Table of Contents

ENSC180-Assignment2

```
% Student Name 1: Jia Jun Li

% Student 1 #: 301238737

% Student 1 userid (email): jjl39 (stu1@sfu.ca)

% Student Name 2: student2

% Student 2 #: 123456782

% Student 2 userid (email): stu2 (stu2@sfu.ca)

% Below, edit to list any people who helped you with the assignment,
%      or put 'none' if nobody helped (the two of) you.

% Helpers:
%Students: Minji Ju, Wenbo Yin, Matt Weins
%Academic: Zhida Li, Hadi Moein, Craig Scratchley
%Online: Mathworks docs, wikipedia, NASA, RedbullStraos Project
```

Instructions:

- Put your name(s), student number(s), userid(s) in the above section.
- Edit the "Helpers" line.
- Your group name should be "A2_<userid1>_<userid2>" (eg. A2_stu1_stu2)
- Form a group as described at: https://courses.cs.sfu.ca/docs/students
- Replace "% your work here" below, or similar, with your own answers and work.
- You can copy your work from your other functions and (live) scripts and as needed.
- Nagvigate to the "PUBLISH" tab (located on top of the editor) * Choose pdf as "Output file format" under "Edit Publishing Options..." * Click "Publish" button. Ensure a report is automatically generated

- You will submit THIS file (assignment2.m), and the PDF report (assignment2.pdf). Craig Scratchley, Spring 2017

main

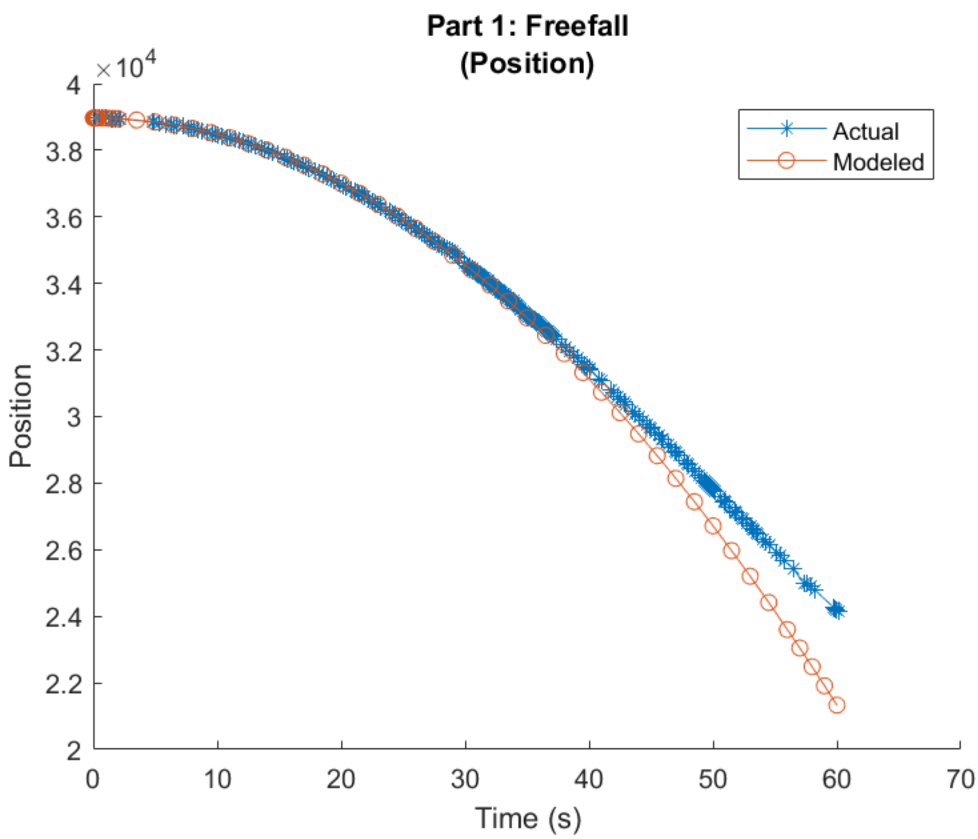
```
function main

% PLEASE use the latest excel data "RedBullStratosData180-draft" for
 marking.

% data prep
filename = 'RedBullStratosData180-draft.xlsx';
dataArray = xlsread(filename);   %reads the data numerically

B = any(dataArray,2);    %logic of rows, if all elements = NaN -> row
 = 0
dataArray = dataArray(B,:);

% constants
A_GRAV_SEA = -9.8;
ABV_SEA = 38969; %in meters

% variables
elapsedTime = dataArray(2:end,11);
altitude = dataArray(2:end,4);                                  %2:end to
 remove rows where time is negative
velocity = (dataArray(2:end,5)) ./ 3.6;                         %setting
 units to m/s
accleration = abs(diff(velocity)) ./ diff(elapsedTime);

% <any() was used here to test along the rows of the data and generate
 a logical col vector.
%This was used to remove all rows of ONLY NaN. Heartrate data ensured
 a row was always non-zero unless all entries were NaN>
```

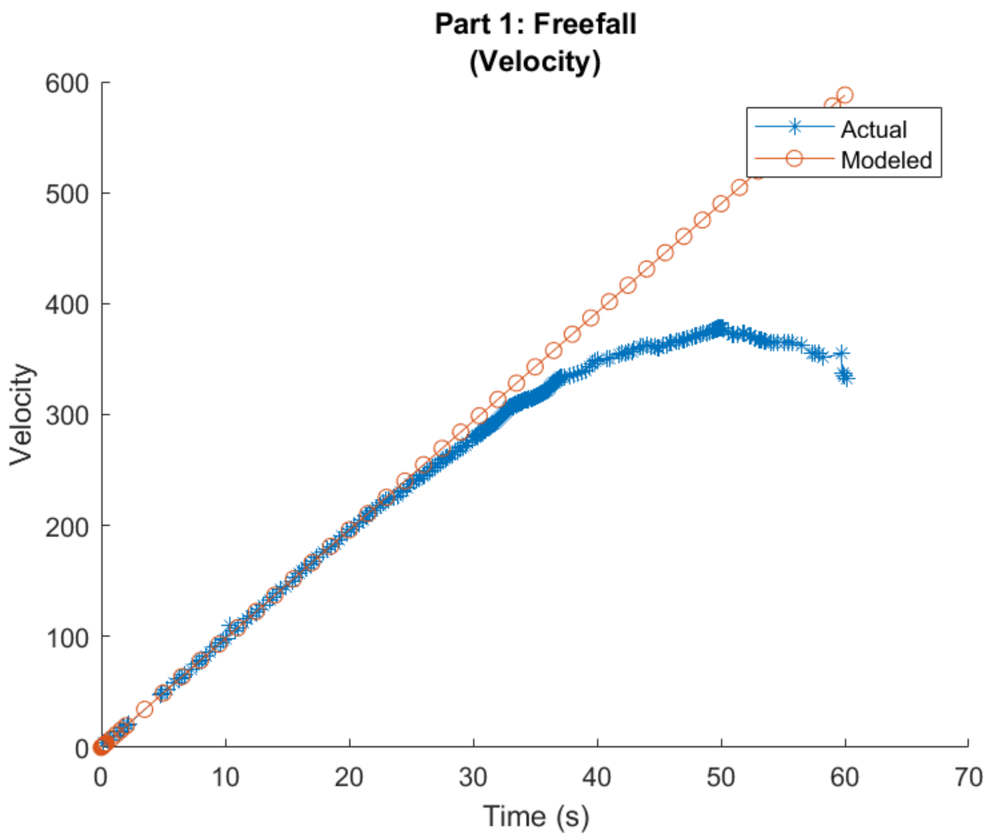
Part 1

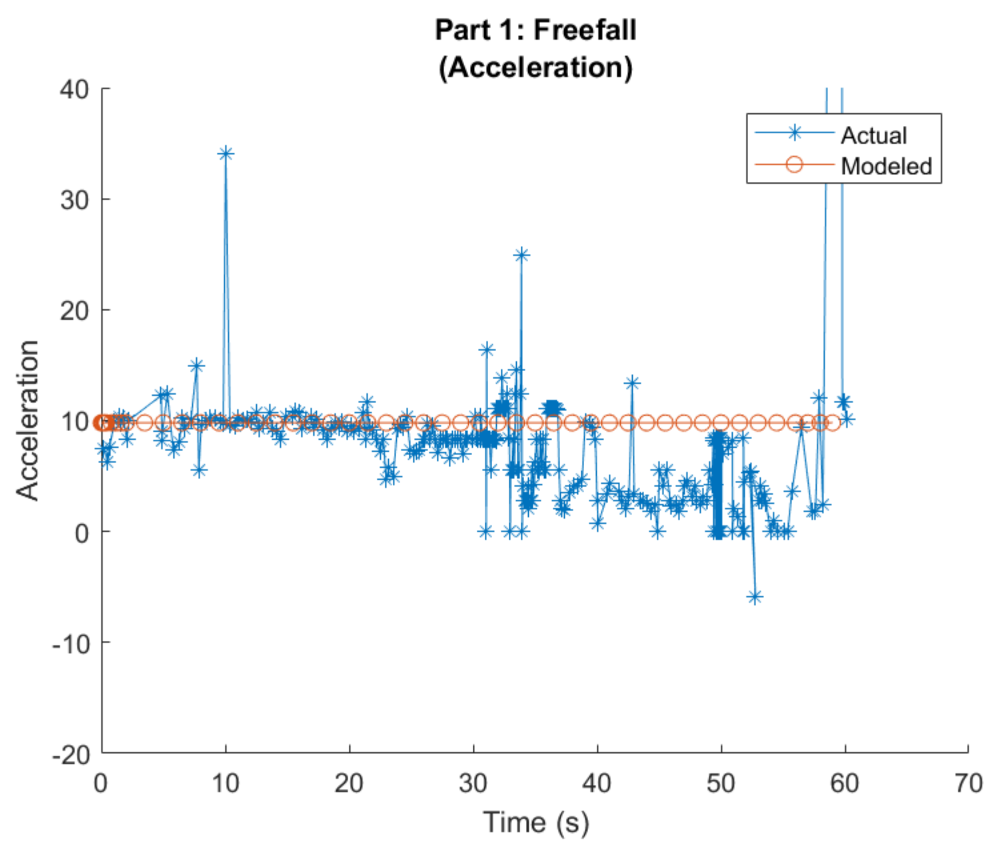
Answer some questions here in these comments...

```
% How accurate is the model for the first portion of the minute?
% <The model is fairly accurate up till 40s, at about 40s, the model
 starts to diverge from actual fall.>

% How accurate is the model for the last portion of that first minute?
% <In the latter part of the first minute we start to see the mdoel
 diverge from the actual fall.>

% Comment on the acceleration calculated from the measured data.
% Is there any way to smooth the acceleration calculated from the
 data?
% <We can use different signal proccessing functions available in
 matlab.
```

```
% There is medfilt1 that used an nth order polynomial to smooth the
 function.
% Another is interp1, which interprolates data points at a queried
 location to smooth the graph.>

part = 1;
plotComparisons(60, 'Part 1: Freefall', 0)
```



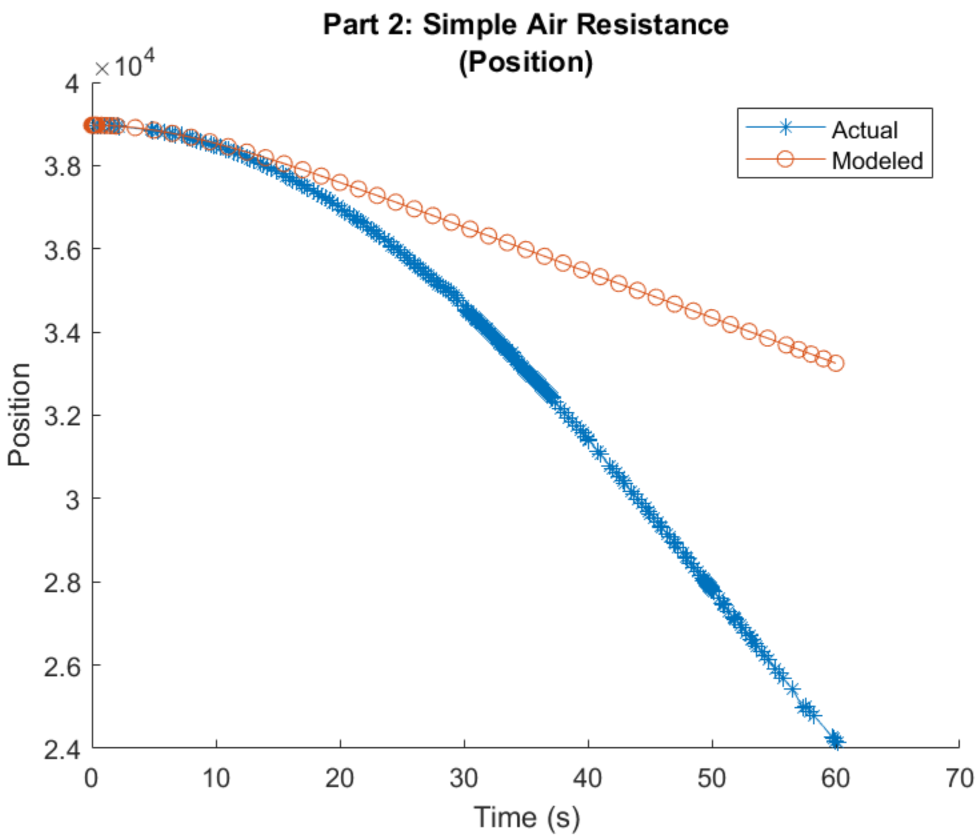

Part 1: Freefall
(Velocity)
Actual
Modeled
Velocity
Time (s)
600
500
400
300
200
100
0
0
10
20
30
40
50
60
70

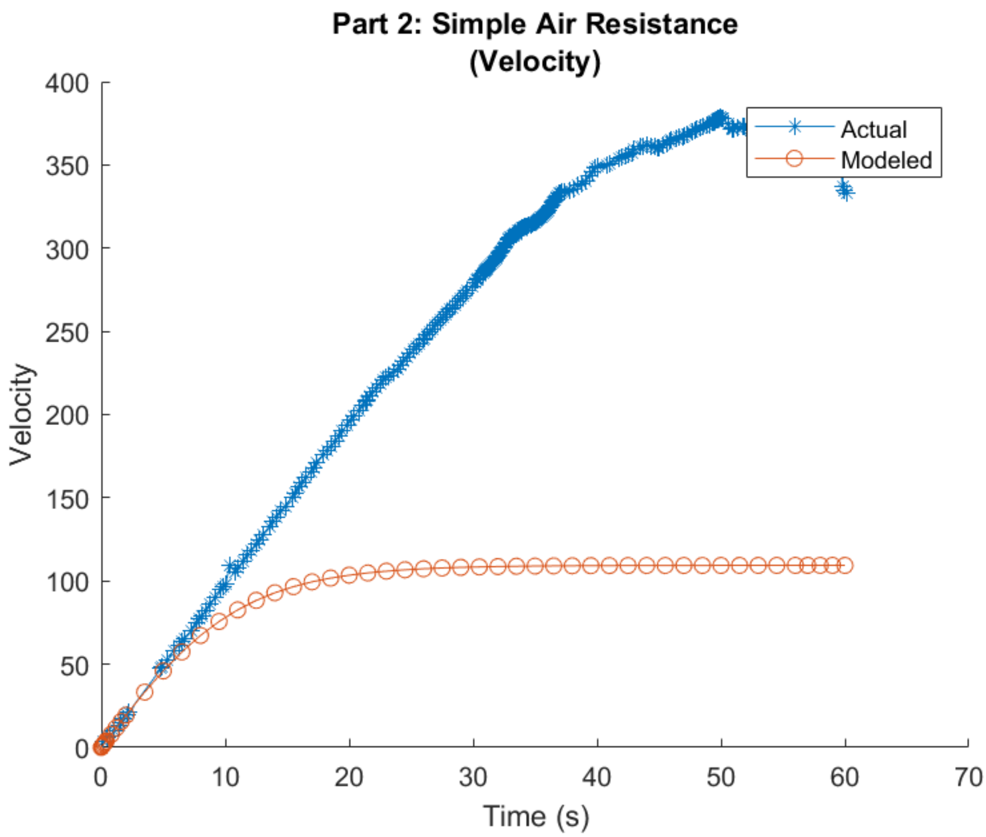

Part 1: Freefall
(Acceleration)
Actual
Modeled
Acceleration
Time (s)
40
30
20
10
0
-10
-20
0
10
20
30
40
50
60
70

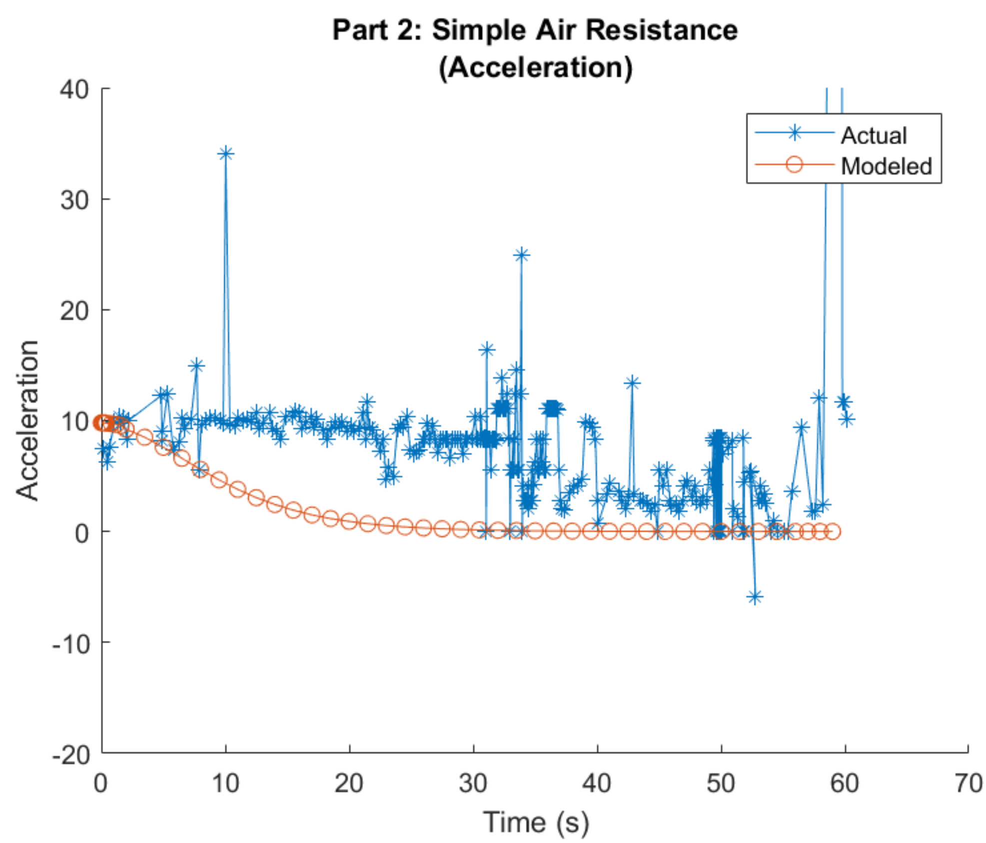
Part 2

Answer some questions here in these comments...

```
% Estimate your uncertainty in the mass that you have chosen (at the
%     beginning of the jump).
% <(244 uncertainty of 5)kg, The error accounted for comes from
 converting some of the values I found online from pounds to kg,
% As well as the mass used for the suit and life support was from a
 similar suit&support from NASA, so Felix's might be different.>

% How sensitive is the velocity and altitude reached after 60 seconds
 to
%    changes in the chosen mass?
% <When I account for the min&max value due to error I notice that
 terminal velocity increases and decreases proportionally
% to mass. The general trend of the plots remained the same, but there
 was a shift along the y axis depending on the mass used.>

part = 2;
plotComparisons(60, 'Part 2: Simple Air Resistance', 0)
```




Part 2: Simple Air Resistance
(Velocity)
Actual
Modeled
Velocity
Time (s)
0
50
100
150
200
250
300
350
400
0
10
20
30
40
50
60
70

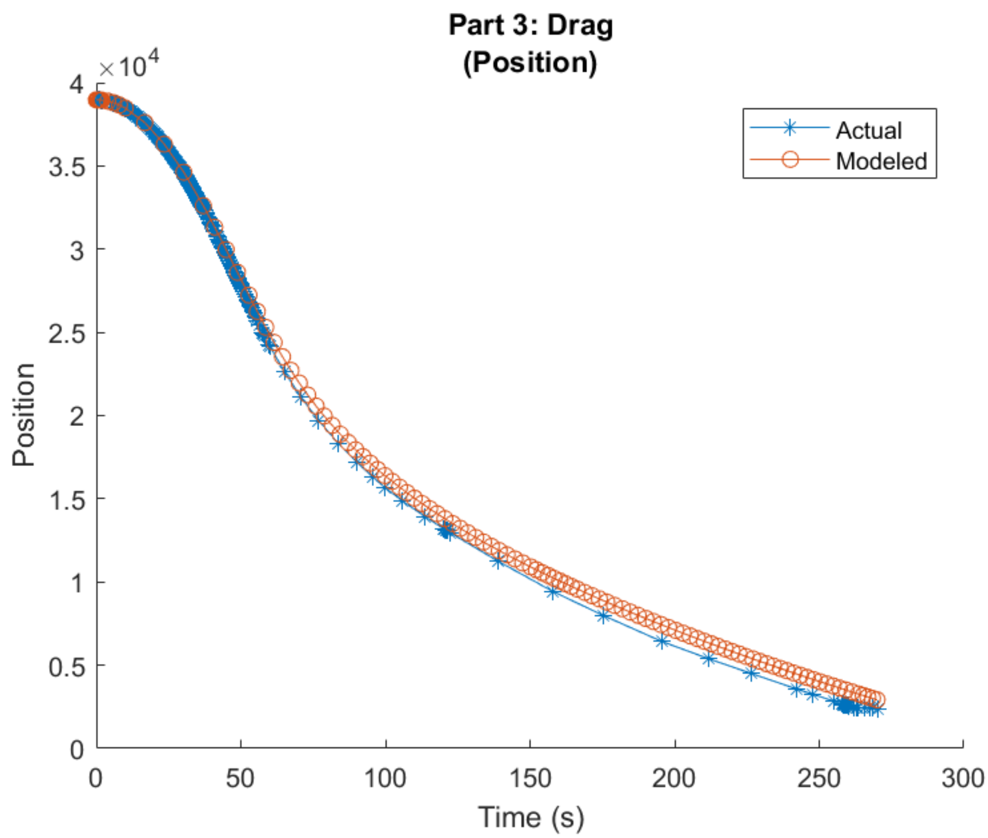

Part 2: Simple Air Resistance
(Acceleration)
Actual
Modeled
Acceleration
Time (s)
-20
-10
0
10
20
30
40
0
10
20
30
40
50
60
70

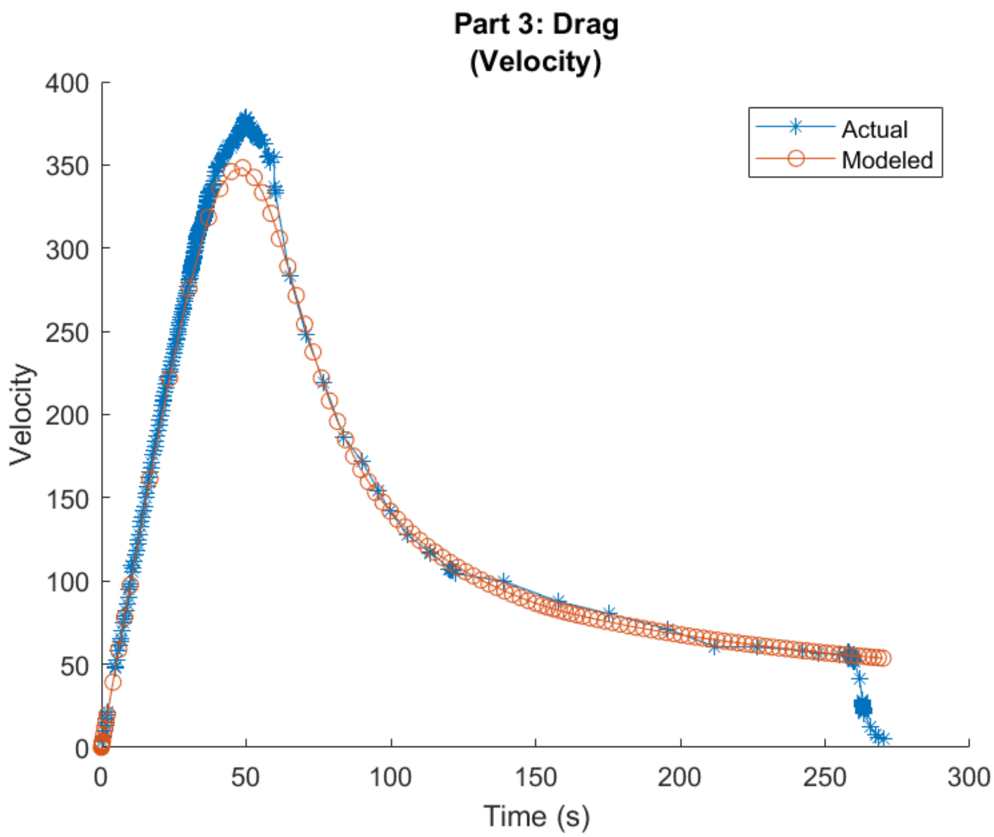
Part 3

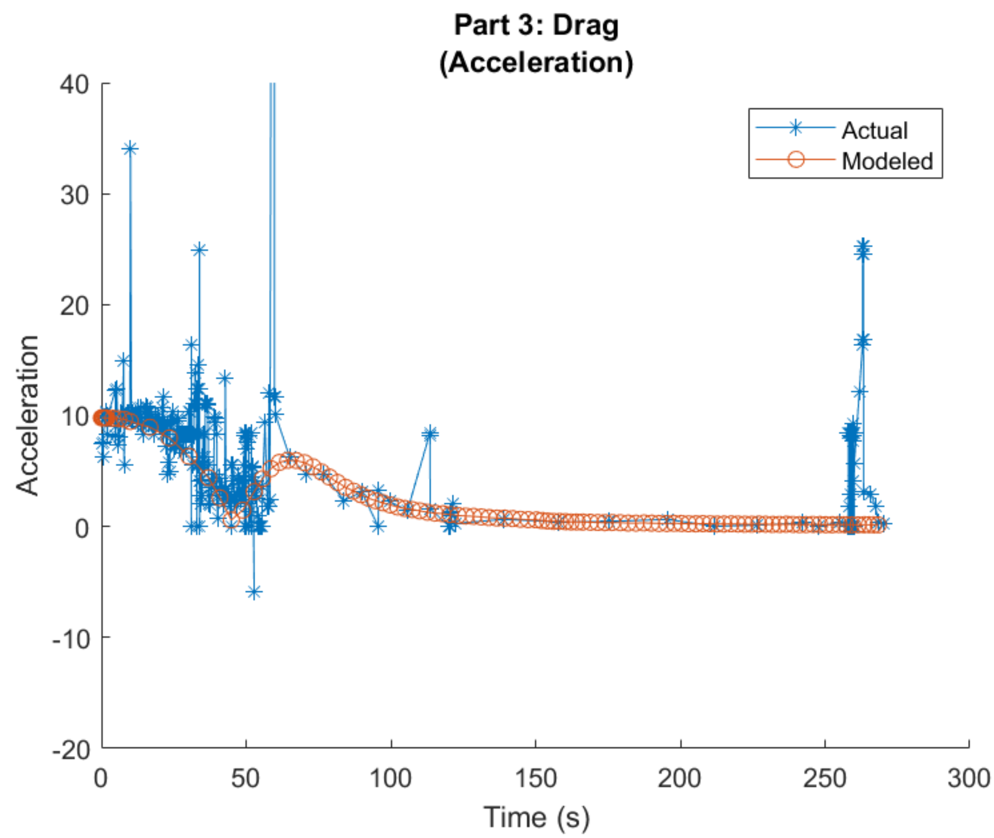
Answer some questions here in these comments...

```
% Felix was wearing a pressure suit and carrying oxygen. Why?
%     What can we say about the density of air in the stratosphere?
%     How is the density of air different at around 39,000 meters than
 it
%     is on the ground?
% <Air density at 39000m is almost 0, compared to on the ground where
 it was close to 1.1 (derived from plot rho=stdatmo(altitude)).
% The lack of atomosphere pressure and air is why Felix needs his own
 oxygen and pressurized suit so he doesn't explode.
% This also means Felix had a far lower drag force at 39000m and was
 able to reach a greater velocity in the upper stratosphere.
% But as he descended, the density of air increases, which increases
 the force due to air resistance, slowing him down. >

% What are the factors involved in calculating the density of air?
%     How do those factors change when we end up at the ground but
 start
%     at the stratosphere?  Please explain how calculating air density
 up
%     to the stratosphere is more complicated than say just in the
 troposphere.
% <Using the formula for air density from wikipedia, the factors
 included:
%    - altitude
%    - std atmosphereic pressure
%    - std temperature
%    - accelereation due to gravity
%    - temperture lapse rate
%    - ideal gas constant
%    - molar mass of dry air
% from wikipedia, we see that std atmosphereic pressure, temperature
 and absolute pressure's ratio all converge to 1 as we get closer
% to ground level. But as we go to higher altitudes these values
 diverge and makes the calculations harder.
% https://upload.wikimedia.org/wikipedia/commons/d/dc/
StandardAtmosphere.png>

% What method(s) can we employ to estimate [the ACd] product?
% < {Fnet = F=ma = 1/2*rho*v^2*A*Cd} where:
%    - At terminal velocity, the drag force = force due to gravity. At
 this interval, acceleration = 0 and we are able to isolate ACd.
%    - Using an altitude we know Felix is moving at terminal velocity,
 we can sub this in to find rho.
%    - rho = air density calculated in function airDensity(terminal
 Velocity Altitude) OR use stdatmo(terminal Velocity Altitude)
%    - now, we have all the unknowns except for the product of ACd,
 which we solve for using
%    - 2*(a - a_grav).*m./(rho.*v.^2) where a=0, a_grav=9.8>
```

```
% What is your estimated [ACd] product?
% <ACd = 1.8448 using function ACdFind(altitude, masss)>
%
% [Given what we are told in the textbook about the simple drag
 constant, b,]
%   does the estimate for ACd seem reasonable?
% <b = 1/2*rho*ACd where rho=0.0253 ACd=1.8448
%  b = 0.02334, which is about a factor of 10 greater than the simple
 drag constant. Because of this factor of 10 error
%  I think this estimate is not reasonable. This idea is re-enforced
 when we compare the graphs of part 2 and 3.>

part = 3;
plotComparisons(270, 'Part 3: Drag', 0)
```




Part 3: Drag
(Velocity)
Actual
Modeled
Velocity
Time (s)

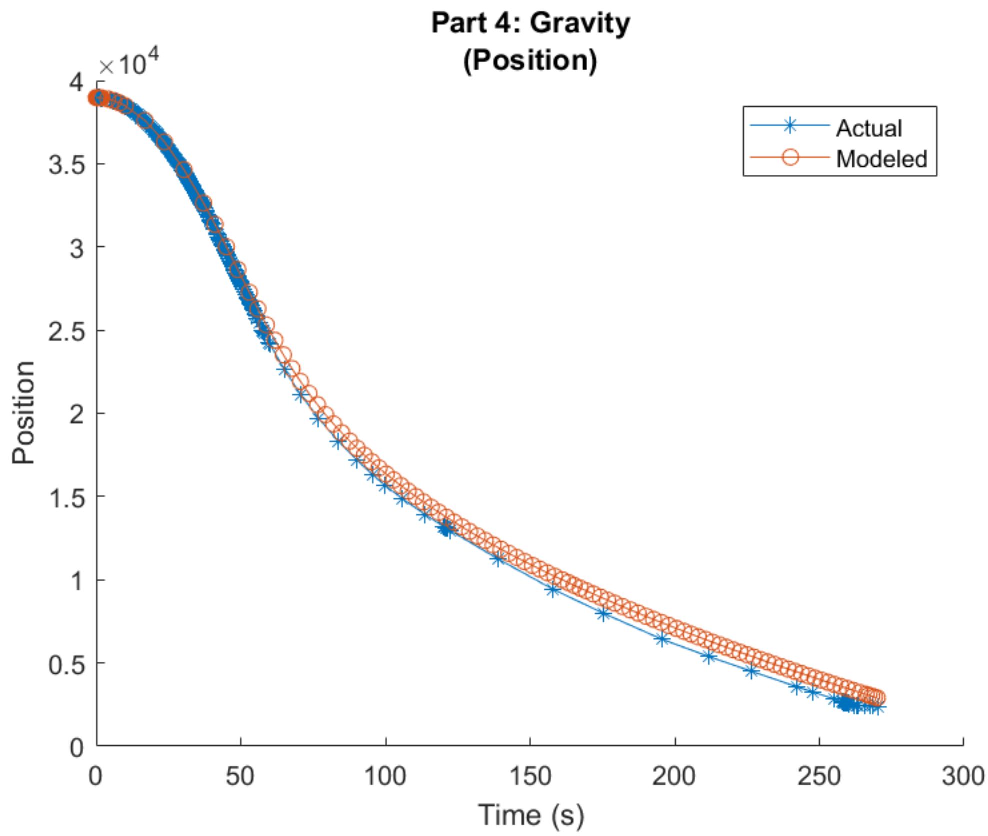

Part 3: Drag
(Acceleration)
Actual
Modeled
Acceleration
Time (s)

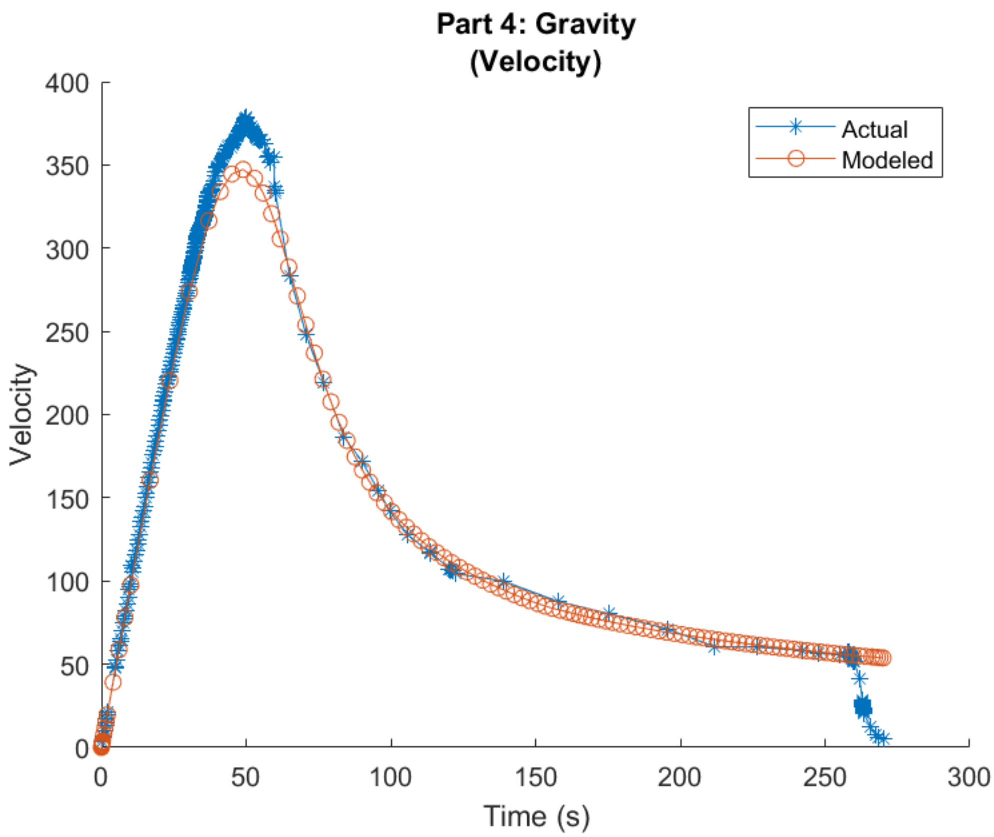
Part 4

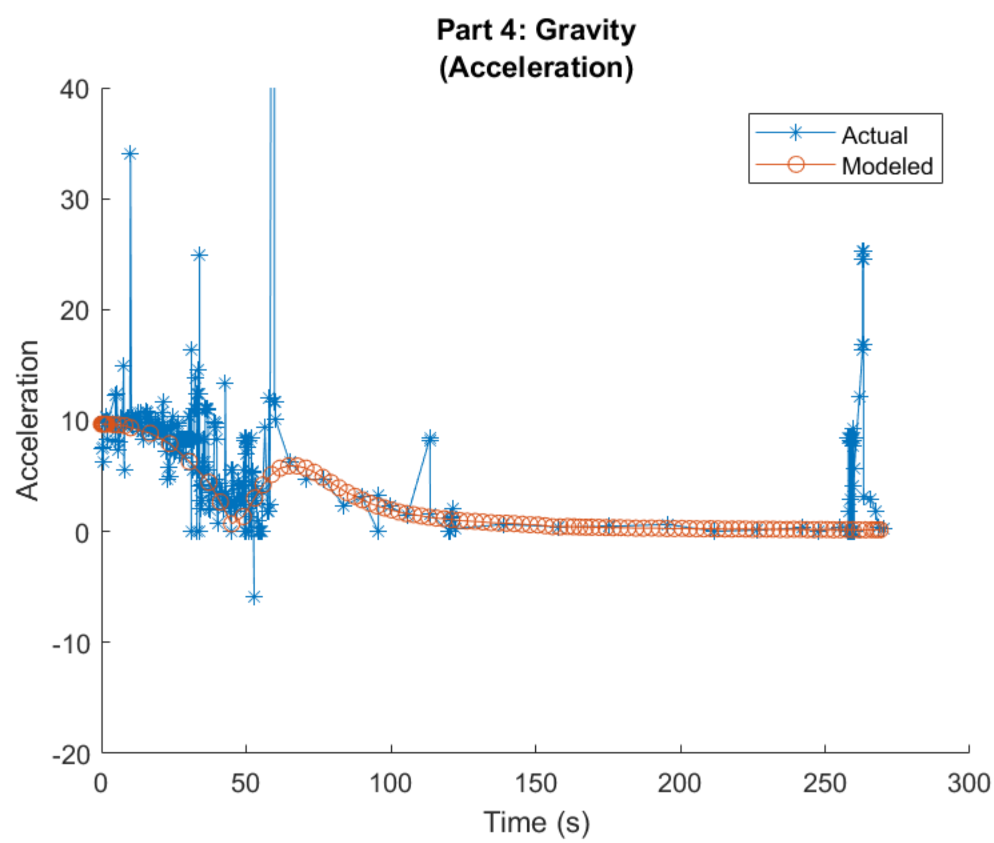
Answer some questions here in these comments...

```
% What is the actual gravitational field strength around 39,000
 meters?
%   (See Tipler Volume 1 6e page 369.)
%<The acceleration due to gravity at 39000m above the surface of Earth
 is about 9.688m/s^2>

% How sensitive is the altitude reached after 4.5 minutes to simpler
 and
%   more complicated ways of modelling the gravitational field
 strength?
% <Not very sensitive, from the formula Gh=g(Re/re+h)^2, we can see
 g=9.8 is being multiplied by a number that is very close
% to 1. Since 39000 is insignificant to 6.371e3, the ratio is 0.98787,
 very close to 1.>

% What other changes could we make to our model? Refer to, or at least
%   attempt to explain, the physics behind any changes that you
 propose.
% <Though relavent to part 6, we could model the rate at which Felix's
 parachute opens as a much slower ramp up.
% The reason for this is that when Felix deploys his parachute, there
 is a period of time where the parachute is flailing.
% So we should apply a rate function that slowly ramps up the cross-
sectional area to account for this flailing.
% In our model, the parachute opening is modeled by a symmetric bump.
 To model the opening better, the first half of the bump
% should be linear, and the second half where accleration returns to 0
 should be a negative logarithm>

% What is a change that we could make to our model that would result
 in
%   insignificant changes to the altitude reached after 4.5 minutes?
% <Accounting for Felix's personal cross sectional area would be an
 insignificant change as by then the parachute's cross
% sectional area would eclipse Felix's.>

% How can we decide what change is significant and what change is
%   insignificant?
% <We can determine if a change is insignificant, if the  value for an
 estimate of a contributing factor
% is small uncertainty from real value. We can see this in the case of
 calculating gravity at different altitudes.

% [What changes did you try out to improve the model?  (Show us your
 changes
%   even if they didn't make the improvement you hoped for.)]
% <My changes can be found in part 6, I was trying to model the rate
 of the parachute opening.
```

```
% I initially used a linear function to model this rate, as if it was
 a growing rectangle.
% I later changed it to a quadratic function, since his parachute is a
 circular and not rectangluar. Circles grow in area quadratically,
% after this implementation allowed the the modeled acclereation to
 huge the actual data better.>

part = 4;
plotComparisons(270, 'Part 4: Gravity', 0)
```



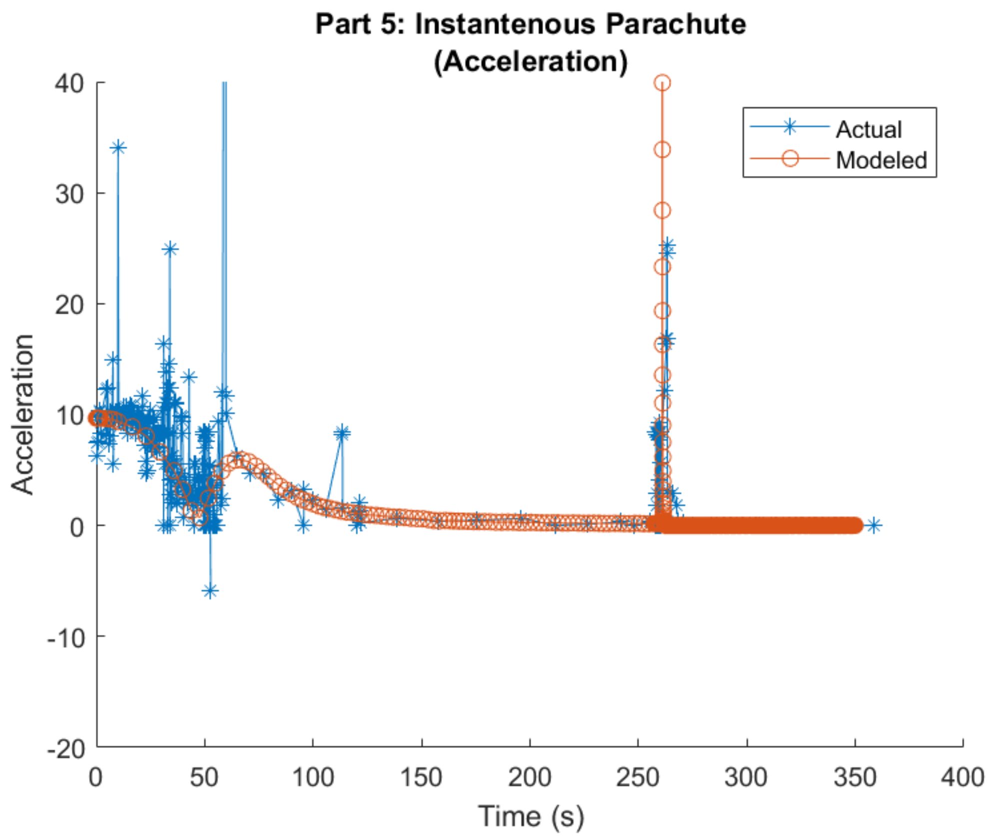

Part 4: Gravity
(Velocity)
Actual
Modeled
Velocity
Time (s)
400
350
300
250
200
150
100
50
0
0
50
100
150
200
250
300


Part 4: Gravity
(Acceleration)
Actual
Modeled
Acceleration
Time (s)
40
30
20
10
0
-10
-20
0
50
100
150
200
250
300

Part 5

Answer some questions here in these comments...

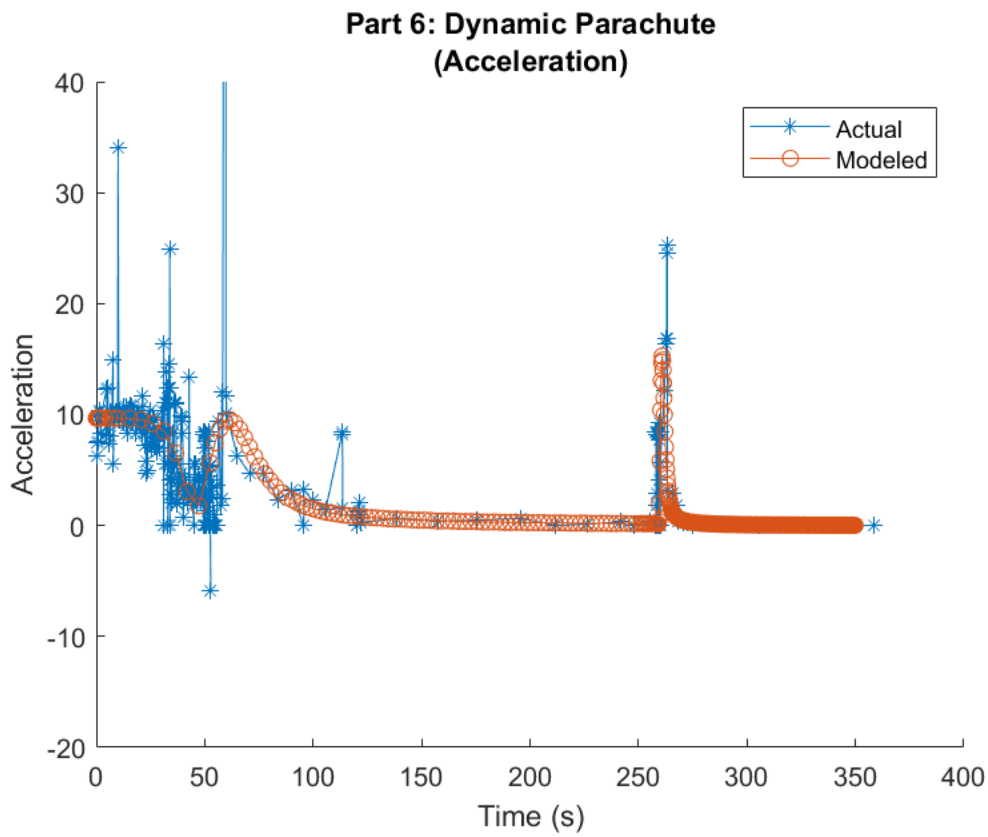
```
% At what altitude does Felix pull the ripcord to deploy his
 parachute?
% <At altitude 2573m>

% Recalculate the ACd product with the parachute open, and modify your
%   code so that you use one ACd product before and one after this
 altitude.
%   According to this version of the model, what is the maximum
 magnitude
%   of acceleration that Felix experiences?
% <Since the parachute opens instantaneously, Felix will experience a
 sharp spike in acclereation.
% Here I found his acceleration to to about 2300m/s^2 via my part
 plot>

%   How safe or unsafe would such an acceleration be for Felix?
% <It would most certainly be lethal as the force produced far exceeds
 the force necessary to rend flesh.
% http://ffden-2.phys.uaf.edu/212_spring2011.web.dir/Brendon_Fuhs/
force.html>

%Make a single acceleration-plot figure that includes, for each of the
%model and the acceleration calculated from measurements, the moment
 when
%the parachute opens and the following 10 or so seconds. If you have
%trouble solving this version of the model, just plot the acceleration
%calculated from measurements.

part = 5;
plotComparisons(350, 'Part 5: Instantenous Parachute', 3)
```



Part 6

Answer some questions here in these comments...

```
% How long does it take for Felix’s parachute to open?
% <roughly 15secounds, this was derived from part 6's plot of
 acceleration.
% This value was determined by checking the duration of Felix's
 accleration
% returning to 0. We know when acclereation = 0 Felix is at terminal
 velocity so it's safe to
% assume the parachute is fully open when he stabilizes on his second
 terminal velocity.

%Redraw the acceleration figure from the previous Part but using the
 new
%   model. Also, using your plotting function from Part 1, plot the
%   measured/calculated data and the model for the entire jump from
%   stratosphere to ground.

part = 6;
plotComparisons(350, 'Part 6: Dynamic Parachute', 3)
```



nested functions

nested functions below are required for the assignment. see Downey Section 10.1 for discussion of nested functions

```
function res = freefall(t, X)
    %FALL <Summary of this function goes here>
    %   <Detailed explanation goes here>

    % do not modify this function unless required by you for some
 reason!

    p = X(1); % the first element is position
    v = X(2); % the second element is velocity

    dpdt = v; % velocity: the derivative of position w.r.t. time
    dvdt = acceleration(t, p, v); % acceleration: the derivative of
 velocity w.r.t. time

    res = [dpdt; dvdt]; % pack the results in a column vector
end

function res = acceleration(t, p, v)
    % P1acceleration = 9.8  P>1accleration = -9.8 + drag acceleration
    % input...
    % t: time
```

```
    % p: position
    % v: velocity
    % output...
    % res: acceleration

    % do not modify this function unless required by you for some
 reason!

    a_grav = gravityFind(p);

    if part == 1 % variable part is from workspace of function main.
        res = a_grav;
    else
        m = mass(t, v);
        b = drag(t, p, v, m);

        f_drag = b * v^2;
        a_drag = f_drag / m;
        res = a_grav + a_drag;
    end
end

function a_grav = gravityFind(p)
    % estimate the acceleration due to gravity as a function of
 altitude, p

    if part <= 3
        a_grav = A_GRAV_SEA;
    else
        r = 6371000; %meters
        a_grav = A_GRAV_SEA*(r^2)./(r+p).^2;
    end
end

function res = mass(t, v)
    % mass in kg of Felix and all his equipment
    res = 3.63+49.9+90.72+27+73;
  %helmet,lifesupport,suit,parachute,felix
end

function res = drag(t, p, v, m)
% calculate the coefficient of drag

    if part == 2
        res = 0.2;
    else
        if part == 6
            rho = airDensity(p);    %produces a better model compared
 to stdatmo
        else
            rho = stdatmo(p);
        end
        ACd = ACdFind(p, m);
        res = 1/2*rho*ACd;
```

```
    end
end
```

Additional nested functions

Nest any other functions below.

```
%Do not put functions in other files when you submit.
function plotComparisons(timeCap,plotName,plotCaller)

    [T, Y] = ode45(@freefall, [0 timeCap], [ABV_SEA 0]);
    timeDataCap = find(elapsedTime > timeCap);
    timeDataCap = timeDataCap(1);

    if plotCaller == 0 || plotCaller == 1
    figure
    hold on
    plot(elapsedTime(1:timeDataCap), altitude(1:timeDataCap), '-*')
    plot(T, Y(:,1),'-o')
    title({plotName; '(Position)'});
    ylabel('Position')
    xlabel('Time (s)')
    legend('Actual', 'Modeled')
    hold off
    end

    if plotCaller == 0 || plotCaller == 2
    figure
    hold on
    plot(elapsedTime(1:timeDataCap), velocity(1:timeDataCap), '-*')
    plot(T, abs(Y(:,2)),'-o')
    title({plotName; '(Velocity)'});
    ylabel('Velocity')
    xlabel('Time (s)')
    legend('Actual', 'Modeled')
    hold off
    end

    if plotCaller == 0 || plotCaller == 3
    figure
    hold on
    ModeledAcc = abs(diff(Y(:,2)) ./ diff(T));
    plot(elapsedTime(1:timeDataCap), acclereation(1:timeDataCap), '-
*')
    plot(T(1:end-1), ModeledAcc, '-o')
    title({plotName; '(Acceleration)'});
    ylim([-20 40])
    ylabel('Acceleration')
    xlabel('Time (s)')
    legend('Actual', 'Modeled')
    hold off
```

```
    end
end

function res = openPara(paraArea, bodyArea, h1, h2, p)
    openArea = paraArea*((h1 - p)/(h1 - h2))^2;

    if openArea > bodyArea
        res = openArea;
    else
        res = bodyArea;
    end
end

function res = getACd(a, h, v, m)

    rho =  airDensity(h);
    a_grav = gravityFind(h);
    res = 2*(a - a_grav).*m./(rho.*v.^2);
end

function res = ACdFind(p, m)
    % free fall before the opening of parachute
    a = 0;

    % calculate body acd
    h = 27833;                    % body terminal velocity height
    v = 1357.6*1e3/(3.6e3);     % body terminal velocity velocity
    bodyACd = getACd(a, h, v, m);  % body ACd

    % calculate parachute acd when full open
    h = 1959;                    % parachute terminal velocity height
    v = 11/3.6;                  % parachute terminal velocity velocity
    paraACd = getACd(a, h, v, m); % parachute ACd

    % parachute opening and fully opened positions
%            h1Open = 2573;
           h1Open = 3400;
%      h1Open = 2673;
    h2Open = 2407;

    res = bodyACd;
    % logic
    if part <= 4
       res = bodyACd;
    elseif part == 5
        if p > h1Open
            % before he opens his parachue use his body acd
            res = bodyACd;
        else
            % after he opens his parachute use the para acd
            res = paraACd;
        end
    elseif part == 6
        if p > h1Open
```

```
            res = bodyACd;
        elseif p > h2Open
            res = openPara(paraACd, bodyACd, h1Open, h2Open, p);
        else
            res = paraACd;
        end
    end
end

function res = airDensity(h)    %https://en.wikipedia.org/wiki/
Density_of_air
    p0 = 101.325;
    T0 = 288.15;
    g = 9.80665;
    L = 0.0065;
    R = 8.31447;
    M = 0.0289644;
    absT = T0 - L*h;
    p = p0*(1-L*h./T0).^(g*M/(R*L))*1e3;
    res = (p*M)./(R*absT);
end

% end of nested functions

end % closes function main.
```

Published with MATLAB® R2017a