Assignment 8 Solution

Problem 1 (6 points)

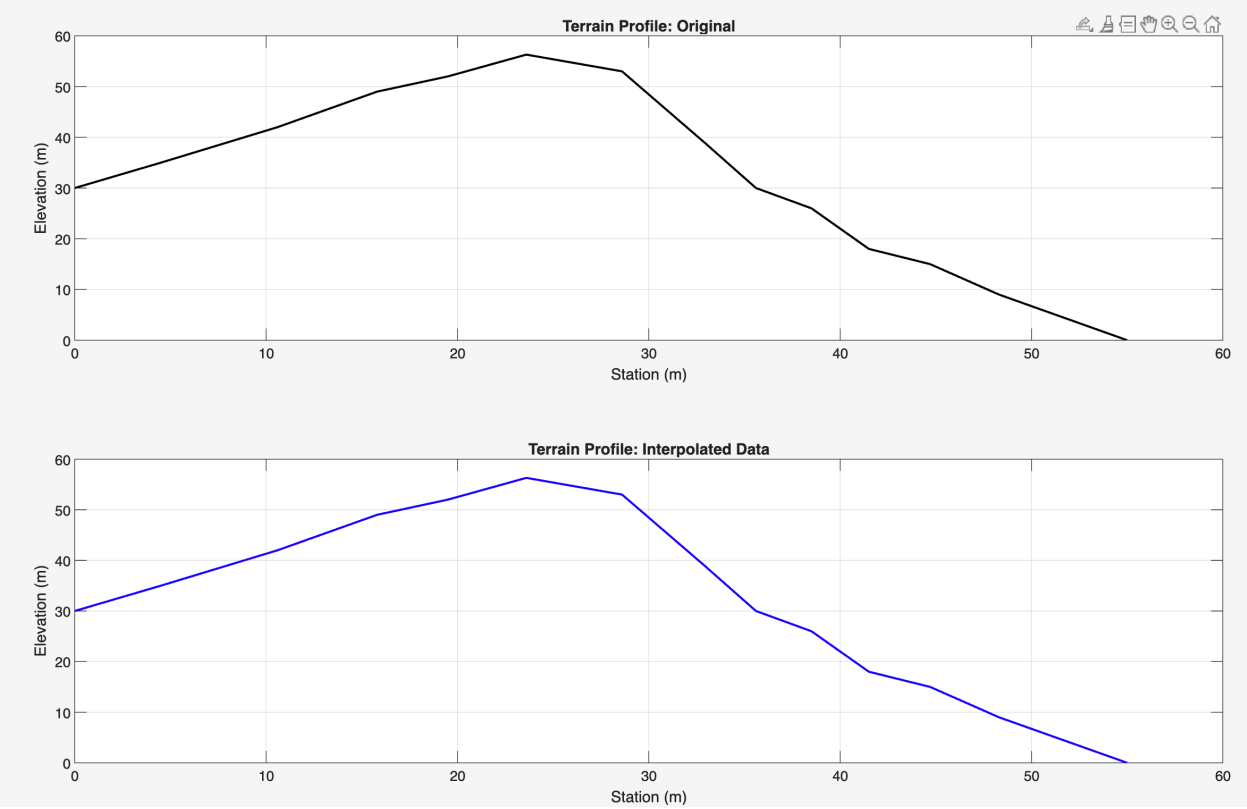
You are task to calculate the volume of earth for a highway project in Virginia. A cross section of the terrain to be removed in the project is shown in Table 1.

Task 1. Create a Matlab script to read the data. Copy the data into another file (Matlab, Excel, etc.).

```
% CEE 3804 Spring 2026 Problem 1 Solution
% Programmer: Jeongwoo Park

clear all
close all
clc

%% TASK 1
CrossSection = load("CrossSection_Terrain.m");

station = CrossSection(:,1); % Unit: meters
elevation = CrossSection(:,2); % Unit: meters
```

Task 2. Add MATLAB code to the script created in Task 1 to interpolate linearly values of elevation from stations 0 to 55 at intervals of 10 centimeters. The goal is to produce a smooth profile of the terrain before calculating the area under the curve in Task 3. Make a plot of the original data and the interpolated data.

```
%% TASK 2
% Range of interpolation
station_interpolation = 0:0.1:55; % 10cm intervals

% Interpolation
elevation_interpolation = interp1(station, elevation, station_interpolation);

% Plot original and interpolated data
figure;
subplot(2,1,1)
plot(station, elevation, 'k-', 'LineWidth', 1.5);
grid on;
xlabel('Station (m)');
ylabel('Elevation (m)');
title('Terrain Profile: Original');

subplot(2,1,2)
plot(station_interpolation, elevation_interpolation, 'b-', 'LineWidth', 1.5);
grid on;
xlabel('Station (m)');
ylabel('Elevation (m)');
title('Terrain Profile: Interpolated Data');
```



Task 3. Estimate the area under the curve projected by the elevation across each station. Use either the Trapezoidal rule. Display the value of the area under the curve in the Command window and also write the value in the title of the plot generated in Task 2.

The area under the curve projected in the original data is 1864.115 sq.m

The area under the curve projected in interpolated data is 1864.115 sq.m

```
37    %% TASK 3
38    CurveProjected = trapz(station, elevation);
39    curveprojected_interpolation = trapz(station_interpolation, elevation_interpolation);
40    disp(['The area under the curve projected in the original data is ', num2str(CurveProjected), ' sq.m'])
41    disp(['The area under the curve projected in interpolated data is ', num2str(curveprojected_interpolation), ' sq.m'])
42
```

Command Window

```
The area under the curve projected in the original data is 1864.115 sq.m
The area under the curve projected in interpolated data is 1864.115 sq.m
>> ✨ Press ⇧⌘P to generate code with Copilot
```

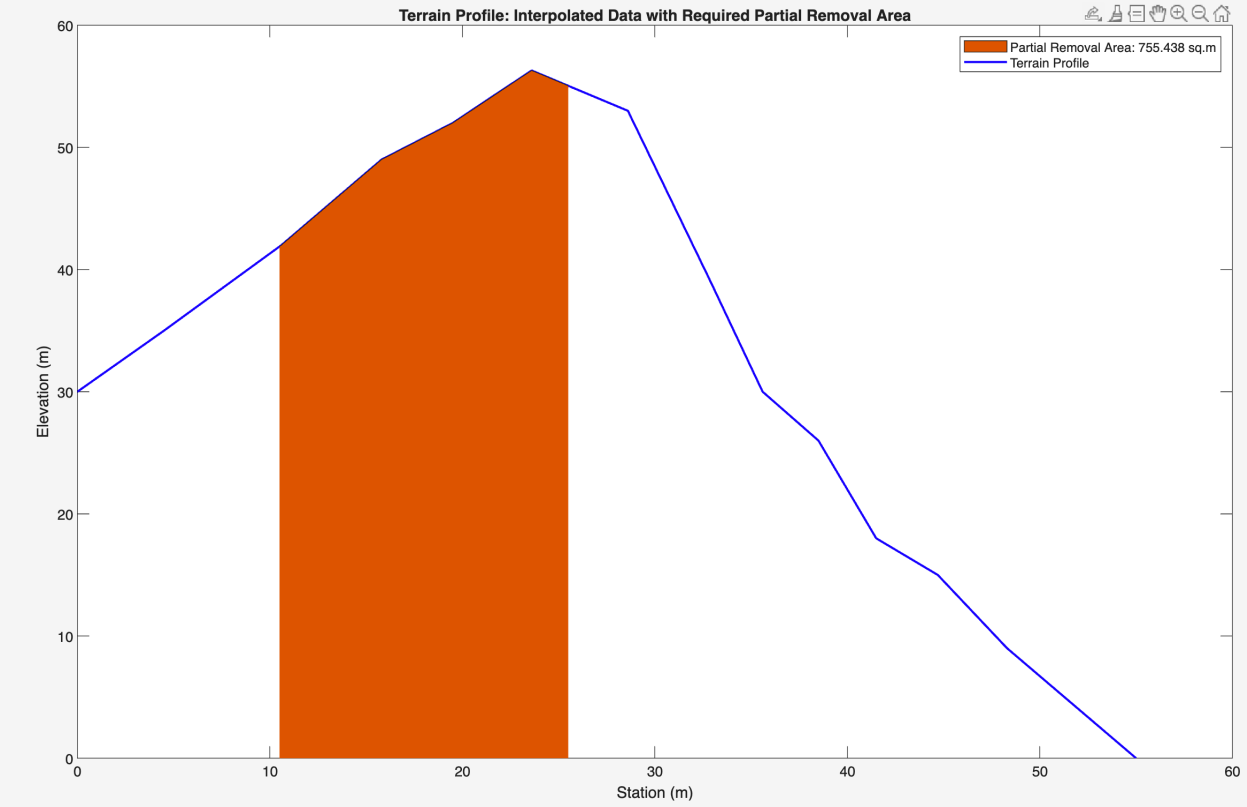
Task 4. Use the Matlab area command to demonstrate a partial removal of terrain between stations 10.5 and 25.5 meters. Show the area under the partial removal in the plot.

The area of partial removal is 755.438 sq.m

```
43  %% TASK 4
44  % Find the indices that required the partial removal
45  Partial_Removal_index = station_interpolation>=10.5&station_interpolation<=25.5;
46
47  % Find the values of each station and elevation that required the partial removal
48  Partial_Removal_Station = station_interpolation(Partial_Removal_index);
49  Partial_Removal_Elevation = elevation_interpolation(Partial_Removal_index);
50
51  % Find the area that required the partial removal
52  partial_removal_area = trapz(Partial_Removal_Station, Partial_Removal_Elevation);
53
54  disp(['The area of partial removal is ', num2str(partial_removal_area), ' sq.m'])
55
56  figure
57  plot(station_interpolation, elevation_interpolation, 'b-', 'LineWidth', 1.5);
58  hold on
59  % Draw a partial removal area
60  area(Partial_Removal_Station, Partial_Removal_Elevation);
61  legend('Terrain Profile', ['Partial Removal Area: ', num2str(partial_removal_area), ' sq.m'])
62  title('Terrain Profile: Interpolated Data with Required Partial Removal Area');
63  xlabel('Station (m)');
64  ylabel('Elevation (m)');
65
66
```

Command Window

```
The area under the curve projected in the original data is 1864.115 sq.m
The area under the curve projected in interpolated data is 1864.115 sq.m
The area of partial removal is 755.438 sq.m
```

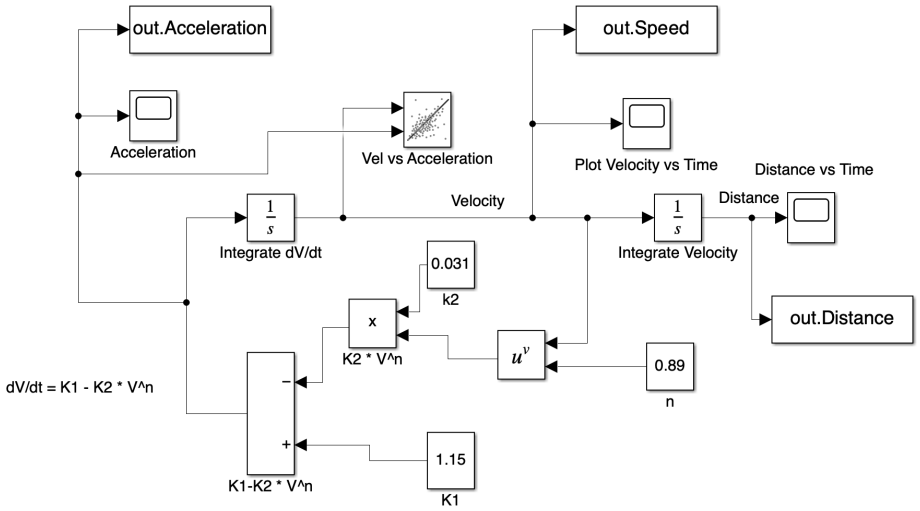


Problem 2 (7 points)

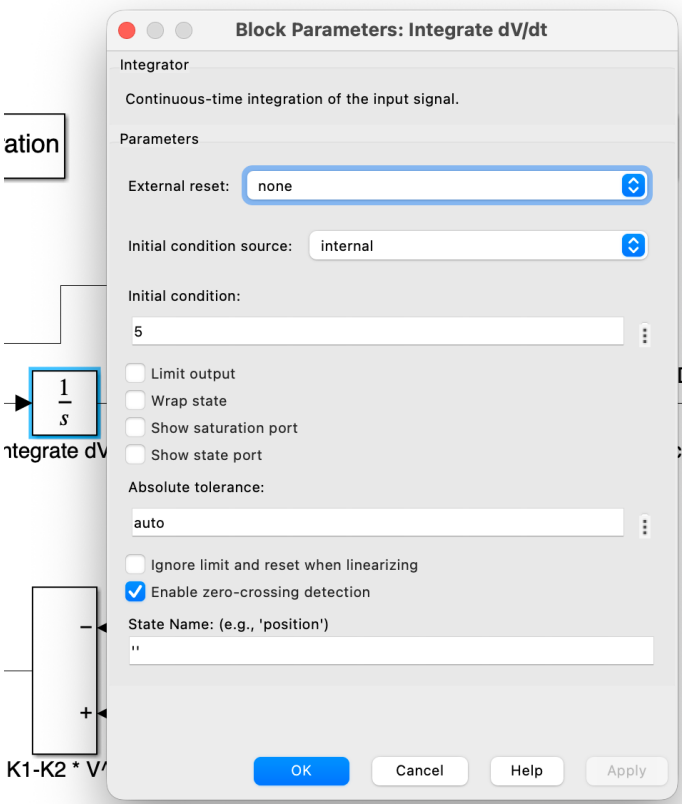
A civil engineer is designing acceleration ramps for a new highway. Figure 1 who's the typical acceleration ramp configuration. The highway is used by a variety of vehicles including cars, light trucks, and heavy trucks. The engineer collected data about three types of vehicles to design the acceleration ramp (L) shown in Figure 1.

Task 1. Create Matlab code (using ODE solver) or a Simulink model to solve the differential equation that estimates vehicle speed as a function of time. You can reuse any of the Matlab code or Simulink models provided in class.

Simulink



Set an initial speed of 5 m/s when integrating the speed.




```
%% TASK 1 and 2
speed = out.Speed.Data;
time = out.Speed.Time;
distance = out.Distance.Data;
acceleration = out.Acceleration.Data;
```

Matlab ODE Solver

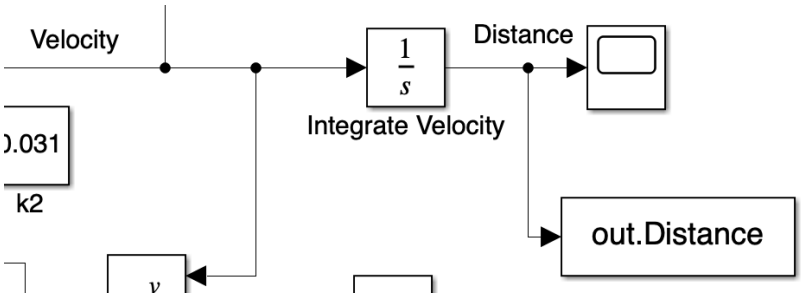
```
% CEE 3804 Spring 2026 Problem 2 Solution
% Programmer: Jeongwoo Park

clear
close all
clc

global k1 k2 n
k1 = 1.15;
k2 = 0.031;
n  = 0.89;

% Range of Speed
tspan = [0 40]; % Unit: m/s
y0 = 5; % Initial Speed

% TASK 1
[time, speed] = ode45(@ODE_ACC, tspan, y0);
```

```
function acceleration = ODE_ACC(t, v)
    global k1 k2 n
    acceleration = k1 - k2 * v.^n;
end
```

Task 2. Enhance the model of Task 1 model to predict the distance traveled by the vehicle as it accelerates on the ramp.

Simulink

Integrate a speed using a integrate block.



Matlab ODE Solver

I used cumtrapz, which is cumulative trapezoidal numerical integration. You can also use the ODE solver again to find the distance.

```
% TASK 2
distance = cumtrapz(time, speed);
```

Task 3. Use the model created in Tasks 1 and 2 to predict the length of the acceleration ramp if a loaded truck is used as the critical vehicle. Assume that the vehicle at location A has an initial speed of 5 m/s (very low speed to simulate a ramp metering light). Compare the acceleration ramp length for a loaded truck and a car.

Simulink

The estimated length of the acceleration ramp for the **heavy truck** is 616 m

The estimated length of the acceleration ramp for the **light truck** is 397 m

The estimated length of the acceleration ramp for the **car** is 331 m

If you haven't set an initial speed of 5 m/s when you integrate, you will get a different distance. (629 meters for heavy truck, 376 meters for light truck, and 325 meters for the car)

```
%% TASK 3
speedmph = speed .* 2.237; % Conver the m/s to mph
speed_limit = 65; % Typical speed limit (Adjustable)

% Find the index where the speed reach the speed limit
idx = find(speedmph >= speed_limit, 1, 'first');
time_target = time(idx); % time of it
dist_ramp = distance(idx); % Distance of it

disp(['The estimated length of the acceleration ramp for the truck is ', num2str(dist_ramp), ' m'])
```

There is a difference between the Simulink and ODE solver because I used cumtrapz (cumulative integral) to calculate a distance, or Simulink may use a different solver. But it is still an acceptable difference.

Matlab ODE Solver

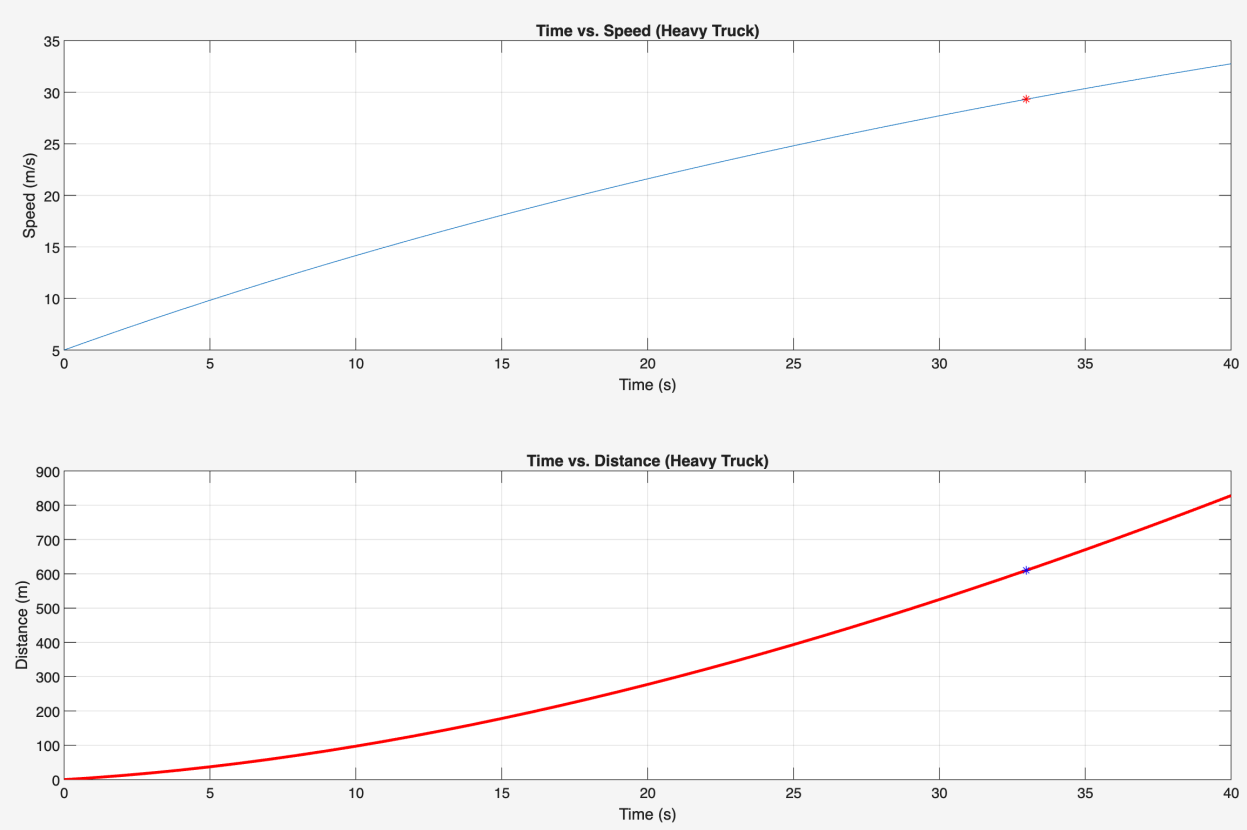
The estimated length of the acceleration ramp for the **heavy truck** is 610.0847 m

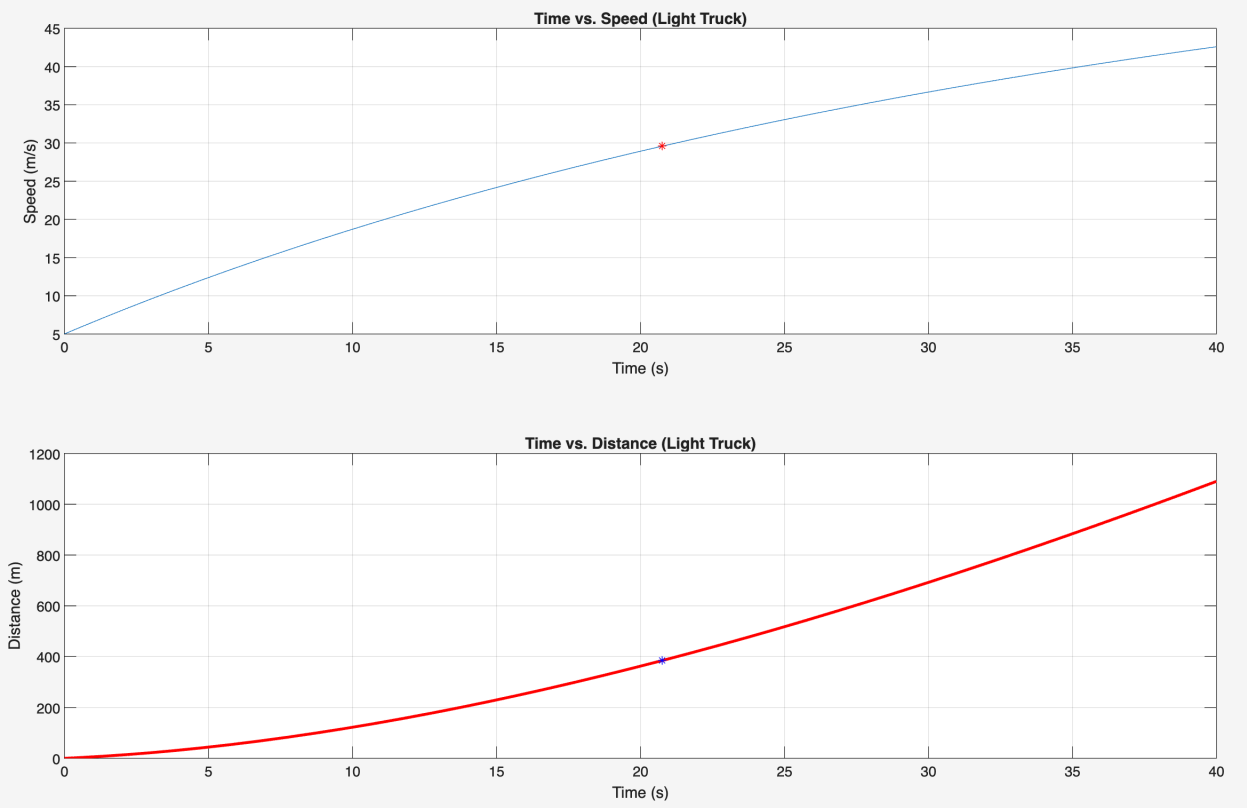
The estimated length of the acceleration ramp for the **light truck** is 384.912 m

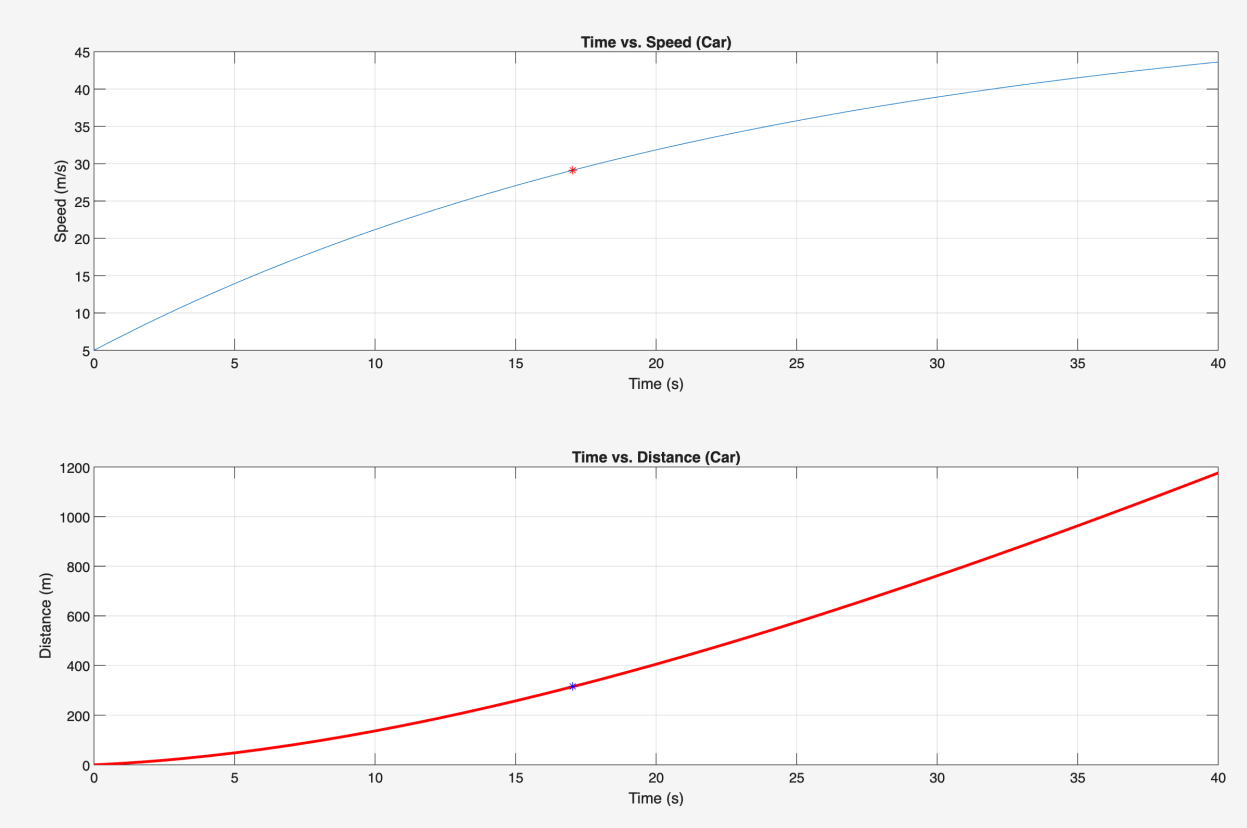
The estimated length of the acceleration ramp for the **car** is 314.9497 m

```
%% TASK 3
speedmph = speed .* 2.237; % Conver the m/s to mph
speed_limit = 65; % Typical speed limit (Adjustable)

% Find the index where the speed reach the speed limit
idx = find(speedmph >= speed_limit, 1, 'first');
time_target = time(idx); % time of it
dist_ramp = distance(idx); % Distance of it

disp(['The estimated length of the acceleration ramp for the truck is ', num2str(dist_ramp), ' m'])

% Plot the data
figure
subplot(2,1,1)
plot(time,speed)
hold on
plot(time(idx), speed(idx), '*r')

title('Time vs. Speed (Truck)')
xlabel('Time (s)')
ylabel('Speed (m/s)')
grid on

subplot(2,1,2)
plot(time, distance, 'r', 'LineWidth', 2)
hold on
plot(time(idx), distance(idx), '*b')

title('Time vs. Distance (Truck)')
xlabel('Time (s)');
ylabel('Distance (m)');
grid on
```

```
Command Window
The estimated length of the acceleration ramp for the truck is 610.0847 m
```


Time vs. Speed (Heavy Truck)
Speed (m/s)
Time (s)
Time vs. Distance (Heavy Truck)
Distance (m)
Time (s)


Time vs. Speed (Light Truck)
Speed (m/s)
Time (s)
Time vs. Distance (Light Truck)
Distance (m)
Time (s)


Time vs. Speed (Car)
Speed (m/s)
Time (s)
Time vs. Distance (Car)
Distance (m)
Time (s)

Problem 3 (7 points)

Figure 2 presents consumption data and emissions is for a small SUV vehicle. The fuel consumption and emissions are presented in columns 3 and 4. The data was collected by the Oak Ridge National Lab.

Task 1. Create Matlab script to read the data. Use the method of your choice.

```
%Speed (m/s) Fuel Consumption (l/s) CO2 Emission(mg/s)

0.00 0.00026 1.89
2.75 0.00035 2.15
4.15 0.00042 3.15
6.00 0.00046 4.78
8.35 0.00057 6.60
11.00 0.000666 9.30
14.20 0.00078 12.80
17.30 0.00092 17.30
19.50 0.00107 21.90
22.25 0.00126 33.60
25.50 0.00145 54.30
28.00 0.00185 97.60
```

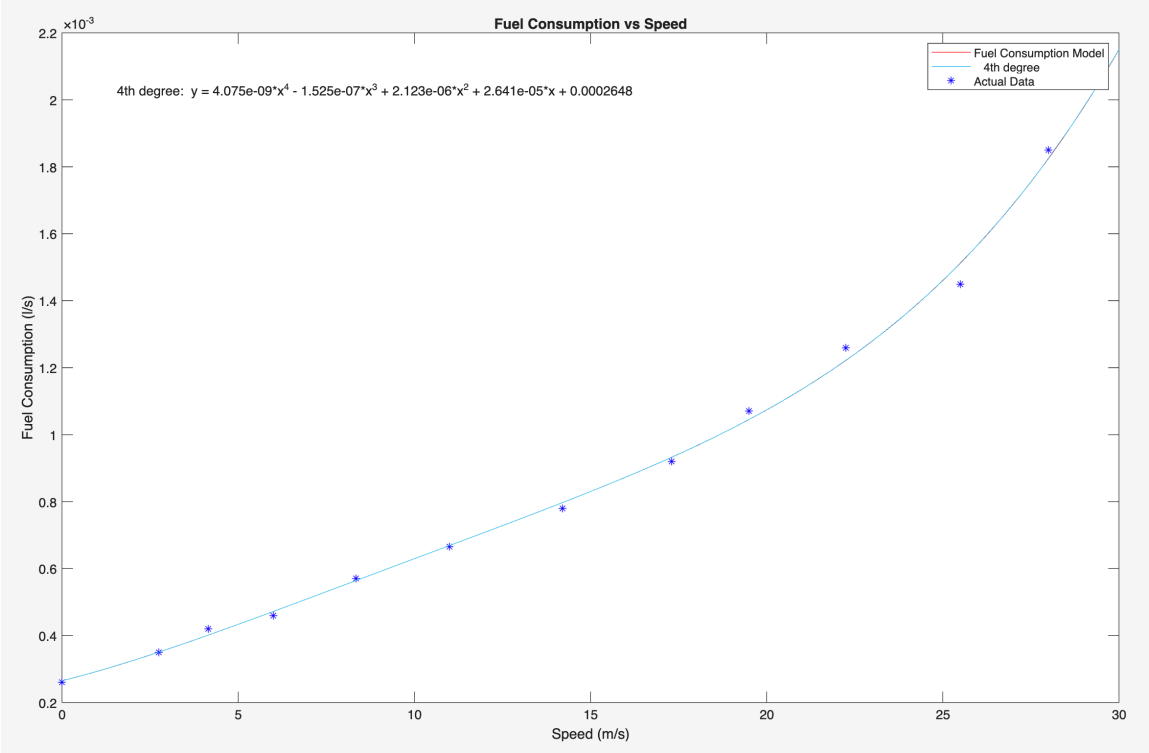
```
% CEE 3804 Spring 2026 Problem 2 Solution
% Programmer: Jeongwoo Park

clear all
close all
clc
```

```
%% TASK 1
Cons_Data = load("Consumption_Data.m");
Speed = Cons_Data(:, 1);
Fuel_Consumption = Cons_Data(:, 2);
CO2_Emissions = Cons_Data(:, 3);
```

Task 2. Add code to the script created in Task 1 to find the best fourth-order polynomial to predict fuel consumption (dependent variable) as a function of speed (independent variable). Display the polynomial in the Command Window and make a screen capture.

Task 3. Make a plot (in code) of the fuel consumption data and also show your polynomial fit in the same plot.

```
%% TASK 2 and 3
% Create a 4th order polynomial to predict fuel consumption
fuel_coefficients = polyfit(Speed, Fuel_Consumption, 4);
Speed_Vector = 0:0.1:30;
fuel_model = polyval(fuel_coefficients, Speed_Vector);

% Plot the data
figure
plot(Speed_Vector, fuel_model, '-r')
hold on
plot(Speed, Fuel_Consumption, 'b*')
xlabel('Speed (m/s)');
ylabel('Fuel Consumption (l/s)');
title('Fuel Consumption vs Speed');
legend('Fuel Consumption Model', 'Actual Data');
```



Task 4. Add code to find the best high-order polynomial to predict vehicle emissions (mg/s) as a function of speed (m/s).

Task 5. Make a plot (in code) of the emissions data and also show your polynomial fit in the same plot. Comment on the quality of the polynomials to fit the data.

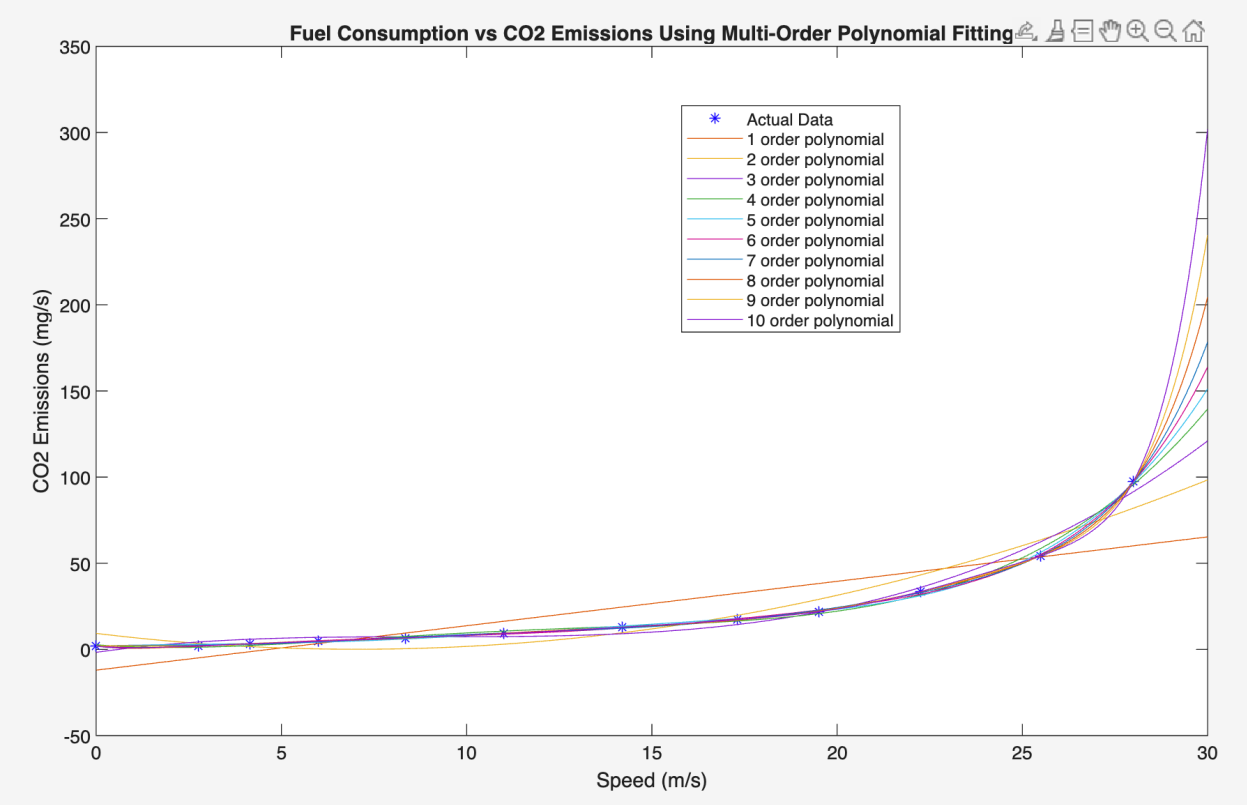
There are several metrics to investigate in the fitted model. I used an adjusted R^2 value to identify the best model for predicting emissions as a function of speed. Because models of higher order are prone to overfitting by returning a high R^2 value in higher orders. Adjusted R^2 is a modified R^2 by accounting for the degrees of freedom, which is the number of independent variables that can be estimated. It is more reliable than the normal R^2 value for higher-order polynomials.

I set a threshold of adjusted R^2 of 0.99, which is sufficient to identify the best model while avoiding overfitting. Therefore, my code finds the polynomial at which the first reaches an adjusted R^2 of 0.99. The results show that a 4th-order polynomial is enough to say it's the best model. Again, higher order will provide a higher R^2 value; however, it overfits. A higher value is not always better. As you can see in the figure below, the polynomial is badly conditioned in higher orders. We are required to add points with distinct X values or reduce the degree of the polynomial.

```
%% TASK 4 and 5
%Create a 4th order polynomial to predict CO2 emissions
figure
plot(Speed, CO2_Emissions, 'b*', Displayname = 'Actual Data')
hold on

n = length(CO2_Emissions);

for i = 1:10
    [Co2_coefficients, ErrorEstimator] = polyfit(Speed, CO2_Emissions, i);
    Co2_model = polyval(Co2_coefficients, Speed_Vector);
    plot(Speed_Vector, Co2_model, Displayname=[num2str(i), ' order polynomial'])

    % Error Estimators from the Polyfit
    R2 = ErrorEstimator.rsquared;
    degreefreedom = ErrorEstimator.df;

    % Calculate Adjusted R^2
    adj_R2(i) = 1 - (1 - R2) * (n - 1) / degreefreedom;
end
xlabel('Speed (m/s)');
ylabel('CO2 Emissions (mg/s)');
title('Fuel Consumption vs CO2 Emissions Using Multi-Order Polynomial Fitting');
legend('Location','best')

% Set a threshold to find the best model
% Find the polynomial where the first reached an adjusted R^2 of 0.99.
% The R^2 value will increase by going higher order (Called overfitting).
% The adjusted R^2 of 0.99 is enough to identify the best model, avoiding overfitting.
adjr2_threshold = 0.99;
idx = find(adj_R2 >= adjr2_threshold, 1, 'first');
disp(['The best higher-order polynomial is a ', num2str(idx), 'th-order polynomial, with an adjust
```

Command Window

```
Warning: Polynomial is badly conditioned. Add points with distinct X values, reduce the degree of the polynomial, or try centering and
scaling as described in HELP POLYFIT.
> In polyfit (line 85)
In Problem3 (line 40)
The best higher-order polynomial is a 4th-order polynomial, with an adjusted R^2 of 0.99403.
```


Fuel Consumption vs CO2 Emissions Using Multi-Order Polynomial Fitting
CO2 Emissions (mg/s)
Speed (m/s)
Actual Data
1 order polynomial
2 order polynomial
3 order polynomial
4 order polynomial
5 order polynomial
6 order polynomial
7 order polynomial
8 order polynomial
9 order polynomial
10 order polynomial
350
300
250
200
150
100
50
0
-50
0
5
10
15
20
25
30