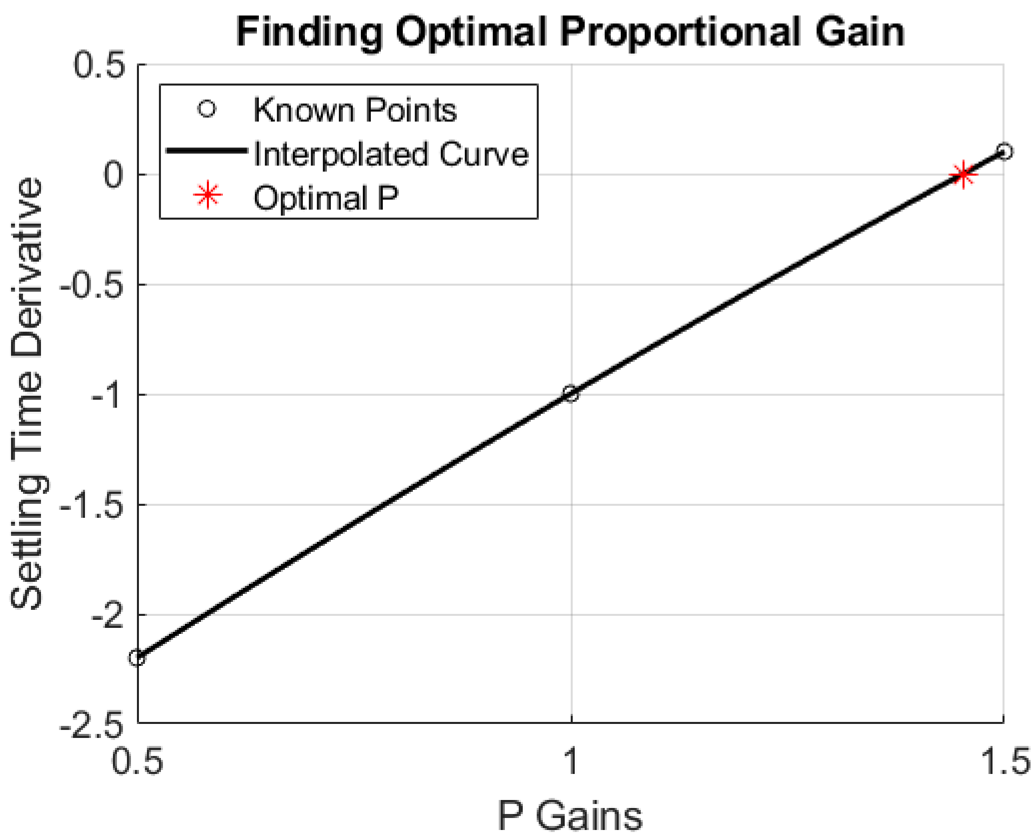
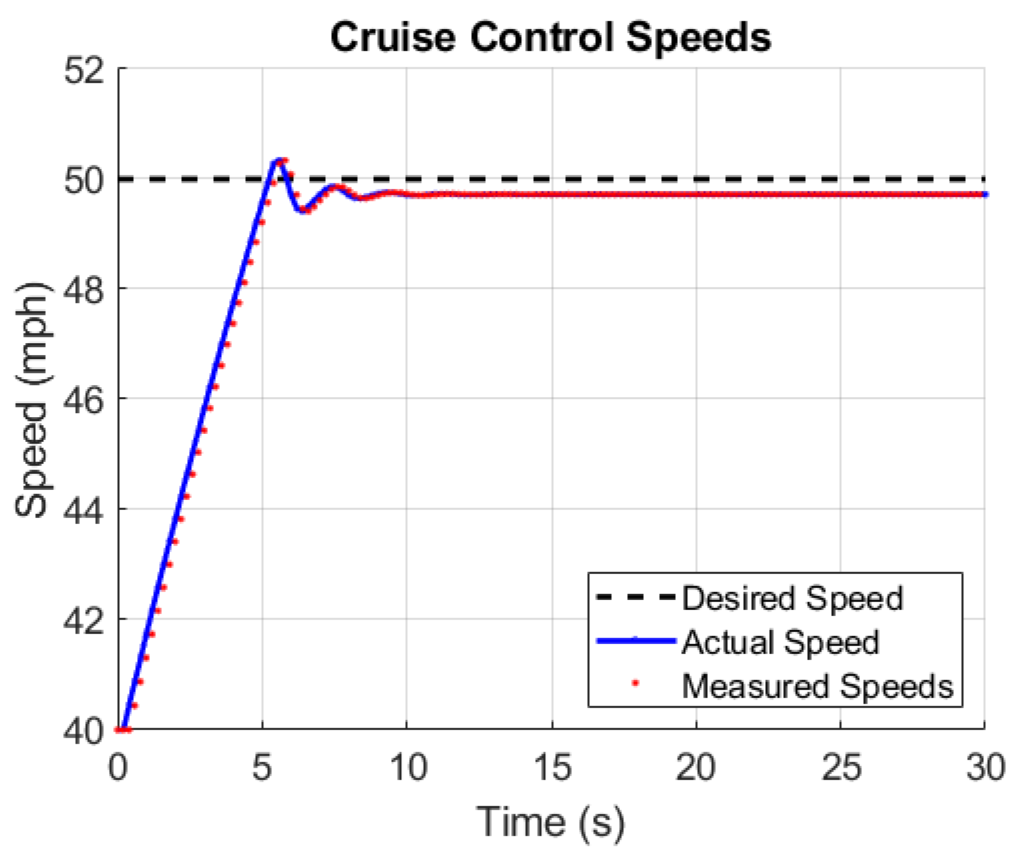
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MATH_151_FinalLab
%-------------------------------------------------------------------------
% C Rocheleau, Colorado State University
% 11/3/2023
%-------------------------------------------------------------------------
% Answer key for MATH-151 Final Lab for the Fall 2023 semester
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all; clear all; clc;
```

Task 1: Going on Cruise Control

```
% Part (a) Solve for P gain that minimizes Settling time
P_vals = [0.5, 1.0, 1.5];
dTdS   = [-2.2, -1, 0.1];

% Secant Method
Pn = P_vals(1); Pm = P_vals(end);
while abs(Newton_interp(P_vals, dTdS, Pm)) > 0.001
    fx = Newton_interp(P_vals, dTdS, [Pn Pm]);
    temp = Pm - fx(2)*(Pm - Pn)/(fx(2) - fx(1));
    Pn = Pm;
    Pm = temp;
end

% Plot to show that this is our Zero
figure(); hold on; grid on;
plot(P_vals, dTdS,'ko');
plot(P_vals(1):0.1:P_vals(end), Newton_interp(P_vals, dTdS,
 P_vals(1):0.1:P_vals(end)),'k-','linewidth',2);
plot(Pm, Newton_interp(P_vals, dTdS, Pm), 'r*','markersize',10);
legend('Known Points','Interpolated Curve','Optimal
 P','location','NorthWest');
set(gca,'fontsize',14);
xlabel('P Gains'); ylabel('Settling Time Derivative');
title('Finding Optimal Proportional Gain');

% Parts b-e, building PID Controller

% Initialize time and velocity vectors
v_start = 40; v_des = 50;
dt = 0.2; t = 0:dt:30;
vel = NaN*t;
vel(1) = v_start; vel(2) = v_start; % Velocity won't change for first 2 steps
vel_meas = vel; % Store our velocity measurements

Im =  3;     % Integral Gain

Dm =  -0.25; % Derivative gain

for ii = 2:length(t)-1
```

```
    vel_meas(ii) = vel(ii-1); % Measure Truth at a step ago

    % Compute our integral estimate
    if ii > 5
        int_est = sum(v_des - vel_meas(ii-4:ii))*dt;
    else
        int_est = sum(v_des - vel_meas(1:ii))*dt;
    end
    deriv_est = 0;
    if ii > 2
        % Finite Difference method to approximate our derivative
        deriv_est = (vel_meas(ii) - vel_meas(ii-2))/(dt);
    end

    % Calculate our components for the controller
    P = Pm*(v_des - vel_meas(ii));
    I = Im*int_est;
    D = Dm*deriv_est;

    % Calculate commanded acceleration
    a_cmd  = P + I + D;
    % Limit acceleration to +/- 3 mph/s
    a_cmd = sign(a_cmd)*min(abs(a_cmd),3);

    % Use current velocity to find our true accelerations
    a = a_cmd - vel(ii)^2/2000;
    % Use acceleration to propagate velocity forward
    vel(ii+1) = vel(ii) + a*dt;
end

% Plot the controller's behavior
figure(); grid on; hold on;
plot( t, v_des*ones(size(t)), 'k--','linewidth',2);
plot( t, vel,'b.-','linewidth',2);
plot( t, vel_meas,'r.');
legend('Desired Speed','Actual Speed','Measured
 Speeds','location','Southeast');
xlabel('Time (s)'); ylabel('Speed (mph)');
title('Cruise Control Speeds')
set(gca,'fontsize',14);
```


Finding Optimal Proportional Gain
Known Points
Interpolated Curve
Optimal P
Settling Time Derivative
P Gains
0.5
0
-0.5
-1
-1.5
-2
-2.5
0.5
1
1.5


Cruise Control Speeds
Desired Speed
Actual Speed
Measured Speeds
Speed (mph)
Time (s)
52
50
48
46
44
42
40
0
5
10
15
20
25
30

Published with MATLAB® R2022b