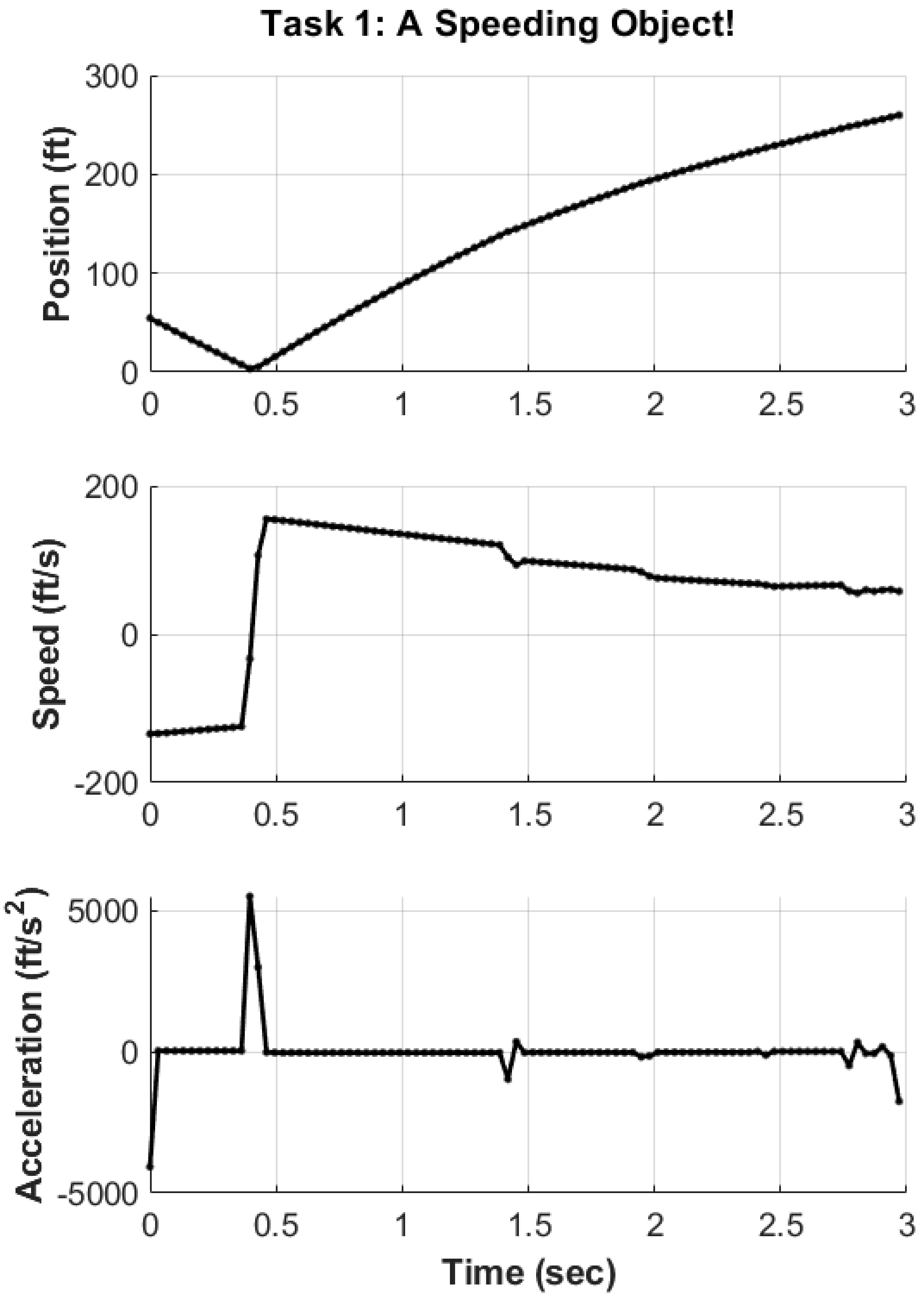
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MATH_151_Lab6
%----------------------------------------------------------------------------
% C Rocheleau, Colorado State University
% 10/08/2023
%----------------------------------------------------------------------------
% Answer key for MATH-151 Lab 6 for the Fall 2023 semester
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all; clear all; clc;
```

Task 1: A Speeding Object

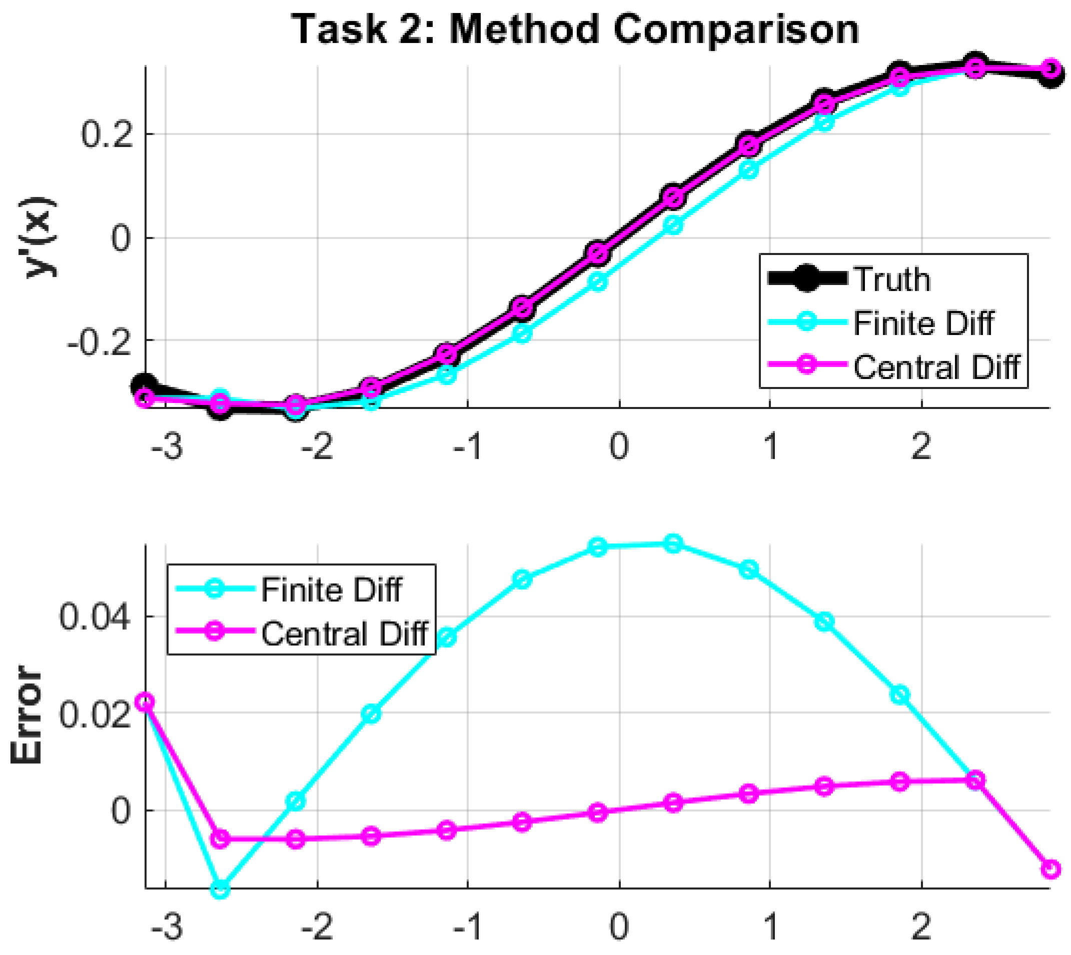
```
% Load in the data (Contains variabels time and pos)
load('Lab6_Data.mat');

% Find object's Speed and acceleration at each time
speed = 0*pos; % Preallocate vectors
accel = 0*pos;
for iT = 1:length(time)
    if iT == 1
        % At our first time step we can't look backwards
        % So we use the forward difference method for our speed
        h = time(iT+1) - time(iT);
        speed(iT) = (pos(iT+1) - pos(iT)) / h;
        % For acceleration, we will just assume it had constant position
        accel(iT) = (pos(iT+1) - 2*pos(iT) + pos(iT)) / h^2;
    elseif iT == length(time)
        % For our last time step we can't look forwards
        % So we use the backward difference method for our speed
        h = time(iT) - time(iT-1);
        speed(iT) = (pos(iT) - pos(iT-1)) / h;
        % For acceleration, we will just assume it stays at its final
        % position
        accel(iT) = (pos(iT) - 2*pos(iT) + pos(iT-1)) / h^2;
    else
        % For every other time step we use central difference method and
        % our second order derivative formula as written
        h = time(iT) - time(iT-1);
        speed(iT) = (pos(iT+1) - pos(iT-1)) / (2*h);
        % For acceleration, we will just assume it stays at its final
        % position
        accel(iT) = (pos(iT+1) - 2*pos(iT) + pos(iT-1)) / h^2;
    end
end

% We'll plot everything on one figure for fun (subplots are cool!)
figure('Outerposition',[100 100 600 900]);
sgtitle('Task 1: A Speeding Object!','fontweight','b','fontsize',16)

% Position Plot
subplot(3,1,1); % This creates a 3x1 grid of axes and we use the first one
```

```
grid on; hold on; set(gca,'fontsize',14)
plot(time, pos, 'k.-','markersize',10,'linewidth',2);
ylabel('Position (ft)','fontweight','b')

% Velocity Plot
subplot(3,1,2); % This tells matlab to move onto the second plot
grid on; hold on; set(gca,'fontsize',14)
plot(time, speed, 'k.-','markersize',10,'linewidth',2);
ylabel('Speed (ft/s)','fontweight','b')

% Acceleration Plot
subplot(3,1,3); % And now the third plot!
grid on; hold on; set(gca,'fontsize',14)
plot(time, accel, 'k.-','markersize',10,'linewidth',2);
xlabel('Time (sec)','fontweight','b')
ylabel('Acceleration (ft/s^2)','fontweight','b')
```


Task 1: A Speeding Object!
Position (ft)
300
200
100
0
0.5
1
1.5
2
2.5
3
Speed (ft/s)
200
0
-200
Acceleration (ft/s^2)
5000
0
-5000
Time (sec)

Task 2: Method Comparison

```
% Create our grid and calculate our function and truth
h = 0.5; x = -pi:h:pi;
f = sin(x/3).^2;
fdot_true = (2/3)*sin(x/3).*cos(x/3); % fdot notation is nicer than prime in
 code...

% Lets loop though and do both of our finite diff and central diff
% simultaneously

fdot_FD = zeros(size(f)); fdot_CD = zeros(size(f)); % Preallocation

for iX = 1:length(x)
    % Take care of endpoints appropriately
    if iX == 1
        fdot_FD(iX) = (f(iX+1) - f(iX)) / h;
        fdot_CD(iX) = fdot_FD(iX); % We only have 2 points so we use Finite
 difference
    elseif iX == length(x)
        fdot_FD(iX) = (f(iX) - f(iX-1)) / h;
        fdot_CD(iX) = fdot_FD(iX);
    else
        fdot_FD(iX) = (f(iX) - f(iX-1)) / h;
        fdot_CD(iX) = (f(iX+1) - f(iX-1)) / (2*h);
    end
end

% I know for part (c) I said in a new figure, but a subplot works well too!
figure('Outerposition',[800 100 600 600]);
sgtitle('Task 2: Method Comparison','fontweight','b','fontsize',16)

% Plot Functions
subplot(2,1,1);
hold on; grid on; set(gca,'fontsize',14)
htrue = plot(x, fdot_true, 'ko-','linewidth',5);
hFD   = plot(x, fdot_FD,   'co-','linewidth',2);
hCD   = plot(x, fdot_CD,   'mo-','linewidth',2);
axis tight;
legend([htrue, hFD, hCD], {'Truth','Finite Diff','Central
 Diff'},'location','SouthEast')
ylabel('y''(x)','fontweight','b')

subplot(2,1,2)
hold on; grid on; set(gca,'fontsize',14)
heFD   = plot(x, fdot_true - fdot_FD,   'co-','linewidth',2);
heCD   = plot(x, fdot_true - fdot_CD,   'mo-','linewidth',2);
axis tight;
legend([heFD, heCD], {'Finite Diff','Central Diff'},'location','NorthWest')
ylabel('Error','fontweight','b')

% We can see that the Central Difference approximation is notably closer to
% the truth! It's largest error (ignoring the endpoints) is about 10 times
% smaller! That is a lot of precision for not a whole lot more work.
```



Published with MATLAB® R2022b