Table of Contents

ENSC180-Assignment2

```
% Student Name 1: Rameshwar Kannnan

% Student 1 #: 301300734

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

% Student Name 2: Santhosh Nandakumar

% Student 2 #: 301300261

% Student 2 userid (email): snandaku@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 (list names or
 put
% ‘none’)
% Cyrus, Jose, Tal, Seli, Jerry, Laurent, Sterling
```

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

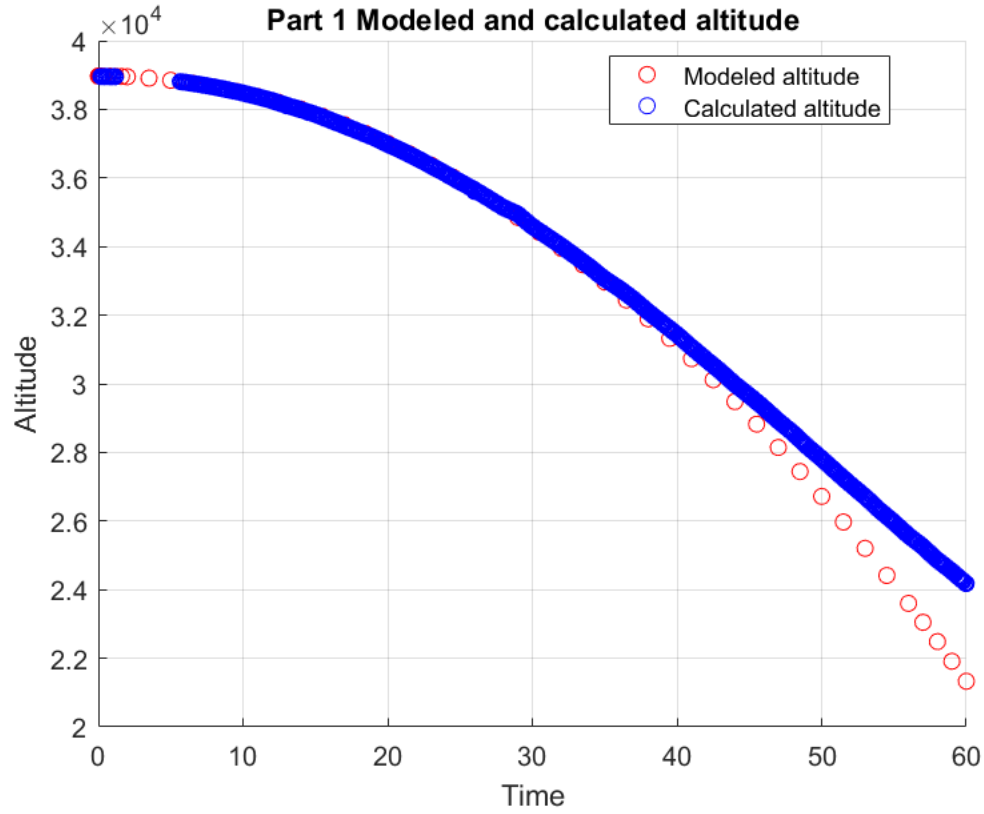
main

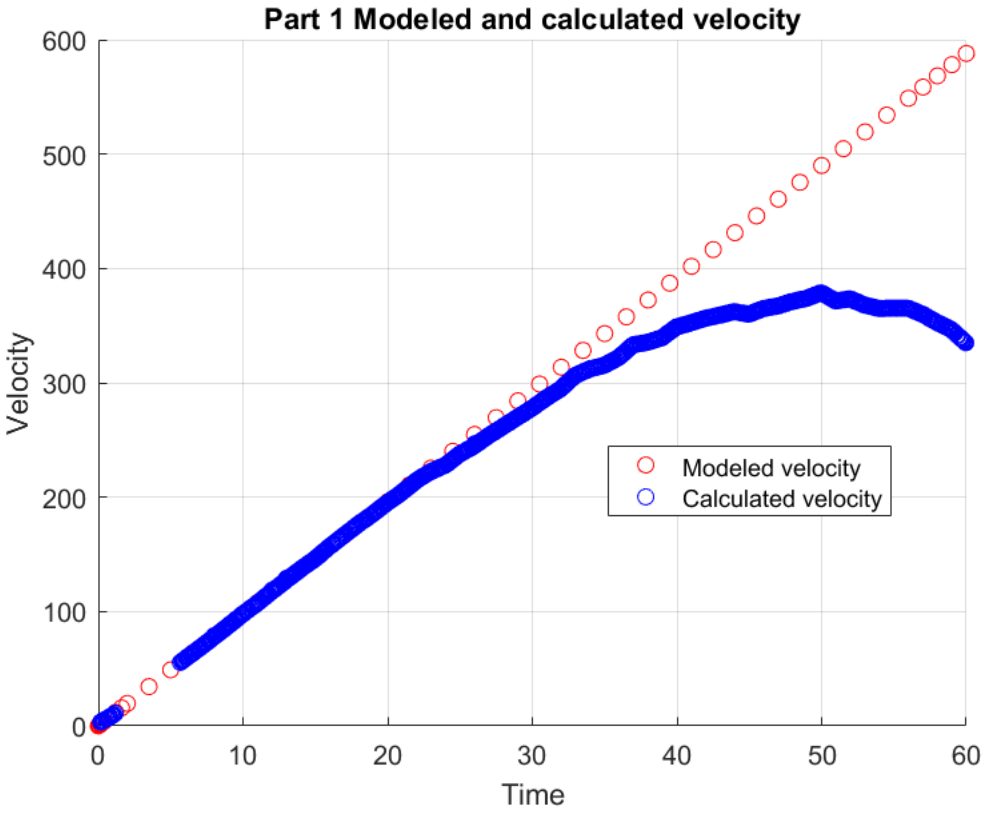
```
function main

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
% <place your work here>
% X=xlsread('180.xlsx','Data','D4:D403')
% X=X(isfinite(X(:, 1)), :)
% v=xlsread('180.xlsx','Data','E4:E403')
% v=v(isfinite(v(:, 1)), :)
% t=xlsread('180.xlsx','Data','K4:K403')
% t=t(isfinite(t(:, 1)), :)
 X=xlsread('data_clean_more_fixed_simplest.xlsx','data_clean_more_fixed_label','B4
%
 v=xlsread('data_clean_more_fixed_simplest.xlsx','data_clean_more_fixed_label','C4
%
t=xlsread('data_clean_more_fixed_simplest.xlsx','data_clean_more_fixed_label','A4:

% ... = xlsread()
% ...
% myVector(isnan(myVector))=[];

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

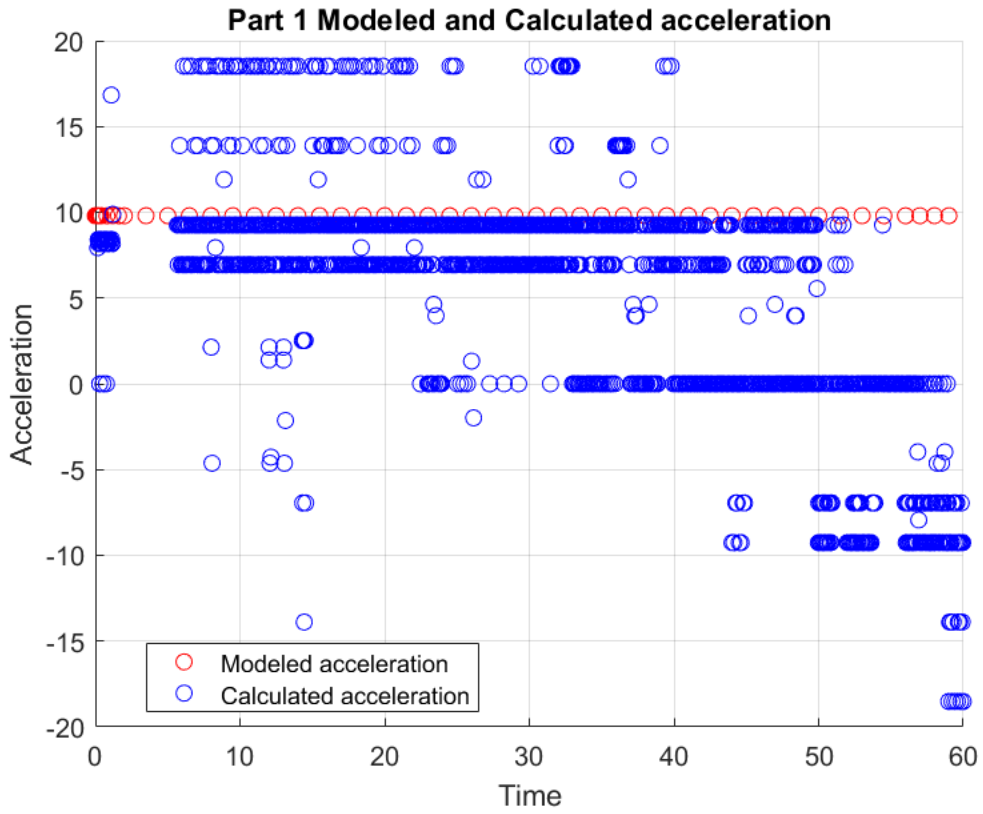
Part 1

Answer some questions here in these comments... How accurate is the model for the first portion of the minute? The model is alomst identical to the calculated data for the first portion of the minute. Therefore it is very accurate.

```
% How accurate is the model for the last portion of that first minute?
% The accuracy of the model decreases over time therefore it is more
% accurate for the first portion of the minute

% Comment on the acceleration calculated from the measured data.
% Is there any way to smooth the acceleration calculated from the
 data?
% This could be done using the smooth function in matlab
% As well as decreasing the increment in the time so that more
 continuous
% data could be extracted <put your answer here in these comments>

part = 1;
%[T,Y] = ode45(@fall, % <...>

% <call here your function to create your plots>
%plotComparisons(60, 'Part1 - Freefall', T, Y <, ...>
calculated_FreeFall(X,v,t);
```


Part 1 Modeled and calculated altitude
×10^4
4
3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
2.2
2
Altitude
0
10
20
30
40
50
60
Time
Modeled altitude
Calculated altitude

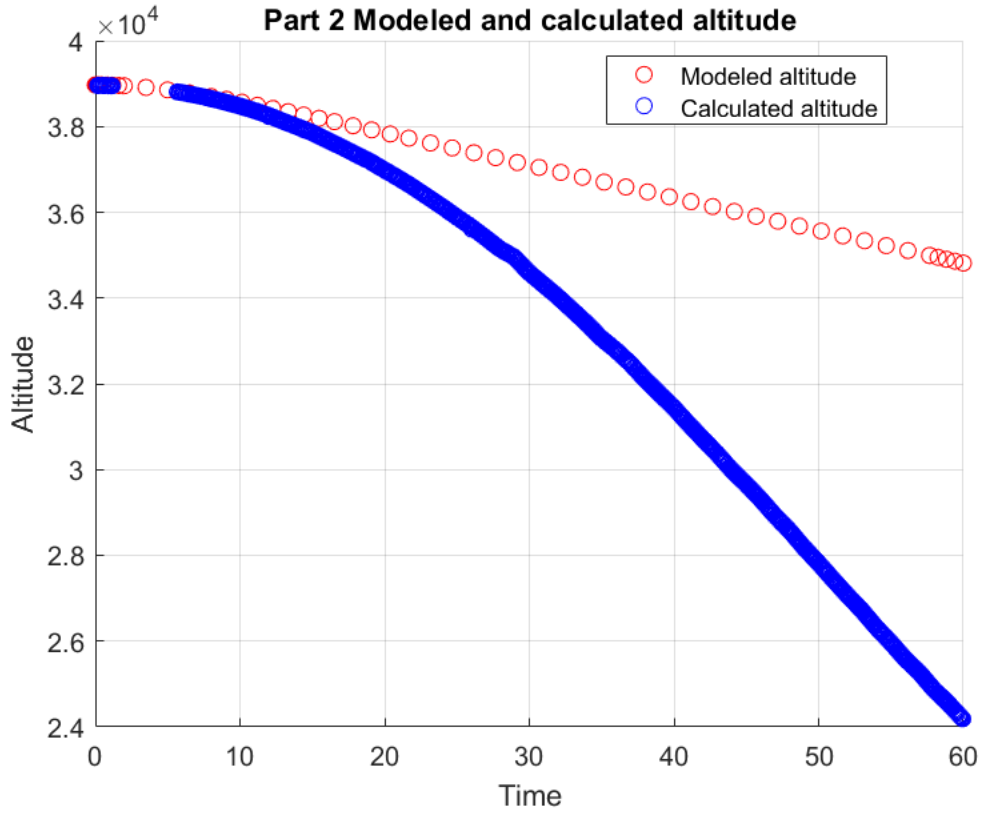

Part 1 Modeled and calculated velocity
600
500
400
300
200
100
0
Velocity
0
10
20
30
40
50
60
Time
Modeled velocity
Calculated velocity

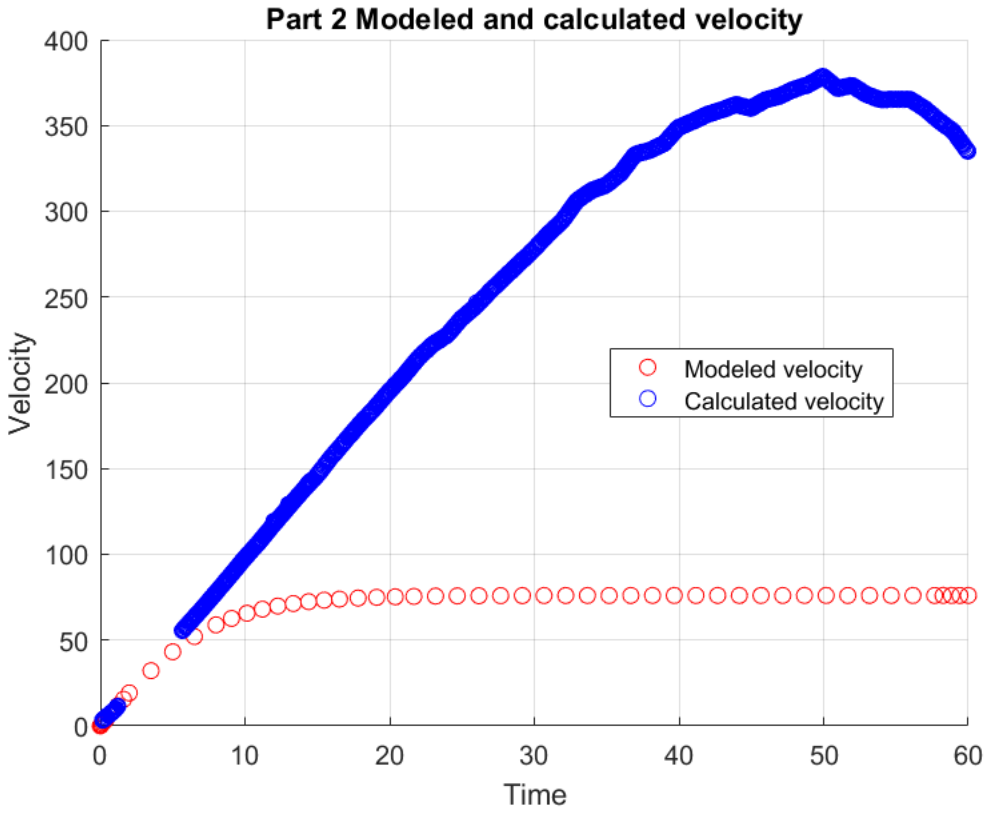


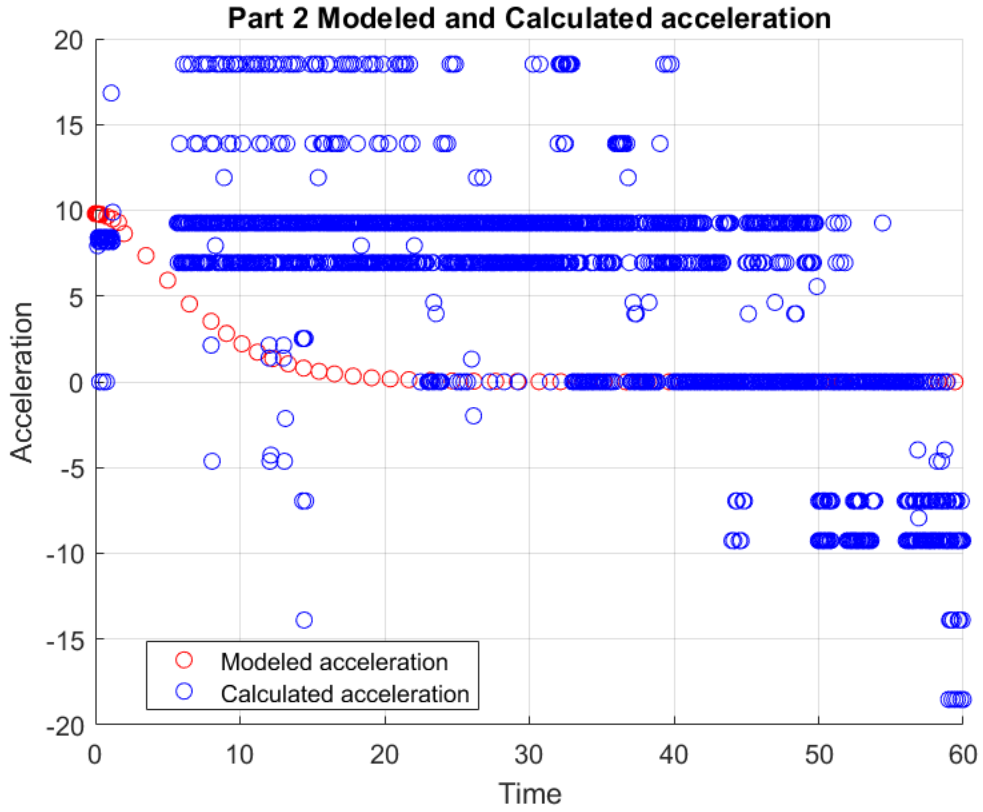
Part 2

Answer some questions here in these comments... Estimate your uncertainty in the mass that you have chosen (at the beginning of the jump). After some research the mass of felix was determined to be 118kg. The mass should be within 10% if his actual mass which includes his suit, oxygen tank and parachute [your answer here in these comments]

```
% How sensitive is the velocity and altitude reached after 60 seconds
 to
%    changes in the chosen mass?
% It is fairly sensitive as the difference between the measured and
% modelled alltitudes and velocities are quiet different.
% <put your answer here in these comments>

part = 2;
%[T,Y] = ode45(@fall, % <...>

% <call here your function to create your plots>
%plotComparisons(<...>, 'Part2 - Simple Air Resistance', T, Y <, ...>
calculated_Simple_Air_Resistence(X,v,t);
```


Part 2 Modeled and calculated altitude
×10^4
4
3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
Altitude
0
10
20
30
40
50
60
Time
Modeled altitude
Calculated altitude


Part 2 Modeled and calculated velocity
400
350
300
250
200
150
100
50
0
Velocity
0
10
20
30
40
50
60
Time
Modeled velocity
Calculated velocity



Part 3

Answer some questions here in these comments... Felix was wearing a pressure suit and carrying oxygen. Why? He needed his suit in order to protect himself from the radiation of the sun due to lack of an ozone layer at that altitutde as well as to shield himself from the cold. He was carrying oxygen due to the decrease in air density in the stratosphere so the need for oxygen was essential in order for him to breath.

```
What can we say about the density of air in the stratosphere?
As mentioned above the air density in the stratosphere is very low.

%     How is the density of air different at around 39,000 meters than
 it
%     is on the ground?
%     Comparitively it is a lot less dense at that altitude. The air
%     density increases with altitude

% 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.
%     Factors such as temperature, air pressure, and altitude are all
 factors
%     that need to be considered when calculating the air density.
```

```
%     As the altitude decreases the pressure increases. Likewise the
%     temperature would decrease due to the ozone layers ability to
 absorb
%     UV rays produced by the sun. All these factors lead to a higher
 overall
%     air density. The troposphere has the highest air density due to
 being
%     compressed by all the other layers of the earths atmosphere.



% What method(s) can we employ to estimate [the ACd] product?
% The approxiamate cross sectional area of Felix and the approxiamate
 drag
% constant for a human were used to estimate the drag constant

% What is your estimated [ACd] product?
% ACd = 0.805
%
% [Given what we are told in the textbook about the simple drag
 constant, b,]
%   does the estimate for ACd seem reasonable?
% It seems reasonable because the ACd product should be around 1.

part = 3;

% <place your work here>
calculated_DragForce(X,v,t);


g =

    9.8066


g =

    9.8066


g =

    9.8066


g =

    9.8066


g =

    9.8066
```

```
g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066
```

```
g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =
```

```
    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =
```

```
    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066
```

```
g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066
```

```
g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066
```

```
g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =
```

```
    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =
```

```
    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066
```

```
g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066
```

```
g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066
```

```
g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =
```

```
    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =
```

```
    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066
```

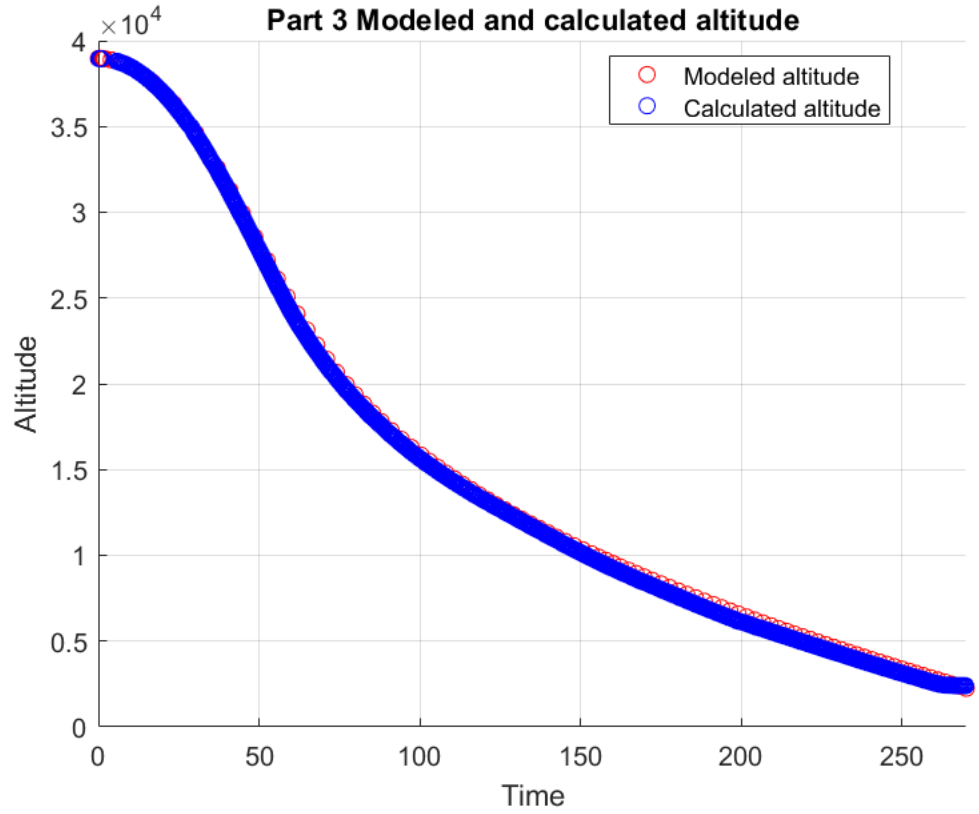
```
g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066
```

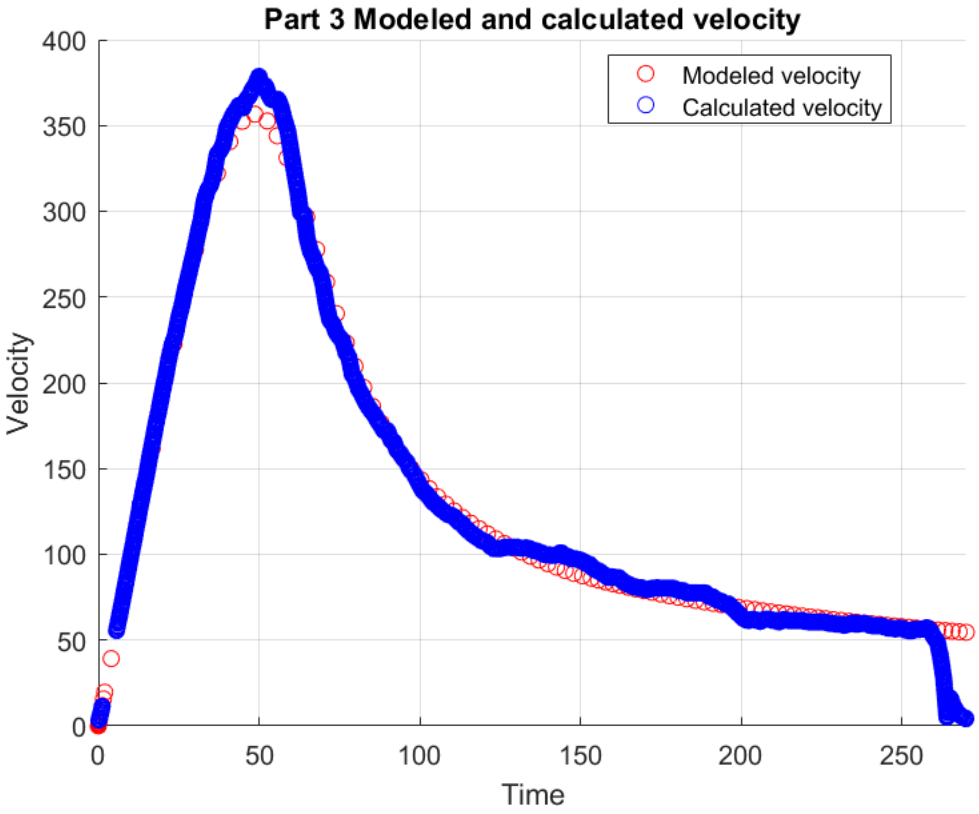
```
g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066
```

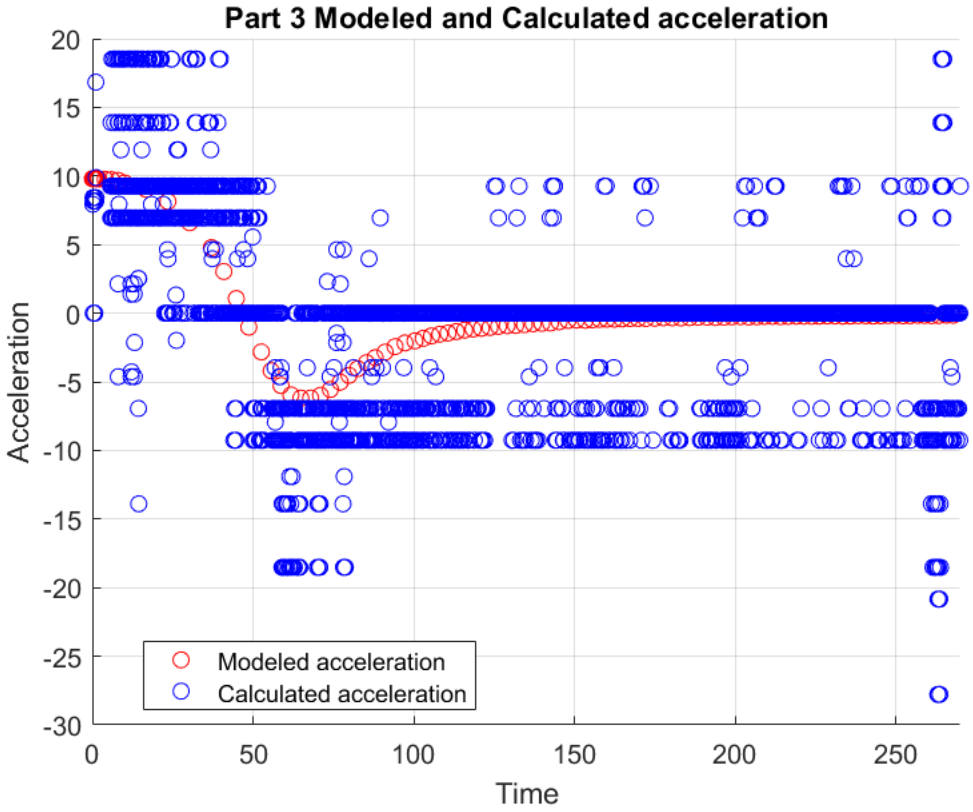
```
g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =
```

```
    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =
```

```
    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066

g =

    9.8066
```


Part 3 Modeled and calculated altitude
×10^4
Modeled altitude
Calculated altitude
Altitude
Time
0
0.5
1
1.5
2
2.5
3
3.5
4
0
50
100
150
200
250


Part 3 Modeled and calculated velocity
Modeled velocity
Calculated velocity
Velocity
Time
0
50
100
150
200
250
300
350
400
0
50
100
150
200
250



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.) 9.69 m/s^2

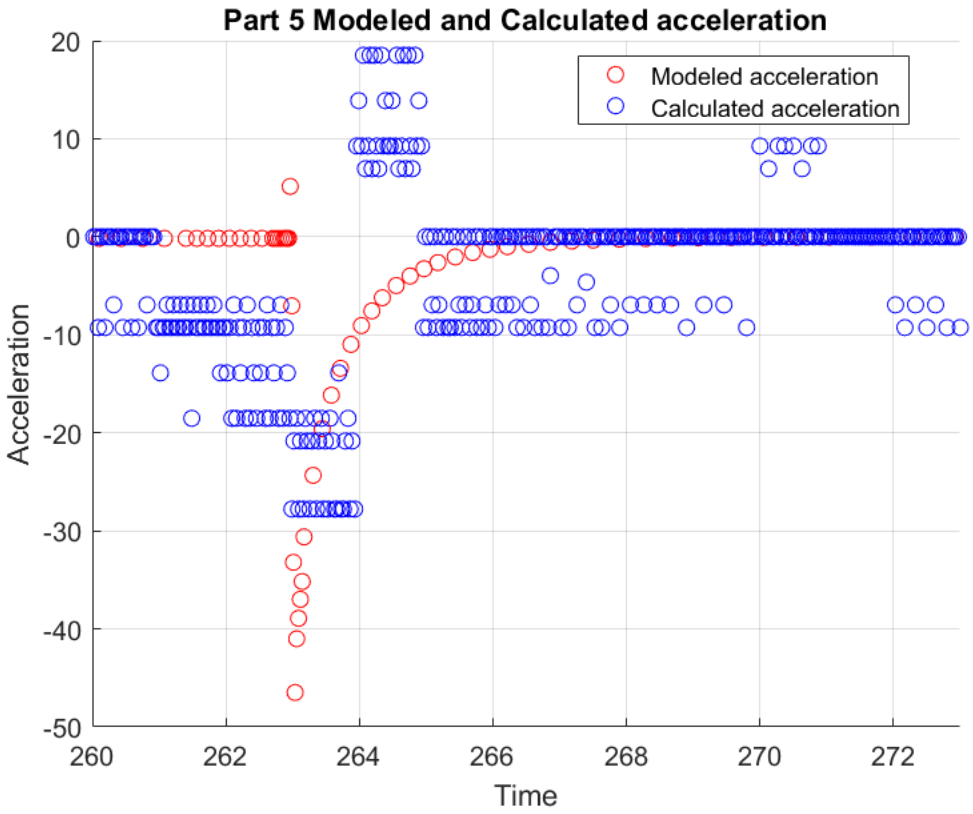
```
% How sensitive is the altitude reached after 4.5 minutes to simpler
 and
%   more complicated ways of modelling the gravitational field
 strength?
% <put your answer here in these comments>

% 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 could take into account the temperature in the stratosphere which
% would change the air density used in the model

% 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?
% Changes to the acceleration due to gravity would be insignificant as
% Felix would be really close to the earth so the acceleration due to
% gravity would be constant.

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

```
% Since the modelled plots for the altitude and velocity were fairly
% close to the calculated plots so the value of the ACd product for
 the
% drag calculation was approxiamated. Factors such as his orientation
% during the fall which would affect his cross sectional area and drag
% constant were not calculated exactly

% [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 tried to closely approxiamate our ACd product to accurately model
% the drag force

part = 4;

% <place your work here>
%calculated_GravitationalForce(X,v,t)
```

Part 5

Answer some questions here in these comments... At what altitude does Felix pull the ripcord to deploy his parachute? He pulls it around 2439 meters

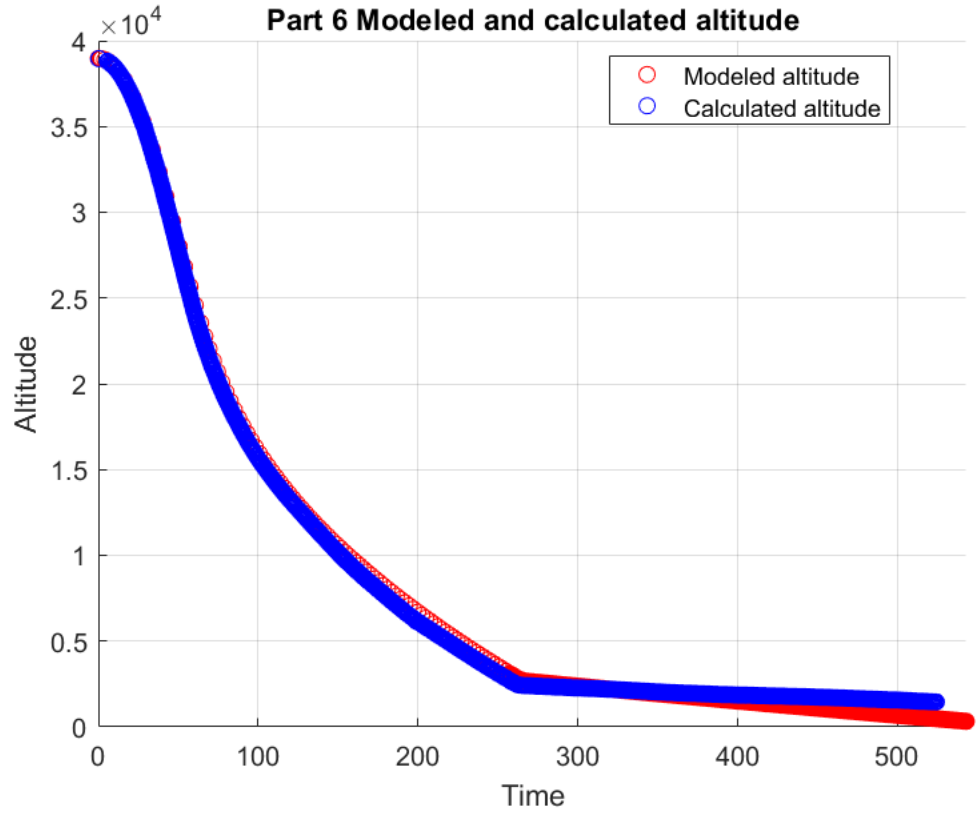
```
%<put your answer here in these comments>

% 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?
% He experiences a maximum acceleration of 48 m/s^2 downwards

%   How safe or unsafe would such an acceleration be for Felix?
% This is a safe acceleration experienced by Felix

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.
% <place your work here>

plotacc(X,v,t);
```

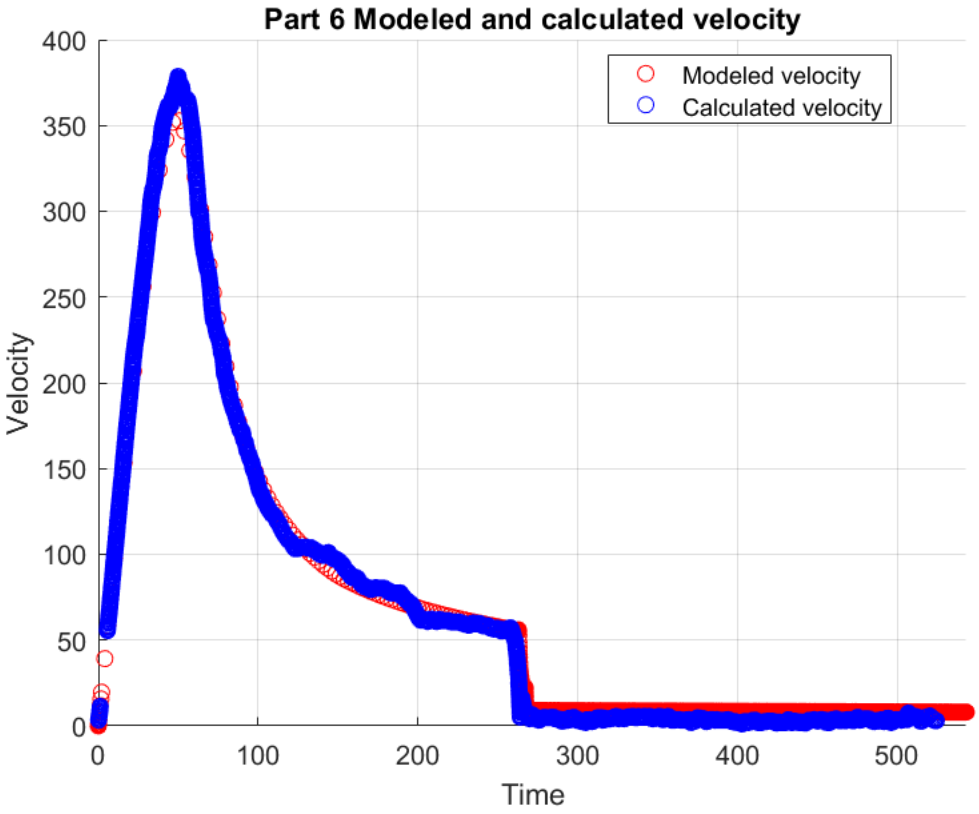


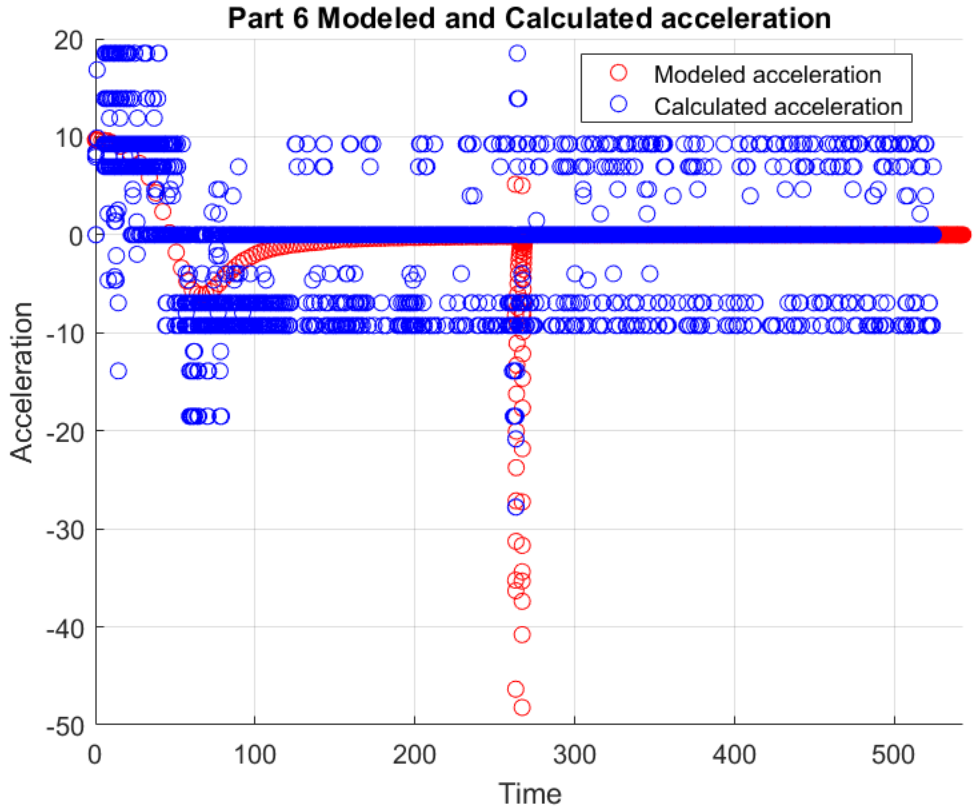
Part 6

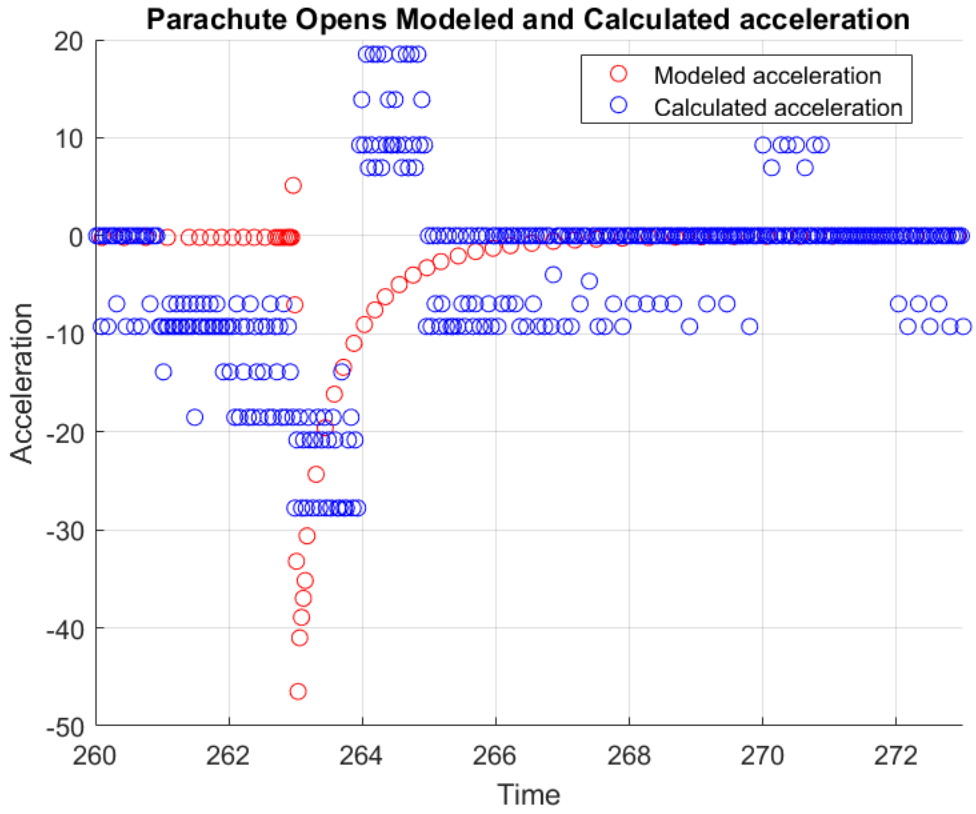
Answer some questions here in these comments... How long does it take for Felix's parachute to open? It takes between 3 to 4 seconds for Felix to open his parachute

```
part = 6;
calculated_StratosJump(X,v,t);
parachute_opens(X,v,t);

%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>
```


Part 6 Modeled and calculated altitude
×10^4
Modeled altitude
Calculated altitude
Altitude
Time
0
0.5
1
1.5
2
2.5
3
3.5
4
0
100
200
300
400
500


Part 6 Modeled and calculated velocity
Modeled velocity
Calculated velocity
Velocity
Time
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500


Part 6 Modeled and Calculated acceleration
Modeled acceleration
Calculated acceleration
Acceleration
Time
20
10
0
-10
-20
-30
-40
-50
0
100
200
300
400
500


Parachute Opens Modeled and Calculated acceleration
Modeled acceleration
Calculated acceleration
Acceleration
Time
20
10
0
-10
-20
-30
-40
-50
260
262
264
266
268
270
272

nested functions

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

```
function res = calculated_FreeFall(X,v,t)
V=abs(v)*0.27777778;
[t1,x1]=ode45(@freefall,[0,60],[38969.4,0]);
x2=x1(:,1);
x3=x1(:,2);
x3=abs(x3);
hold on;
grid on;
plot(t1,x2,'ro');
plot(t,X,'bo');
xlim([0,60]);
title('Part 1 Modeled and calculated altitude');
xlabel('Time');
ylabel('Altitude');
legend('Modeled altitude','Calculated altitude','Location','best');
hold off;
figure;
hold on;
grid on;
plot(t1,x3,'ro');
plot(t,V,'bo');
xlim([0,60]);
title('Part 1 Modeled and calculated velocity');
xlabel('Time');
ylabel('Velocity');
legend('Modeled velocity','Calculated velocity','Location','best');
hold off;
figure;
hold on;
grid on;
delta_V= diff(V);
delta_T= diff(t);
A=delta_V./delta_T;
delta_x3= diff(x3);
delta_t1= diff(t1);
a=delta_x3./delta_t1;
final_Time = t1(1:end-1);
t=t(1:end-1);
plot(final_Time,a,'ro');
plot(t,A,'bo');
xlim([0,60]);
title('Part 1 Modeled and Calculated acceleration');
xlabel('Time');
ylabel('Acceleration');
legend('Modeled acceleration','Calculated
 acceleration','Location','best');
hold off;
end
```

```
function res = calculated_Simple_Air_Resistence(X,v,t)
V=abs(v)*0.27777778;
[t1,x1]=ode45(@freefall1,[0,60],[38969.4,0]);
x2=x1(:,1);
x3=x1(:,2);
x3=abs(x3);
figure;
hold on;
grid on;
plot(t1,x2,'ro');
plot(t,X,'bo');
xlim([0,60]);
title('Part 2 Modeled and calculated altitude');
xlabel('Time');
ylabel('Altitude');
legend('Modeled altitude','Calculated altitude','Location','best');
hold off;
figure;
hold on;
grid on;
plot(t1,x3,'ro');
plot(t,V,'bo');
xlim([0,60]);
title('Part 2 Modeled and calculated velocity');
xlabel('Time');
ylabel('Velocity');
legend('Modeled velocity','Calculated velocity','Location','best');
hold off;
figure;
hold on;
grid on;
delta_V= diff(V);
delta_T= diff(t);
A=delta_V./delta_T;
delta_x3= diff(x3);
delta_t1= diff(t1);
a=delta_x3./delta_t1;
final_Time = t1(1:end-1);
t=t(1:end-1);
plot(final_Time,a,'ro');
plot(t,A,'bo');
xlim([0,60]);
title('Part 2 Modeled and Calculated acceleration');
xlabel('Time');
ylabel('Acceleration');
legend('Modeled acceleration','Calculated
 acceleration','Location','best');
hold off;
end

function res = calculated_DragForce(X,v,t)
V=abs(v)*0.27777778;
[t1,x1]=ode45(@freefall2,[0,270],[38969.4,0]);
```

```
x2=x1(:,1);
x3=x1(:,2);
x3=abs(x3);
figure;
hold on;
grid on;
plot(t1,x2,'ro');
plot(t,X,'bo');
xlim([0,270]);
title('Part 3 Modeled and calculated altitude');
xlabel('Time');
ylabel('Altitude');
legend('Modeled altitude','Calculated altitude','Location','best');
hold off;
figure;
hold on;
grid on;
plot(t1,x3,'ro');
plot(t,V,'bo');
xlim([0,270]);
title('Part 3 Modeled and calculated velocity');
xlabel('Time');
ylabel('Velocity');
legend('Modeled velocity','Calculated velocity','Location','best');
hold off;
figure;
hold on;
grid on;
delta_V= diff(V);
delta_T= diff(t);
A=delta_V./delta_T;
delta_x3= diff(x3);
delta_t1= diff(t1);
a=delta_x3./delta_t1;
final_Time = t1(1:end-1);
t=t(1:end-1);
plot(final_Time,a,'ro');
plot(t,A,'bo');
xlim([0,270]);
title('Part 3 Modeled and Calculated acceleration');
xlabel('Time');
ylabel('Acceleration');
legend('Modeled acceleration','Calculated
 acceleration','Location','best');
hold off;
end

function res = plotacc(X,v,t)
V=abs(v)*0.277777778;
[t1,x1]=ode45(@freefalla,[0,543.043],[38969.4,0]);
x2=x1(:,1);
x3=x1(:,2);
x3=abs(x3);
figure;
```

```
hold on;
grid on;
delta_V= diff(V);
delta_T= diff(t);
A=delta_V./delta_T;
delta_x3= diff(x3);
delta_t1= diff(t1);
a=delta_x3./delta_t1;
final_Time = t1(1:end-1);
t=t(1:end-1);
plot(final_Time,a,'ro');
plot(t,A,'bo');
xlim([260,273]);
title('Part 5 Modeled and Calculated acceleration');
xlabel('Time');
ylabel('Acceleration');
legend('Modeled acceleration','Calculated
 acceleration','Location','best');
hold off;
end

function res = calculated_StratosJump(X,V,T)
V=abs(V)*0.27777778;
[t1,x1]=ode45(@freefall5,[0,543.043],[38969.4,0]);
x2=x1(:,1);
x3=x1(:,2);
x3=abs(x3);
figure;
hold on;
grid on;
plot(t1,x2,'ro');
plot(T,X,'bo');
xlim([0,543.043]);
title('Part 6 Modeled and calculated altitude');
xlabel('Time');
ylabel('Altitude');
legend('Modeled altitude','Calculated altitude','Location','best');
hold off;
figure;
hold on;
grid on;
plot(t1,x3,'ro');
plot(T,V,'bo');
xlim([0,543.043]);
title('Part 6 Modeled and calculated velocity');
xlabel('Time');
ylabel('Velocity');
legend('Modeled velocity','Calculated velocity','Location','best');
hold off;
figure;
hold on;
grid on;
delta_V= diff(V);
delta_T= diff(T);
```

```
A=delta_V./delta_T;
delta_x3= diff(x3);
delta_t1= diff(t1);
a=delta_x3./delta_t1;
final_Time = t1(1:end-1);
T=T(1:end-1);
plot(final_Time,a,'ro');
plot(T,A,'bo');
xlim([0,543.043]);
title('Part 6 Modeled and Calculated acceleration');
xlabel('Time');
ylabel('Acceleration');
legend('Modeled acceleration','Calculated
 acceleration','Location','best');
hold off;
end

function res = parachute_opens(X,v,t)
V=abs(v)*0.277777778;
[t1,x1]=ode45(@freefalla,[0,543.043],[38969.4,0]);
x2=x1(:,1);
x3=x1(:,2);
x3=abs(x3);
figure;
hold on;
grid on;
delta_V= diff(V);
delta_T= diff(t);
A=delta_V./delta_T;
delta_x3= diff(x3);
delta_t1= diff(t1);
a=delta_x3./delta_t1;
final_Time = t1(1:end-1);
t=t(1:end-1);
plot(final_Time,a,'ro');
plot(t,A,'bo');
xlim([260,273]);
title('Parachute Opens Modeled and Calculated acceleration');
xlabel('Time');
ylabel('Acceleration');
legend('Modeled acceleration','Calculated
 acceleration','Location','best');
hold off;
end

end

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

    function res = freefall(t,X)
               p=X(1);
               v=X(2);
               dpdt=v;
```

```
                dvdt=-9.8;
        res=[dpdt;dvdt];
 end

function res = freefall1(t,X)
                p=X(1);
                v=X(2);
                dpdt=v;
                dvdt=acceleration(t,p,v);
        res=[dpdt;dvdt];
end

function res = freefall2(t,X)
                p=X(1);
                v=X(2);
                dpdt=v;
                dvdt=Density(t,p,v);
        res=[dpdt;dvdt];
 end

function res = freefall3(t,X)
                 p=X(1);
                 v=X(2);
                 dpdt=v;
                 dvdt=acceleration1(t,p,v);
                 res=[dpdt;dvdt];
end

function res = freefalla(t,X)
                p=X(1);
                v=X(2);
                dpdt=v;
                dvdt=acceleration5(t,p,v);
        res=[dpdt;dvdt];
end

function res = freefall5(t,X)
    p=X(1);
    v=X(2);
    dpdt=v;
    dvdt=acceleration2(t,p,v);
    res=[dpdt;dvdt];
end




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

%   % a_grav = gravityEst(X);
%       m=118;
%     if part == 1 % variable part is from workspace of function main.
%         res = -a_grav;
```

```
%     else
%    % acceleration(t,p,v)
%     %end
%
%       % b = drag(t, X, v, m);
%        %f_drag = b * v^2;
%        %a_drag = f_drag / m;
%        %res = -a_grav + a_drag;
%     %end
%
%
% function a_grav = gravityEst(X)
%     % 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 <= % fill in
%        a_grav = A_GRAV_SEA;
%      %else
%         %...
%     %end
%end

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

%function res = drag(t, X, v, m)
% <insert description of function here>
%char part;
 %   if part == 2
  %      res = 0.2;
   %else
        % air resistance drag = 1/2*rho*A*Cd
        %%%%%%%%%%% input your code here %%%%%%%%%%%%%%%%
        %ACd = ...

  %end
%end
```

Additional nested functions

Nest any other functions below.

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

function res = acceleration(t,p,v)
    a_g = -9.8;
    b = 0.2;
    m = 118;
    f_d = b * v^2;
```

```
    a_d = f_d/m;
    res = a_g + a_d;
end

function res = Density(t,p,v)
     a_g = -9.8;
     C= 1.15;
  g=9.80665
     m = 118;
 M=0.0289644;
  T0=288.15;
  p0=101.325;
  L=0.0065;
  R=8.31447;
  x=(g*M)/(R*L);
  T=T0-L*x;
  P=p0*(1-(L*p/T0))^x;
  rho = (P*M)/(R*T);
    A = 0.7;
    f_d = 0.5*C* v^2*stdatmo(p)*A;
    a_d = f_d/m;
    res =a_g+ a_d;
end


function res =acceleration1(t,p,v)
    G=6.67*10^(-11);
    M=5.98*10^(24);
    r=6370000+p;
    R=6370000;
    g=-9.81;
    res1=(g*(R)^2)/r^2;
    res2=Density(t,p,v);
    res = res1+res2;
end


function res =acceleration2(t,p,v)
G=6.67*10^(-11);
if t<=263;
b=0.82;
end
if t<267 && t>263
 b=5.2;
end
if t>267
   b=30.333;
end
m=118;
M=5.98*10^(24);
r=6370000+p;
R=6370000;
g=-9.81;
res1=g*(R)^2/r^2;
```

```
res2=0.5*v^2*b*stdatmo(p);
res2=res2/m;
res=res1+res2;
end


function res = acceleration5(t,p,v)
if t<=263
 b=0.867;
end
if t>263
b=4.862;
end
g=-9.8;
m=118;
res1=g;
res2=0.5*b*v^2*stdatmo(p);
res2=res2/m;
res3=res1+res2;
res=res3;
end
% end of nested functions
% closes function main.
```

Published with MATLAB® R2016b