Table of Contents

ENSC180-Assignment2

```
% Student Name 1:  Vincent Le

% Student 1 #: 301301495

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

% Student Name 2: Sirpreet Kaur Dhillon

% Student 2 #: 301318486

% Student 2 userid (email): skd24 (skd24@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: _everybody helped us/me with the assignment:__
%           1) TA's
%           2) Jaime Lopez Raichs (student)
%           3) Andy Liu (student)
%           4) Hamlet Jiang Su (student)
```

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

clf

% constants
T_PART1 = 60;
HEIGHT_1 = 38969.4;
V_INIT = 0;
FILE = 'data_clean_more_fixed.xlsx';
COLS = 3;
T_PART3 = 270; % 4.5 minutes * 60 seconds / 1 minute (conversion)
T_PART5 = 524.49; % change this to the time that Felix stayed in fall
T_PART6 = (9 * 60) + 2;

% variables
TimeData = [];
HeightData = [];
VelocityData = [];
AllData = [];


% prepare the data
AllData = xlsread(FILE);

il = 1;
j = 1;

% in the following loop we remove all the rows with missing or nan
 values
% thus cleaning the data

    for xx = 1: length(AllData)
        for yy = 1:COLS
            y = AllData(il,j);
            if (isnan(y))
                AllData(il,:) = [];
                il = il - 1;
            end
            j = j+1;
        end
        il = il +1;
        j = 1;
    end
```

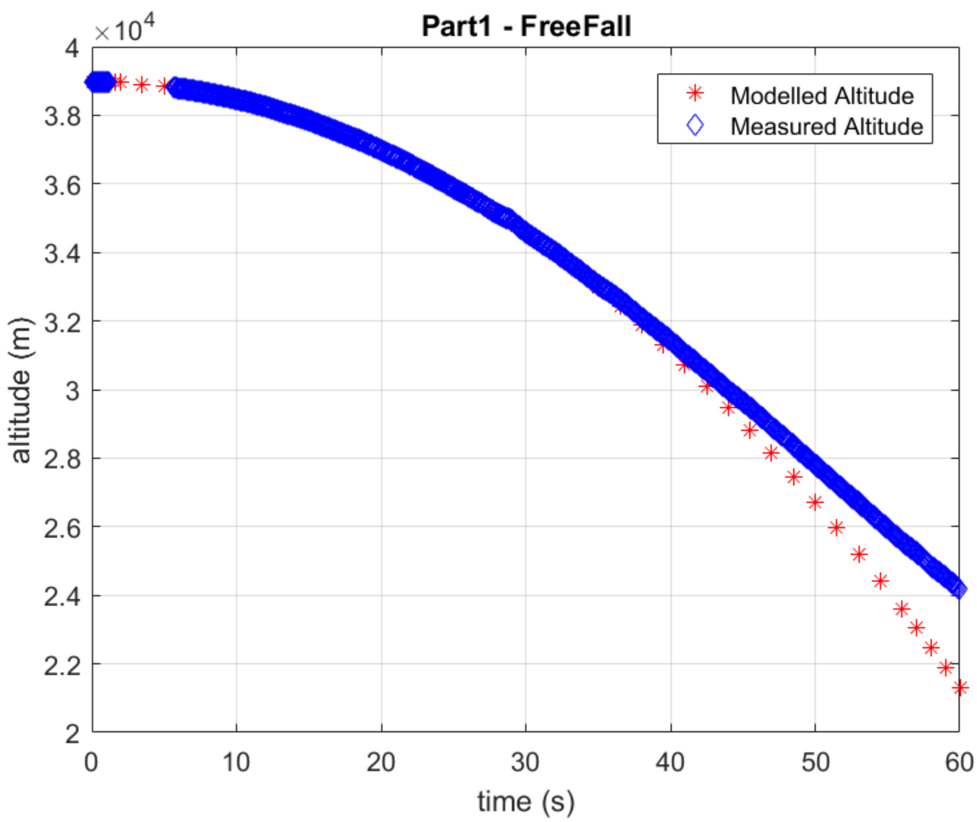
```
TimeData = AllData(:,1);  %extracting the time elapsed column in
 seconds from the spreadsheet
HeightData = AllData(:,2); %extracting the height column in meter from
 the spreadsheet
Velocity = AllData(:,3)*1000/3600; %extracting the velocity/airspeed
 column in kph from the spreadsheet and at the same time converting it
 into mps
YData = [HeightData, Velocity];
v_t = max(Velocity); % constant terminal velocity taken from the table
 given



% We converted velocity from kph to mps when we took it out of the
 table I
% believe these values are consistent now
```

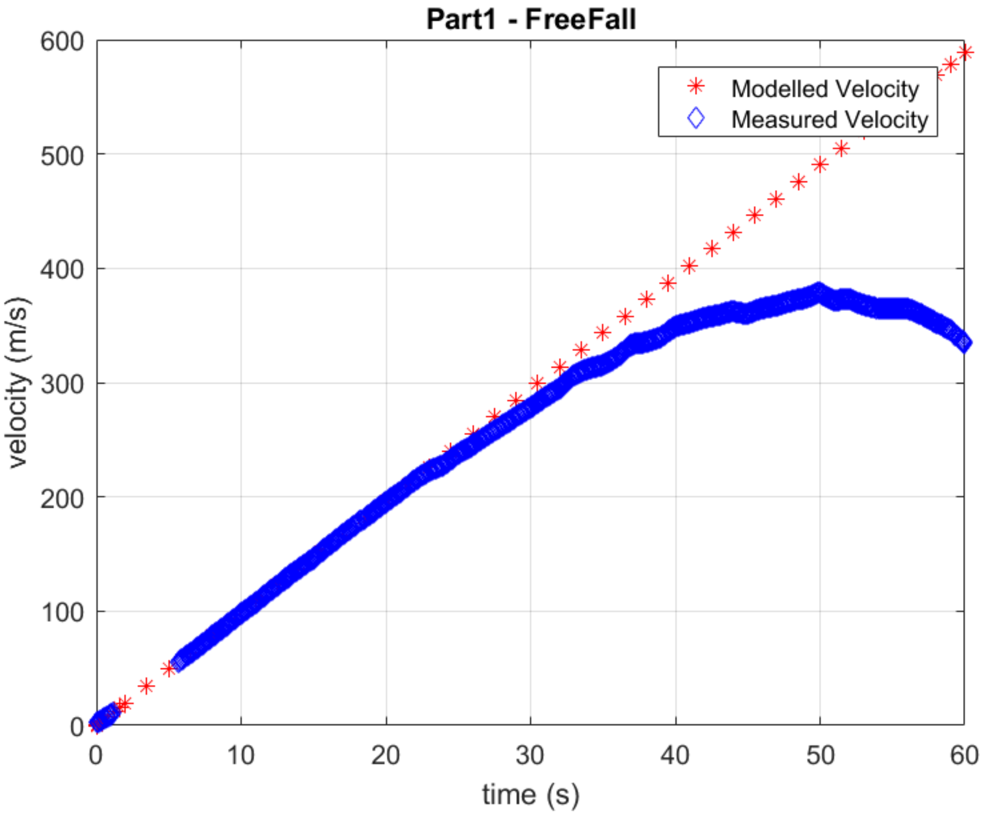
Part 1

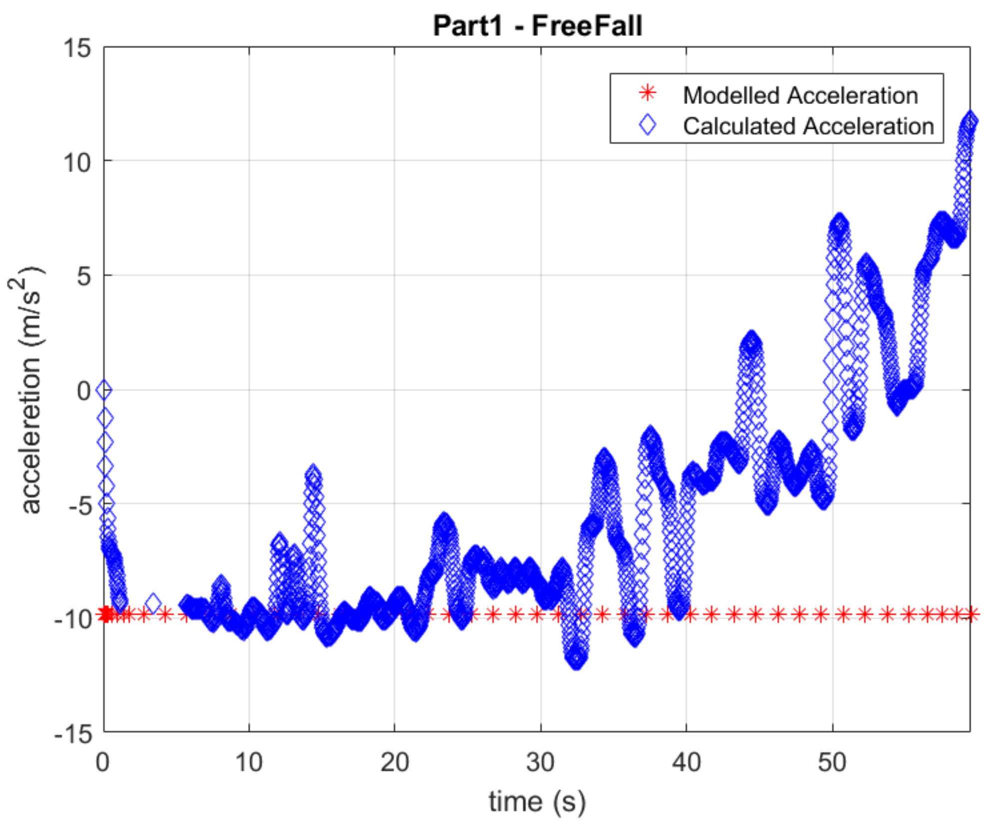
How accurate is the model for the first portion of the minute? The model is very accurate to the first half of the minute as there almost no deflection from the measured data.

```
% How accurate is the model for the last portion of that first minute?
% The model is not very accurate to the last half of the minute as
 there
% is significant deflection from the measured data
```

```
% Comment on the acceleration calculated from the measured data.
% We used the relation that change in velocity divided by the change
 in
% time to calculate acceleration. The plot for acceleration was had a
 lot
% peeks and valleys so we used the smooth function to smooth out the
 plot
% to some extent.

part = 1;
%setting up the data
[T_model,Y_model] = ode45(@fall, [0,T_PART1], [HEIGHT_1, V_INIT]); %
 here we call the ode45 to model Felix's fall
string1 = 'Part1 - FreeFall';
t_end = T_model(end);
% <call here your function to create your plots>
plotComparisons(t_end, string1 , T_model, TimeData, Y_model, YData)
```



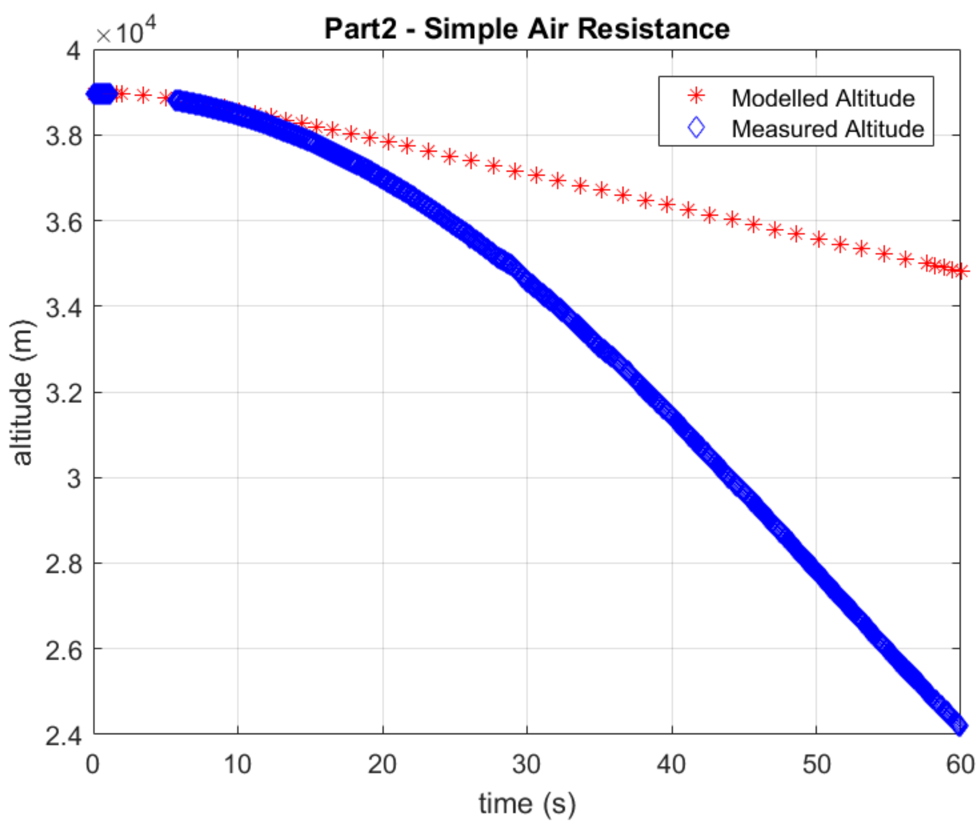

Part1 - FreeFall
Modelled Velocity
Measured Velocity
velocity (m/s)
time (s)
0
100
200
300
400
500
600
0
10
20
30
40
50
60

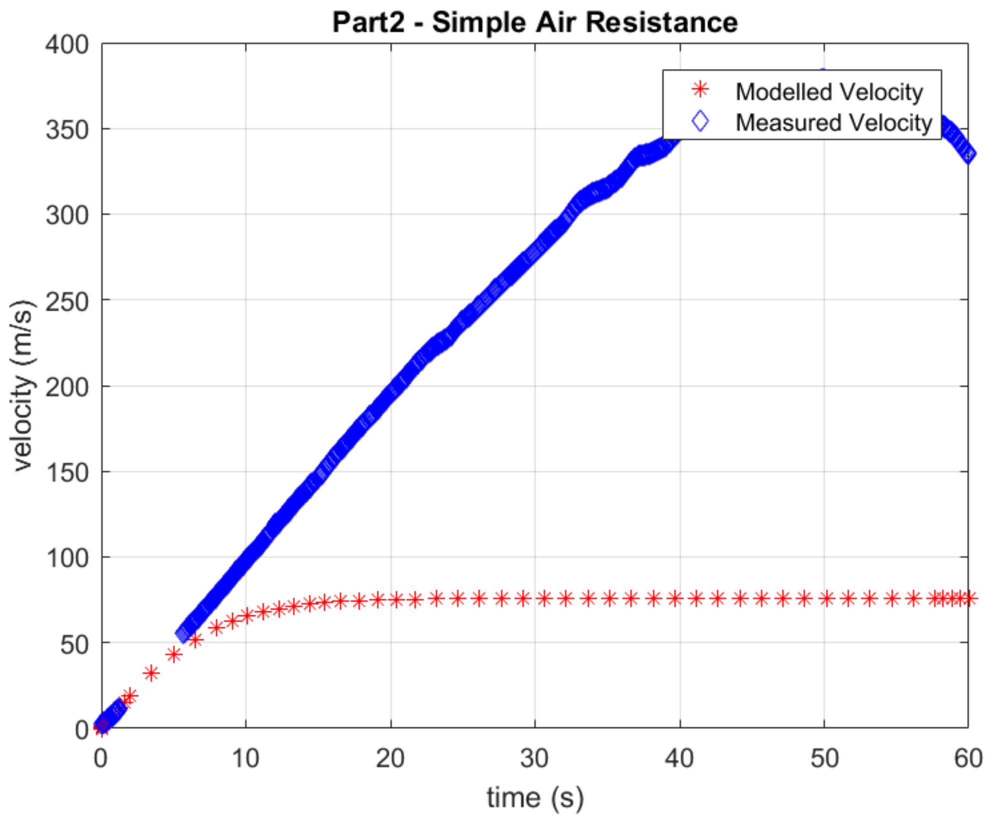

Part1 - FreeFall
Modelled Acceleration
Calculated Acceleration
acceleretion (m/s2)
time (s)
15
10
5
0
-5
-10
-15
0
10
20
30
40
50

Part 2

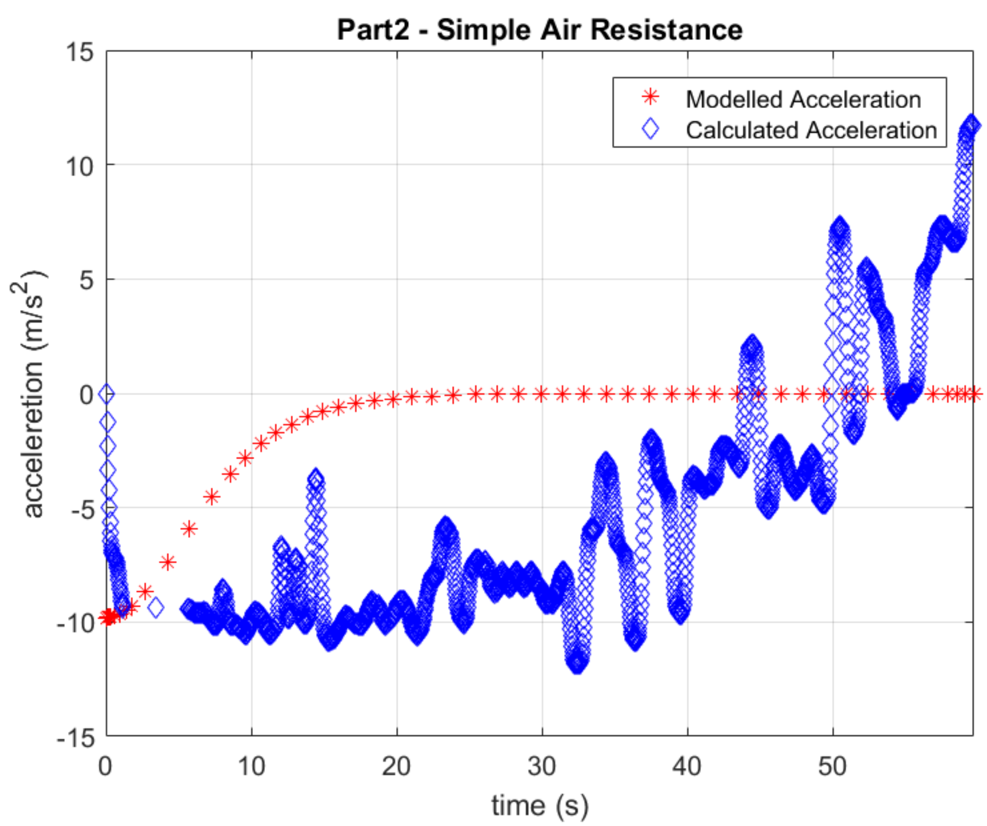
Estimated uncertainty in the mass that you have chosen (at the beginning of the jump): Found on the red bull stratos jump site and a scientific report by the University of Leicester that the mass of Felix along with the suit was 260 pounds or 117.93 kg. The uncertainity could be plus minus 1 pound.

```
% How sensitive is the velocity and altitude reached after 60 seconds
 to
%    changes in the chosen mass?
% Altitude sensitivity at 60 seconds: 34815.386 - 21316.8 = 13498.586m
% Velocity sensitivity at 60 seconds: 588.42 - 76.044 = 512.376m/s

part = 2;

[T_model,Y_model] = ode45(@fall, [0,T_PART1], [HEIGHT_1, V_INIT]); %
 here we call the ode45 to model Felix's fall
string2 = 'Part2 - Simple Air Resistance';
t_end = T_model(end);
% <call here your function to create your plots>
plotComparisons(t_end, string2 , T_model, TimeData, Y_model, YData)
```





Part2 - Simple Air Resistance
Modelled Velocity
Measured Velocity
velocity (m/s)
time (s)
0
50
100
150
200
250
300
350
400
0
10
20
30
40
50
60

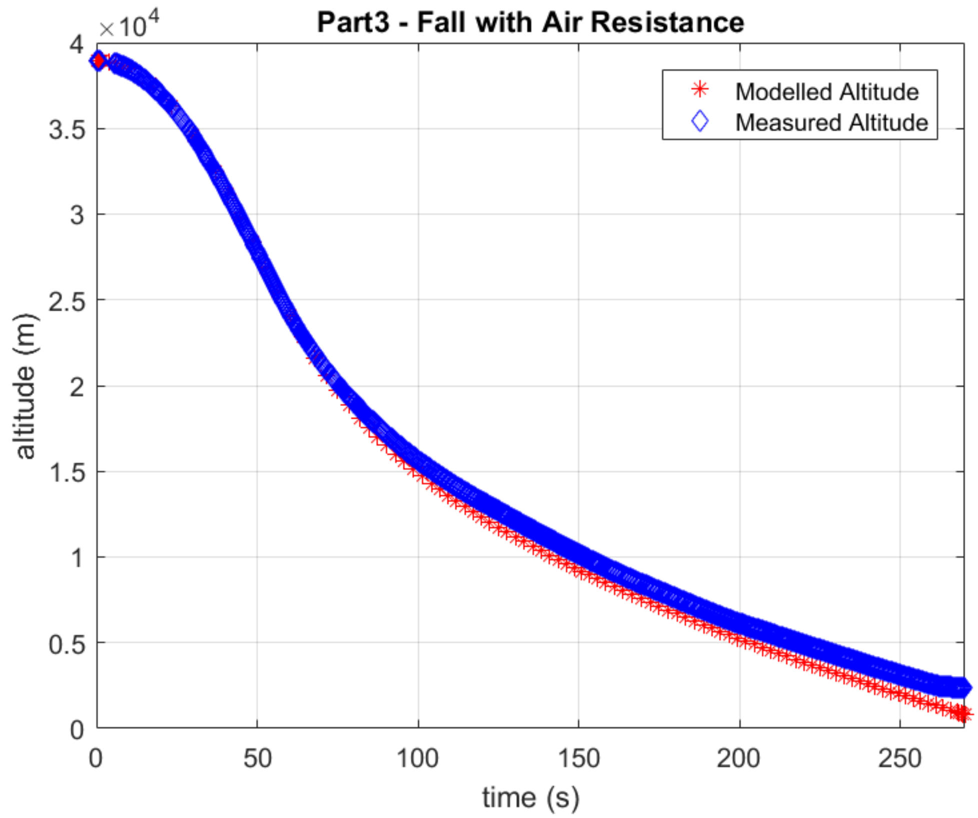


Part2 - Simple Air Resistance
Modelled Acceleration
Calculated Acceleration
acceleretion (m/s2)
time (s)
15
10
5
0
-5
-10
-15
0
10
20
30
40
50

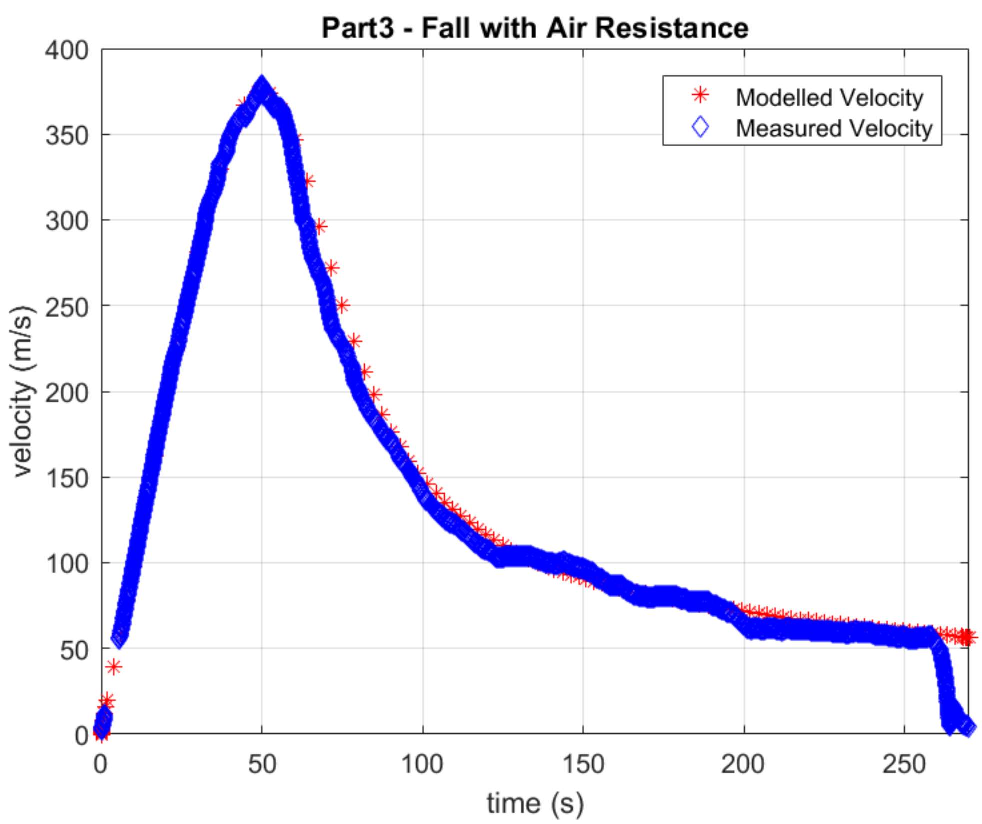
Part 3

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? Humans aren't conditioned for the density at the top of the stratosphere. Plus the oxygen levels at that height are really low so he would have died from the lack of oxygen. pressure inside his body would be greater than the atmospheric pressure and due to science, he would have exploded. density in the stratosphere is less than that 39,000 meters below on Earth. the suit for safety because he surpassed the speed of sound, protected from the heat

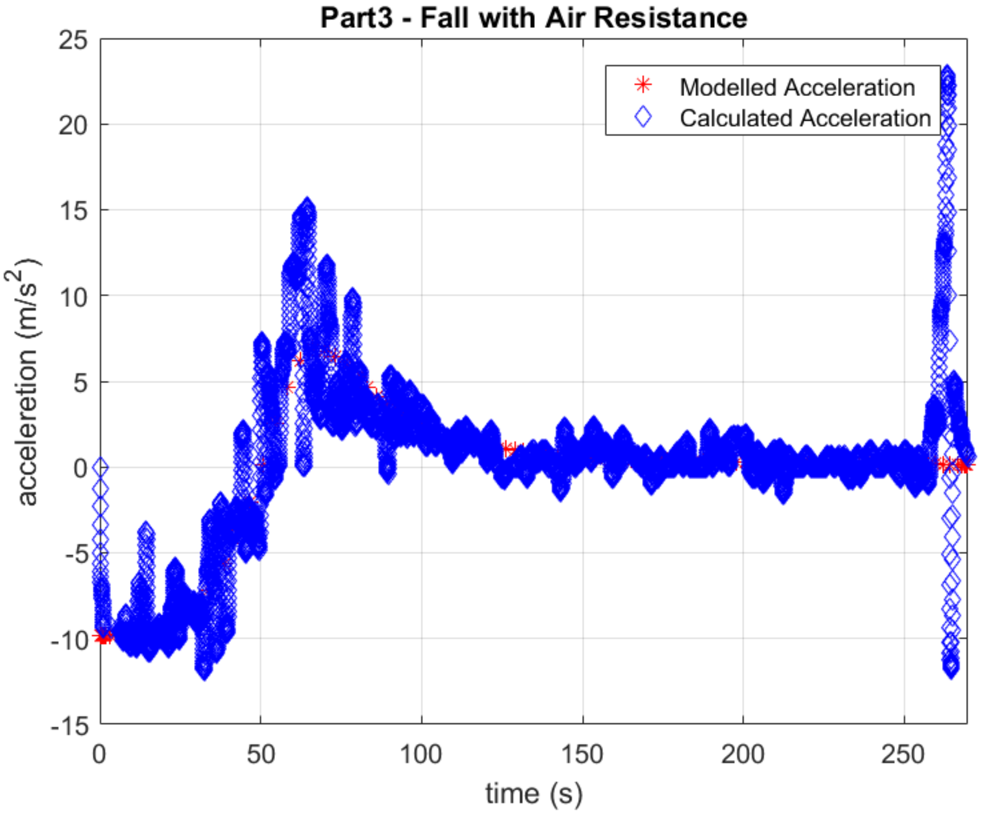
```
% What are the factors involved in calculating the density of air?
%     How do those factors change when we end up at the ground but
 start
%     at the stratosphere?  Please explain how calculating air density
 up
%     to the stratosphere is more complicated than say just in the
 troposphere.
% <The factors in calculating air density are the altitude and the
 temperature offset which are used to calculate the pressure which
 then gets the density.>
% <Altitude decreases from the stratosphere to the ground, temperature
 increases as well. Pressure will increases due to this.>
% <the temperature formula: T = T0 - L*h (Stoke's law) is only valid
 for the troposphere. And therefore to calculate density in the
 stratosphere we used different equations which were more complicated>

% What method(s) can we employ to estimate [the ACd] product?
% <When terminal velocity is reached, force due to drag is equal to
 force due to gravity. This simplifies the equation F_net equals Fg -
 Fd = 0 which in turn finds us the ACd product from Fd.>
% <We assumed it as constant up until parachute opening>

% What is your estimated [ACd] product?
% <0.65>
%
% [Given what we are told in the textbook about the simple drag
 constant, b,]
%   does the estimate for ACd seem reasonable?
% <Yes it seems reasonable>

part = 3;

% <place your work here>
[T_model,Y_model] = ode45(@fall, [0,T_PART3], [HEIGHT_1, V_INIT]); %
 here we call the ode45 to model Felix's fall
string3 = 'Part3 - Fall with Air Resistance';
t_end = T_model(end);
% % <function is called to create plots>
plotComparisons(t_end, string3 , T_model, TimeData, Y_model, YData)
```



Part3 - Fall with Air Resistance
×10^4
altitude (m)
time (s)
Modelled Altitude
Measured Altitude



Part3 - Fall with Air Resistance
velocity (m/s)
time (s)
Modelled Velocity
Measured Velocity

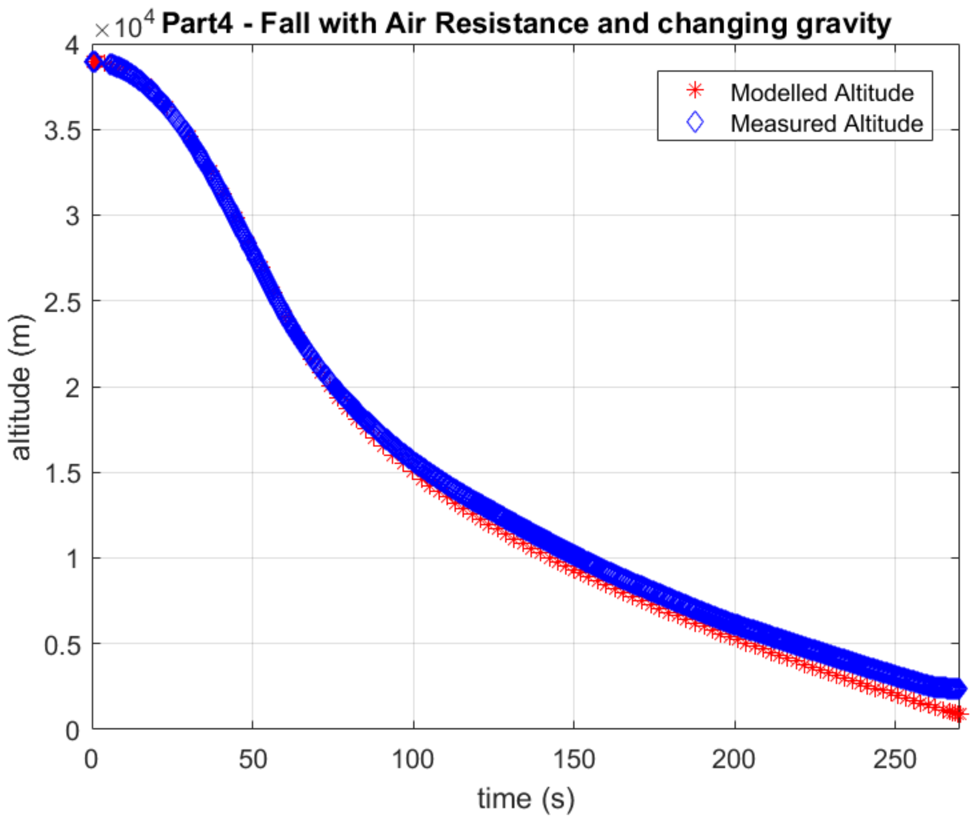


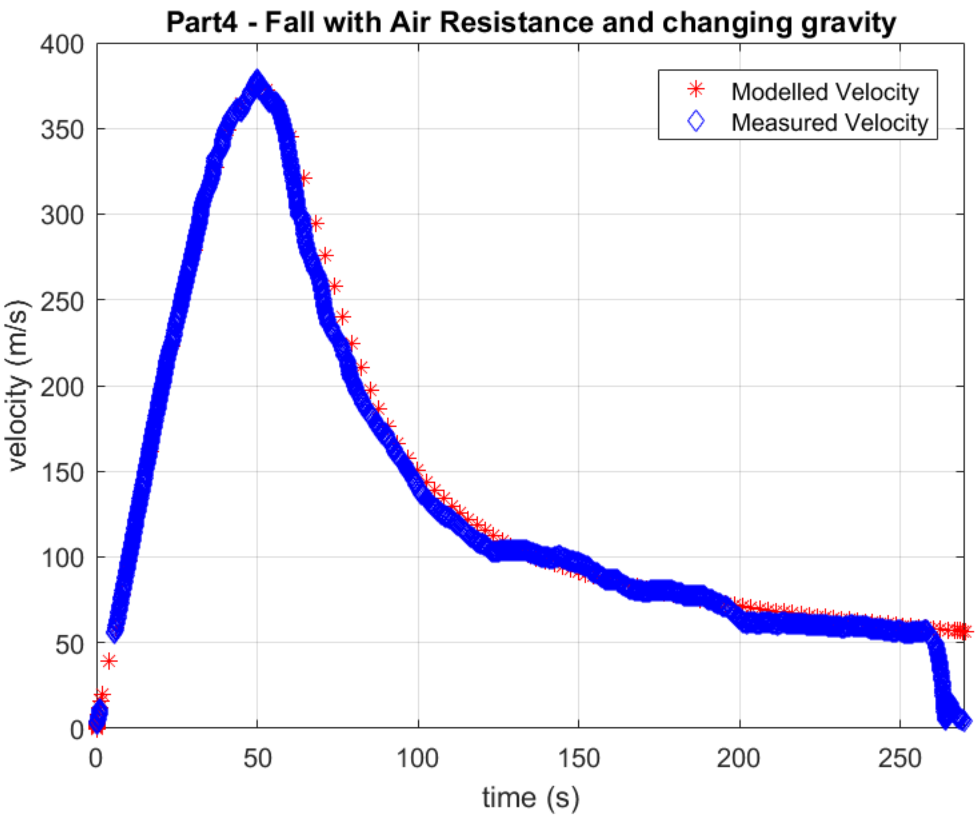
Part 4

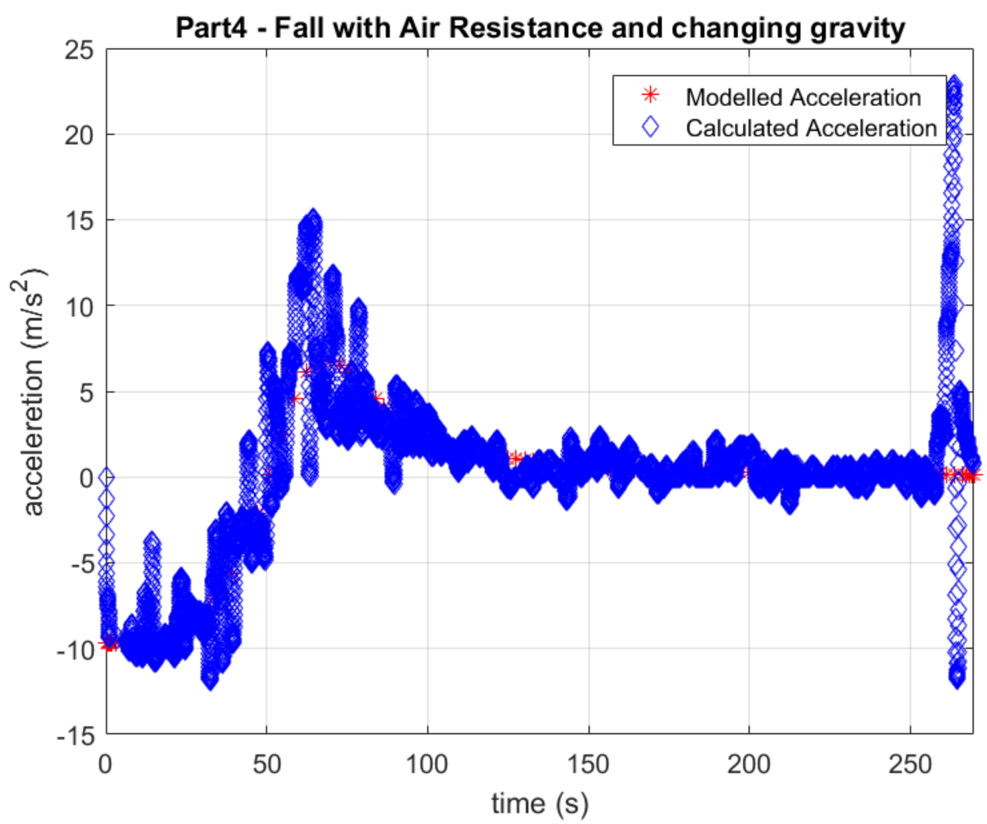
Answer some questions here in these comments... What is the actual gravitational field strength around 39,000 meters? (See Tipler Volume 1 6e page 369.) [9.74m/s^2]

```
% How sensitive is the altitude reached after 4.5 minutes to simpler
 and
%   more complicated ways of modelling the gravitational field
 strength?
% <857.9 m - 819 m = 38.9 m>

% 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.
% <We changed the gravitatonal field strength. GM1M2/R^2. As R
 decreases because Felix is falling, the gravitational field strength
 increases>

% 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?
% <The change in gravitational field strength did not make much of a
 difference>

% How can we decide what change is significant and what change is
%   insignificant?
```

```
% <If safety is assured then the change is insignificant>

% [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.)]
% <We attempted finding each variable to obtain more precise values.
 The attempt proved to be difficult.>

part = 4;

% <place your work here>
[T_model,Y_model] = ode45(@fall, [0,T_PART3], [HEIGHT_1, V_INIT]); %
 here we call the ode45 to model Felix's fall
string4 = 'Part4 - Fall with Air Resistance and changing gravity';
t_end = T_model(end);
% % <function is called to create plots>
plotComparisons(t_end, string4 , T_model, TimeData, Y_model, YData)
```



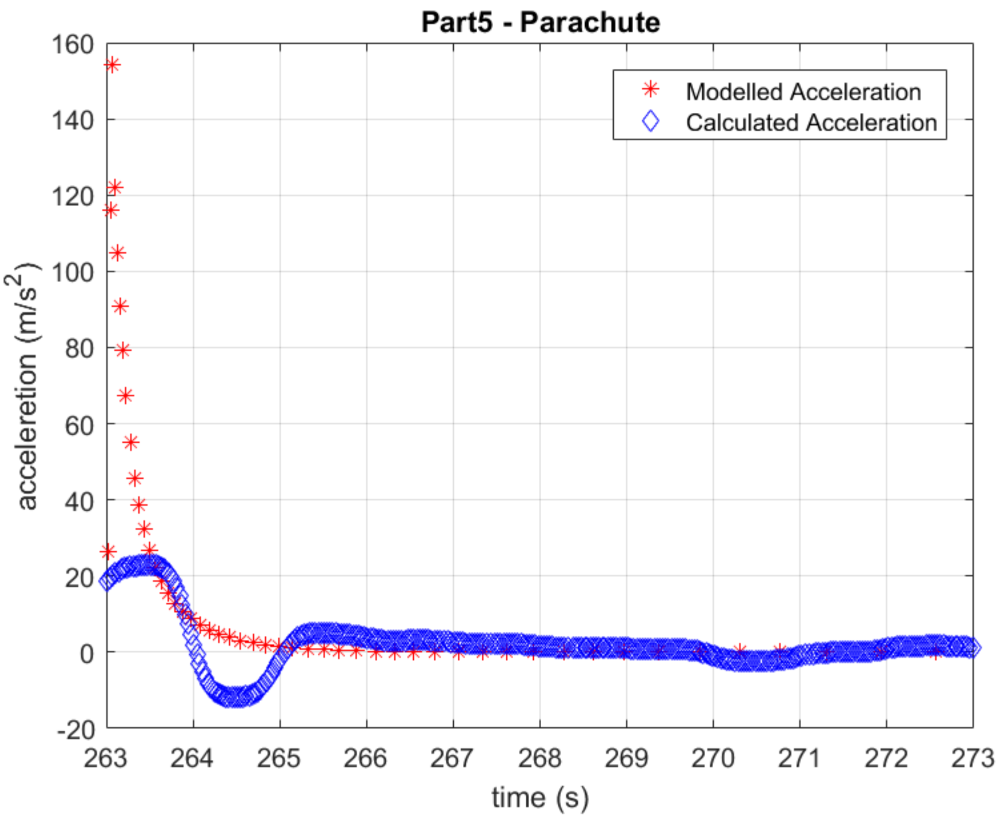

Part4 - Fall with Air Resistance and changing gravity
Modelled Velocity
Measured Velocity
velocity (m/s)
time (s)
400
350
300
250
200
150
100
50
0
0
50
100
150
200
250


Part4 - Fall with Air Resistance and changing gravity
Modelled Acceleration
Calculated Acceleration
acceleretion (m/s2)
time (s)
25
20
15
10
5
0
-5
-10
-15
0
50
100
150
200
250

Part 5

Answer some questions here in these comments... At what altitude does Felix pull the ripcord to deploy his parachute? 2750 meters

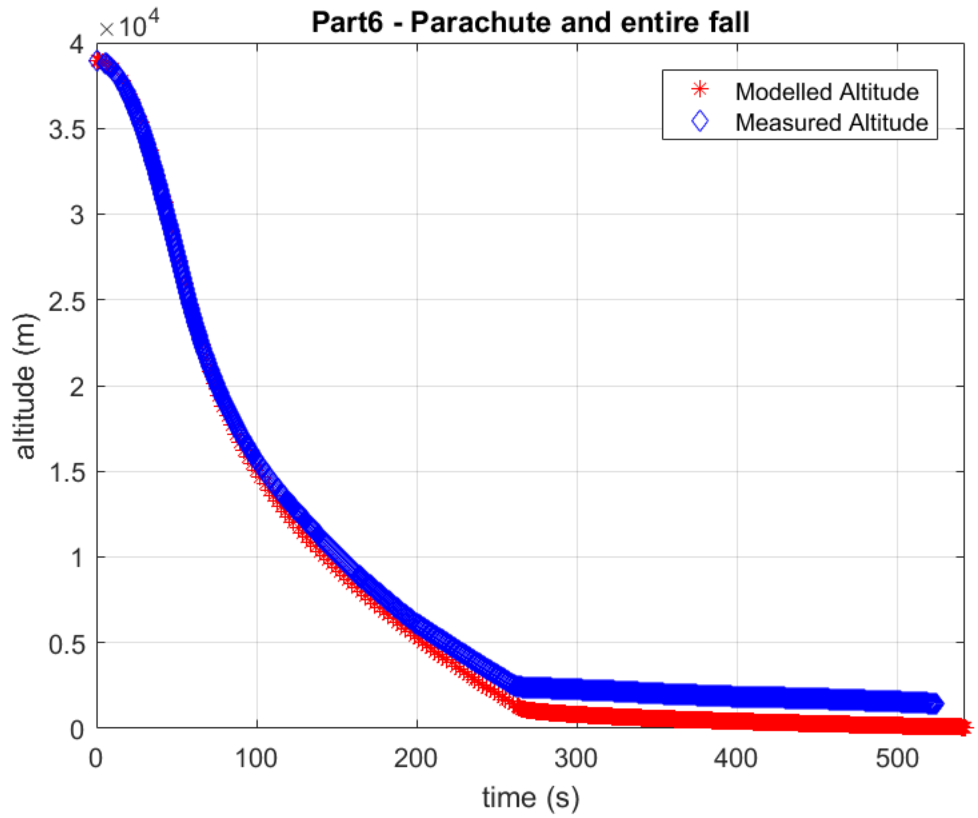
```
% Recalculate the ACd product with the parachute open, and modify your
%   code so that you use one ACd product before and one after this
 altitude.
%   According to this version of the model, what is the maximum
 magnitude
%   of acceleration that Felix experiences?
% <154.35m/s^2>

%   How safe or unsafe would such an acceleration be for Felix?
% <Very unsafe according to the number above>

part = 5;

%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.
%increment initial value of acd <siri>
% <place your work here>
[T_model,Y_model] = ode45(@fall, [0,T_PART5], [HEIGHT_1, V_INIT]); %
 here we call the ode45 to model Felix's fall
string5 = 'Part5 - Parachute';
t_end = T_model(end);
% % <function is called to create plots>
plotComparisons(t_end, string5 , T_model, TimeData, Y_model, YData)
```

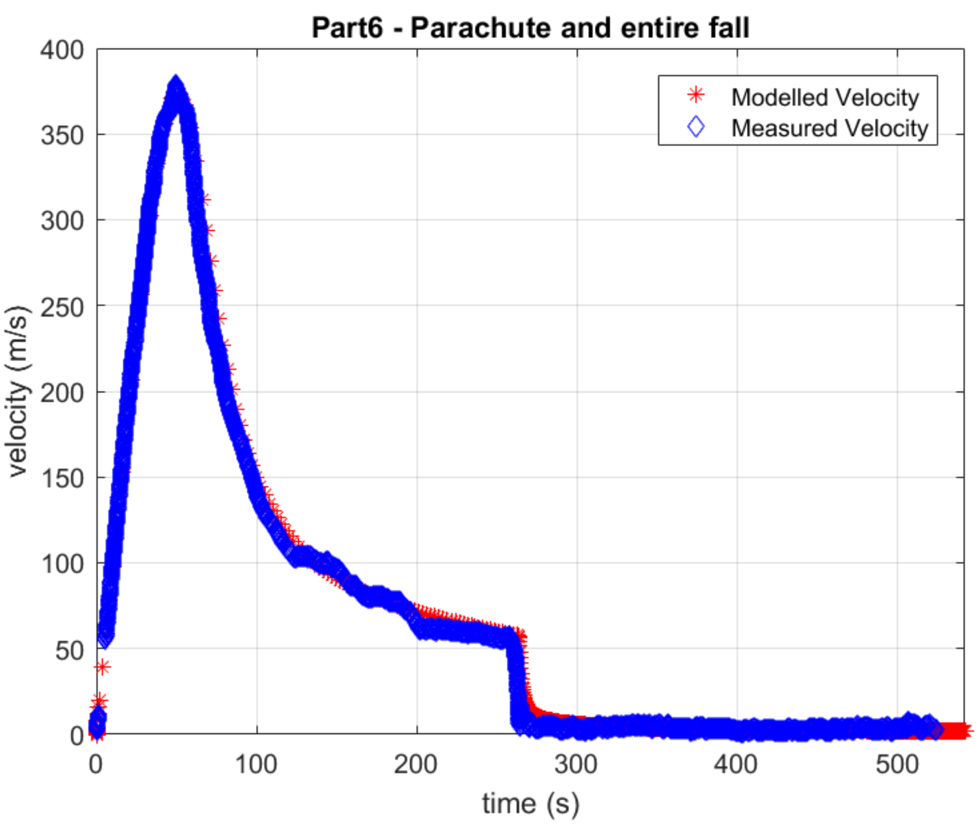


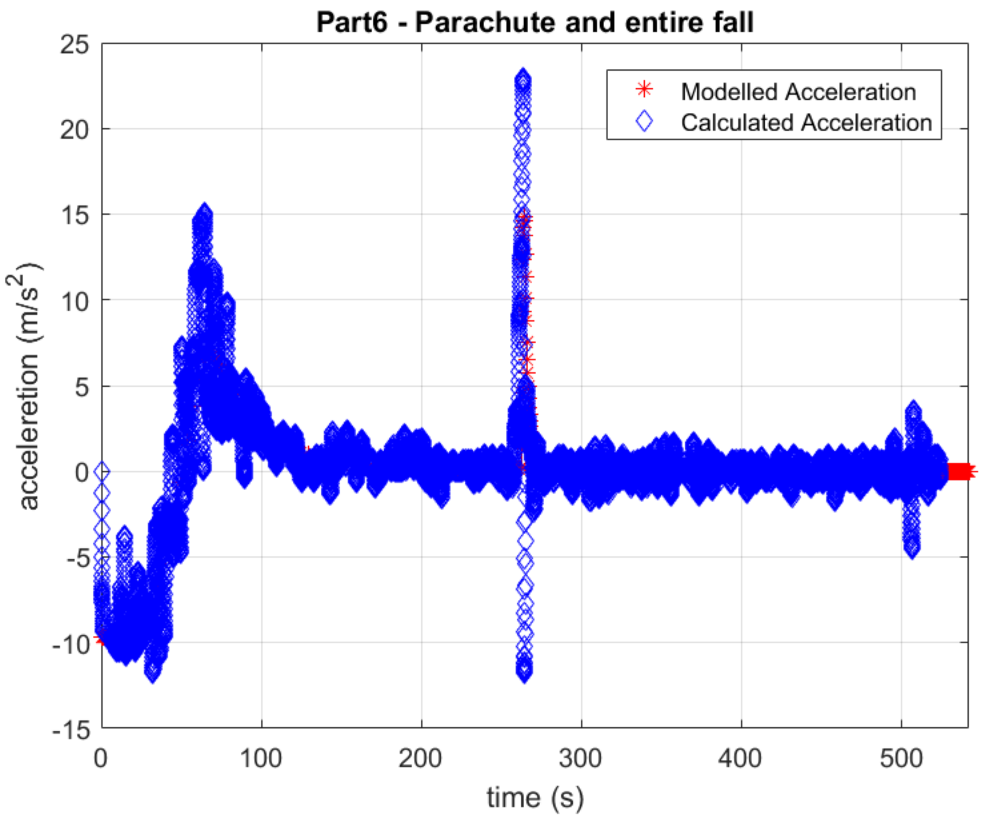
Part 6

Answer some questions here in these comments... How long does it take for Felix's parachute to open? It takes 7 seconds for Feix's parachute to open

```
part = 6;

%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.
% <place your work here>
[T_model,Y_model] = ode45(@fall, [0,T_PART6], [HEIGHT_1, V_INIT]); %
 here we call the ode45 to model Felix's fall
string6 = 'Part6 - Parachute and entire fall';
t_end = T_model(end);
% % <function is called to create plots>
plotComparisons(t_end, string6 , T_model, TimeData, Y_model, YData)
```


Part6 - Parachute and entire fall
Modelled Altitude
Measured Altitude
altitude (m)
×10^4
4
3.5
3
2.5
2
1.5
1
0.5
0
100
200
300
400
500
time (s)


Part6 - Parachute and entire fall
Modelled Velocity
Measured Velocity
velocity (m/s)
400
350
300
250
200
150
100
50
0
100
200
300
400
500
time (s)



nested functions

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

```
function res = fall(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)
    % This function calculates the acceleration  and returns it to the
 fall function
    % input:                   units:
```

```
    % t: time                 s
    % p: position             m
    % v: velocity             m/s
    % output:
    % res: acceleration       m/s^2

    % do not modify this function unless required by you for some
 reason!
    % no changes were made to this function

    a_grav = gravityEst(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

% Please paste in or type in code into the below functions as may be
 needed.

function a_grav = gravityEst(p)
    % estimate the acceleration due to gravity as a function of
 altitude, p
    A_GRAV_SEA = 9.807;  % acceleration of gravity at sea level in m/
s^2
    RAD_EARTH = 6400000; % radius of Earth in meter
    TEMP = (RAD_EARTH/(RAD_EARTH + p));
     if (part == 1 || part == 2 || part == 3)
        a_grav = A_GRAV_SEA;
     else
        a_grav = A_GRAV_SEA * power(TEMP,2);
     end
end

function res = mass(t, v)
    % mass in kg of Felix and all his equipment
    res = 117.93;
end

function res = drag(t, p, v, m)
% <insert description of function here>
    DRAG_CONST = 0.2;
    if part == 2
        res = DRAG_CONST;
    else
        % air resistance drag = 1/2*rho*A*Cd
        ACd = AC(t,p,v);
```

```
        rho = density(p);
        res = 0.5 * rho * ACd;
    end
end
```

Additional nested functions

Nest any other functions below.

```
%Do not put functions in other files when you submit.

    function plotComparisons(t_end, string1 , T_model, TimeData,
 Y_model, YData)
        if (part ==1)
            f = 1;
            g = 2;
            h = 3;
        else if(part == 2)
               f = 4;
               g = 5;
               h = 6;
            else if(part == 3)
                    f = 7;
                    g = 8;
                    h = 9;
                else if (part == 4)
                        f = 10;
                        g = 11;
                        h = 12;
                    else if (part == 5)
                            h = 13;
                        else if (part == 6)
                              f = 14;
                              g = 15;
                              h = 16;
                            end
                        end
                    end
                end
            end
        end



            %plotting Felix's altitude/height in m
            P = Y_model(:,1);
            P1 = YData (:,1);
            T = T_model(:,1);
            T1 = TimeData(:,1);
            V = abs(Y_model(:,2));
            V1 = abs(YData(:,2));
            [dT,A] = myacceleration(T, V);
```

```
            [dT1,A1] = myacceleration(T1, V1);

        if (part <= 4 || part == 6)
            figure(f)
            plot(T,P,'r*');
            hold on
            plot(T1,P1,'bd');
            title( string1 )
            xlim([0 t_end])
            xlabel('time (s)');
            ylabel('altitude (m)')
            grid on
            legend('Modelled Altitude','Measured Altitude')
            hold off

            %plotting Felix's velocity in mps

            figure(g)
            plot(T,V,'r*');
            hold on
            plot(T1,V1,'bd');
            title( string1 )
            xlim([0 t_end])
            xlabel('time (s)');
            ylabel('velocity (m/s)')
            grid on
            legend('Modelled Velocity','Measured Velocity')
            hold off

            % now we are plotting the calculated acceleration

            c =
smooth(smooth(smooth(smooth(smooth(smooth(smooth(smooth(smooth(smooth(smoo
            len = length(A1);
            C1 = reshape(c,len,1);
            figure(h)
            t_endTemp = dT(end);
            plot(dT,A,'r*');
            hold on
            plot(dT1,C1,'bd');
            title( string1 )
            xlim([0 t_endTemp])
            xlabel('time (s)');
            ylabel('acceleretion (m/s^2)')
            grid on
            legend('Modelled Acceleration','Calculated Acceleration')
            hold off
        end

        if (part == 5)
            c =
smooth(smooth(smooth(smooth(smooth(smooth(smooth(smooth(smooth(smooth(smooth(smoo
            len = length(A1);
            C1 = reshape(c,len,1);
```

```
            figure(h)
            plot(dT,A,'r*');
            hold on
            plot(dT1,C1,'bd');
            title( string1 )
            xlim([263 273])
            xlabel('time (s)');
            ylabel('acceleretion (m/s^2)')
            grid on
            legend('Modelled Acceleration','Calculated Acceleration')
            hold off
         end



    end

    function [T2,A] = myacceleration(T, V)
        T_f = [];
        T = T(:,1);
        A = [];
        dT = diff(T);
        dV = diff(V);
        A = [A; (-1)*(dV./dT)];

        for il = 1:(length(T)-1)
            T_f = [T_f; ((T(il,1)+T((il+1),1))/2)];
        end
        T2 = T_f;
    end

    function ACd = AC(t,p,v)
         %terminal velocity v_t is used in this function
        g_const_vt = 9.757504543 ;%Meters per second squared
        ma = 117.93;
        ro = density(27873);
        b = g_const_vt * ma / power(v_t,2);
        ACd1 = (b*2)/ro;
        Cd = ACd1/(1.7 * 0.8); % trying to take out the Cd by assuming
Felix's area
        ACd2 = 25 * Cd;

         if(t>263.02 && part == 5)
             ACd = ACd2;
         else if (part == 6 && t>263.02)
                 ACd = chute(t, ACd1, ACd2);
             else
                 ACd = ACd1;
             end
         end
    end

    function res = density(p)
```

```
        P_NOT = 101.325 * 10^3; %Pressure in Pascal
        T_NOT = 288.15 ;%In Kelvin
        g_const = 9.80665 ;%Meters per second squared
        L = 0.0065 ;%Kelvin per meters
        R_CONST = 8.31447 ;%joules per mole * kelvin | universal gas
constant
        M = 0.0289644 ;%kg per mole | molar mass of air

        pressure_power = (g_const*M)/(R_CONST*L);
        pressure_base = 1-(L*p/T_NOT);
        if (p > 25000) % upper stratosphere
            Temp = -131.21 + (0.00299 * p);
            Pressure = 2.488 * power( (((Temp + 273.1)/216.6)),
(-11.388) );
            rho = Pressure/(0.2869 * (Temp + 273.1));
        else if (p <=25000 && p > 11000) % lower stratospere
                Temp = - 56.46;
                Pressure = 22.65 * exp(1.73 - 0.000157 * p);
                rho = Pressure/(0.2869 * (Temp + 273.1));
            else if (p  < 11000) % troposphere
                    Temp = T_NOT - (L*p);
                    Pressure =
P_NOT*power(pressure_base,pressure_power);
                    rho = (Pressure * M) / (R_CONST * Temp);
                end
            end
        end
        res = rho;
    end

    function ACd_chute = chute(t,ACdi, ACdf)
       % use this function to return the ACd value
       ti = 263.02;
       tf = 270;
       slope = (ACdf - ACdi)/(tf - ti);
       ACd_chute = ACdi + slope * (t - ti);

    end


% end of nested functions

end % closes function main.
```

Published with MATLAB® R2016b