Table of Contents

ENSC180-Assignment2

```
% Student Name 1: Joseph Dillman

% Student 1 #: 301317623

% Student 1 userid (email): jdillman@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: George Lertzman, Sterling Smith
```

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
```

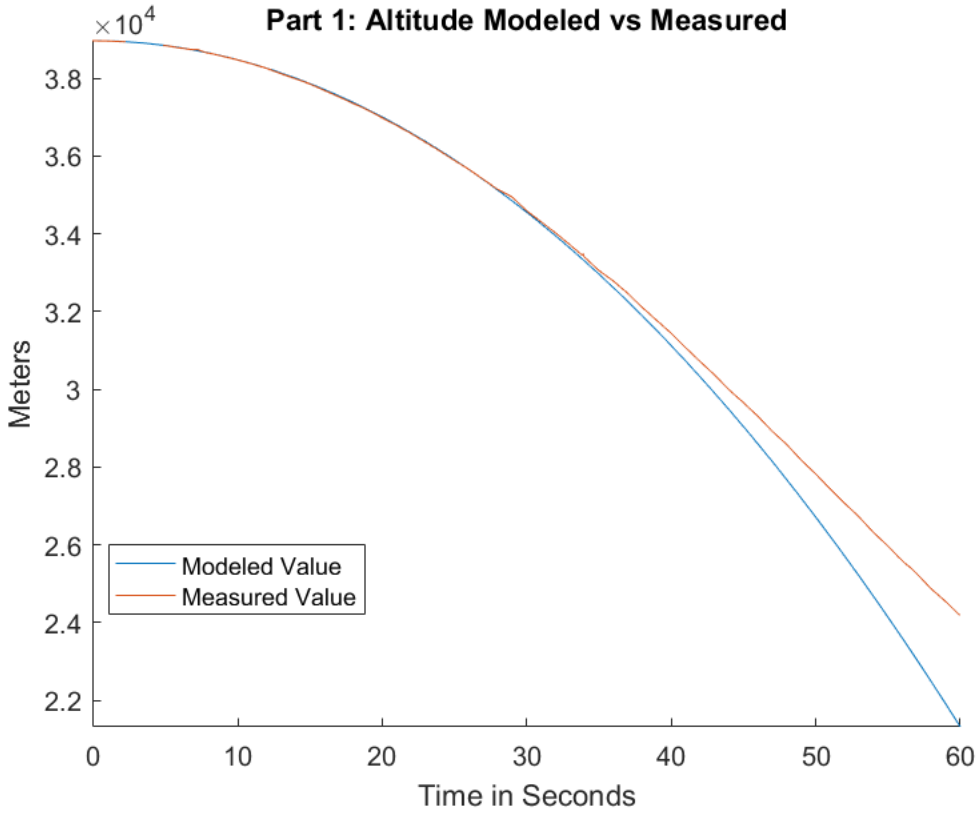
```
% constants -- you can put constants for the program here
%MY_CONST = 123;

% variables -- you can put variables for the program here
%myVar = 456;

% prepare the data
%the following vectors are used to create graphs of length 60 seconds
%or 4.5 minutes. Calculaing acceleration is done manually using the
 diff()
%function so each vector must be one element longer
filename = 'RedBullStratosData180.xlsx';
altitudetotal = xlsread(filename,'Data','D4:D403');
airspeedtotal = (xlsread(filename,'Data','E4:E403'))/3.6;
timetotal = xlsread(filename,'Data','K4:K403');
acceltotal = diff((xlsread(filename,'Data','E4:E404'))/3.6)./
diff(xlsread(filename,'Data','K4:K404'));

altitude1min = xlsread(filename,'Data','D4:D284');
airspeed1min = (xlsread(filename,'Data','E4:E284'))/3.6;
time1min = xlsread(filename,'Data','K4:K284');
accel1min = diff((xlsread(filename,'Data','E4:E285'))/3.6)./
diff(xlsread(filename,'Data','K4:K285'));

altitude270sec = xlsread(filename,'Data','D4:D396');
airspeed270sec = (xlsread(filename,'Data','E4:E396'))/3.6;
time270sec = xlsread(filename,'Data','K4:K396');
accel270sec = diff((xlsread(filename,'Data','E4:E397'))/3.6)./
diff(xlsread(filename,'Data','K4:K397'));

% <put here any conversions that are necessary>
```

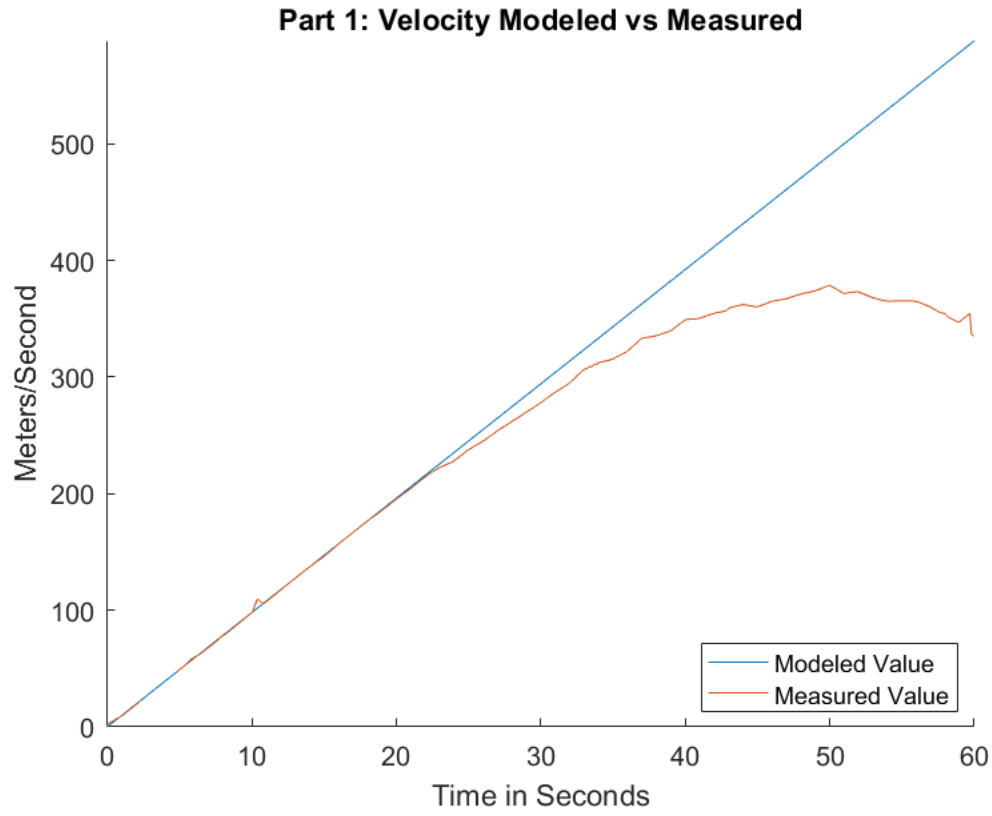
Part 1

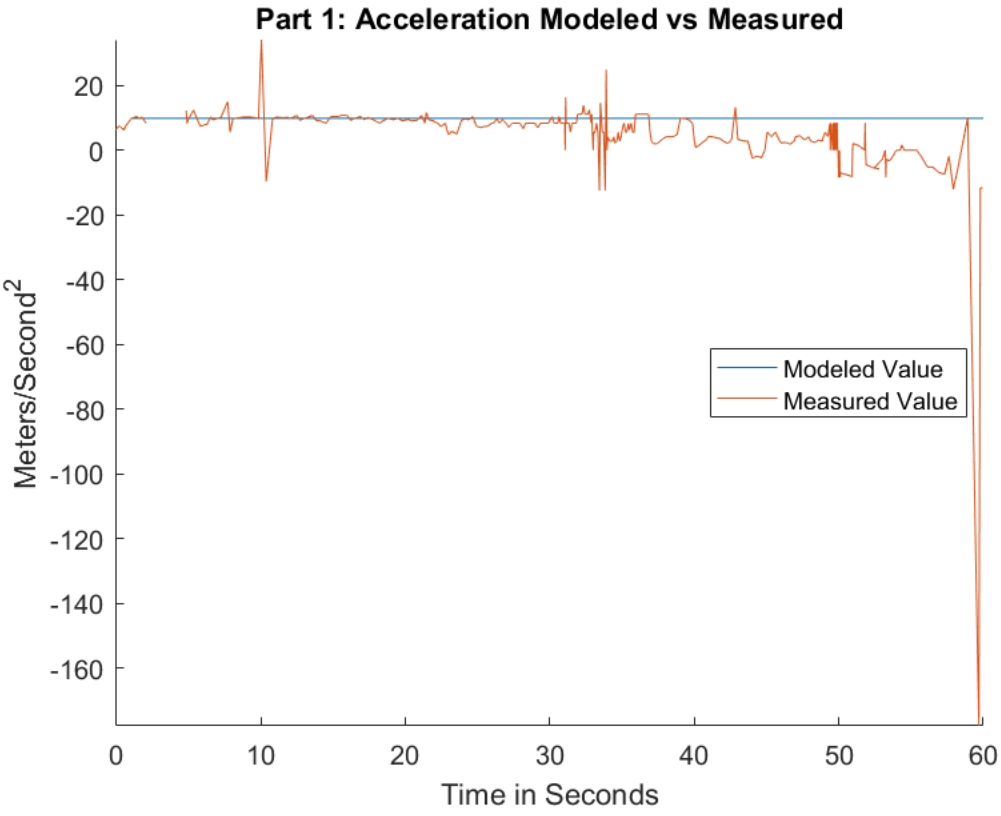
Answer some questions here in these comments... How accurate is the model for the first portion of the minute?

```
    %The model is reasonably accurate for the first portion of the
 minute,
    %however at around the 30-40 second mark it starts to trail off

% How accurate is the model for the last portion of that first minute?
    %The model starts to trail off as it accelerates however the
 measured
    %data is affected by air resistance which lowers the acceleration
 and
    %maintains velocity.

% Comment on the acceleration calculated from the measured data.
% Is there any way to smooth the acceleration calculated from the
 data?
    %One can remove outliers from the data manually or use smoothing
    %functions to take the running averages of the values to prevent
 large
    %jumps in acceleration

part = 1;
 [T, M]= ode45(@fall, [0, 60], [38969.4, 0]);
```

```
%calling plotcomparisons function for altitude, velocity, acceleration
plotComparisons(1,'Part 1: Altitude Modeled vs Measured', 'Time in
 Seconds'...
    , 'Meters', T, M(:,1), time1min, altitude1min)
plotComparisons(2,'Part 1: Velocity Modeled vs Measured', 'Time in
 Seconds'...
    , 'Meters/Second', T, abs(M(:,2)), time1min, airspeed1min)
plotComparisons(3,'Part 1: Acceleration Modeled vs Measured', 'Time in
 Seconds'...
    , 'Meters/Second^2', [1 60], [9.81 9.81], time1min, accel1min)
```



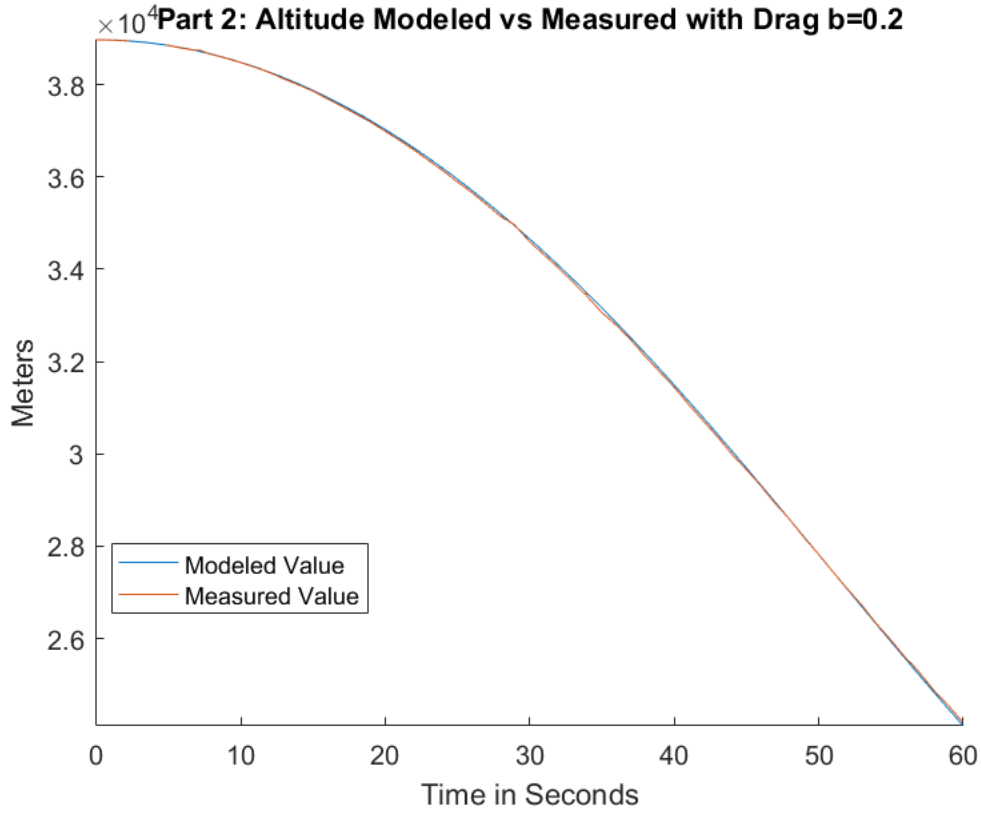

Part 1: Velocity Modeled vs Measured
Meters/Second
Time in Seconds
Modeled Value
Measured Value
0
100
200
300
400
500
0
10
20
30
40
50
60

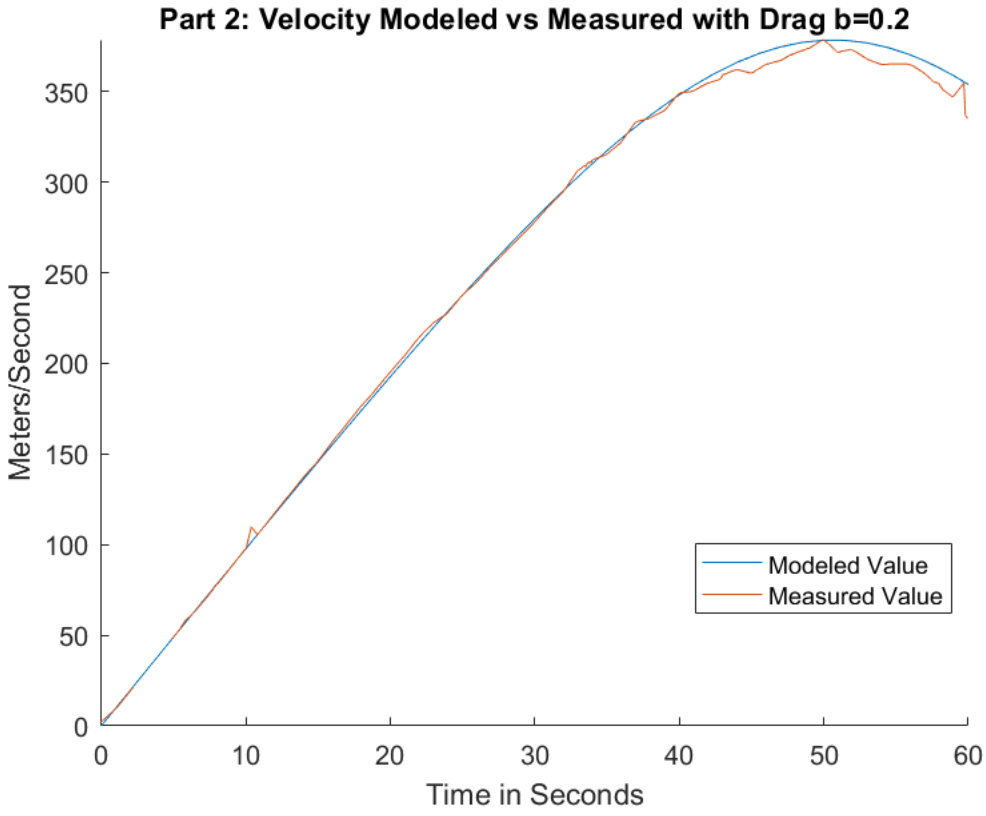

Part 1: Acceleration Modeled vs Measured
Meters/Second2
Time in Seconds
Modeled Value
Measured Value
20
0
-20
-40
-60
-80
-100
-120
-140
-160
0
10
20
30
40
50
60

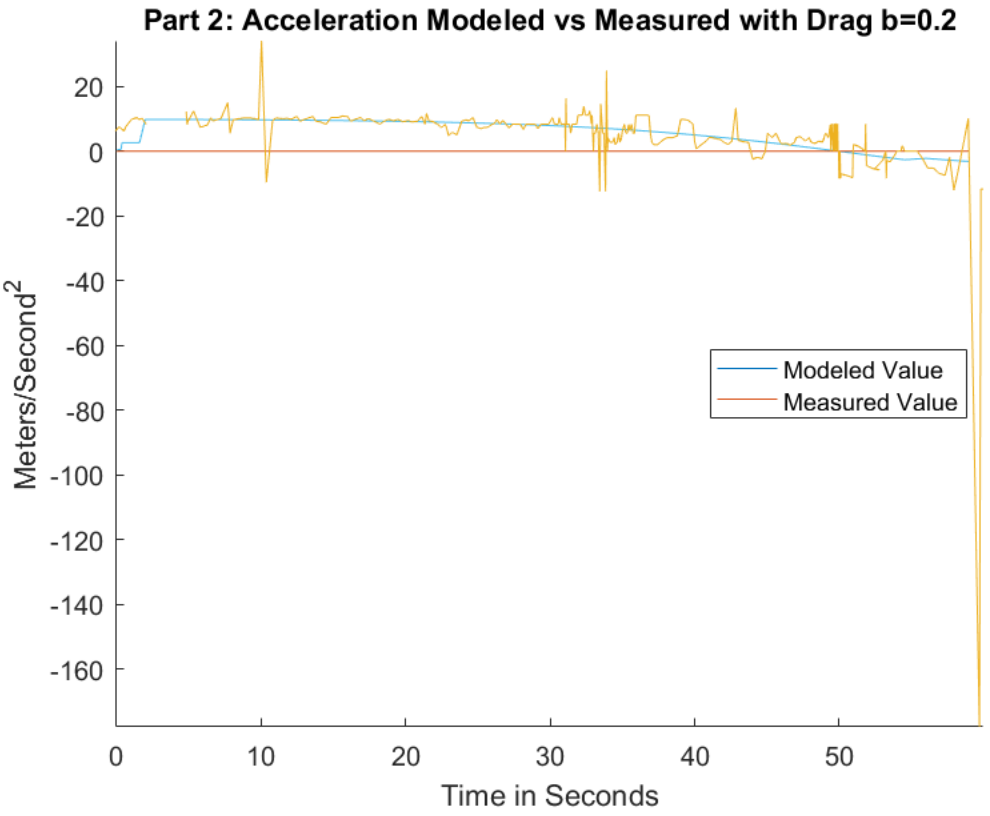
Part 2

Answer some questions here in these comments... Estimate your uncertainty in the mass that you have chosen (at the beginning of the jump).

```
    %I estimate the weight of Felix Baumgartner and the suit combined
 to
    %be 100 kg plus/minus 10 kg

% How sensitive is the velocity and altitude reached after 60 seconds
 to
% changes in the chosen mass?
    %The changes in mass did not have a noticeable effect on the
 velocity
    %and altitude when eyeballing results from the graph.

part = 2;
  [T, M]= ode45(@fall, [0, 60], [38969.4, 0]);

%calling plotcomparisons function for altitude, velocity, acceleration
plotComparisons(4,'Part 2: Altitude Modeled vs Measured with Drag
 b=0.2', 'Time in Seconds'...
    , 'Meters', T, M(:,1), time1min, altitude1min)
plotComparisons(5,'Part 2: Velocity Modeled vs Measured with Drag
 b=0.2', 'Time in Seconds'...
    , 'Meters/Second', T, abs(M(:,2)), time1min, airspeed1min)

%The following is a manual calculation of acceleration from the model
 for u = 1:(size(T)-1)
   TT(u) = T(u);
 end
plotComparisons(6,'Part 2: Acceleration Modeled vs Measured with Drag
 b=0.2', 'Time in Seconds'...
    , 'Meters/Second^2', TT, ((diff(abs(M(:,2))))/(diff(T))),
 time1min, accel1min)
```


Part 2: Altitude Modeled vs Measured with Drag b=0.2
×10^4
Meters
Time in Seconds
Modeled Value
Measured Value
3.8
3.6
3.4
3.2
3
2.8
2.6
0
10
20
30
40
50
60


Part 2: Velocity Modeled vs Measured with Drag b=0.2
Meters/Second
Time in Seconds
Modeled Value
Measured Value
350
300
250
200
150
100
50
0
0
10
20
30
40
50
60

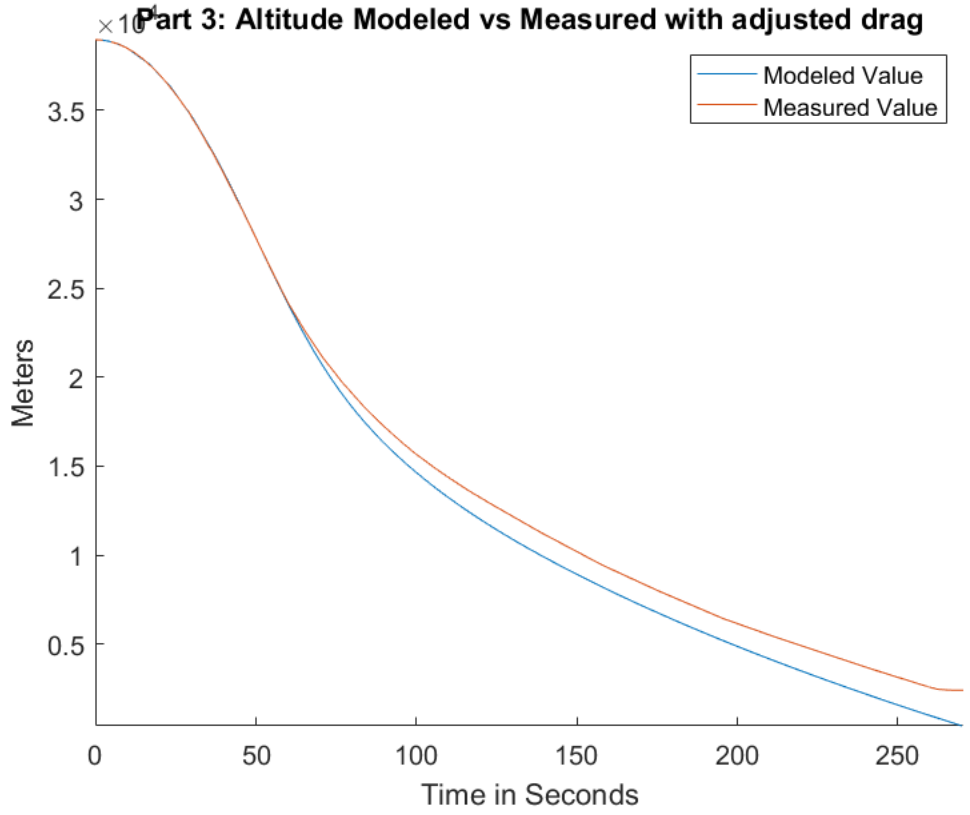


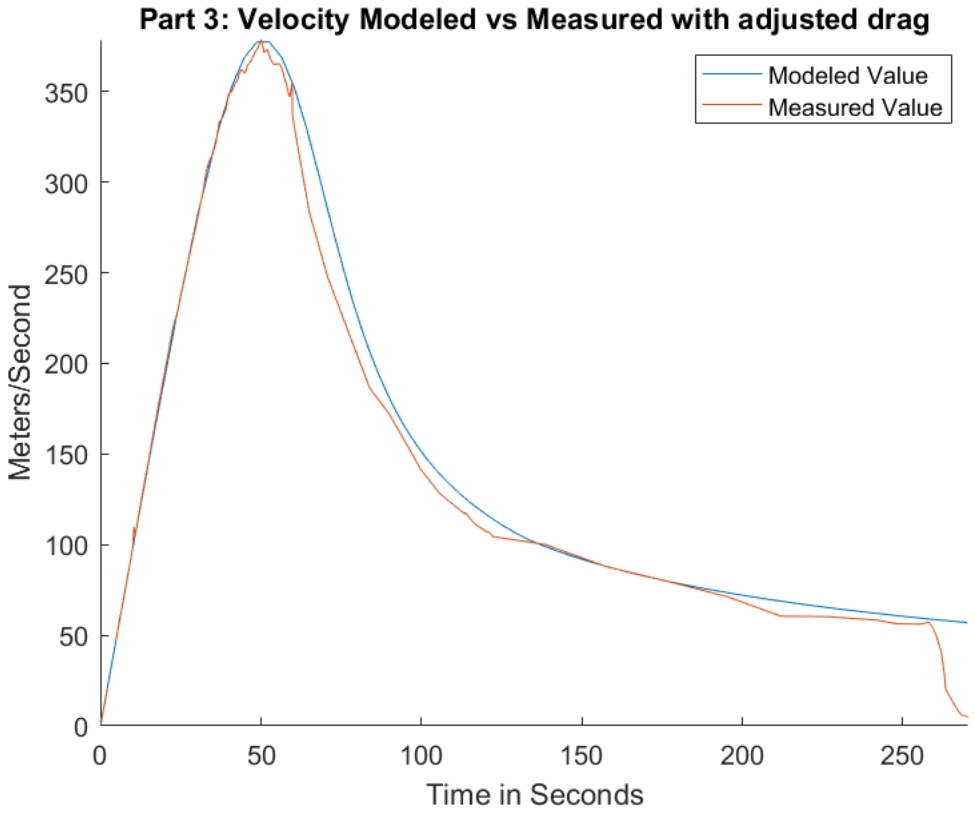
Part 3

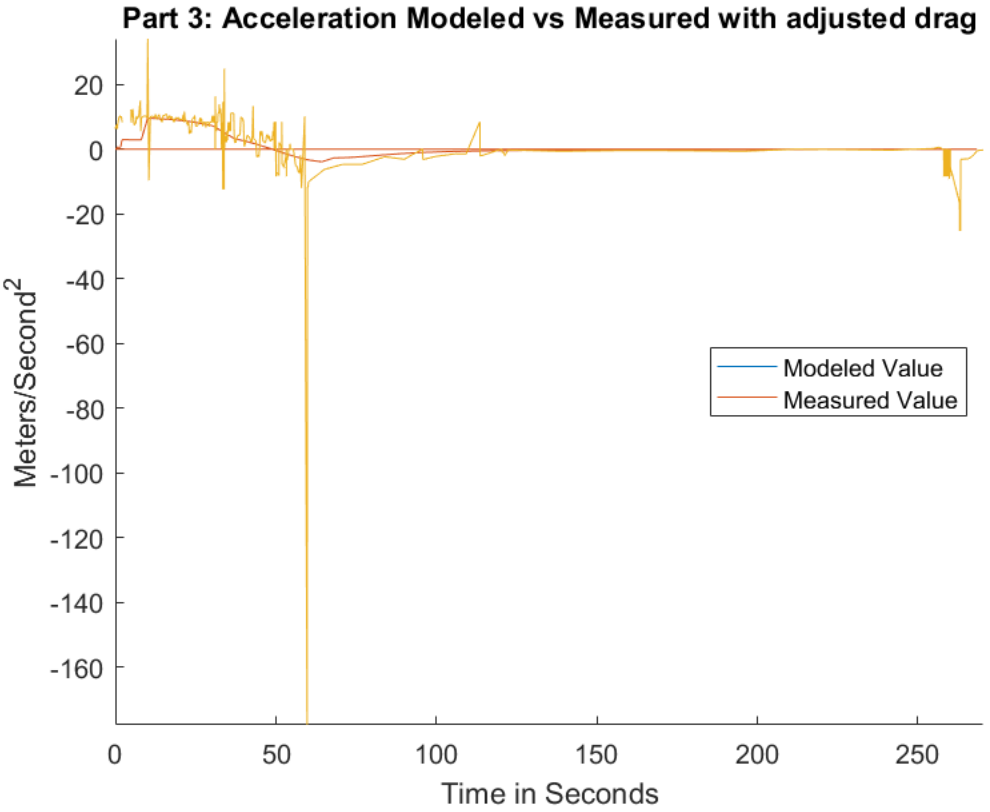
Answer some questions here in these comments... Felix was wearing a pressure suit and carrying oxygen. Why? What can we say about the density of air in the stratosphere? How is the density of air different at around 39,000 meters than it is on the ground?

```
    %Air becomes more dense the closer it is to sea-level because air
 is a fluid
    %that compresses under the weight of the air above it. At 40,000
 meters
    %the air pressure and levels of oxygen arent sustainable for a
 human. Therefore
    %Felix required a suit to maintain pressure and oxygen

% 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.
    %Since we are only dealing with 1 dimension, we can see that the
 drag force
    %is equal to 1/2*p*v^2*Cd*A where p is the density of the fuild
 (changes with
    %the altitude), v is velocity, Cd is drag coefficient(does not
 change), while
```

```
    %A is the cross sectional area (does not change). The troposphere
 does not
    %have a pressure and air density difference as high as the
 stratosphere when
    %compared with sea-level. Therefore calculations involving the
 troposphere
    %would be easier since one would not have to take into account
    %pressure/density changes.

% What method(s) can we employ to estimate [the ACd] product?
    %The cross sectional area (A)of Felix can be estimated to be 0.4m,
 taken
    %roughly from the dimensions of his suit. The Cd value can be
 estimated
    %to be 1.3 from http://www.engineeringtoolbox.com/drag-
coefficient-d_627.html
    %a normal human is around 1.0-1.3 and given Felix's suit we can
 round up to 1.3

% What is your estimated [ACd] product?
    %ACd = 0.4m x 1.3 = 0.52
%
% [Given what we are told in the textbook about the simple drag
 constant, b,]
% does the estimate for ACd seem reasonable?
    %Yes it seems about right given Felix's large suit, the drag
 equation is
    %also multiplied by 1/2 and so 0.52/2 is very close to b =0.2 from
 the textbook


part = 3;

[T, M]= ode45(@fall, [0, 270], [38969.4, 0]);

%calling plotcomparisons function for altitude, velocity, acceleration
plotComparisons(7,'Part 3: Altitude Modeled vs Measured with adjusted
 drag', 'Time in Seconds'...
    , 'Meters', T, M(:,1), time270sec, altitude270sec)
plotComparisons(8,'Part 3: Velocity Modeled vs Measured with adjusted
 drag', 'Time in Seconds'...
    , 'Meters/Second', T, abs(M(:,2)), time270sec, airspeed270sec)

%The following is a manual calculation of acceleration from the model
 for u = 1:(size(T)-1)
   TT(u) = T(u);
 end
plotComparisons(9,'Part 3: Acceleration Modeled vs Measured with
 adjusted drag', 'Time in Seconds'...
    , 'Meters/Second^2', TT, ((diff(abs(M(:,2))))/(diff(T))),
 time270sec, accel270sec)
```


Part 3: Altitude Modeled vs Measured with adjusted drag
×10^4
Modeled Value
Measured Value
Meters
3.5
3
2.5
2
1.5
1
0.5
0
50
100
150
200
250
Time in Seconds


Part 3: Velocity Modeled vs Measured with adjusted drag
Modeled Value
Measured Value
Meters/Second
350
300
250
200
150
100
50
0
0
50
100
150
200
250
Time in Seconds



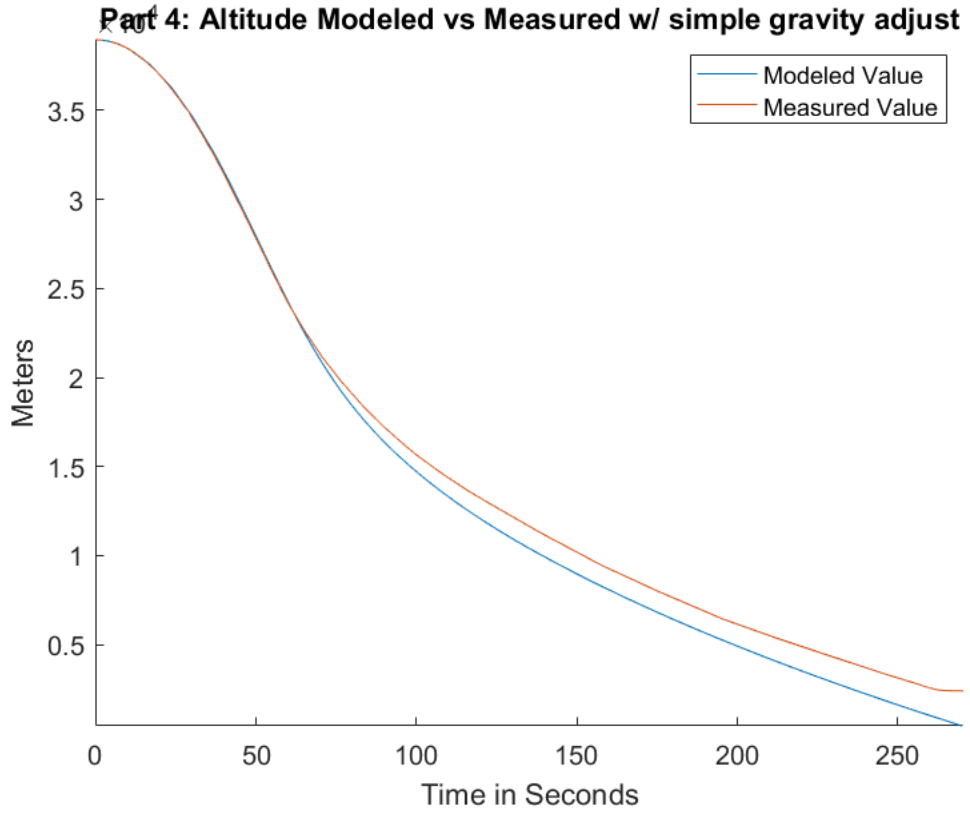
Part 4

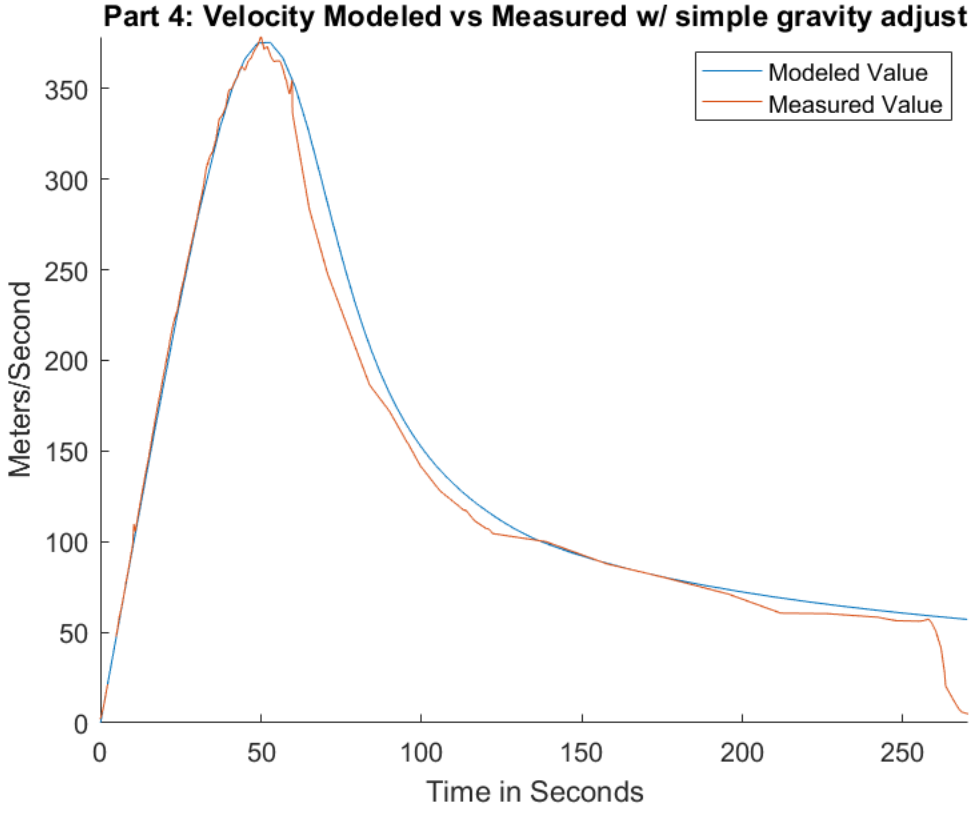
Answer some questions here in these comments... What is the actual gravitational field strength around 39,000 meters? (See Tipler Volume 1 6e page 369.)

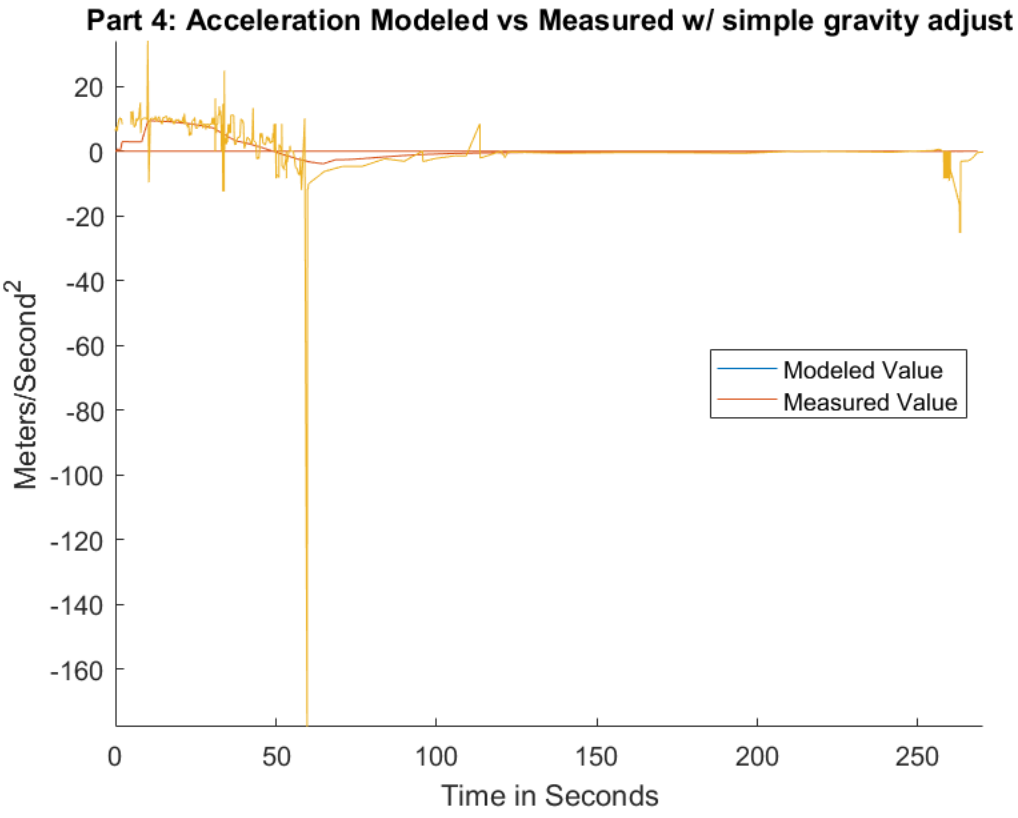
```
    %The gravitational field strength is about 9.68N/kg at 39,000m

% How sensitive is the altitude reached after 4.5 minutes to simpler
 and
% more complicated ways of modelling the gravitational field strength?
    %I have chosen two methods to adjust the gravitional fiels
 strength.
    %The first simple method was to linearly adjust the field strength
    %from 9.68 to 9.81 as Felix reached earth. The second more
 advanced
    %method was to use Netwon's law of gravitation g=GM/r^2 to find
 the
    %field strength. I have calculated the final altitudes for the
 simple
    %and advanced method (which should show up in the command window
 after
    %running the assignment2 script) and found that the simple method
 gave
    %a final altitude of 436.1398 compared to 4.28.4161 from the
 advanced
```

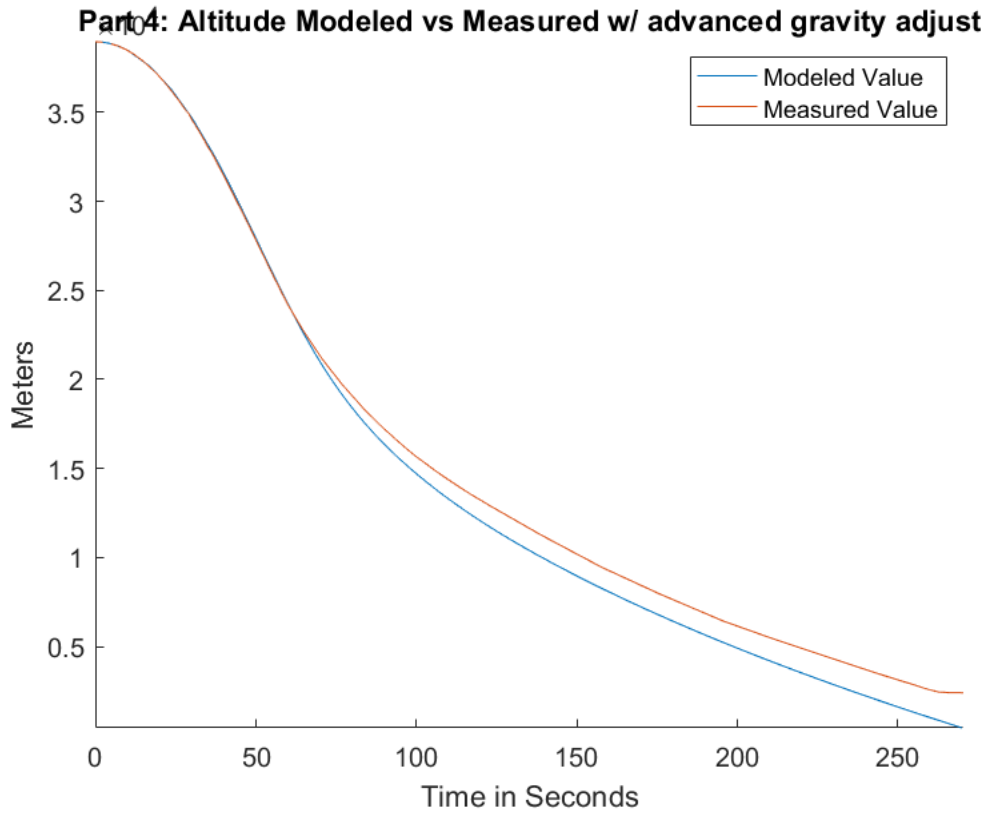
```
    %method. A change in 8m is very very small given the initial
 conditions

% 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.
    %one of the reasons that the final altitude value is so low is
 because
    %the parachute hasn't been factored in yet. Another possible
 change one
    %could make is the factor of wind or water which would be a
 varying
    %force acting against gravity, however this would be extremely
    %difficult to calculate.

% 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?
    %Adjusting for changes in gravitational field strength did not
 have a
    %"significant" change in fnial altitude after 4.5 minutes. Small
    %changes in mass also do have very significant effects. A
 seriously
    %useless change would be factoring in Felix's Lorenz contraction
 and
    %time dialation from his speed and factoring that into his drag
 and
    %airspeed.

% How can we decide what change is significant and what change is
%   insignificant?
    %One could model taking into account a change and compare it to a
    %measured value or another model without the change. Or one can
    %conceptually think about it using physics equations. Obviously
 changes
    %in air density will provide significant drag to counter the force
 of
    %gravity given f=ma but Lorenz contractions or time dialation are
 far
    %too negligible to have any effect

% [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.)]
    %I tried to adjust the gravitational field strength in two ways as
    %mentioned above. The first method was a linear change that ranges
 the
    %gravitational field strength from 9.68 to 9.81 depending on
 altitude.
    %The second method was to calculate the graviational field
 strength at
    %each individual point using Newton's law of Gravity g=GM/r^2

part = 4;
mode_grav = 1; %the simple linear calculation for gravity
```

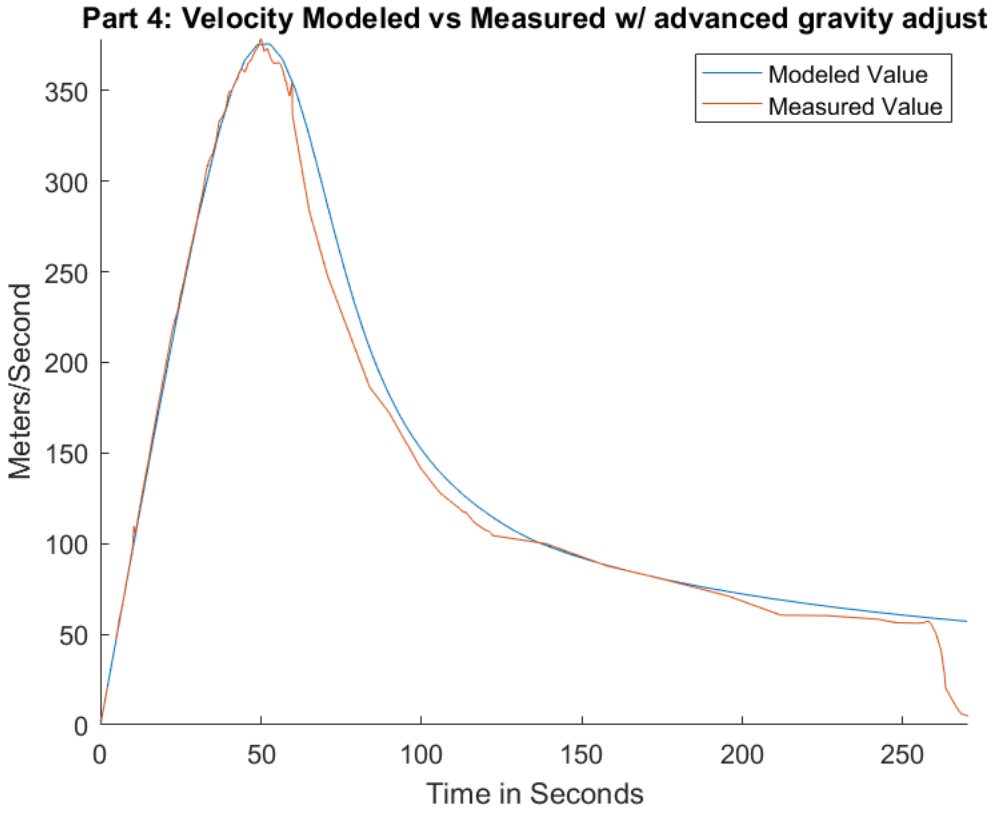
```
[T, M]= ode45(@fall, [0, 270], [38969.4, 0]);
plotComparisons(10,'Part 4: Altitude Modeled vs Measured w/ simple
 gravity adjust', 'Time in Seconds'...
    , 'Meters', T, M(:,1), time270sec, altitude270sec)
plotComparisons(11,'Part 4: Velocity Modeled vs Measured w/ simple
 gravity adjust', 'Time in Seconds'...
    , 'Meters/Second', T, abs(M(:,2)), time270sec, airspeed270sec)

%The following is a manual calculation of acceleration from the model
 for u = 1:(size(T)-1)
   TT(u) = T(u);
 end
plotComparisons(12,'Part 4: Acceleration Modeled vs Measured w/ simple
 gravity adjust', 'Time in Seconds'...
    , 'Meters/Second^2', TT, ((diff(abs(M(:,2))))/(diff(T))),
 time270sec, accel270sec)
finalAltitudeWithSimpleGravity = M(end, 1)



mode_grav = 2; %the more complicated calculation for gravity (GM/r^2)
[T, M]= ode45(@fall, [0, 270], [38969.4, 0]);
plotComparisons(13,'Part 4: Altitude Modeled vs Measured w/ advanced
 gravity adjust', 'Time in Seconds'...
    , 'Meters', T, M(:,1), time270sec, altitude270sec)
plotComparisons(14,'Part 4: Velocity Modeled vs Measured w/ advanced
 gravity adjust', 'Time in Seconds'...
    , 'Meters/Second', T, abs(M(:,2)), time270sec, airspeed270sec)

%The following is a manual calculation of acceleration from the model
 for u = 1:(size(T)-1)
   TT(u) = T(u);
 end
plotComparisons(15,'Part 4: Acceleration Modeled vs Measured w/
 advanced gravity adjust', 'Time in Seconds'...
    , 'Meters/Second^2', TT, ((diff(abs(M(:,2))))/(diff(T))),
 time270sec, accel270sec)
finalAltitudeWithAdvancedGravity = M(end, 1)


finalAltitudeWithSimpleGravity =

  436.1398


finalAltitudeWithAdvancedGravity =

  428.4161
```

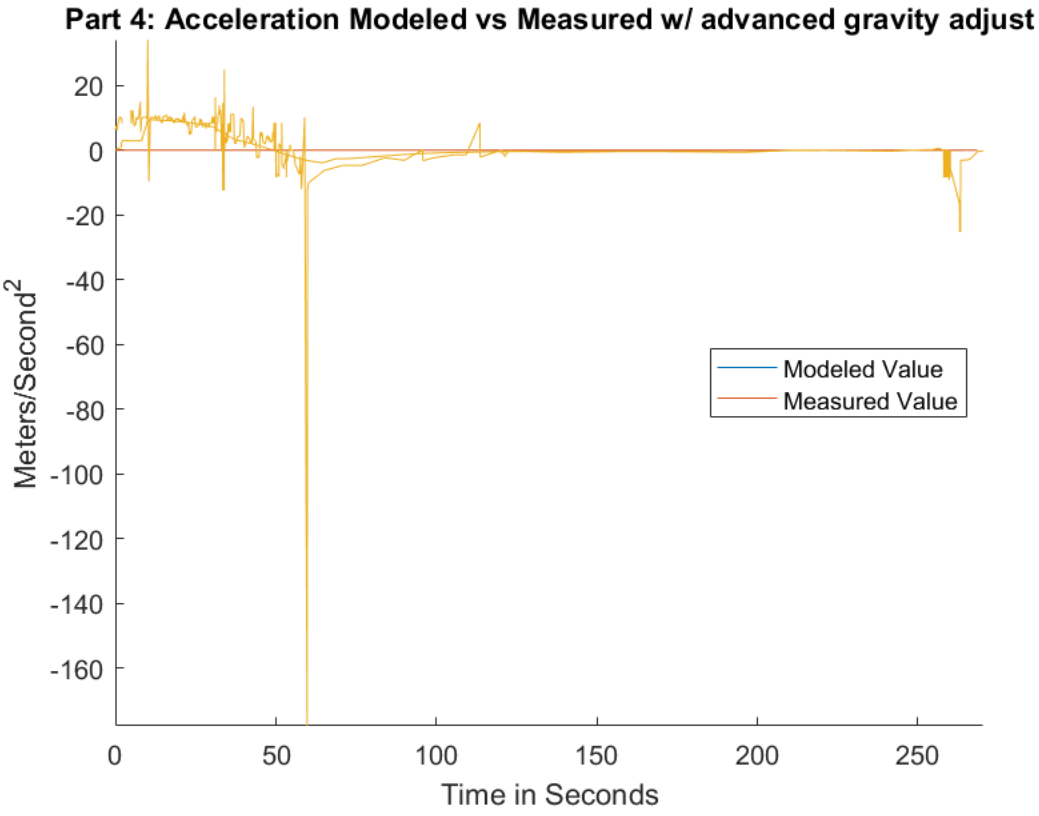

Part 4: Altitude Modeled vs Measured w/ simple gravity adjust
Modeled Value
Measured Value
Meters
Time in Seconds
0
50
100
150
200
250
0.5
1
1.5
2
2.5
3
3.5


Part 4: Velocity Modeled vs Measured w/ simple gravity adjust
Modeled Value
Measured Value
Meters/Second
Time in Seconds
0
50
100
150
200
250
300
350


Part 4: Acceleration Modeled vs Measured w/ simple gravity adjust
Meters/Second^2
Time in Seconds
Modeled Value
Measured Value
20
0
-20
-40
-60
-80
-100
-120
-140
-160
0
50
100
150
200
250


Part 4: Altitude Modeled vs Measured w/ advanced gravity adjust
x 10^4
Meters
Time in Seconds
Modeled Value
Measured Value
3.5
3
2.5
2
1.5
1
0.5
0
50
100
150
200
250


Part 4: Velocity Modeled vs Measured w/ advanced gravity adjust
Modeled Value
Measured Value
Meters/Second
Time in Seconds
0
50
100
150
200
250
300
350


Part 4: Acceleration Modeled vs Measured w/ advanced gravity adjust
Modeled Value
Measured Value
Meters/Second2
Time in Seconds
20
0
-20
-40
-60
-80
-100
-120
-140
-160

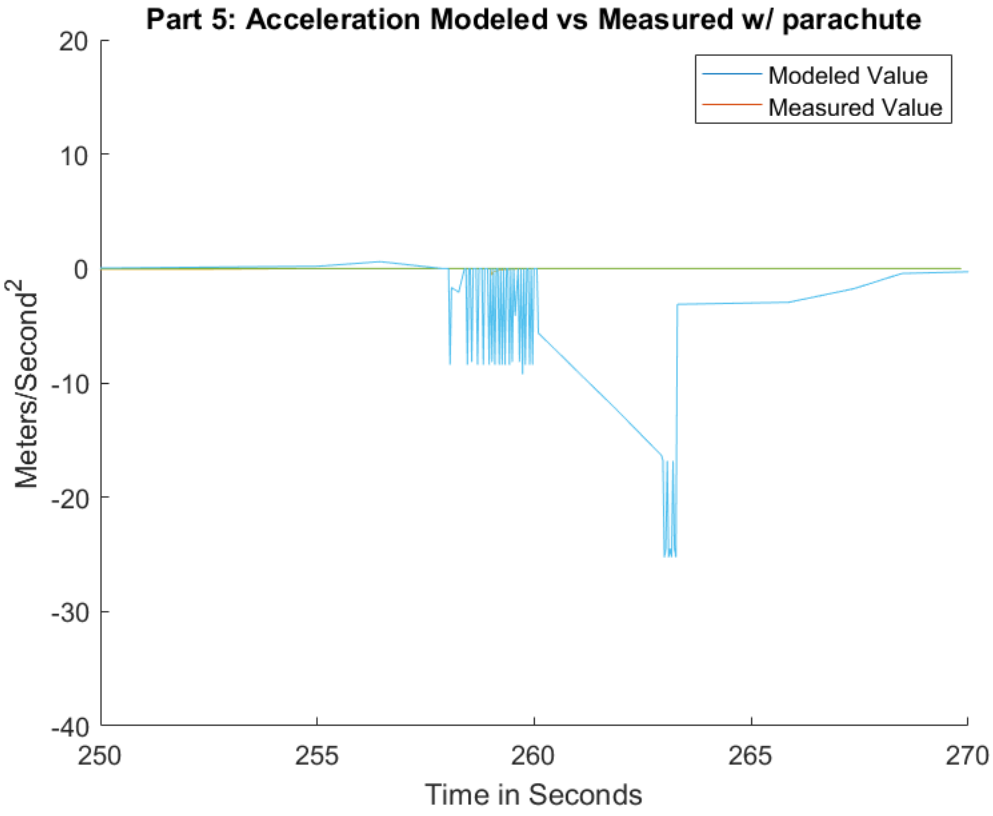
Part 5

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

```
    %at 4min 18 seconds or 258 second mark

% 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?
    %As found from google, I choose my new A value to be 33m^2 and the
 Cd
    %value to be 1.75. The ACd product is now 57.75. However, for
 reasons I
    %cannot understand, the modeled change in acceleration is
 extremely
    %small which does not seem right. However, looking at the measured
    %value I can see that the maximum magnitude of acceleration was
 about
    %-27 m/s^2

%   How safe or unsafe would such an acceleration be for Felix?
    %-27 m/s^2 would be less than 3 Gs which isn't actually that bad
 and is very
    %survivable

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.
[T, M]= ode45(@fall, [0, 270], [38969.4, 0]);
%The following is a manual calculation of acceleration from the model
 for u = 1:(size(T)-1)
   TT(u) = T(u);
 end
 %this plot had to be done manually because it required special axis
 %options that I couldn't add to the usual plot function
 figure(16)
        hold on
        title('Part 5: Acceleration Modeled vs Measured w/ parachute')
        xlabel('Time in Seconds')
        ylabel('Meters/Second^2')
        plot(TT, ((diff(abs(M(:,2))))/(diff(T))))
        plot( time270sec, accel270sec)
        axis([250,270,-40,20]);
        legend('Modeled Value', 'Measured Value', 'Location', 'best')
```

```
        hold off
```

```
Warning: One or more altitudes above upper limit.
Warning: One or more altitudes above upper limit.
```



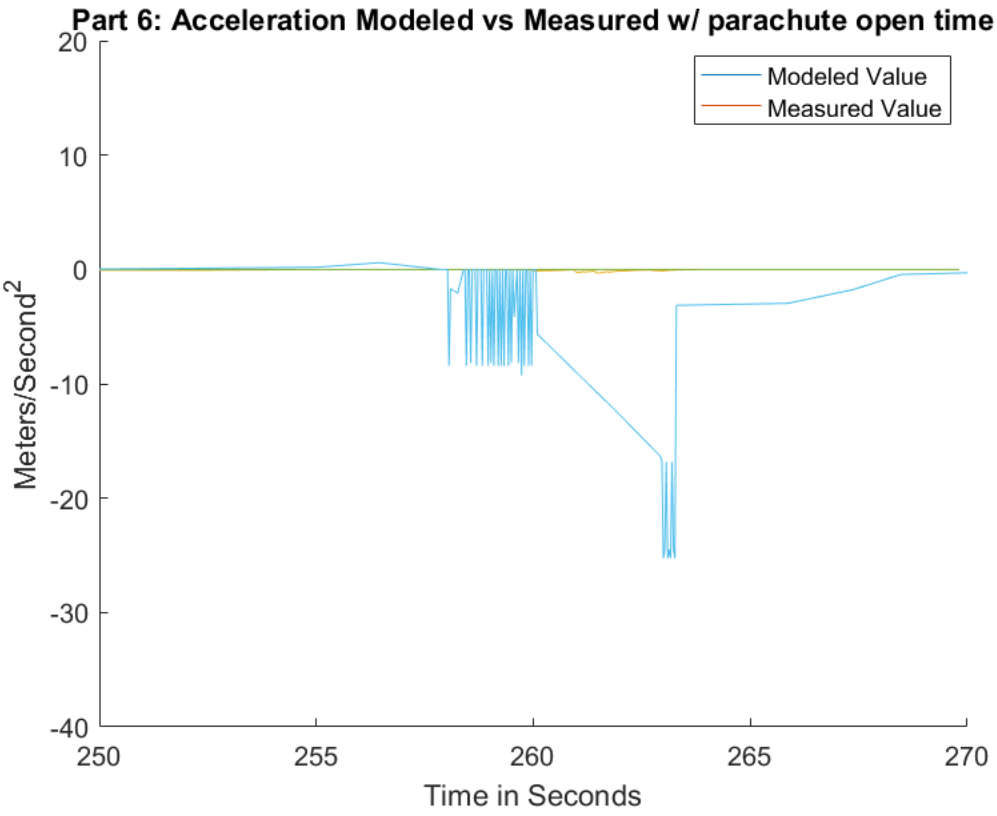
Part 6

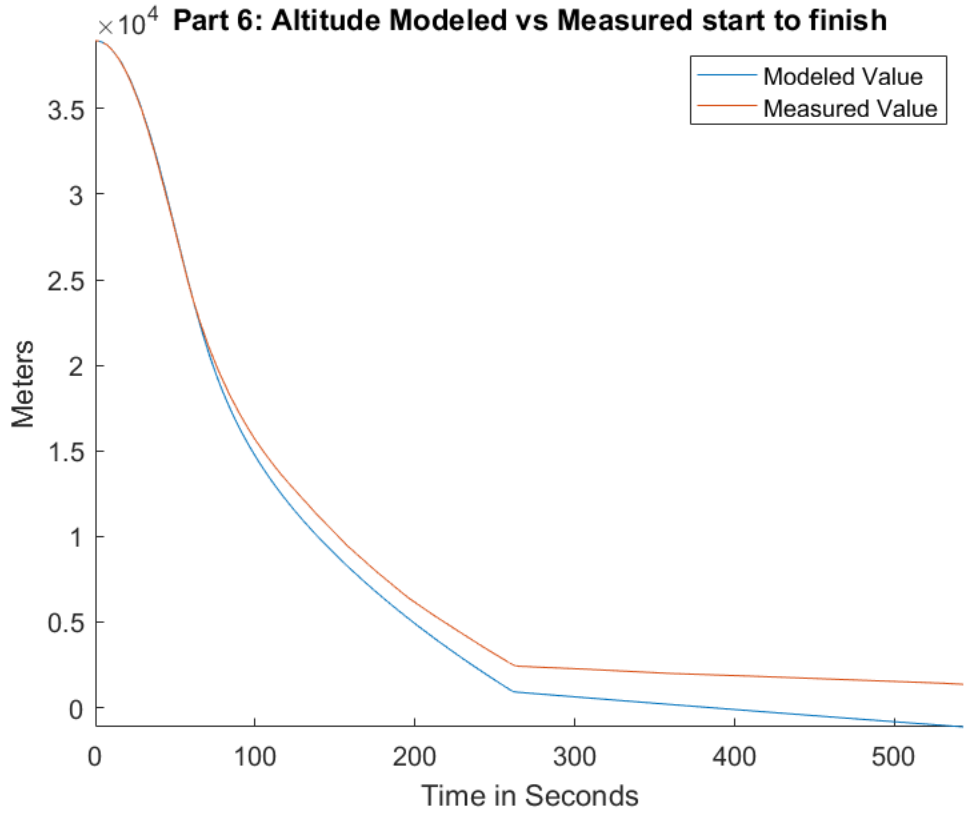
Answer some questions here in these comments... How long does it take for Felix's parachute to open?

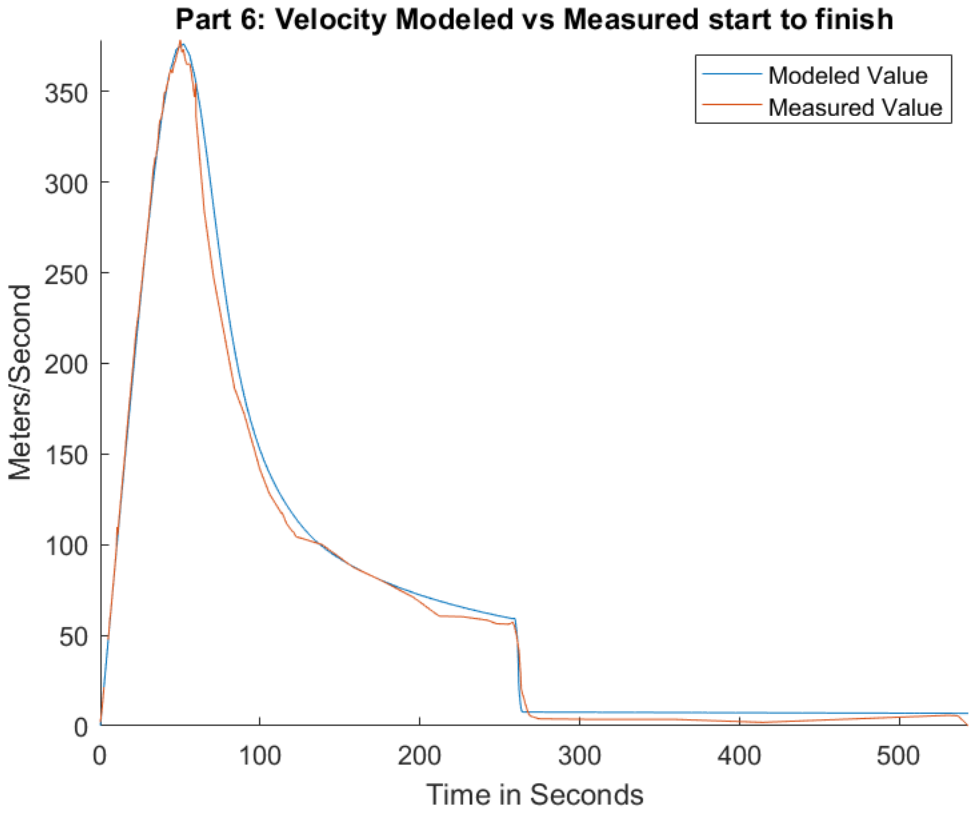
```
    %From looking at the video of his jump and the data I would have
 to
    %estimate the time to be around 4-5 seconds
part = 6;

%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.

%Although the acceleration due to the parachute opening was unusually
 low
%for my part 5 answer, I will still try to spread out the acceleration
 over
%a period of 4-5 seconds for the sake of this assignment. You can see
 this
```

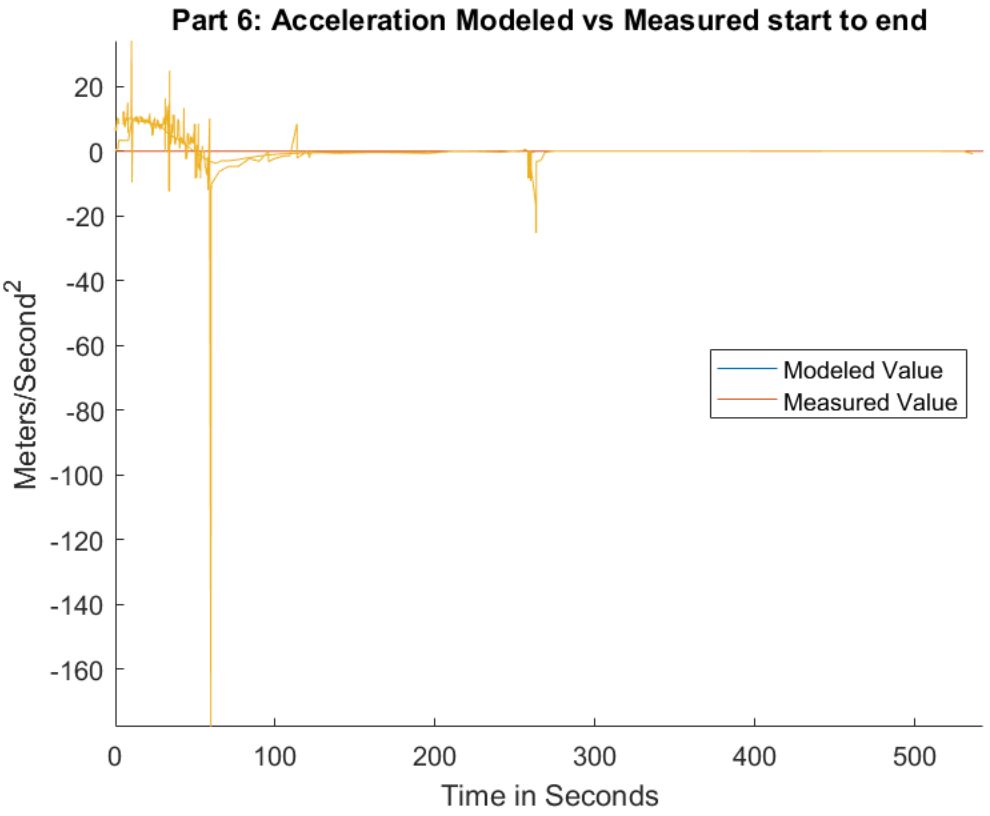
```
%work in the drag function. What's strange is that despite my part 5
 answer
%not showing a large jump in acceleration, my part 6 answer seems to
 look
%just fine.

%now to plot with parachute opening
[T, M]= ode45(@fall, [0, 270], [38969.4, 0]);
%The following is a manual calculation of acceleration from the model
 TT=T;
TT(end)=[];
 %this plot had to be done manually because it required special axis
 %options that I couldn't add to the usual plot function
 size(T)
 size(TT)
 size(M)
 figure(17)
        hold on
        title('Part 6: Acceleration Modeled vs Measured w/ parachute
 open time')
        xlabel('Time in Seconds')
        ylabel('Meters/Second^2')
        plot(TT, ((diff(abs(M(:,2))))/(diff(T))))
        plot( time270sec, accel270sec)
        axis([250,270,-40,20]);
        legend('Modeled Value', 'Measured Value', 'Location', 'best')
        hold off


%Finally, to plot the entire jump from start to finish

[T, M]= ode45(@fall, [0, 543], [38969.4, 0]);
plotComparisons(18,'Part 6: Altitude Modeled vs Measured start to
 finish', 'Time in Seconds'...
    , 'Meters', T, M(:,1), timetotal, altitudetotal)
plotComparisons(19,'Part 6: Velocity Modeled vs Measured start to
 finish', 'Time in Seconds'...
    , 'Meters/Second', T, abs(M(:,2)), timetotal, airspeedtotal)

%The following is a manual calculation of acceleration from the model
 for u = 1:(size(T)-1)
   TT(u) = T(u);
   end
 timetotalpart6accel = timetotal;
 timetotalpart6accel(end) = [];
plotComparisons(20,'Part 6: Acceleration Modeled vs Measured start to
 end', 'Time in Seconds'...
    , 'Meters/Second^2', TT, ((diff(abs(M(:,2))))/(diff(T))),
 timetotalpart6accel, acceltotal)


ans =

   237     1
```

```
ans =

   236     1

ans =

   237     2
```




Part 6: Altitude Modeled vs Measured start to finish
×10^4
Modeled Value
Measured Value
Meters
Time in Seconds
0
0.5
1
1.5
2
2.5
3
3.5
0
100
200
300
400
500


Part 6: Velocity Modeled vs Measured start to finish
Modeled Value
Measured Value
Meters/Second
Time in Seconds
0
50
100
150
200
250
300
350
0
100
200
300
400
500



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 <This function is used with ode45 to calculate altitude
    % and acceleration while Felix falls>

    % 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 is used will fall and ode45 to calculate Felix's
    % 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 = 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

    if part < 4
        a_grav = A_GRAV_SEA;
    else
        if mode_grav == 1
            a_grav = 9.807 - (p*(9.807-9.68)/38969);
            %a simple linear method to slowly reduce the gravitational
            %field strength as the altitude decreases
        end
        if mode_grav == 2
             a_grav = ((6.67*10^(-11))*(5.972*10^24))/((6371000+p)^2);
             %this method uses the gravity equation g=GM/r^2 where G
 is
             %the gravity constant, M is the mass of earth, and r is
 the
             %radius
        end
    end

end

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

```
    res = 100;
end

function res = drag(t, p, v, m)
% <This function calculates the drag force on Felix as he falls>
    airdata = stdatmo(p);
    density = airdata(1);
    if part == 2
        res = 0.2;
    end
    if part == 5
        A=0.4;
        Cd=1.3;
        if t > 259
        A=33;
        Cd=1.75;
        end
        res=  0.5*Cd*stdatmo(p)*A;
    else
        res = (1/2)*0.52*density;
    end
    if part == 6
         if t <= 260
        A= 0.4;
        Cd=1.3;
        end
        a= [4,8,12,16,20,25];
        CD= [0.3,0.6,0.9,1.2,1.5,1.75];
        if t > 260
        A=a(1);
        Cd=CD(1);
        end
        if t > 260.5
        A=a(1);
        Cd=CD(1);
        end
        if t > 261
        A=a(2);
        Cd=CD(2);
        end
        if t > 261.5
        A=a(3);
        Cd=CD(3);
        end
        if t > 262
        A=a(3);
        Cd=CD(3);
        end
        if t > 262.5
        A=a(4);
        Cd=CD(4);
        end
        if t > 263
        A=a(5);
```

```
        Cd=CD(5);
        end
        if t > 263.5
        A=a(5);
        Cd=CD(5);
        end
        res=  0.5*Cd*stdatmo(p)*A ;
    end
end
```

Additional nested functions

Nest any other functions below.

```
%Do not put functions in other files when you submit.
    function res = plotComparisons(fignumber, graphtitle, x_label,
 y_label, T, M,...
            modelx, modely)
        figure(fignumber)
        hold on
        title(graphtitle)
        xlabel(x_label)
        ylabel(y_label)
        plot(T, M)
        plot(modelx, modely)
        axis tight;
        legend('Modeled Value', 'Measured Value', 'Location', 'best')
        hold off
    end
function
 [rho,a,temp,press,kvisc,ZorH]=stdatmo(H_in,Toffset,Units,GeomFlag)
%  STDATMO Find gas properties in earth's atmosphere.
%   [rho,a,T,P,nu,ZorH] = STDATMO(H,dT,Units,GeomFlag)
%
%   STDATMO by itself gives the atmospheric properties at sea level on
 a
%   standard day.
%
%   STDATMO(H) returns the properties of the 1976 Standard Atmosphere
 at
%   geopotential altitude H (meters), where H is a scalar, vector,
 matrix,
%   or ND array.
%
%   STDATMO(H,dT) returns properties when the temperature is dT
 degrees
%   offset from standand conditions. H and dT must be the same size or
 else
%   one must be a scalar.
%
%   STDATMO(H,dT,Units) specifies units for the inputs outputs.
 Options are
```

```
%   SI (default) or US (a.k.a. Imperial, English). For SI, set Units
 to []
%   or 'SI'. For US, set Units to 'US'. Input and output units may be
%   different by passing a cell array of the form {Units_in
 Units_out},
%   e.g. {'US' 'SI'}. Keep in mind that dT is an offset, so when
 converting
%   between Celsius and Fahrenheit, use only the scaling factor (dC/dF
 =
%   dK/dR = 5/9). Units are as follows:
%       Input:                      SI (default)    US
%           H:     Altitude         m               ft
%           dT:    Temp. offset     °C/°K           °F/°R
%       Output:
%           rho:   Density          kg/m^3          slug/ft^3
%           a:     Speed of sound   m/s             ft/s
%           T:     Temperature      °K              °R
%           P:     Pressure         Pa              lbf/ft^2
%           nu:    Kinem. viscosity m^2/s           ft^2/s
%           ZorH:  Height or altitude m             ft
%
%   STDATMO(H,dT,u), where u is a structure created by the UNITS
 function,
%   accepts variables of the DimVar (Dimensioned Variable) class as
 inputs.
%   Outputs are of the DimVar class. If a DimVar is not provided for
 an
%   input, STDATMO assumes SI input.
%
%   STDATMO(H,dT,Units,GeomFlag) with logical input GeomFlag returns
%   properties at geometric altitude input H instead of the normal
%   geopotential altitude.
%
%   [rho,a,T,P,nu] = STDATMO(H,dT,...) returns atmospheric properties
 the
%   same size as H and/or dT (P does not vary with temperature offset
 and
%   is always the size of H)
%
%   [rho,a,T,P,nu,ZorH] = STDATMO(H,...) returns either geometric
 height,
%   Z, (GeomFlag not set) or geopotential height, H, (GeomFlag set).
%
%   Example 1: Find atmospheric properties at every 100 m of geometric
%   height for an off-standard atmosphere with temperature offset
 varying
%   +/- 25°C sinusoidally with a period of 4 km.
%       Z = 0:100:86000;
%       [rho,a,T,P,nu,H] = stdatmo(Z,25*sin(pi*Z/2000),'',true);
%       semilogx(rho/stdatmo,H/1000)
%       title('Density variation with sinusoidal off-standard
 atmosphere')
%       xlabel('\sigma'); ylabel('Altitude (km)')
%
```

```
%   Example 2: Create tables of atmospheric properties up to 30000 ft
 for a
%   cold (-15°C), standard, and hot (+15°C) day with columns
%   [h(ft) Z(ft) rho(slug/ft³) sigma a(ft/s) T(R) P(psf) μ(slug/ft-s)
 nu(ft²/s)]
%   using 3-dimensional array inputs.
%       [~,h,dT] = meshgrid(0,-5000:1000:30000,-15:15:15);
%       [rho,a,T,P,nu,Z] = stdatmo(h,dT*9/5,'US',0);
%       Table = [h Z rho rho/stdatmo(0,0,'US') T P nu.*rho nu];
%       format short e
%       ColdTable       = Table(:,:,1)
%       StandardTable   = Table(:,:,2)
%       HotTable        = Table(:,:,3)
%
%   Example 3: Use the unit consistency enforced by the DimVar class
 to
%   find the SI dynamic pressure, Mach number, Reynolds number, and
%   stagnation temperature of an aircraft flying at flight level FL500
%   (50000 ft) with speed 500 knots and characteristic length of 80
 inches.
%       u = units;
%       V = 500*u.kts; c = 80*u.in;
%       [rho,a,T,P,nu] = stdatmo(50*u.kft,[],u);
%       Dyn_Press = 1/2*rho*V^2;
%       M = V/a;
%       Re = V*c/nu;
%       T0 = T*(1+(1.4-1)/2*M^2);
%
%   This atmospheric model is not recommended for use at altitudes
 above
%   86 km geometric height (84852 m/278386 ft geopotential) and
 returns NaN
%   for altitudes above 90 km geopotential.
%
%   See also ATMOSISA, ATMOSNONSTD, ATMOS, TROPOS,
%     DENSITYALT - http://www.mathworks.com/matlabcentral/
fileexchange/39325,
%     UNITS      - http://www.mathworks.com/matlabcentral/
fileexchange/38977.
%
%   [rho,a,T,P,nu,ZorH] = STDATMO(H,dT,Units,GeomFlag)

%   Copyright 2010-2014 Sky Sartorius
%   www.mathworks.com/matlabcentral/fileexchange/authors/101715
%
%   References: ESDU 77022;  www.pdas.com/atmos.html


if nargin == 0
    H_in = 0;
end

if nargin < 2 || isempty(Toffset)
    Toffset = 0;
```

```
end

% global u
U = false;
if nargin >= 3 && isstruct(Units)

%     u = Units;
end
if isa(H_in,'DimVar')
    U = true;
    H_in = H_in/u.m;
    Units = 'si';
end
if isa(Toffset,'DimVar')
    Toffset = Toffset/u.K;
end


% else
%
% end

% if nargin <= 2 && all(H_in(:) <= 11000) %quick troposphere-only code
%     TonTi=1-2.255769564462953e-005*H_in;
%     press=101325*TonTi.^(5.255879812716677);
%     temp = TonTi*288.15 + Toffset;
%     rho = press./temp/287.05287;
%
%     if nargout > 1
%         a = sqrt(401.874018 * temp);
%         if nargout >= 5
%             kvisc = (1.458e-6 * temp.^1.5 ./ (temp + 110.4)) ./ rho;
%             if nargout == 6 % Assume Geop in, find Z
%                 ZorH = 6356766*H_in./(6356766-H_in);
%             end
%         end
%     end
%     return
% end

% index   Lapse rate   Base Temp     Base Geopo Alt        Base
 Pressure
%   i      Ki (°C/m)    Ti (°K)         Hi (m)              P (Pa)
D =[1       -.0065      288.15          0                   101325
    2       0           216.65          11000
 22632.0400950078
    3       .001        216.65          20000
 5474.87742428105
    4       .0028       228.65          32000
 868.015776620216
    5       0           270.65          47000
 110.90577336731
    6       -.0028      270.65          51000
 66.9385281211797
```

```
    7       -.002       214.65          71000
 3.9563921603966
    8       0           186.94590831019 84852.0458449057
 0.373377173762337  ];

% Constants
R=287.05287; %N-m/kg-K; value from ESDU 77022
% R=287.0531;   %N-m/kg-K; value used by MATLAB aerospace toolbox
 ATMOSISA
gamma=1.4;
g0=9.80665;     %m/sec^2
RE=6356766;     %Radius of the Earth, m
Bs = 1.458e-6;  %N-s/m2 K1/2
S = 110.4;      %K

K=D(:,2); %°K/m
T=D(:,3); %°K
H=D(:,4); %m
P=D(:,5); %Pa

temp=zeros(size(H_in));
press=temp;
hmax = 90000;

if nargin < 3 || isempty(Units)
    Uin = false;
    Uout = Uin;
elseif isnumeric(Units) || islogical(Units)
    Uin = Units;
    Uout = Uin;
else
    if ischar(Units) %input and output units the same
        Unitsin = Units; Unitsout = Unitsin;
    elseif iscell(Units) && length(Units) == 2
        Unitsin = Units{1}; Unitsout = Units{2};
    elseif iscell(Units) && length(Units) == 1
        Unitsin = Units{1}; Unitsout = Unitsin;
    else
        error('Incorrect Units definition. Units must be ''SI'',
 ''US'', or 2-element cell array')
    end

    if strcmpi(Unitsin,'si')
        Uin = false;
    elseif strcmpi(Unitsin,'us')
        Uin = true;
    else error('Units must be ''SI'' or ''US''')
    end

    if strcmpi(Unitsout,'si')
        Uout = false;
    elseif strcmpi(Unitsout,'us')
        Uout = true;
    else error('Units must be ''SI'' or ''US''')
```

```
    end
end

% Convert from imperial units, if necessary.
if Uin
    H_in = H_in * 0.3048;
    Toffset = Toffset * 5/9;
end



% Convert from geometric altitude to geopotental altitude, if
 necessary.
if nargin < 4
    GeomFlag = false;
end
if GeomFlag
    Hgeop=(RE*H_in)./(RE+H_in);
else
    Hgeop=H_in;
end

n1=(Hgeop<=H(2));
n2=(Hgeop<=H(3) & Hgeop>H(2));
n3=(Hgeop<=H(4) & Hgeop>H(3));
n4=(Hgeop<=H(5) & Hgeop>H(4));
n5=(Hgeop<=H(6) & Hgeop>H(5));
n6=(Hgeop<=H(7) & Hgeop>H(6));
n7=(Hgeop<=H(8) & Hgeop>H(7));
n8=(Hgeop<=hmax & Hgeop>H(8));
n9=(Hgeop>hmax);

% Troposphere
if any(n1(:))
    i=1;
    TonTi=1+K(i)*(Hgeop(n1)-H(i))/T(i);
    temp(n1)=TonTi*T(i);
    PonPi=TonTi.^(-g0/(K(i)*R));
    press(n1)=P(i)*PonPi;
end

% Tropopause
if any(n2(:))
    i=2;
    temp(n2)=T(i);
    PonPi=exp(-g0*(Hgeop(n2)-H(i))/(T(i)*R));
    press(n2)=P(i)*PonPi;
end

% Stratosphere 1
if any(n3(:))
    i=3;
    TonTi=1+K(i)*(Hgeop(n3)-H(i))/T(i);
    temp(n3)=TonTi*T(i);
```

```
    PonPi=TonTi.^(-g0/(K(i)*R));
    press(n3)=P(i)*PonPi;
end

% Stratosphere 2
if any(n4(:))
    i=4;
    TonTi=1+K(i)*(Hgeop(n4)-H(i))/T(i);
    temp(n4)=TonTi*T(i);
    PonPi=TonTi.^(-g0/(K(i)*R));
    press(n4)=P(i)*PonPi;
end

% Stratopause
if any(n5(:))
    i=5;
    temp(n5)=T(i);
    PonPi=exp(-g0*(Hgeop(n5)-H(i))/(T(i)*R));
    press(n5)=P(i)*PonPi;
end

% Mesosphere 1
if any(n6(:))
    i=6;
    TonTi=1+K(i)*(Hgeop(n6)-H(i))/T(i);
    temp(n6)=TonTi*T(i);
    PonPi=TonTi.^(-g0/(K(i)*R));
    press(n6)=P(i)*PonPi;
end

% Mesosphere 2
if any(n7(:))
    i=7;
    TonTi=1+K(i)*(Hgeop(n7)-H(i))/T(i);
    temp(n7)=TonTi*T(i);
    PonPi=TonTi.^(-g0/(K(i)*R));
    press(n7)=P(i)*PonPi;
end

% Mesopause
if any(n8(:))
    i=8;
    temp(n8)=T(i);
    PonPi=exp(-g0*(Hgeop(n8)-H(i))/(T(i)*R));
    press(n8)=P(i)*PonPi;
end

if any(n9(:))
    warning('One or more altitudes above upper limit.')
    temp(n9)=T(8); % Modified by Craig Scratchley, February 2017
    press(n9)=0; % Modified by Craig Scratchley, February 2017
end

temp = temp + Toffset;
```

```
rho = press./temp/R;

if nargout >= 2
    a = sqrt(gamma * R * temp);
    if nargout >= 5
        kvisc = (Bs * temp.^1.5 ./ (temp + S)) ./ rho; %m2/s
        if nargout == 6
            if GeomFlag % Geometric in, ZorH is geopotential altitude
 (H)
                ZorH = Hgeop;
            else % Geop in, find Z
                ZorH = RE*Hgeop./(RE-Hgeop);
            end
        end
    end
end

if Uout %convert to imperial units if output in imperial units
    rho = rho / 515.3788;
    if nargout >= 2
        a = a / 0.3048;
        temp = temp * 1.8;
        press = press / 47.88026;
        if nargout >= 5
            kvisc = kvisc / 0.09290304;
            if nargout == 6
                ZorH = ZorH / 0.3048;
            end
        end
    end
end

if U
    rho = rho*u.kg/(u.m^3);
    if nargout >= 2
        a = a*u.m/u.s;
        temp = temp*u.K;
        press = press*u.Pa;
        if nargout >= 5
            kvisc = kvisc*u.m^2/u.s;
            if nargout == 6
                ZorH = ZorH*u.m;
            end
        end
    end
end

end
% end of nested functions

end % closes function main.
```

Published with MATLAB® R2016b