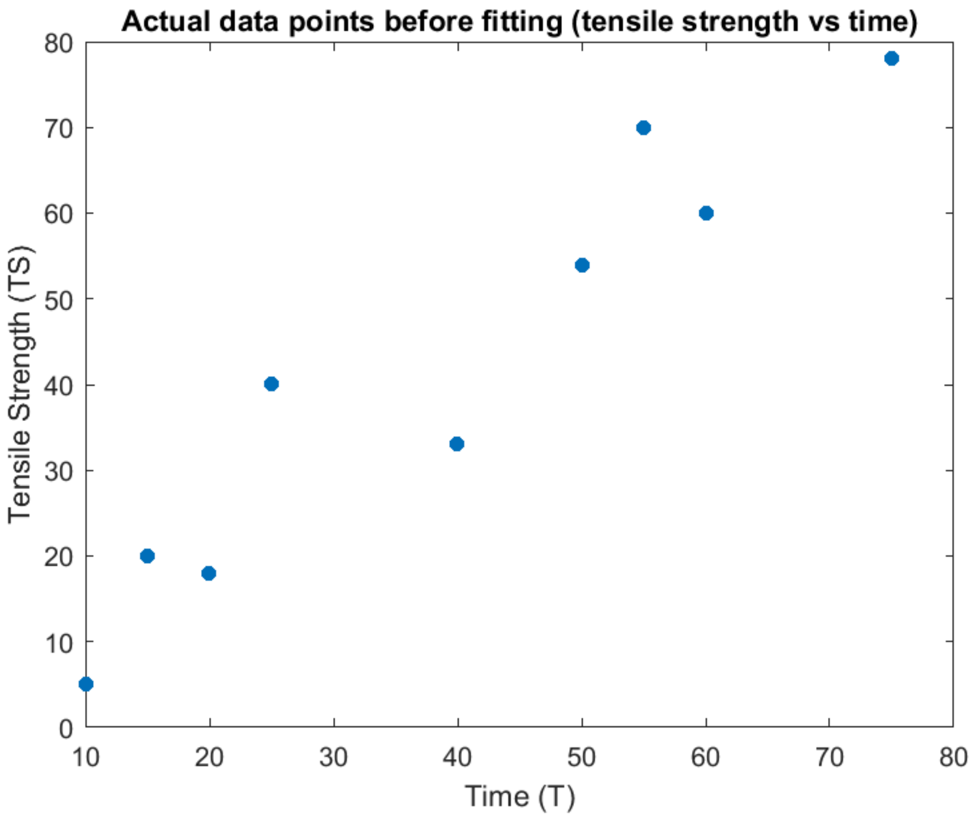
Solutions with marking scheme by Ameya Datta and Katie Allison 2021-12-13

```
% preliminaries for legend appearance
opengl software
opengl('save','software')

% cleanup
clear
close all
clc
```

Question 1: Least Squares Fitting

0.5 marks for setting up data correctly

```
% input data given in test document
T = [10; 15; 20; 25; 40; 50; 55; 60; 75];
TS = [5; 20; 18; 40; 33; 54; 70; 60; 78];

% plot initial data for visualization
figure
plot(T,TS,".",'MarkerSize',20)
xlabel("Time (T)")
ylabel("Tensile Strength (TS)")
title("Actual data points before fitting (tensile strength vs time)")
```



Part a) fitting curves to data

For all data curve fits, the least squares solution for the vector of coefficients is:

$$\overrightarrow{x_{\text{LS}}} = (A^T A)^{-1}(A^T(\text{TS}))$$

0.25 marks for each A matrix and 0.25 marks for each b vector (1.5 marks total for all A matrices and b vectors set up correctly)

0.5 marks for computing each set of coefficients correctly (1.5 marks total for all sets of coefficients)

```
% get number of rows in A for use in making column of 1s
m = length(T);

% fit TS_fit1=a_1+b_1 T (curve 1)
A = [ones(m,1) T];
prod1 = A'*A;
prod2 = A'*TS;
x_LS = prod1\prod2;
a_1 = x_LS(1)
```

```
a_1 = 0.8179
```

```
b_1 = x_LS(2)
```

```
b_1 = 1.0590
```

```
% fit TS_fit2=a_2+b_2 T^2 (curve 2)
A = [ones(m,1) T.^2];
prod1 = A'*A;
prod2 = A'*TS;
x_LS = prod1\prod2;
a_2 = x_LS(1)
```

```
a_2 = 17.8626
```

```
b_2 = x_LS(2)
```

```
b_2 = 0.0123
```

```

% fit TS_fit3=a_3+b_3 √T (curve 3)
A = [ones(m,1) sqrt(T)];
prod1 = A'*A;
prod2 = A'*TS;
x_LS = prod1\prod2;
a_3 = x_LS(1)
```

```
a_3 = -32.8554
```

```
b_3 = x_LS(2)
```

```
b_3 = 12.5397
```

Part b) plotting fit curves

0.25 marks for computing each vector of fit data correctly (0.75 marks total for all fit data)

```
% make vector of many x points at which to plot
T_plot = T(1):0.1:T(end);

% make vectors of TS_fit data for fits 1, 2, and 3
TS_fit1 = a_1 + b_1*T_plot;
TS_fit2 = a_2 + b_2*(T_plot.^2);
TS_fit3 = a_3 + b_3*sqrt(T_plot);
```

0.25 marks for plotting each fit curve correctly (0.75 marks total for all fit curves)

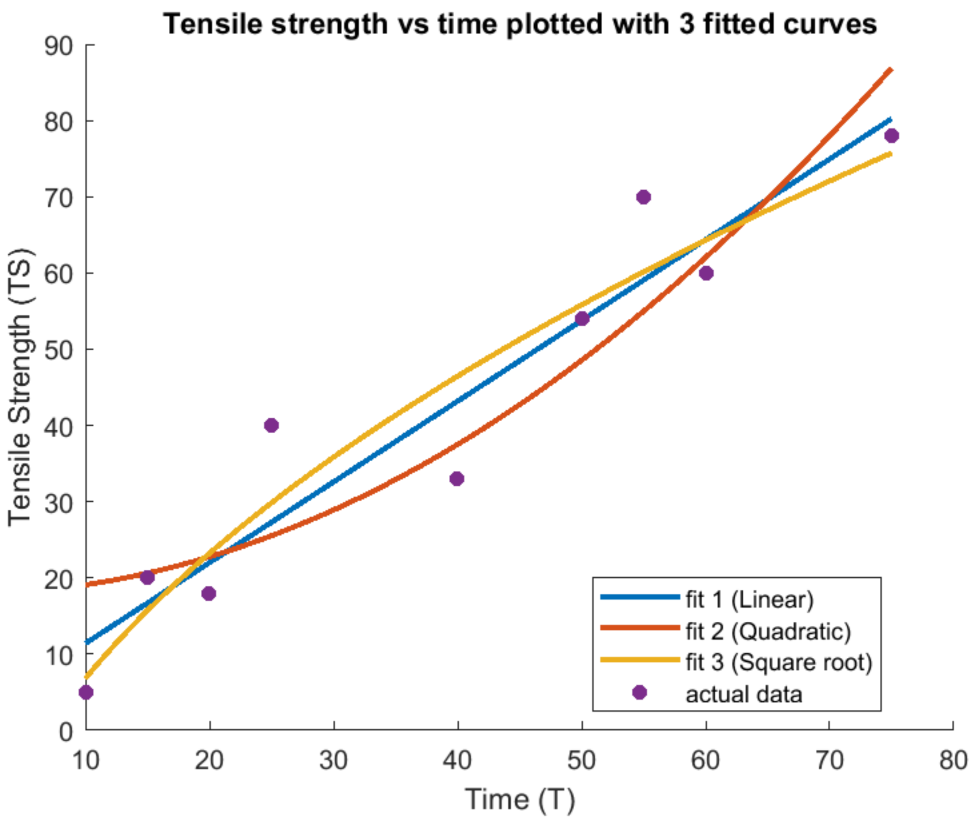
```
% plot data
figure
hold on
plot(T_plot,TS_fit1,'LineWidth',2)
plot(T_plot,TS_fit2,'LineWidth',2)
plot(T_plot,TS_fit3,'LineWidth',2)
```

1 mark for plotting data points

```
plot(T,TS,".",'MarkerSize',20)
% plot data last to ensure it shows on top of lines
% can also use scatter(T,TS) to plot data points
```

1 mark for annotating plot correctly with labels and legend

```
% add labels, legend, and title
xlabel("Time (T)")
ylabel("Tensile Strength (TS)")
legend("fit 1 (Linear)","fit 2 (Quadratic)","fit 3 (Square root)","actual data",'Location','bes
title("Tensile strength vs time plotted with 3 fitted curves")
```



Part c) squared magnitude of the error vector ||e||^2

The squared magnitude is equal to the sum of the squares of the component values of the error vector; i.e., for

$\overrightarrow{e} = \begin{bmatrix} e_1 & e_2 & \cdots & e_n \end{bmatrix}$, the squared magnitude is:

$$||e||^2 = e_1^2 + e_2^2 + \cdots + e_n^2$$

1.25 marks for correct approach (0.5 for error vector, 0.5 for magnitude, 0.25 for squaring magnitude)

0.25 for each squared magnitude error value (0.75 marks in total)

```
% get values of fit curves at each measured T and subtract from measured TS
e1 = TS - (a_1 + b_1*T);
e2 = TS - (a_2 + b_2*(T.^2));
e3 = TS - (a_3 + b_3*sqrt(T));

% note: we could also get the error vector with the A matrix and x_LS
% solution directly i.e. e = TS - A*x_LS for each of our 3 A and x_LS sets
% compute squared magnitude of each error vector
e1_sq_mag = sum(e1.^2)
```

```
e1_sq_mag = 476.6712
```

```
e2_sq_mag = sum(e2.^2)
```

```
e2_sq_mag = 789.6823
```

```
e3_sq_mag = sum(e3.^2)
```

```
e3_sq_mag = 456.9278
```

The fit that gives the lowest squared error magnitude is fit 3!

Part d) tensile strength at T = 32

We will use fit 3 (the function involving the square root of T) since this fit gave the lowest squared error magnitude.

0.5 for correct approach to interpolation

0.5 for correct value (based on fit used by student, regardless of whether this fit selected correctly)

```
% here we just need to use our fit function for fit 3 with T = 32
a = a_3;
b = b_3;
TS_32 = a + b*sqrt(32)
```

```
TS_32 = 38.0796
```

```
% we could also do this by setting up an anonymous function to make it
% easier to test multiple values
fit_fcn = @(t) a + b*sqrt(t);
TS_32 = fit_fcn(32);
```

ESC 103F Engineering Mathematics and Computation

MATLAB Test 2

Name (last name, first name):

Student #:

Tutorial Section Number:

Test Instructions

(Note to Class of 2T6: these are the test instructions used last year. These instructions are currently being reviewed by ECF staff and will be revised for 2022 as necessary.)

1. Sign into one of the ECF lab computers using the username and password given here (**,**).
2. Right click on the Start Menu found in the lower left-hand corner of the screen and select File Explorer. Then select This PC. Make sure that you can access both the W:\ drive (for working) and the S:\ drive (for submission). If you do not see both drives, notify a TA.
3. Return to the Start Menu and open MATLAB R2021a.
4. Include your **name and student number** in a comment at the top of your live script file.
5. Make sure to save your code periodically on the W:\ drive as you work through the test.
6. Save your live script file using the naming convention **TUT01##_lastname_studentnumber.mlx** where TUT01## is your tutorial section number, TUT0101-TUT0112.
7. Once you have finished the test, **copy and paste** your final live script file to:

   S:\courses\ESC103\Submit\username

   Note: you **cannot** save your file directly to the S:\ drive.

8. After submission to the S:\drive, if you want to submit a better version, do **not** try to edit file in the S:\drive. Instead, copy and paste the better version using the following naming convention **TUT01##_lastname_studentnumber_v2.mlx**. Your better (i.e. v2) version is the one that will be marked.
9. Right click on the Start Menu and select sign out.
10. Complete and hand in this test paper to the TA and have the TA check to be sure they can see your submitted file before you leave the computer lab.

Given information

Left endpoint formula: $L_n = \sum_{i=1}^{n} f(x_{i-1})\Delta x$

Trapezoidal formula: $T_n = \sum_{i=1}^{n} (\frac{f(x_{i-1})+f(x_i)}{2})\Delta x$

Q1: Consider the following integral equation:

$$F = \int_0^2 (4x^5 + 3x^3 - 2)dx$$

a) This integral can be evaluated analytically. Determine the analytical value for F and give it here_____.

b) Develop a MATLAB live script that numerically estimates F using (i) a left endpoint approximation and (ii) a trapezoidal approximation.

c) Within your live script, produce a single plot showing both the left endpoint results and the trapezoidal results. Plot the numerical estimates for F on the y-axis as a function of the number of subdivisions n on the x-axis. Use different line types for the two different approximations. Also include a horizontal line showing the analytical solution. For both methods, determine estimates for F with n ranging from 10 to 100. Make sure your plot has appropriate axis labels and includes a legend.

d) Within your live script, determine the minimum number of subdivisions (n_{min}) required for the trapezoidal approximation to produce an error with an absolute value less than 0.01 and give the value for n_{min} here_____. Then, within your live script, determine the absolute value of the error obtained with the left endpoint approximation when using the same number of subdivisions (n_{min}) and give the absolute value of the error here_____.

e) In a second plot within your live script, plot the integrand $f(x) = 4x^5 + 3x^3 - 2$ over the interval $[0,2]$ using a step size of 0.01 to ensure a smooth curve. Make sure your plot has appropriate axis labels.

f) Give below the solid line at the bottom of the page, a concise explanation as to why, for this particular example, the trapezoidal approximation gives a much more accurate estimate of the integral than does the left endpoint approximation when using the same number of subdivisions over the interval $[0,2]$.

```
% ESC103 MATLAB Test 2 Solution 2021

% Patch legends graphics error
opengl software
opengl('save','software')

clear; clc; close all;
```

```
% a) The analytical value is 50.667
res_A = 50.667;
```

```
% b) leftpoint and trapezoid numerical approximations. functions below

% generate x and F vectors
a = 0;                              % starting x value
b = 2;                              % ending x value
n = 100;                            % number of subdivisions
dx = (b-a)/n;
x = linspace(a,b,n+1);
F = 4*x.^5 + 3*x.^3 - 2;

% i) left end approximation
F_L = F(1:end-1);                   % F at left edge of n rectangles
res_L = sum(F_L * dx)
```

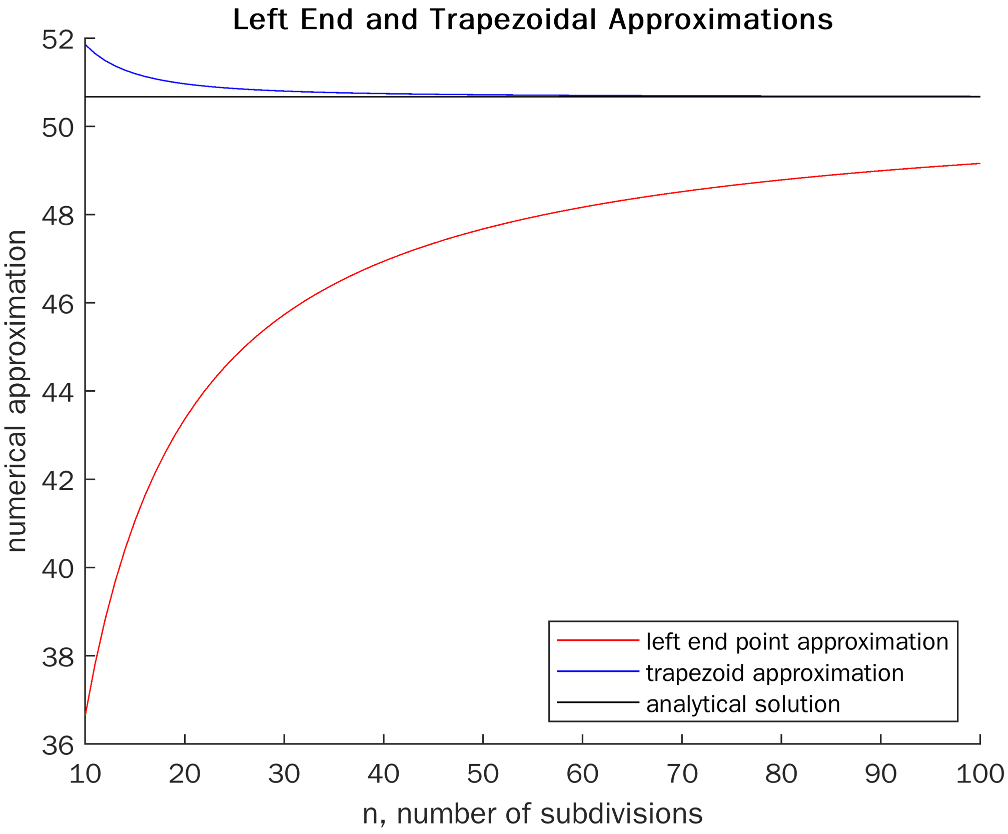
```
res_L = 49.1585
```

```


% ii) trapezoid approximation
F_R = F(2:end);                     % F at right edge of n rect/traps
res_T = sum((F_L + F_R)/2*dx)
```

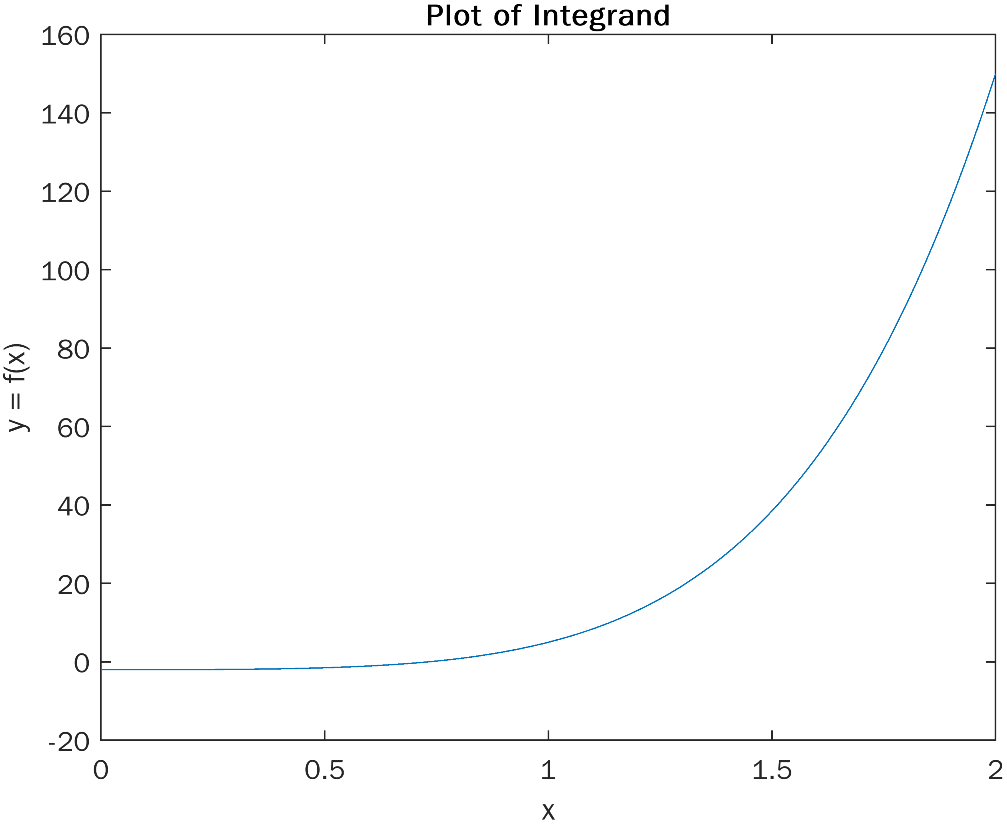
```
res_T = 50.6785
```

```
% c) loop through n = 10:100
res_Ls = nan(1,91);
res_Ts = nan(1,91);

for n = 10:100
    % calculate and save results. Write function or copy and paste from b)
    res_Ls(n-9) = left_approx(n);
    res_Ts(n-9) = trap_approx(n);
end

% plotting
figure(1)
hold on
plot(10:100, res_Ls, 'r')
plot(10:100, res_Ts, 'b')
plot([10,100], [res_A, res_A], 'k')
xlim([10,100])
title('Left End and Trapezoidal Approximations')
```

```
xlabel('n, number of subdivisions')
ylabel('numerical approximation')
legend({'left end point approximation', ...
        'trapezoid approximation', ...
        'analytical solution'}, 'location','southeast')
```



```
% d) plot F(x)
dx_2 = 0.01;
x_2 = 0:dx_2:2;
F_2 = 4*x_2.^5 + 3*x_2.^3 - 2;

figure(2)
plot(x_2, F_2)
title('Plot of Integrand')
xlabel('x')
ylabel('y = f(x)')
ylim([-20,160])
```



```
% e) n_min for trapezoidal error < 0.01
for n = 10:150
    res_T = trap_approx(n);
    if abs(res_T - res_A) <= 0.01
        break
    end
end

n_min = n
```

```
n_min = 108
```

```

res_L = left_approx(n_min);

error_L = abs(res_A - res_L)
```

```
error_L = 1.3976
```

```
% f) Explanation for the success of the trapezoidal rule
%
% The left hand rule significantly underestimates
% high degree polynomials, in this case we have x^5 leading,
% and when x > 1, this polynomial increases drastically.
% At the same time, trapezoidal rule is able to better approximate
% this increase due to use the use of the right hand point.
```

```
function [res] = left_approx(n)
    % generate x and F vectors for n divisions (n+1 points)
    dx = (2-0)/n;
    x = linspace(0,2,n+1);
    F = 4*x.^5 + 3*x.^3 - 2;

    % i) left end approximation
    F_L = F(1:end-1);                   % F at left edge of rect
    res = sum(F_L * dx);
end

function [res] = trap_approx(n)
    % generate x and F vectors for n divisions (n+1 points)
    dx = (2-0)/n;
    x = linspace(0,2,n+1);
    F = 4*x.^5 + 3*x.^3 - 2;

    % i) trapezoidal approximation
    F_L = F(1:end-1);                   % F at left edge of trap
    F_R = F(2:end);                     % F at right edge of trap
    res = sum((F_L + F_R)/2 * dx);
end
```

ESC 103F Engineering Mathematics and Computation

MATLAB Test 3

Name (last name, first name):

Student #:

Tutorial Section Number:

Test Instructions

(Note to Class of 2T6: these are the test instructions used last year. These instructions are currently being reviewed by ECF staff and will be revised for 2022 as necessary.)

1. Sign into one of the ECF lab computers using the username and password given here (**,**).
2. Right click on the Start Menu found in the lower left-hand corner of the screen and select File Explorer. Then select This PC. Make sure that you can access both the W:\ drive (for working) and the S:\ drive (for submission). If you do not see both drives, notify a TA.
3. Return to the Start Menu and open MATLAB R2021a.
4. Include your **name and student number** in a comment at the top of your live script file.
5. Make sure to save your code periodically on the W:\ drive as you work through the test.
6. Save your live script file using the naming convention **TUT01##_lastname_studentnumber.mlx** where TUT01## is your tutorial section number, TUT0101-TUT0112.
7. Once you have finished the test, **copy and paste** your final live script file to:

   S:\courses\ESC103\Submit\username

   Note: you **cannot** save your file directly to the S:\ drive.

8. After submission to the S:\drive, if you want to submit a better version, do **not** try to edit file in the S:\drive. Instead, copy and paste the better version using the following naming convention **TUT01##_lastname_studentnumber_v2.mlx**. Your better (i.e. v2) version is the one that will be marked.
9. Right click on the Start Menu and select sign out.
10. Complete and hand in this test paper to the TA and have the TA check to be sure they can see your submitted file before you leave the computer lab.

Given information

Euler's Method (EM):

$$t_{n+1} = t_n + \Delta t$$

$$Z_{n+1} = Z_n + \Delta t A Z_n$$

Improved Euler's Method (IEM):

$$t_{n+1} = t_n + \Delta t$$

$$Z_{n+1} = Z_n + \Delta t A Z_n$$

$$Z_{n+1} = Z_n + \frac{\Delta t}{2}(AZ_n + AZ_{n+1})$$

Q1: Consider the following initial value problem (IVP)

$$y'' = -\frac{19}{4}y - 10y' \quad y(0) = -9 \quad y'(0) = 0$$

The analytical solution to this IVP is given by

$$y(t) = -\frac{19}{2}e^{-\frac{t}{2}} + \frac{1}{2}e^{-\frac{19t}{2}}$$

a) By defining $Z = \begin{bmatrix} y \\ y' \end{bmatrix}$, determine the 2x2 matrix A in the state-space representation of the system described by the above second order differential equation, i.e., $Z' = AZ$, and give the value for matrix A here:

(Note to Class of 2T6: state-space representation corresponds to $Z' = AZ$.)

$$A = [\qquad]$$

b) Develop a MATLAB live script that numerically solves the IVP for $t \in [0,10]$ using both Euler's Method (EM) and Improved Euler's Method (IEM).

c) Within your live script, produce five separate plots using the following five different numbers of time steps $N = 45, 55, 75, 100, 250$, where $\Delta t = 10/N$. Each plot should include for each value of N the EM results, the IEM results, and the analytical solution, using different line types or colours, with the value of y plotted on the y-axis as a function of time on the x-axis. For plotting the analytical solution use a value of $\Delta t = 0.01$ in order to produce a smooth curve. Be sure to include appropriate axis labels, a legend, and a title with each plot.

d) Based on the values for N used in part (c), recommend and briefly explain your choice for the number of time steps needed for EM to provide a reasonable balance between accuracy and computational efficiency:

e) Based on the values for N used in part (c), recommend and briefly explain your choice for the number of time steps needed for IEM to provide a reasonable balance between accuracy and computational efficiency:

```
opengl software
opengl('save', 'software')
```

+1: Correct A matrix written on sheet

+1: Correct setup parameters including initial conditions and analytical solution

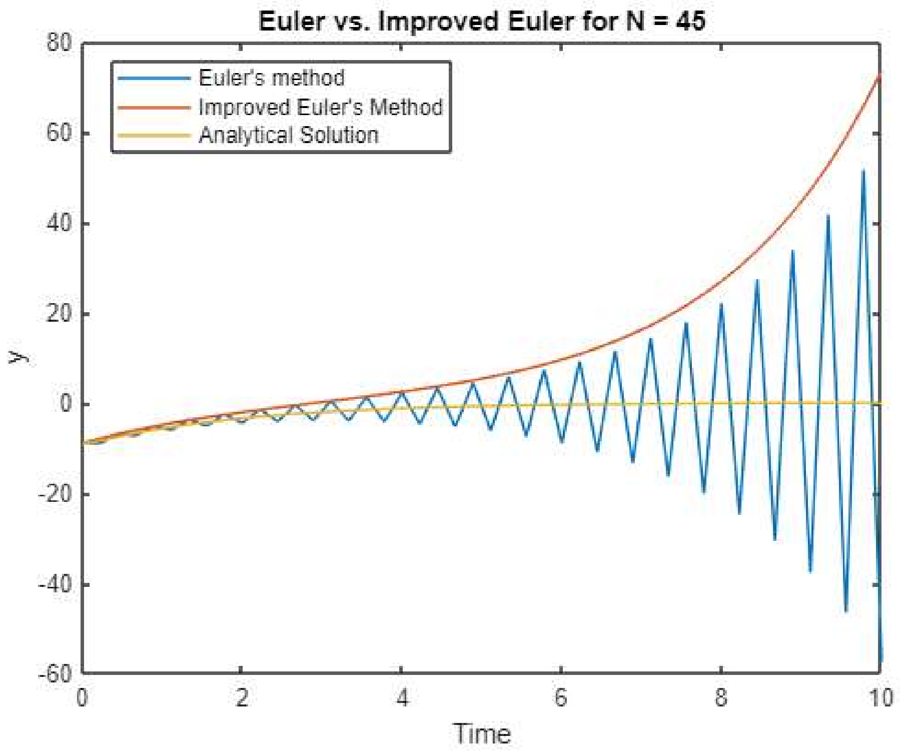
```
% Parameters
A = [0 1; -19/4 -10];
y0 = -9; yprime0 = 0;
N_arr = [45, 55, 75, 100, 250];
t_bounds = [0, 10];
delta_t_analytical = 0.01;

% Set up initial variables
t_analytical = t_bounds(1):delta_t_analytical:t_bounds(2);
y_analytical = -19/2*exp(-t_analytical/2) + 1/2*exp(-19/2*t_analytical);
```

+1: Iterates through all required time steps

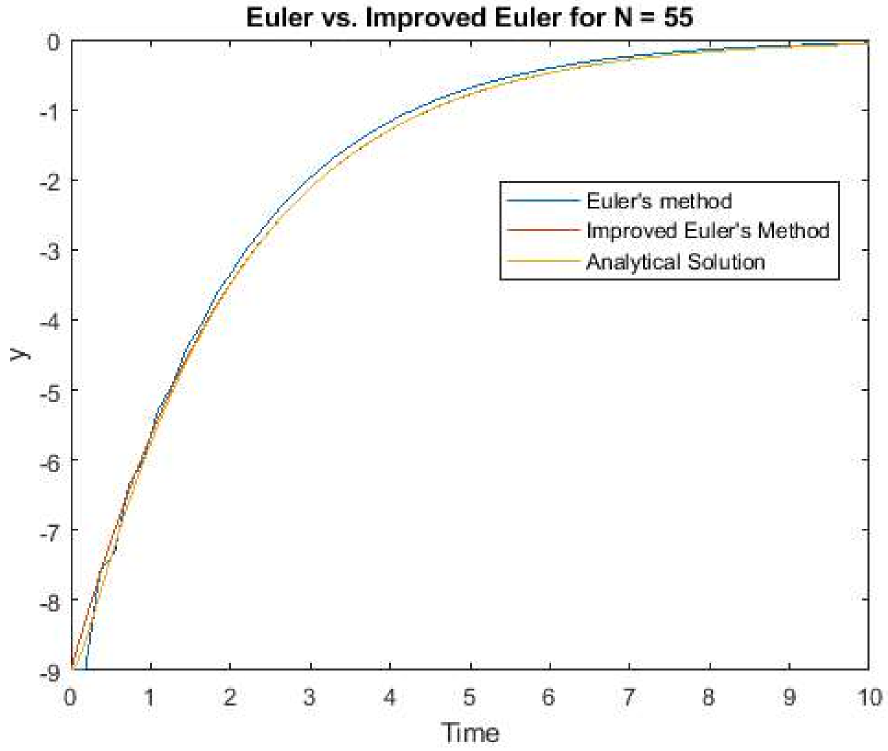
+1: Plots EM and IEM solution

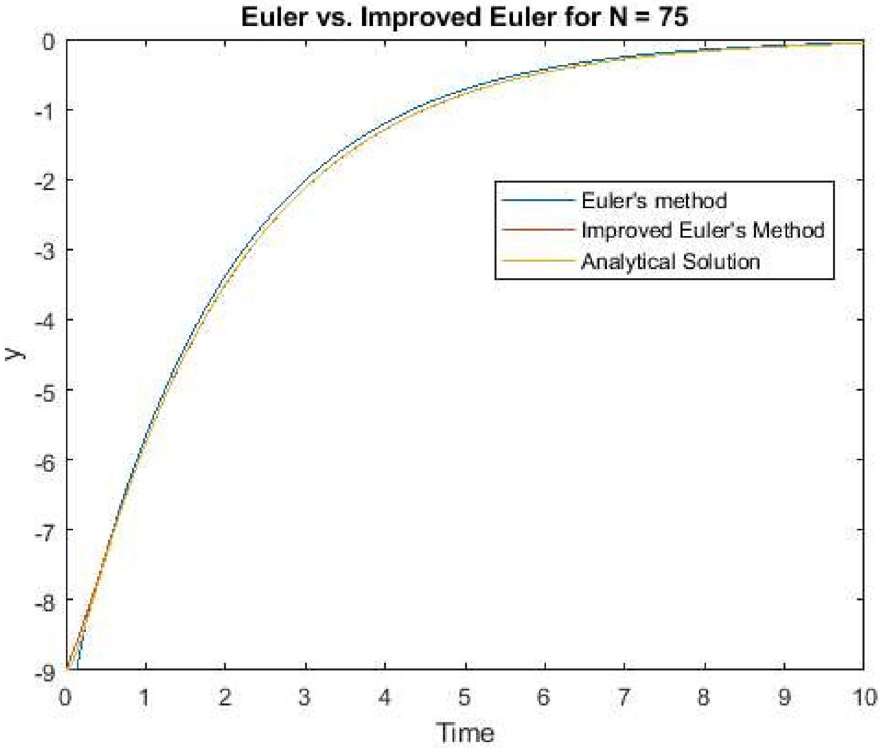
+1: Plots analytical solution

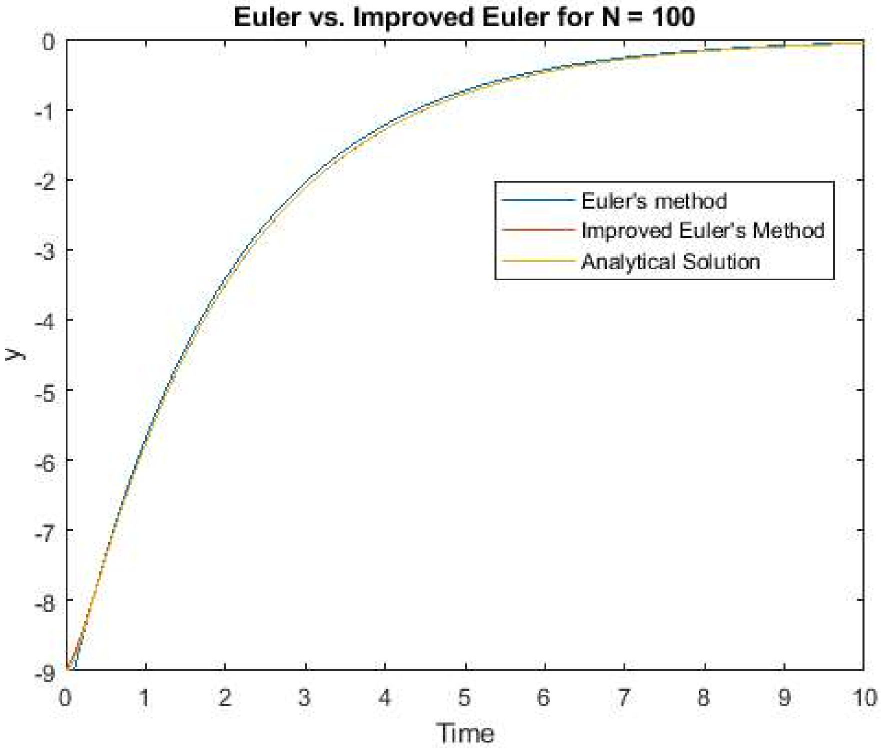
```
% Compute EM and IEM solutions

i = 1;
for N = N_arr
    figure(i)
    %Approximating solutions
    t = linspace(t_bounds(1), t_bounds(2), N + 1);
    y_EM = EM(A, y0, yprime0, t_bounds, N);
    y_IEM = IEM(A, y0, yprime0, t_bounds, N);

    %Plotting each figure
    plot(t, y_EM)
    hold on
    plot(t, y_IEM)
    hold on
    plot(t_analytical, y_analytical)
    title(sprintf('Euler vs. Improved Euler for N = %d', N))
    xlabel('Time')
    ylabel('y')
    legend("Euler's method", "Improved Euler's Method", "Analytical Solution", 'Location', 'best')

    i = i + 1;
end
```

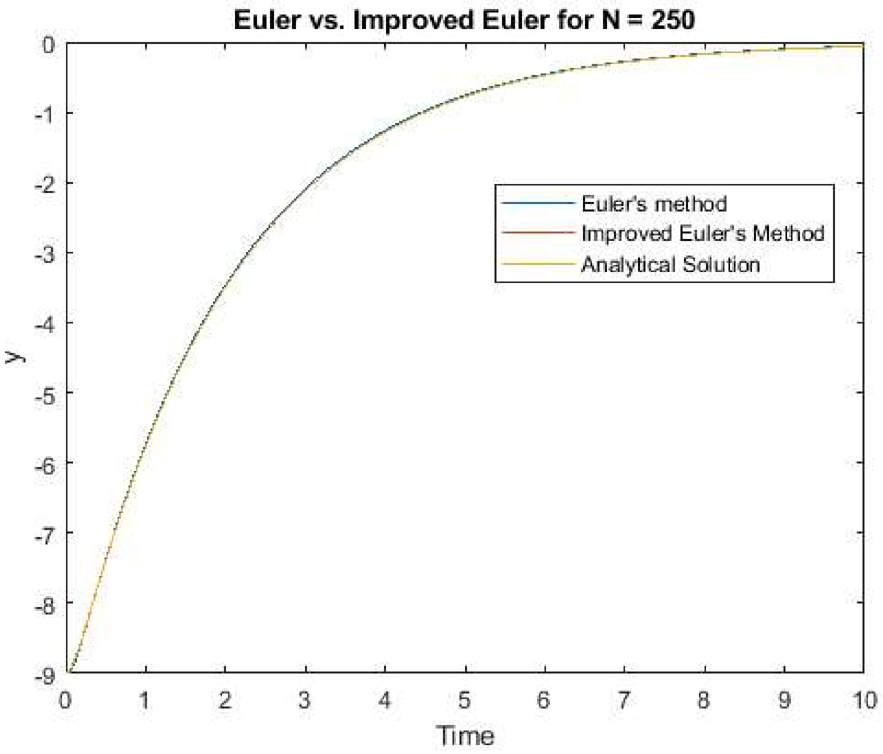


+1: Plots meet specifications
-0.5 for missing axis label on any plot
-0.5 for missing legend entry on any plot




Euler vs. Improved Euler for N = 75
Euler's method
Improved Euler's Method
Analytical Solution
y
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
Time
0
1
2
3
4
5
6
7
8
9
10


Euler vs. Improved Euler for N = 100
Euler's method
Improved Euler's Method
Analytical Solution
y
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
Time
0
1
2
3
4
5
6
7
8
9
10



Functions to solve for Euler's method and Improved Euler's method

```
function y = EM(A, y0, yprime0, t_bounds, N)
    delta_t = (t_bounds(2) - t_bounds(1)) /  N;
    Z = zeros(2, N + 1);
    Z(:, 1) = [y0; yprime0];

    for n = 1:N
        Z(:, n + 1) = Z(:, n) + delta_t*A*Z(:, n);
    end

    y = Z(1, :);
end
```

+1: Implements EM correctly

```
function y = IEM(A, y0, yprime0, t_bounds, N)
    delta_t = (t_bounds(2) - t_bounds(1)) /  N;
    Z = zeros(2, N + 1);
    Z(:, 1) = [y0; yprime0];

    for n = 1:N
        Z_nplus1 = Z(:, n) + delta_t*A*Z(:, n);
        Z(:, n + 1) = Z(:, n) + delta_t/2 * A * (Z(:, n) + Z_nplus1);
    end

    y = Z(1, :);
end
```

+1: Implements IEM correctly

+1:Correct written response to d): N=100 or N=250

+1:Correct written response to e): N=55