Title: High Altitude Skydiver
Programmer: Andrew Sack
December 20, 2018
ES-55

Description:

This program models a high-altitude sky diver. It uses real air density data from the 1976 U.S. Standard Atmosphere and Newton's law of universal gravitation to model relevant parameters of the atmosphere at different altitudes. It calculates the position, velocity, acceleration, and jerk of the skydiver for their descent. It also models parachute deployment at a specified altitude. An animation and plots of these results are then generated.

Inputs:

The user-configurable parameters are:

- Starting Height – the height the skydiver starts falling from in meters. The maximum is 85,000 meters. [Default: 85000]
- Skydiver Mass – the mass of the skydiver and equipment in kilograms [Default: 90]
- Skydiver Drag Coefficient – the skydiver's drag coefficient before parachute deployment. [Default: 1]
- Skydiver Cross Sectional Area – the skydiver's cross-sectional area normal to the earth in square meters before parachute deployment. [Default: 0.7]
- Parachute Drag Coefficient – the parachute's drag coefficient. [Default: 0.75]
- Parachute Cross Sectional Area – the parachute's cross-sectional area in square meters. [Default: 20]
- Parachute Deploy Altitude – the altitude in meters the parachute is deployed at. [Default: 1500]

Outputs:

The program does the following:

- Collects and validates user input with validate_input()
- Uses airdensity_for_altitude() and the file density_table.csv to import air density values from the 1976 U.S. Standard Atmosphere and uses spline to interpolate for a requested altitude.
- Creates ODE events with heightevent() for the specified parachute deploy height and zero height.
- Uses ode45 to calculate the position and velocity overtime in 2 segments, pre- and post-parachute deploy.
- Calculates acceleration by plugging the position and velocity found by ode45 back into the dv/dt equation.

- Computes jerk with the diff function
- Interpolates the results with interp1 to produce vectors with constant time intervals between points
- Animates the skydiver's descent with values displayed in the figure
- Plots Altitude, Velocity, Acceleration, and Jerk vs. Time and Velocity vs. Altitude.

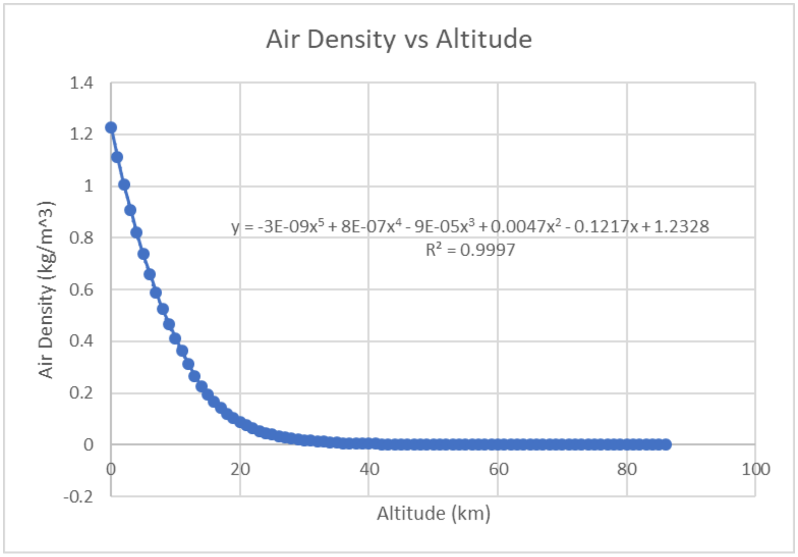
Numerical Methods:

- ODE with events (ode45)
- Differentiation (diff)
- Interpolation (interp1 and spline)
- Animation

Validation of Results:

When looking at the graphs of the motion, the results are as expected. For Sample 1 in Appendix 1, the acceleration is initially approximately -9.5 m/s^2. This is roughly what the acceleration due to gravity at 80 km is, which makes sense since the atmosphere is very thin at this altitude, so the drag force is negligible. As shown in the plot below, the density of air above roughly 30 km is negligible.

As the skydiver approaches 30 km, the acceleration starts to increase to a maximum of 26 m/s^2. At first this was confusing, as I expected the acceleration to be more linear and of a much smaller magnitude. However, upon plotting the air density vs altitude data from the 1976 U.S. Standard Atmosphere in Excel, I noticed that the curve that best approximated it was a 5th order polynomial. Since drag, and hence acceleration, are proportional to air density, the high order of air density explains why the acceleration is so large.



After 30 kilometers is passed, the acceleration remains just above 0 for most the remaining descent. This is because terminal velocity has been reached. As altitude decreases, air density and drag increase slightly, causing the terminal velocity to gradually decrease as well. This is why acceleration is not exactly 0. A large spike in acceleration is observed when the parachute is deployed, due to the significant increase in cross-sectional area and, hence, drag force. This corresponds with a step in the velocity plot and a change of slop in the altitude plot.

Since all interesting points in the graphs can be explained by expected physical occurrences, the simulation makes sense. It is hard to compare with numbers, since the highest anyone has actually skydived from was only 39 km. This world record jump was performed by Felix Baumgartner in October 2012. This jump was simulated in Sample 2 of Appendix 1 with the parameters specified in the Appendix. These parameters were gathered from a paper on the jump. This simulation calculated a maximum speed of 392.8 m/s, while the actual maximum speed was measured to be 377.1 m/s. The paper also said that the maximum speed occurred at 50 seconds into the jump, while the simulation has it occurring at 53.57 seconds. Both of these values are reasonable accurate, however total flight time is less accurate. The parachute was deployed at 2566 m, which was 4:20 minutes into the flight, however the simulation has this occurring at 3:34 minutes into the flight.

The reason for this discrepancy is that both skydiver area and drag coefficient are assumed to be constant. However, drag coefficient is dependent on velocity. Felix Baumgartner had a drag coefficient of approximately 2 when he was supersonic and of approximately 1 when he was subsonic. Furthermore, skydivers regularly change their body position which changes both their area and drag coefficient. These parameters are also hard to determine exact values for, so the defaults are rough estimates based on a combination of values published online.

References:

"1976 Standard Atmosphere Table." Accessed December 9, 2018. https://www.digitaldutch.com/atmoscalc/tableatmosphere.htm.

Chapra, S.C. 2007. Applied Numerical Methods with MATLAB for Engineers and Scientists. McGraw-Hill, New York.

Colino, Jose M., Antonio J. Barbero, and Francisco J. Tapiador. "Dynamics of a Skydiver's Epic Free Fall." *Physics Today*, April 2014.

"Earth Atmosphere Model - Metric Units." Accessed December 8, 2018. https://www.grc.nasa.gov/www/k-12/rocket/atmosmet.html.

"Law of Gravity." Accessed December 15, 2018. http://physics.weber.edu/amiri/physics1010online/WSUonline12w/OnLineCourseMovies/CircularMotion&Gravity/reviewofgravity/ReviewofGravity.html.

Mills N, Landell. "Buoyancy Explains Terminal Velocity in Skydiving." *Journal of Aeronautics & Aerospace Engineering* 06, no. 02 (2017). https://doi.org/10.4172/2168-9792.1000189.

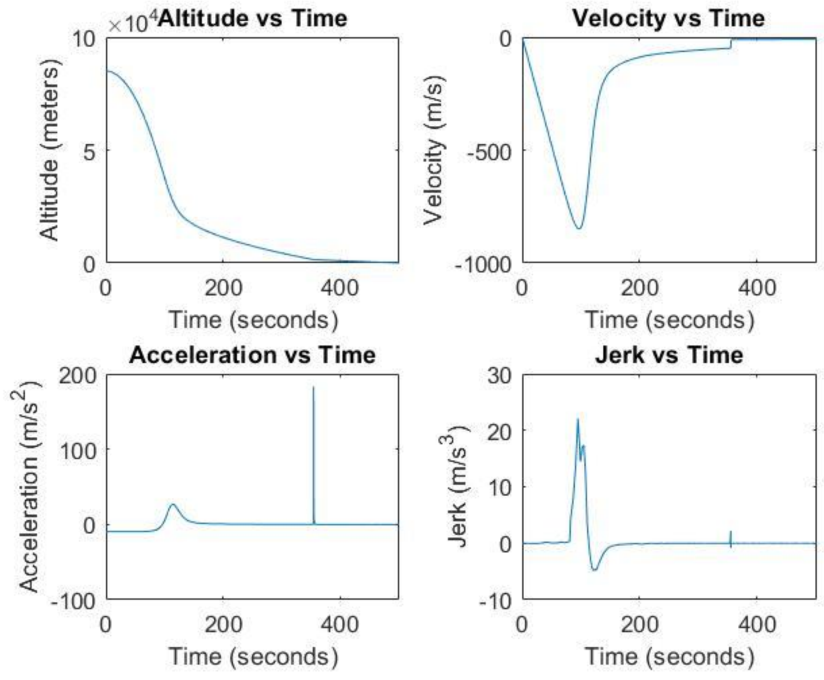
"Newton's Law of Universal Gravitation." Accessed December 15, 2018. https://www.physicsclassroom.com/class/circles/Lesson-3/Newton-s-Law-of-Universal-Gravitation.

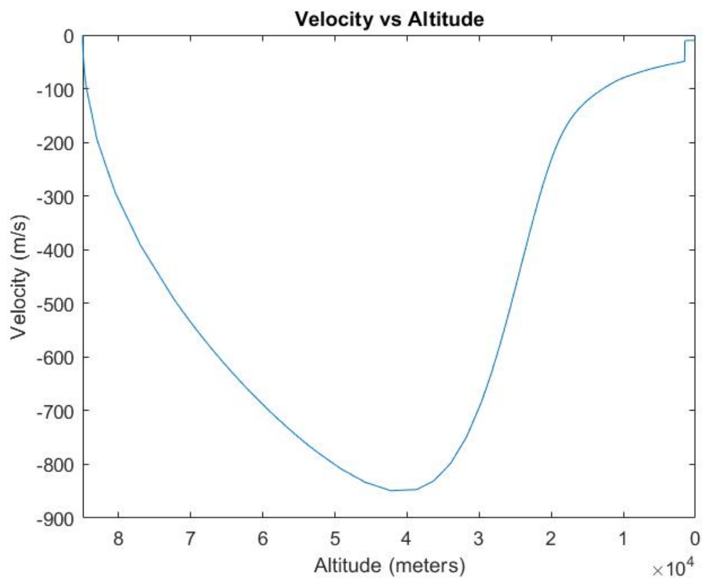
"Parachute | Red Bull Stratos." Accessed December 15, 2018. http://www.redbullstratos.com/technology/parachute/index.html.

"Speed Skydiving." *Wikipedia*, November 29, 2018. https://en.wikipedia.org/w/index.php?title=Speed_skydiving&oldid=871119786.

Appendix 1: Sample Output

Sample 1 (Default Settings):



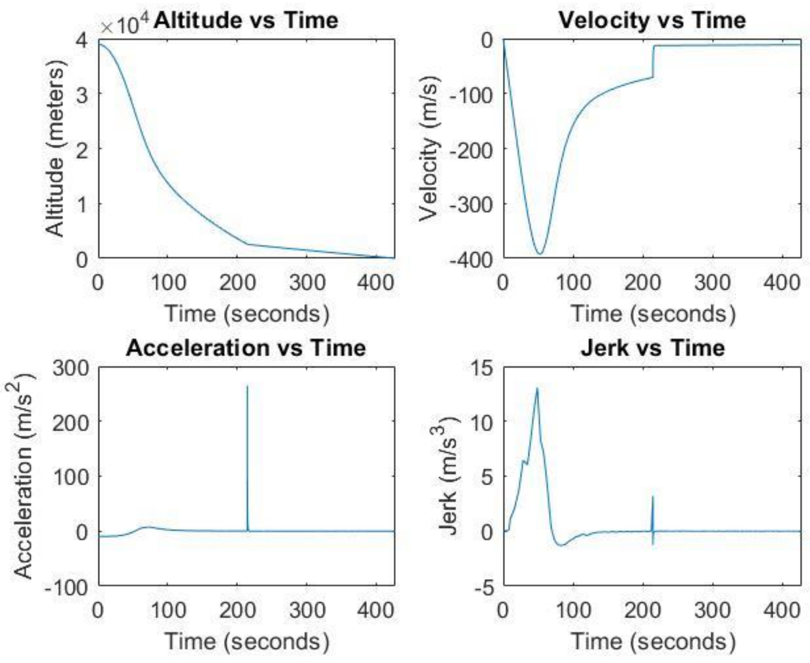


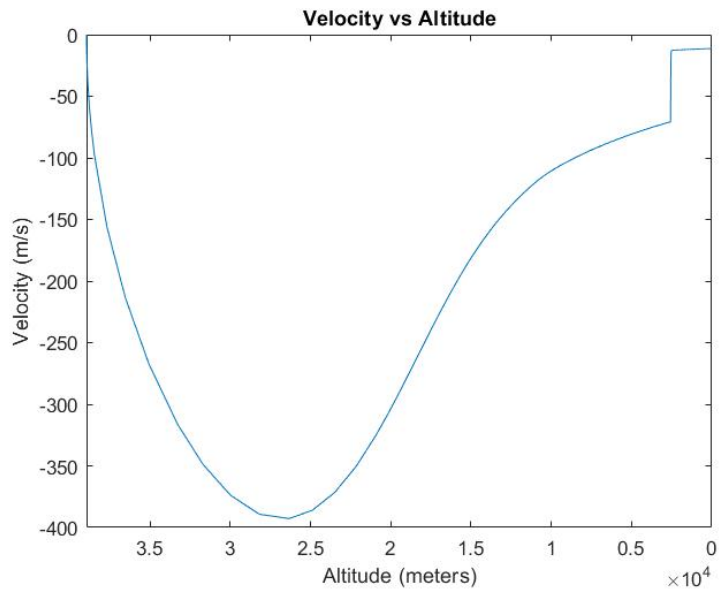
Sample 2 (Felix Baumgartner Jump):

Parameters:

- Starting Height – 38969 m

- Skydiver Mass – 118 kg
- Skydiver Drag Coefficient – 2
- Skydiver Cross Sectional Area – 0.25 m^2
- Parachute Drag Coefficient – 0.75
- Parachute Cross Sectional Area – 20 m^2
- Parachute Deploy Altitude – 2566 m





Appendix 2: Code

Sack_Project2018.m (Script)

```
% Andrew Sack
% ES 55
% Final Project
% 12/20/18
clear all; close all; clc;

%% User Input
prompt = {'Starting Height (meters) - max 85000','Skydiver Mass (kg)',...
    'Skydiver Drag Coefficient','Skydiver Cross Sectional Area (square
meters)', ...
    'Parachute Drag Coefficient','Parachute Cross Sectional Area (square
meters)', ...
    'Parachute Deploy Altitude (meters)'};
title1 = 'Parameters';
dims = [1 50];
definput = {'85000','90', '1', '0.7', '0.75', '20', '1500'};

isValid = false;
while ~isValid % Loop to reprompt if input is invalid

    answer = inputdlg(prompt,title1,dims,definput);

    height = str2num(answer{1});
    mass = str2num(answer{2});
    Cd_man = str2num(answer{3});
    area_man = str2num(answer{4});
    Cd_par = str2num(answer{5});
    area_par = str2num(answer{6});
    h_deploy = str2num(answer{7});

    isValid = validate_input(height, mass, Cd_man, area_man, Cd_par, ...
        area_par, h_deploy);
    if ~isValid
        errordlg('Inputs are not Valid')
    end
end

% global so file input isn't repeated every time airdensity_for_altitude is
called
global M;
M = csvread('density_table.csv'); % Read in air density values

%% Calculations
% Pre Parachute
tspan_1 = [0 inf];
init_1 = [height 0];
parDeployFcn = @(T, Y) heightevent(T, Y, h_deploy);
opts=odeset('Events', parDeployFcn);

[t1, y1, te1, ye1] = ode45(@(t, y) skydiving_diffeq(t, y, Cd_man, area_man,
mass), tspan_1, init_1, opts);
```

```
% After Parachute is deployed
tspan_2 = [t1(end) inf];
init_2 = [h_deploy y1(end)];
landFcn = @(T, Y) heightevent(T, Y, 0);
opts=odeset('Events', landFcn);

[t2, y2, te2, ye2] = ode45(@(t, y) skydiving_diffeq(t, y, Cd_par, area_par,
mass), tspan_2, init_2, opts);

% Merge pre and post parachute together
t = [t1; t2(2:end)];
pos = [y1(:,1); y2(2:end,1)];
vel = [y1(:,2); y2(2:end,2)];
acc1 = skydiving_accel(y1(:,1),y1(:,2), Cd_man, area_man, mass);
acc2 = skydiving_accel(y2(:,1),y2(:,2), Cd_par, area_par, mass);
acc = [acc1; acc2(2:end)];

% calculate jerk
for n = 1:length(t)-1
    dt(n) = t(n+1) - t(n);
end
jerk = diff(acc) .* dt';

%% Animation
% Interpolate everything so time step is constant
time_step = 5;
t_interp = 0:time_step:te2;

p_interp = interp1(t, pos, t_interp);
v_interp = interp1(t, vel, t_interp);
a_interp = interp1(t, acc, t_interp);
j_interp = interp1(t(1:end-1), jerk, t_interp);

f = figure;
hold on;
ylim([-1000, height])
xlim([-1 1])
axis manual
rectangle('Position', [-1 -1000 2, 1000],'FaceColor', 'g')% 'land'

% Animation loop
for n = 1:length(t_interp)
    title(sprintf('Time: %0.0f Seconds', t_interp(n)));

    str = {sprintf('Altitude: %0.0f m', p_interp(n)), ...
        sprintf('Velocity: %0.1f m/s', v_interp(n)), ...
        sprintf('Acceleration: %0.3f m/s^2', a_interp(n)), ...
        sprintf('Jerk: %0.3f m/s^3', j_interp(n))};
    t_handle = text(0.32,0.87*height,str);

    if t_interp(n) < te1 % Plot circle for pre parachute
        h = plot(0, p_interp(n), 'or',
'MarkerFaceColor',[1,0,0],'MarkerSize',5);
    else % Plot upside down triangle to represent parachute
        h = plot(0, p_interp(n), 'vb',
'MarkerFaceColor',[0,0,1],'MarkerSize',10);
    end
    pause(0.1);
    delete(t_handle);
    delete(h)

end
hold off;
delete(f)

%% Plotting
figure;
subplot(2, 2, 1)
plot(t,pos)
title('Altitude vs Time')
xlabel('Time (seconds)')
ylabel('Altitude (meters)')
xlim([0 te2]);

subplot(2, 2, 2)
plot(t, vel)
title('Velocity vs Time')
xlabel('Time (seconds)')
ylabel('Velocity (m/s)')
xlim([0 te2]);

subplot(2, 2, 3)
plot(t, acc)
title('Acceleration vs Time')
xlabel('Time (seconds)')
ylabel('Acceleration (m/s^2)')
xlim([0 te2]);

subplot(2, 2, 4)
plot(t(1:length(t)-1), jerk)
title('Jerk vs Time')
xlabel('Time (seconds)')
ylabel('Jerk (m/s^3)')
xlim([0 te2]);

figure;
plot(pos, vel)
title('Velocity vs Altitude')
xlabel('Altitude (meters)')
ylabel('Velocity (m/s)')
xlim([0 height]);
set(gca, 'XDir','reverse')
```

skydiving_diffeq.m (Function)

```
function [dy] = skydiving_diffeq(t, y, Cd, A, m)
%function [dy] = skydiving_diffeq(t, y, Cd, A, m)
%Differential equation used by ode45
%Inputs:
%   t: time
%   y: vector representing position and velocity
%   Cd: Drag Coefficient
%   A: Area
%   m: mass of skydiver
%Output:
%   dy: vector representing derivative of position and velocity

% Constants
r_earth = 6.371 * 10^6; %m
m_earth = 5.9722 * 10^24;%kg
G = 6.67384 * 10^-11; %m^3/kg*s^2

dy(1, 1) = y(2);
dy(2, 1) = -G .* m_earth ./ (r_earth + y(1)).^2 + ...
    (airdensity_for_altitude(y(1)) .* y(2).^2 .* Cd .* A) ./ (2.*m);

end
```

skydiving_accel.m (Function)

```
function [accel] = skydiving_accel(x, v, Cd, A, m)
%function [accel] = skydiving_accel(x, v, Cd, A, m)
%Calculate acceleration for a given altitude and velocity
%Inputs:
%   x: position
%   v: velocity
%   Cd: Drag Coefficient
%   A: Area
%   m: mass of skydiver
%Output:
%   accel: acceleration

% Constants
r_earth = 6.371 * 10^6; %m
m_earth = 5.9722 * 10^24;%kg
G = 6.67384 * 10^-11; %m^3/kg*s^2

accel = -G .* m_earth ./ (r_earth + x).^2 + ...
    (airdensity_for_altitude(x) .* v.^2 .* Cd .* A) ./ (2.*m);

end
```

airdensity_for_altitude.m (Function)

```
function [density] = airdensity_for_altitude(altitude)
%function [density] = airdensity_for_altitude(altitude)
%Uses Spline to calculate density for a given altitude from a table
%Input:
%   altitude: Altitude in meters to find density at
%Output:
%   density: Air density at the input altitude

global M;
density = spline(M(:,1), M(:,2), altitude);

end
```

height_event.m (Function)

```
function [detect, stopint, direct]=heightevent(t, y, offset)
%function [detect, stopint, direct]=heightevent(t, y, offset)
% Locate the time when height passes through offset
% and stop integration.
%Inputs:
%   t: time
%   y: vector of position and velocity
%   offset: height to stop ODE at
%Outputs:
%   detect: stop ODE when it equals 0. When height = offset
%   stopint: 1 to stop integration
%   direct: 0 because Direction does not matter

    detect = y(1) - offset; % stop when detect = 0
    stopint = 1; % Stop the integration
    direct = 0; % Direction does not matter
end
```

validate_input.m (Function)

```
function [isValid] = validate_input(height, mass, Cd1, A1, Cd2, A2, h_dep)
%function [isValid] = validate_input(height, mass, Cd1, A1, Cd2, A2, h_dep)
%Checks that all user inputs are valid
%Inputs:
%   height - user inputed height
%   mass - user inputed mass
%   Cd1 - user inputed Cd_man
%   A1 - user inputed area_man
%   Cd2 - user inputed Cd_par
%   A2 - user inputed area_par
%   h_dep - user inputed h_deploy
%Output:
%   isValid - True if all inputs are valid, else false
isValid = true;

% Check everything is a number
if isempty(height) || isempty(mass) || isempty(Cd1) || isempty(A1) || ...
        isempty(Cd2) || isempty(A2) || isempty(h_dep)
    isValid = false;
    return;
% Check everything is positive
elseif height <= 0 || mass <= 0 || Cd1 <= 0 || A1 <= 0 ||...
        Cd2 <= 0 || A2 <= 0 || h_dep <=0
    isValid = false;
% Check heights are valid
elseif height > 85000 || h_dep >= height
    isValid = false;
end

end
```

Appendix 3: Mathematical Details

There are 2 forces that act on a skydiver, Drag and Gravity.

$$F_{net} = F_{drag} - F_{gravity}$$

The sum of these two forces (in opposite directions) is the net force on the skydiver and is equal to their mass * acceleration in accordance with Newton's 2nd Law of motion.

$$F_{net} = ma$$

$$a = \frac{dv}{dt} = \frac{F_{drag}}{m} - g$$

Drag Force:

$$F_{drag} = \frac{1}{2}\rho v^2 C_D A$$

Where:

- ρ = air density [kg/m^3] (non-constant)
- v = velocity [m/s]
- C_D = Drag Coefficient [unitless]
- A = Cross-sectional Area [m^2]

Since we are concerned with high altitudes, gravitational acceleration is not constant, so we must derive it from Newton's law of universal gravitation.

$$F_{gravity} = G * \frac{m_1 m_2}{r^2}$$

Rearranging and accounting for the radius of the earth we get:

$$a_{grav} = g = G * \frac{m_{earth}}{(r_{earth} + h)^2}$$

Where:

- G = 6.67384 * 10^{-11} m^3/kg*s^2 (Gravitational Constant)
- m_{earth} = 5.9722 * 10^{24} kg
- r_{earth} = 6.371 * 10^6 m
- h = height from surface of the Earth [m]

We can then form our 2 ordinary differential equations for position and velocity

Position:

$$v = \frac{dx}{dt}$$

Velocity:

$$\frac{\rho(h) v^2 C_D A}{2m} - G * \frac{m_{earth}}{(r_{earth} + h)^2} = \frac{dv}{dt}$$

Acceleration can then be found by plugging the position and velocity results back into the ODE for velocity.

Jerk is then calculated by taking the derivative with respect to time of acceleration.

$$jerk = \frac{da}{dt}$$