Table of Contents


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CODE CHALLENGE 4 - Linear Least-Squares Fit
%
% The purpose of this program is to calculate the equation of the best
 fit
% line for a data set using linear least-squares fitting.
%
% To complete the challenge, finish the code below to:
% 1) load data from csv file
% 2) find linear best fit coefficients and associated uncertainty
% 3) plot the original data along with the best fit line
% 4) add errorbars for fit uncertainty to this plot from the data and
 from
%    the linear regression parameters
%
% NOTE: DO NOT change any variable names already present in the code.
%
% Upload your team's script to Gradescope when complete.
%
% NAME YOUR FILE AS Challenge4_Sec{section number}_Group{group
 breakout #}.m
% ***Section numbers are 1 or 2***
% EX File Name: Challenge4_Sec1_Group15.m
%
% STUDENT TEAMMATES
% 1) Derek Blunt
% 2) Kyle Roche
% 3) Levi Reveles
% 4) Mikaela Felix
% 5) Caleb Bristol
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Housekeeping (Please don't "clear all" or "clearvars", it makes grading difficult)

```
close all   % Close all open figure windows
```

```
clc           % Clear the command window
```

Load and extract the time and velocity vectors from the data

```
data = readtable("Challenge4_data.csv");
t = data.Time_s_;    % [s]
v = data.Velocity_m_s_;    % [m/s]
```

```
Warning: Column headers from the file were modified to make them valid
 MATLAB
identifiers before creating variable names for the table. The original
 column
headers are saved in the VariableDescriptions property.
Set 'PreserveVariableNames' to true to use the original column headers
 as table
variable names.
```

Calculations

Find number of data points in the vectors

```
N = length(t);

% Find linear best fit coefficients A and B
% Create H matrix
H = [ones(N,1),t];

% Create y matrix
y = v;

% Create W matrix (hint: type <help diag> in command line)
W = eye(N);

% Solve for P matrix
P = (H' * W * H)^-1;

% Solve for x_hat matrix and extract A and B parameters
x_hat = P * H' * W * y;
A = x_hat(1);
B = x_hat(2);

% extract uncertainty in A and uncertainty in B from P matrix
Deviation = y - (A + B .*t);
SigY = sqrt((1/(length(y) - length(x_hat))) * sum(Deviation .*
 Deviation));

delta_y (1:length(y)) = SigY;
Diagonal = 1 ./ (delta_y .* delta_y);

W = diag(Diagonal);
```

```
%Error Covariance Matrix
Sigma_xHat = (H' * W * H)^-1;

A_error = sqrt(Sigma_xHat(1,1));
B_error = sqrt(Sigma_xHat(2,2));
```

Display acceleration with associated uncertainty and the intial velocity with associated uncertainty

Make sure to use and display with CORRECT SIG FIGSd

```
fprintf("Acceleration (m/s^2): ");
disp(round(B,2));
fprintf("Initial Velocity (m/s): ");
disp(round(A,2));
fprintf("Uncertainty of Data: ");
disp(round(SigY,2));
```

Acceleration (m/s^2):   -1.5200

Initial Velocity (m/s):   8.2400

Uncertainty of Data: 0.1300

Find predicted velocity values using your linear fit equation

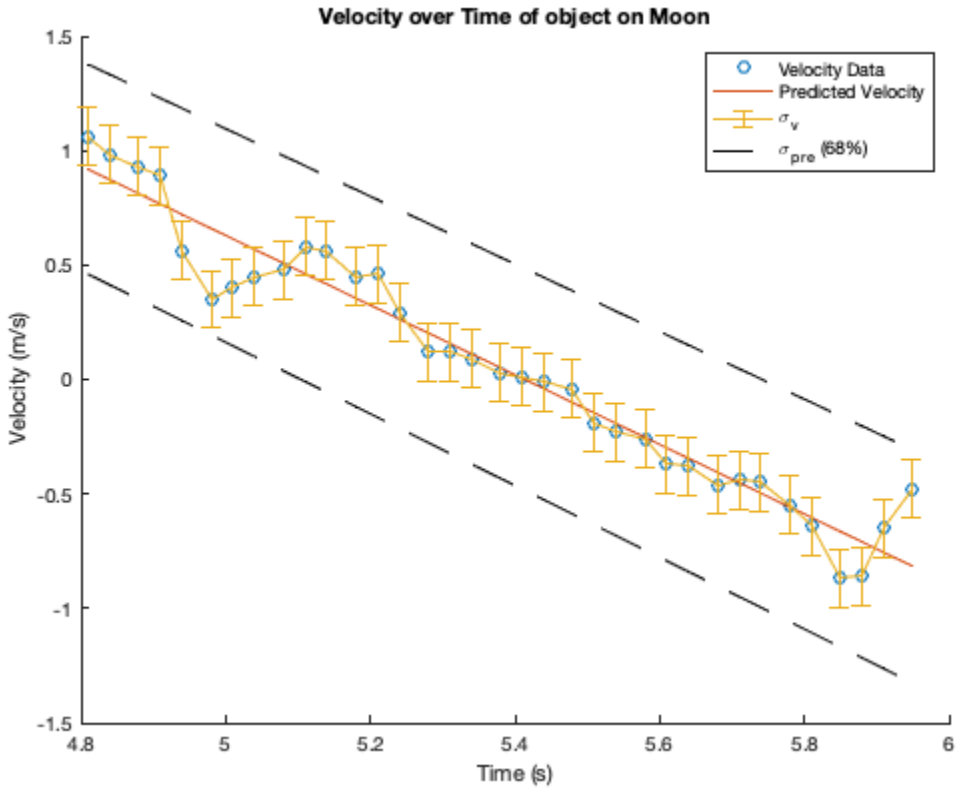
```
v_predicted = A + B .* t;
```

Ploting and Error Calculations

On the same plot, do the following: 1. plot the velocity data vs time as a scatter plot 2. plot predicted velocity vs time as a line 3. title your plot so that it indicates what celestial body this data simulates 4. Add measured velocity error bars and predicted velocity error bars to the plot (hint - this will involve error propagation calculations

```
%Plotting
scatter(t,v); hold on
plot(t,v_predicted);
title("Velocity over Time of object on Moon");
xlabel("Time (s)");
ylabel("Velocity (m/s)");
xlim([4.8,6]);

%Error Calculations
v_err = SigY .* ones(N,1);
SigQ = sqrt(Sigma_xHat(1,1) + t.^2 .*Sigma_xHat(2,2));
```

```
v_predicted_error = [v_predicted + SigQ, v_predicted - SigQ];

%Plotting Error
errorbar(t,v,v_err);
plot(t,v_predicted_error,'--','color','black');

legend('Velocity Data','Predicted Velocity','\sigma_v','\sigma_p_r_e
 (68%)')

hold off;
```



Published with MATLAB® R2020a