1. Consider the following definite integral:

$$\int_0^{30} 200\left(\frac{x}{5+x}\right) \exp\left(\frac{-2x}{30}\right) dx$$

This integral is difficult to evaluate analytically using the Fundamental Theorem of Calculus. Instead, you are asked to numerically estimate its value using a trapezoidal approximation (see formula given below).

In a single plot, show your trapezoidal results. Plot the estimates for the integral on the y-axis as a function of the number of subdivisions n on the x-axis with n ranging from 100 to 5000. Make sure your plot has appropriate axis labels.

Using the Text button at the top of the Live Editor tab (or type Alt + Enter), at the end of your file, indicate what you believe to be the true value for the definite integral (to two decimal places) based on your results.

Coding requirements:

- You must use Live Editor to create a script for solving this problem;
- Your live script must contain and make use of a function called TRAPsolver(n) that solves the numerical integration problem using the trapezoidal rule and returns the approximate area for n subdivisions;
- Your live script must contain a section where you plot your results.

Given information:

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

Exponential function in MATLAB: exp

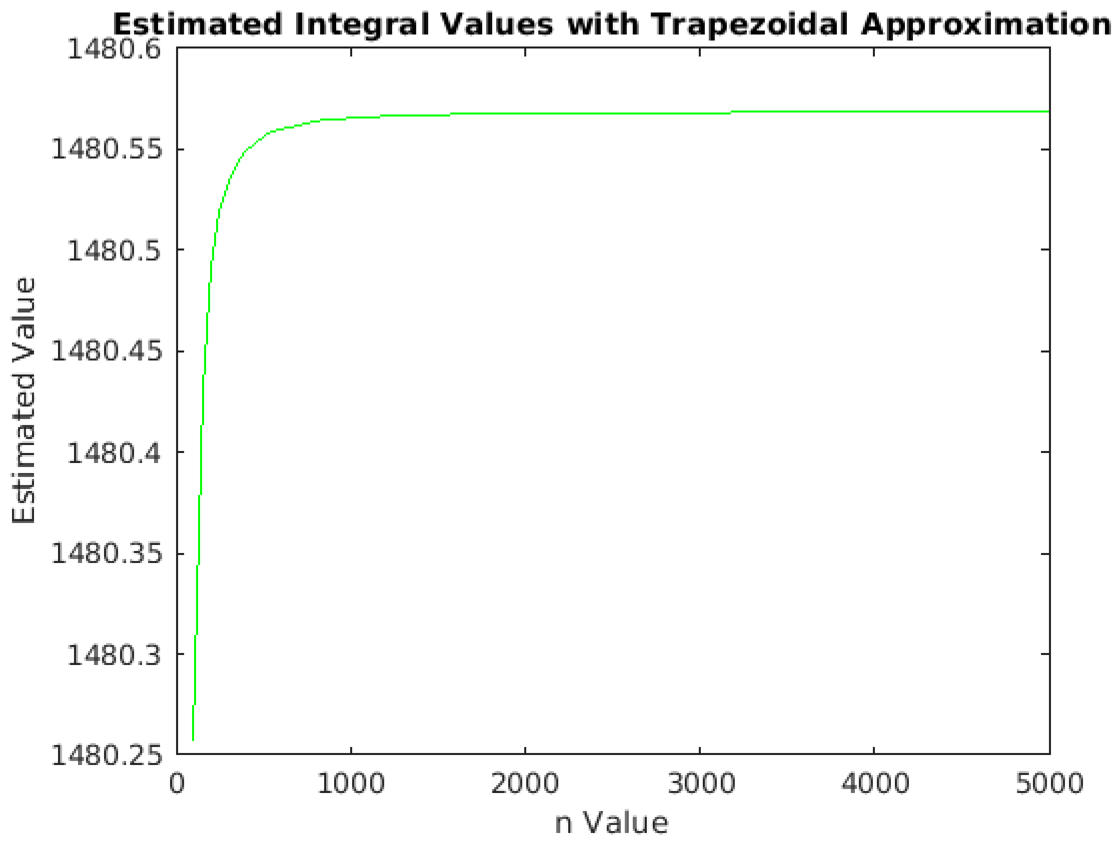
```
format long
```

```
n_values = 100:50:5000
```

```
n_values = 1×99
         100         150         200         250         300         350 ···
```

```
totals = zeros(length(n_values), 1);
for i=1:length(n_values)
    n = n_values(i);
    totals(i) = TRAPsolver(n);
end
```

```
figure
plot(n_values, totals, '-g')
xlabel("n Value")
ylabel("Estimated Value")
title("Estimated Integral Values with Trapezoidal Approximation")
```



```
% To get a good estimate, we just use an absurdly large n
TRAPsolver(100000)
```

```
ans = 
     1.480568479775100e+03
```

```
function value = TRAPsolver(n)
    start_x = 0;
    end_x = 30;
    dx = (end_x - start_x) / n;
    x_values = start_x:dx:end_x;
    y_values = 200 * (x_values ./ (5 + x_values)) .* exp((-2*x_values)/30);
    value = 0;
    for j=1:length(x_values)-1
        value = value + ((y_values(j) + y_values(j+1)) / 2)*dx;
    end
end
```

Given the value of the integral estimated with a very large n of 100000, the true value of the integral is approximately 1480.57.

2. The rate of cooling of a body can be expressed in terms of the following first order differential equation:

$$\frac{dT(t)}{dt} = -k(T(t) - T_a)$$

where

$$T(t) = temperature\ of\ the\ body\ (degrees\ C)$$

$$T_a = temperature\ of\ the\ surrounding\ medium\ (degrees\ C)$$

$$k = a\ proportionality\ constant\ (min^{-1})$$

The differential equation specifies that the rate of cooling of the body is proportional to the difference in temperature between the body and the surrounding medium.

Assume that a metal ball (the body) is heated to 90 degrees C and is dropped at $t = 0$ into water (the surrounding medium) that is being held at a constant value of $T_a = 20$ degrees C. Use Improved Euler's Method (IEM) to numerically solve this initial value problem for $t \in [0,20\ min]$ with $k = 0.25\ min^{-1}$.

Provide a single plot showing the estimated temperature of the metal ball T on the y-axis as a function of time on the x-axis. Use different line types for different time steps $N = 3, 5, 10, 20$ and include appropriate axis labels and a legend.

Using the Text button at the top of the Live Editor tab (or type Alt + Enter), at the end of your file, indicate your recommended choice for the number of time steps that provides reasonable accuracy and is computationally efficient.

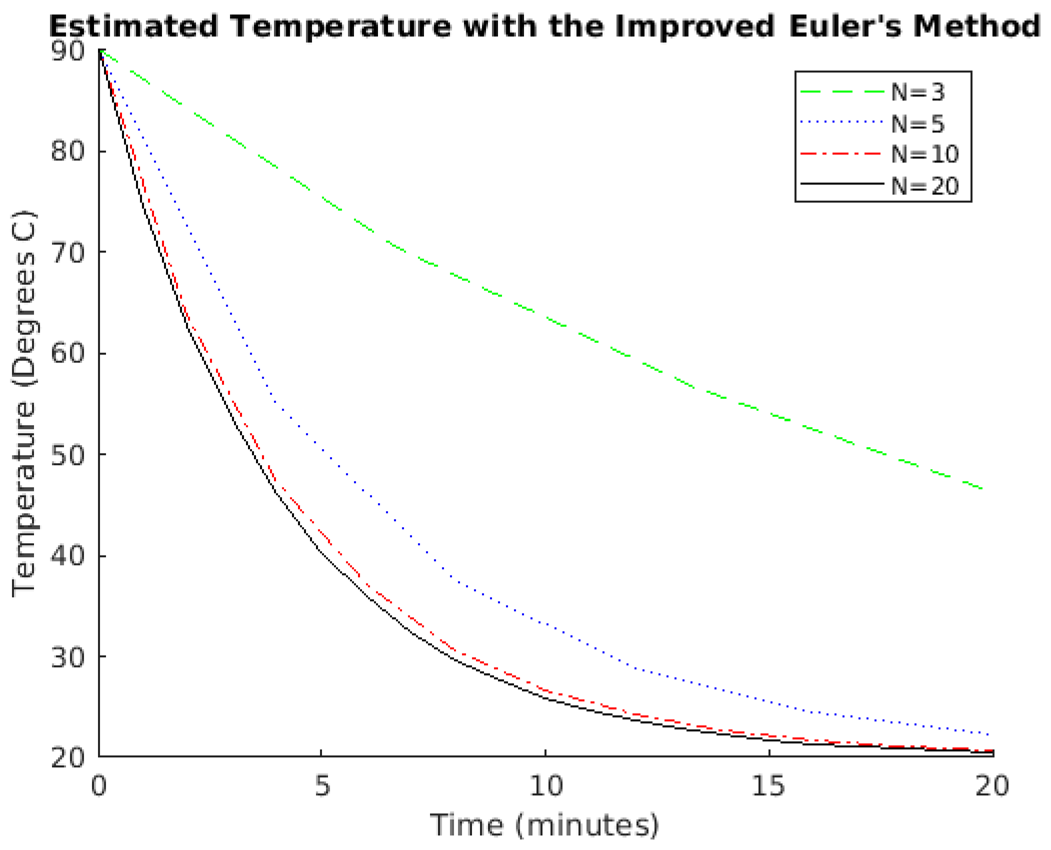
In a second text field at the end of your file, provide an estimate of how long it takes the ball to cool from its initial temperature of 90 degrees C to 40 degrees C.

Coding requirements:

- You must use Live Editor to create a script for solving this problem;
- Your live script must contain a function called IEMsolver(N) that solves this IVP using IEM for N time steps and returns two arrays of length N, one for time and the other for temperature;
- Your live script must contain a section where you plot your results.

```
[t_3, temp_3] = IEMsolver(3);
[t_5, temp_5] = IEMsolver(5);
[t_10, temp_10] = IEMsolver(10);
[t_20, temp_20] = IEMsolver(20);
```

```
% Honestly, the insructions said to use different line types and I wasn't
% sure whether you wanted different colors or different line types as
% well so I just did both.
figure
hold on
plot(t_3, temp_3, '--g')
plot(t_5, temp_5, ':b')
plot(t_10, temp_10, '-.r')
plot(t_20, temp_20, '-k')
hold off
legend("N=3", "N=5", "N=10", "N=20")
xlabel("Time (minutes)")
ylabel("Temperature (Degrees C)")
title("Estimated Temperature with the Improved Euler's Method")
```



```
function [t, T] = IEMsolver(N)
    T_0 = 90;
    T_a = 20;
    k = 0.25;
    t_start = 0;
    t_end = 20;
    dt = (t_end - t_start) / N;
    t = t_start:dt:t_end;
    SOL = NaN(1,length(t));
    SOL(1) = T_0;

    for j = 2:length(t)
        pos_0 = SOL(j-1);
        m = -1*k*(pos_0 - T_a);
        pos_1 = pos_0 + dt*m;
        m_1 = -1*k*(pos_1 - T_a);
        SOL(j) = pos_0 + (dt/2)*(m + m_1);
    end

    T = SOL(:);
end
```

Both N=10 and N=20 produce very similar solutions which indicates that using anything above N=10 is probably overkill for general usage. At N=5 and below, the solution will eventually come to 20 degrees, but the intermediate time steps are no longer accurate so I would not advise using anything less than N=10.

It appears that it takes almost exactly 5 minutes to reach 40 degrees.

3. Experimental data from a physics lab has been collected and five experiments have been performed to determine if there is a relationship between a particular independent variable and a dependent response variable. The table below shows the different values of the independent variable (v_1) used for each experiment and the corresponding dependent response variable (v_2).

Experiment #	$\boldsymbol{v_1}$	$\boldsymbol{v_2}$
1	2	5
2	3	7
3	4	8
4	5	11
5	6	12

Let us say that the objective is to fit a straight-line model for the relationship, $v_2 = c_1 + c_2 v_1$, using all five data points. Start by setting up the A matrix and $\vec{b}$ vector for the corresponding system of linear equations ($A\vec{x} = \vec{b}$). Then, find the least squares fit of a straight line.

Provide a single plot that shows all five data points and the straight-line fit, with v_1 on the x-axis and v_2 on the y-axis. Remember to include appropriate axis labels and a legend.

Using the Text button at the top of the Live Editor tab (or type Alt + Enter), at the end of your file, indicate your predicted value for v_2 using the straight-line model corresponding to $v_1 = 4.6$.

Coding requirements:

- You must use Live Editor to create a script for solving this problem;
- Your live script must contain a function called LSsolver(A, b) that solves for the least squares estimates of c_1 and c_2 and returns c_1 and c_2 in an array;
- Your live script must contain a section where you plot your results.

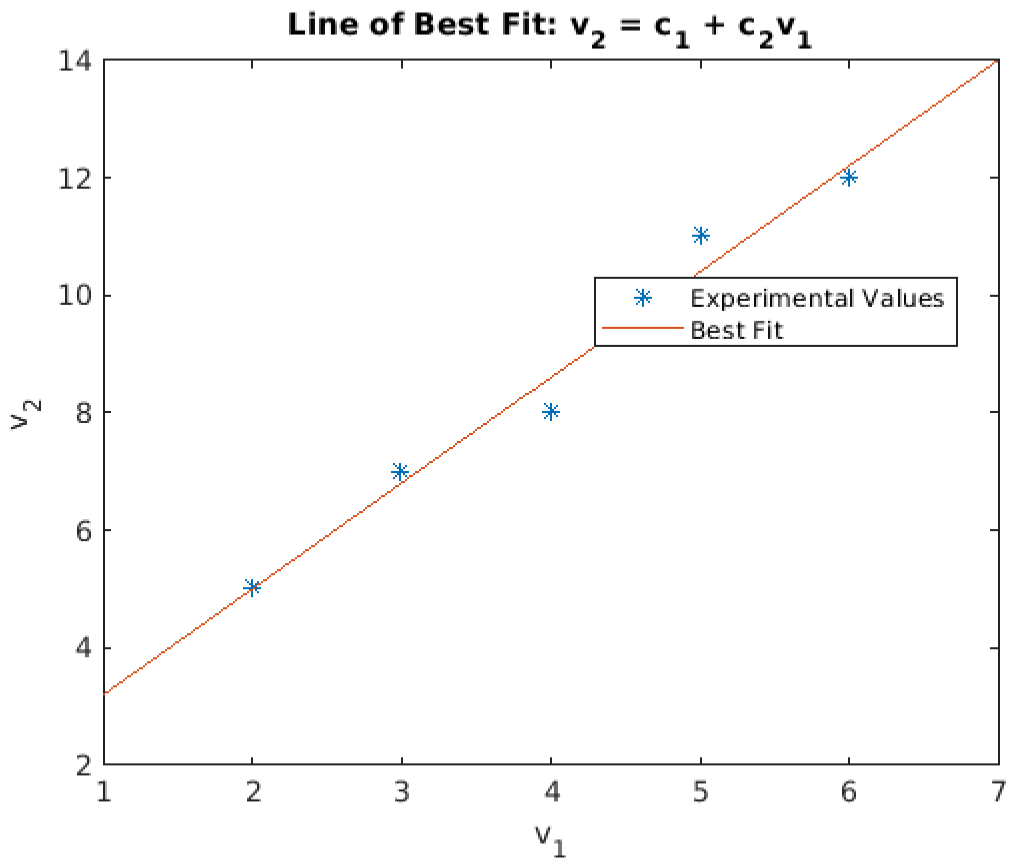
```
v_1 = [2, 3, 4, 5, 6];
v_2 = [5, 7, 8, 11, 12];
```

```
% We will first solve for a linear equation
A = ones(length(v_1), 2);
b = zeros(length(v_2), 1);

for i=1:length(v_1)
    A(i, 2) = v_1(i);
    b(i) = v_2(i);
end

c = LSsolver(A, b);
x = [1, 7];
y = c(1) + c(2)*x;
```

```
figure
plot(v_1, v_2, '*')
hold on
plot(x, y)
legend("Experimental Values", "Best Fit", "Location", "best")
xlabel("v_1")
ylabel("v_2")
title("Line of Best Fit: v_2 = c_1 + c_2v_1")
```



```
predicted_value = c(1) + c(2) * 4.6
```

```
predicted_value =
   9.680000000000000
```

```
function c = LSsolver(A, b)
    A_transpose = A';
    new_A = A_transpose * A;
    new_b = A_transpose * b;
    params = linsolve(new_A, new_b);
    c = params;
end
```

Our predicted v_2 for an input of v_1=4.6 is 9.68