Dynamics of a Skydiver

Simulation of Freefall and Parachute Deployment

Nil Valls

Physics 309 – Spring 2006

Prof. P. Johnson

Illinois Institute of Technology

1 Introduction

Skydiving has become a popular amusement activity among people of all ages. As in most thrills, skydivers become aroused by the release of adrenaline while experiencing a dangerous situation. This practice would be a lot more fatal if it were not by the safety mechanisms developed after analyzing the dynamics of skydivers and implementing the results into the design of skydiving gear, such as the canopy, suspension lines, harness and attachments.

A study of the general dynamics governing the motion of an idealized skydiver jumping off a moving airplane and deploying the parachute at a safe altitude is performed in this project. This will likely allow for the determination of critical points of acceleration and velocity which will help understand the engineering requirements or the skydiving gear and the physiological limits of an average person.

2 Procedure

For the purpose of this project, a single idealized skydiver will be considered. It will be assumed the skydiver does not perform any maneuvers or change the geometry of his body. Furthermore, the geometry of the skydiver's body will be approximated to be a rectangular plate with the same surface area as an average person, and its orientation will be perpendicular to the direction of motion at all times. The circular (traditional) parachute will be deployed at a safe altitude of about 1200m above ground. The kinematics of the skydiver can be obtained from the kinetic relationships, i.e. Newton's second law. Equation 1, in which $\vec{F}$ is any force vector acting upon a body, m is the mass, $\vec{a}$ is the acceleration vector of the center of mass and $\vec{r}$ is the position vector with respect to the origin, illustrates such a relationship.

$$\sum \vec{F} = m\vec{a} = m\frac{d^2\vec{r}}{dt^2}$$

Equation 1: *Newton's second law.*

In addition, the relationship relating the net force the angular acceleration, given by equation 2, will hold for a more realistic case. However, it will be ignored in this simulation. In the equation below, $\vec{s}$ is the position vector of any particle on the surface of the body with respect to its center of mass, I_{ij} is the moment of inertia tensor about the center of mass, $\vec{\alpha}$ is the angular acceleration vector and $\vec{\theta}$ is the orientation vector.

$$\sum \vec{s} \times \vec{F} = I_{ij}\vec{\alpha} = m\frac{d^2\vec{\theta}}{dt^2}$$

Equation 2: *Newton's second law for rotating system.*

The scale of the skydiver's speed requires the consideration of air resistance. Knowing the drag coefficient C of the medium, the density of the medium ρ, the cross-sectional area A of the moving body perpendicular to the direction of motion and its instantaneous speed v, the magnitude of the drag force F_D caused by the air resistance will be given by the relationship in equation 3. The direction of the drag force will always be opposite to the direction of motion.

$$F_D = \frac{1}{2}C\rho Av^2$$

Equation 3: *Drag force.*

A close look at the relationship above reveals the complexity of performing a two dimensional analysis with air resistance, as the drag force can not be computed by obtaining its orthogonal components from each velocity component.

Figure 1 shows the free body diagram of the idealized skydiver a few seconds after leaving the airplane. The system of coordinates chosen associates the x direction with the horizontal and y direction with the vertical.

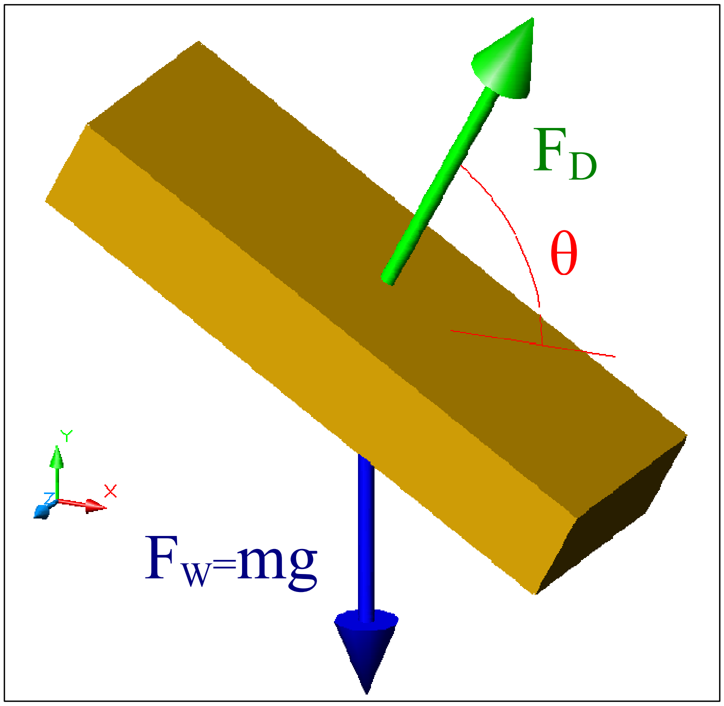


Figure 1: *Free body diagram of the idealized skydiver, a few seconds after jumping of the airplane. F_D is the drag force, F_w is the weight, m is the mass and g the acceleration due to gravity.*

The free body diagram, along with equations 1 and 3, allows the formulation of the following relationships describing the motion of the skydiver in the x-y plane:

$$\rightarrow \sum F_x = -F_D \cos\left[\theta(t)\right] = \frac{1}{2} C\rho A\left(v_x^2 + v_y^2\right)\cos\left[\theta(t)\right] = ma_x$$

$$\uparrow \sum F_y = F_D \sin\left[\theta(t)\right] - F_W = \frac{1}{2} C\rho A\left(v_x^2 + v_y^2\right)\sin\left[\theta(t)\right] - gm = ma_y$$

Equations 4, 5: *Kinetics.*

It is convenient to make use of the equality $\gamma = \frac{C\rho A}{2m}$ and to substitute in for all the time dependent variables except for the angle of attack $\theta(t)$. This leads to the kinematic equations, as presented below.

$$\gamma\left[\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right]\cos\left[\theta(t)\right] = -\frac{d^2x}{dt^2}$$

Equation 6: *Horizontal kinematic equation.*

$$\gamma\left[\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right]\sin\left[\theta(t)\right] - g = \frac{d^2y}{dt^2}$$

Equation 7: *Vertical kinematic equation.*

$$\theta(t) = \tan^{-1}\left[\frac{\left(\frac{dx}{dt}\right)}{\left(\frac{dy}{dt}\right)}\right]$$

Equation 8: *Angle of attack.*

As it can be seen above, the kinematic relationships consist of a system of two second order non linear, non homogeneous differential equations, which present a difficult challenge. Since the purpose of this project is to obtain a fair approximation of the skydiver's motion with decent time resolution, a numerical iterative algorithm was created and run in MatLab® 7.0.5 for Linux. The complete source code can be found in the appendix.

The algorithm made use of the simple kinematic relationships derived from the definitions of position, velocity and acceleration as functions of time. A small time increment of 100 ms was chosen for each iteration. The equations below were used in the iterative loop.

$$(x,y)_{n+1} = (x,y)_n + \left(v_x,v_y\right)_n \cdot (dt) + \frac{1}{2}\left(a_x,a_y\right)_n \cdot (dt)^2$$

Equation 9: *Horiz'l position.*

$$\left(v_x,v_y\right)_{n+1} = \left(v_x,v_y\right)_n + \left(a_x,a_y\right)_n \cdot (dt)$$

Equation 10: *Vertical position.*

$$\left(a_x\right)_{n+1} = -\gamma \cos\left[\theta_n\right] \cdot \left[\left(v_x\right)_n^2 + \left(v_y\right)_n^2\right]$$

Equation 11: *Vertical position.*

$$\left(a_y\right)_{n+1} = \gamma \cos\left[\theta_n\right] \cdot \left[\left(v_x\right)_n^2 + \left(v_y\right)_n^2\right] - g$$

Equation 12: *Vertical position.*

$$\theta_{n+1} = \tan^{-1}\left[\frac{\left(v_y\right)_{n+1}}{\left(v_x\right)_{n+1}}\right]$$

Equation 13: *Angle of attack.*

The right hand side of equations 9 through 12 can be used in a series to compute each quantity at a specific moment in time, considering that the time step dt is 100ms. For example, the horizontal position as a discrete function of time will be given by equation 14.

$$x(t) = \sum_{n=1}^{t/dt} x_{n-1} + \left(v_x\right)_n \cdot (dt) + \frac{1}{2}\left(a_x\right) \cdot (dt)^2$$

Equation 14: *Horizontal position as a discrete function of time.*

The simulation started at time zero, at which moment the skydiver just cleared the airplane, and it stops when the altitude is effectively zero.

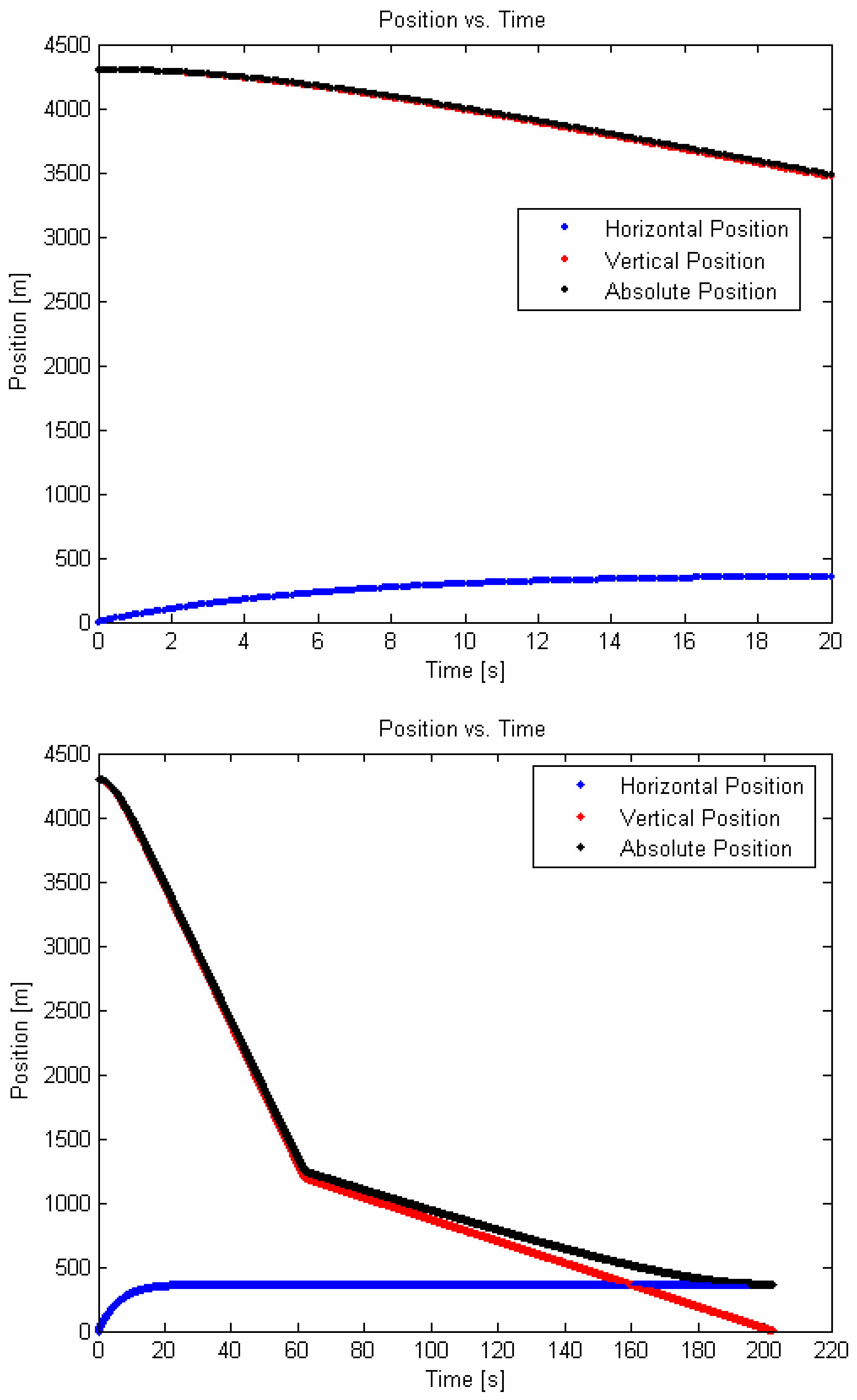
3 Results

Table 1 contains simulation input parameters and initial conditions.

Simulation Input Parameters and Initial Conditions		
Parameter	**Value in SI units**	**Value in units used in Aeronautics**
Time step	0.1 s	
Total mass	80 kg	
Skydiver’s surface area	0.75 m^2	
Canopy surface area	30 m^2	
Air density at jump altitude	1.1 kg/m^3	
Air density at deployment alt.	1.25 kg/m^3	
Air’s drag coefficient	0.7	
Jump altitude	4300 m	14000 ft
Parachute deployment altitude	1200 m	4000 ft
Airplane horizontal velocity	60 m/s	120 kts
Airplane vertical velocity	0 m/s	0 kts

Table 1: *Simulation input parameters and initial conditions.*

The time and horizontal and vertical components of the position, velocity and acceleration for each instant in time were stored in a two dimensional matrix. Two dimensional plotting of the quantities described led to the figures in the following discussion.



Figures 2, 3: *Position vs. Time.*

Figures 2 and 3 illustrate the position of the skydiver between the jump and the landing. Contrary to the motion of a regular projectile without air resistance, the horizontal position quickly converges asymptotically to a value of about 350m from the jump site. In approximately 20 seconds, the skydiver virtually stops moving horizontally. On the other hand, the altitude does not vary very rapidly until about 10 seconds after clearing the airplane. This coincides with the moment at which the speed of the skydiver can be considered has reached the value of terminal velocity (figure 5), even though it is actually an asymptotic approximation. The discontinuity at about 60s after the jump corresponds to the deployment of the parachute. About 2.5 minutes after that, the skydiver lands safely. A failure in the deployment mechanism would only allow for additional 20 seconds for the deployment of the emergency parachute.

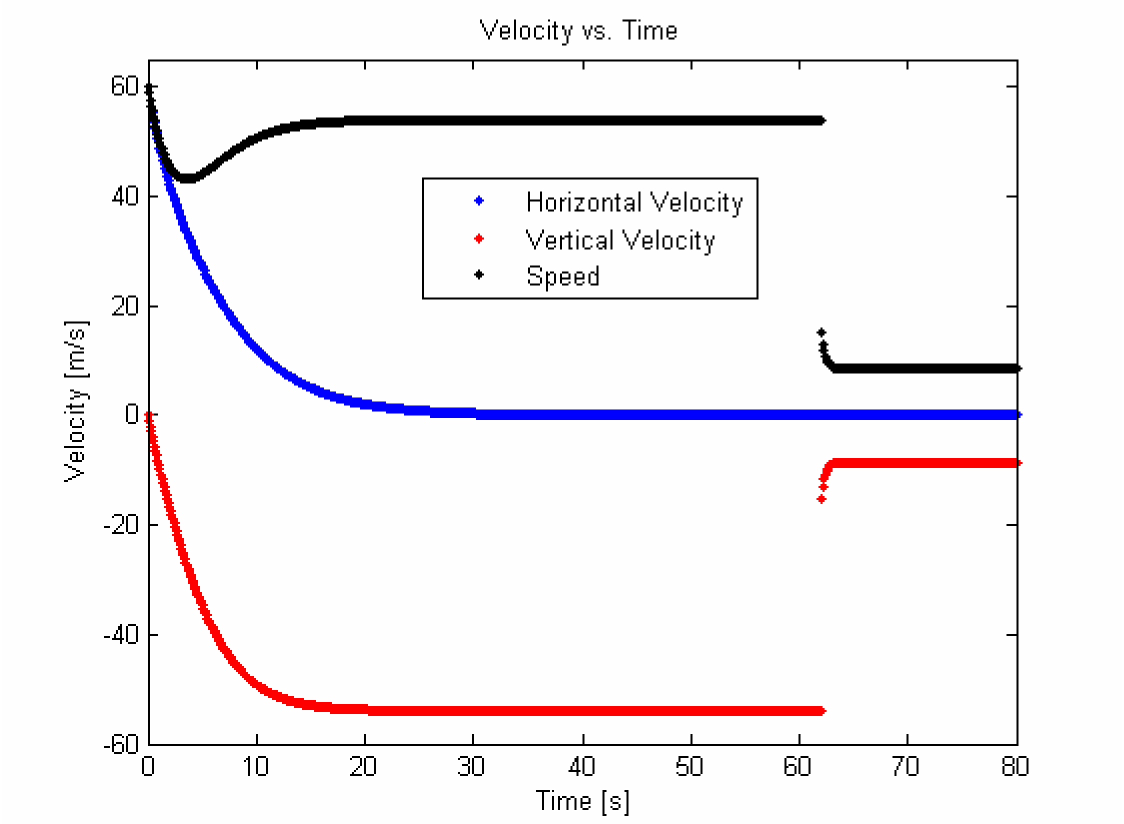
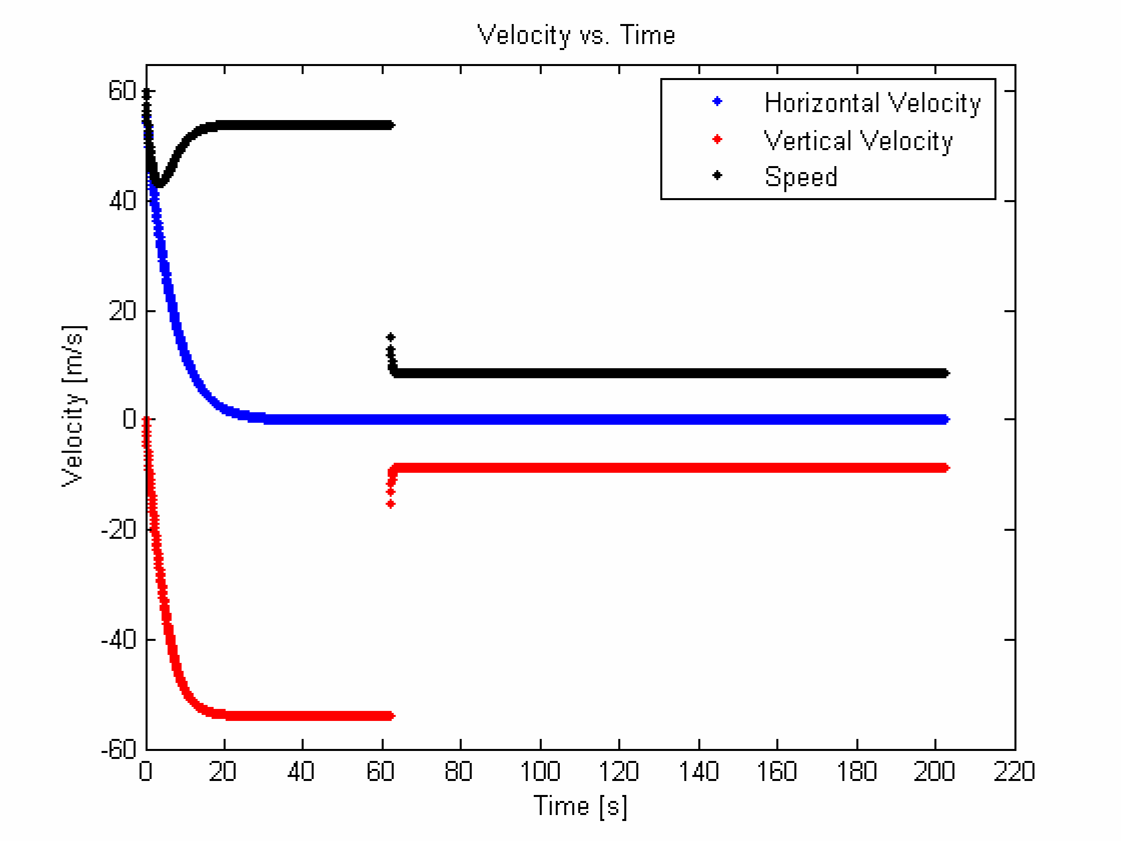


Figure 4: *Velocity vs. Time (initial 80 seconds).*



Figures 5: *Velocity vs. Time (full range).*

Figures 4 and 5 show velocity vs. time. It can be seen that the vertical terminal velocity is approximately 55m/s (107kts). At this moment, the drag force and the weight are the equal in magnitude. This value is consistent with the literature. As expected, the horizontal velocity approaches zero, since the skydiver starts with the horizontal velocity of the plane (60m/s) and then is submitted to air resistance. Due to the lack of any force, other than the opposing air resistance and weight, the skydiver virtually stops moving horizontally about 30 seconds later. After about 60 seconds, the parachute is suddenly deployed, implying a drastic reduction of the vertical velocity. However, the figure does not show an instantaneous change; essentially, the skydiver reaches a new terminal velocity, since the effective cross sectional area increases by a factor of 40 in less than 2 seconds. The horizontal velocity remains the same, as expected. If a modern canopy (rectangular) were considered in this analysis, the horizontal velocity would likely change, as the open canopy would glide. The skydiver safely lands at a comfort speed of about 7m/s.

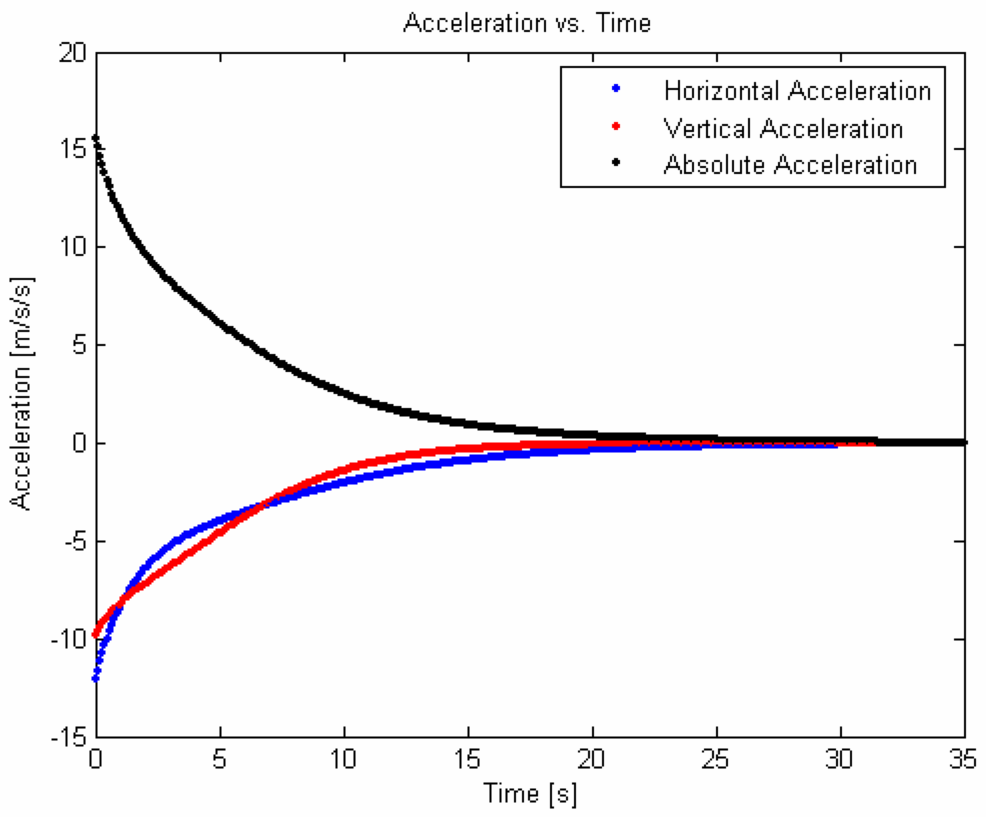


Figure 6: *Acceleration vs. Time (initial 35 seconds).*

The acceleration vs. time curve reveals that the skydiver might feel a free falling sensation during the first 5 or 6 seconds. Surprisingly enough, the horizontal acceleration due to the air resistance as the skydiver leaves the airplane initially exceeds that of gravity. At about 1 second after leaving the plane, the horizontal acceleration drops below the vertical. This is due to the fact that at that time, the horizontal velocity is much greater than the vertical. Hence, the angle of attack is minimal, and the drag force is mostly exerted in the horizontal direction. As the skydiver acquires vertical speed, the air resistance starts to have a more serious effect in the vertical direction. Both accelerations converge to zero as the terminal velocity is reached about 20 seconds after jumping.

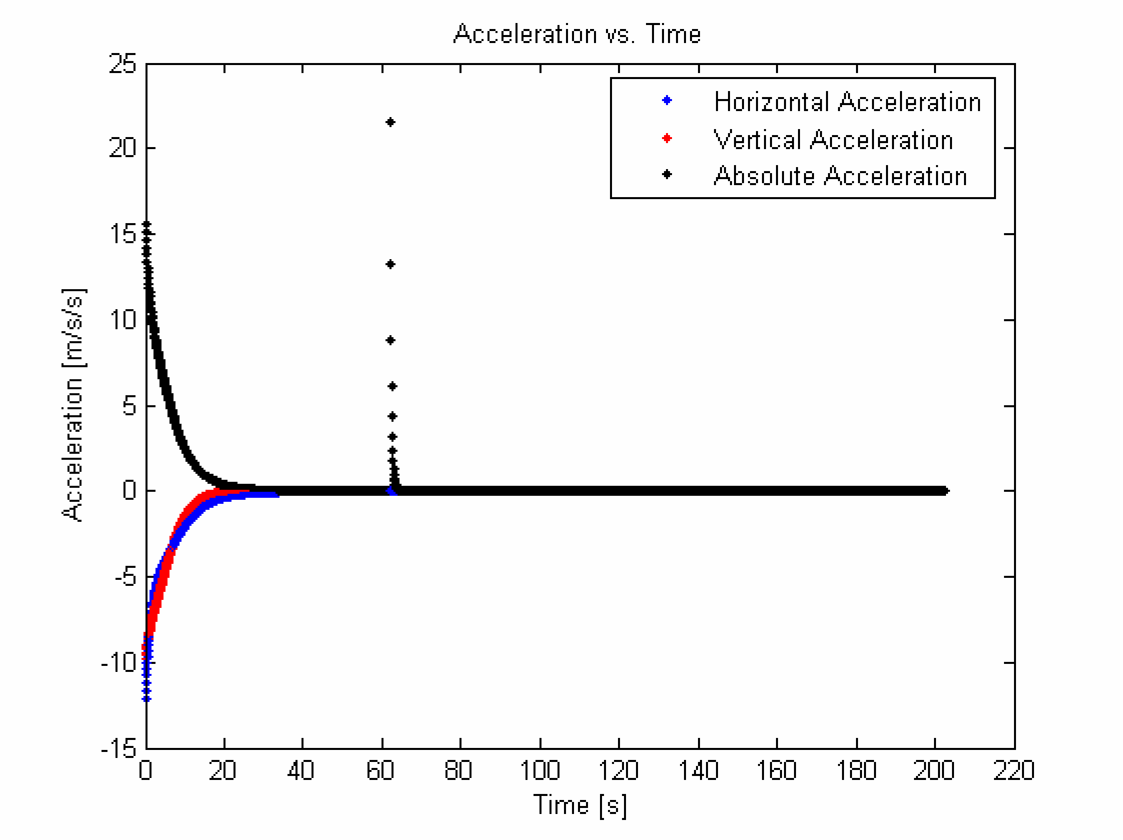
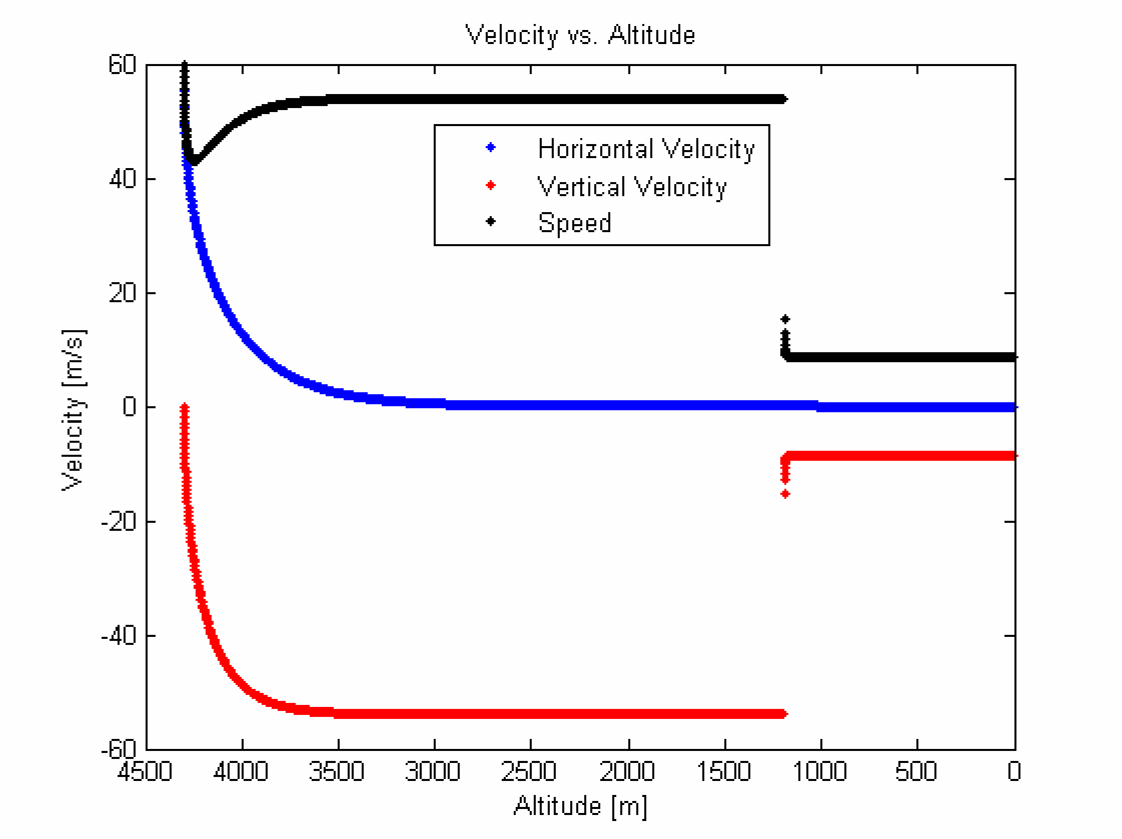


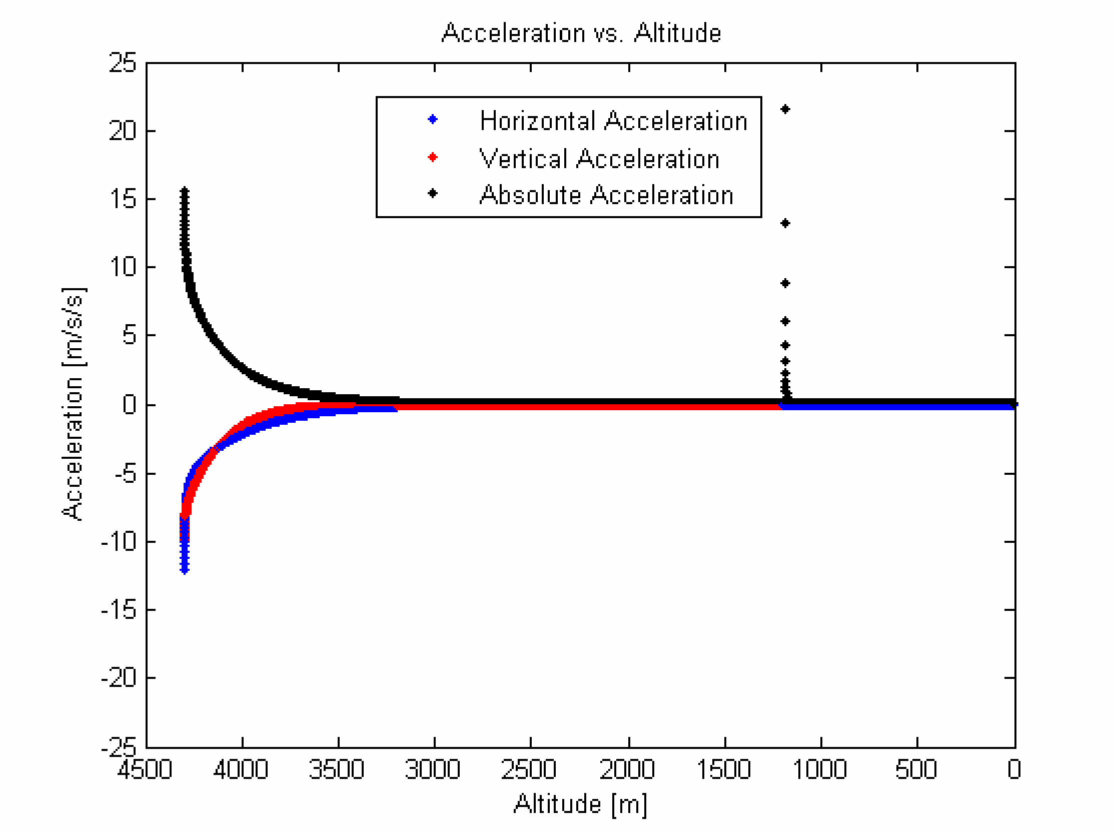
Figure 7: *Acceleration vs. Time (full range).*

As the parachute is deployed, the surface area of the body in question increases by a factor of 40. In a more realistic case, the semi-smooth opening of the canopy should be taken into account. Nevertheless, in this simplified simulation it can be seen how the vertical acceleration (total by that time) suddenly increases to its maximum, of about 25m/s/s, approximately 2.5 g's. For a body of mass 80kg, this corresponds to 2000 N of force. If, for instance, a flat strap in the harness of approximately 1 by 40 mm is submitted to this level of tensile force, it would have to be able to support a normal stress of at least 50 kPa. This level of normal stress is considerable, and it is a clear reason why the design and choice of materials for skydiving gear is so important.

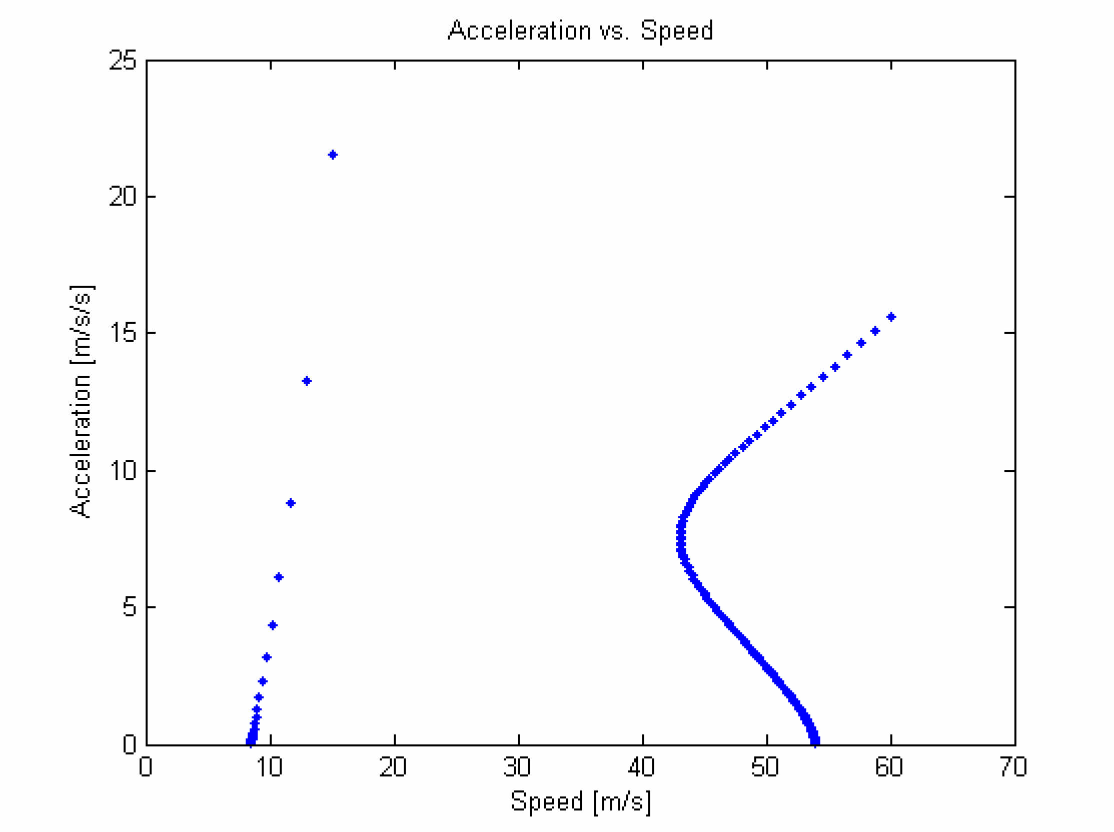


Figures 8: *Velocity vs. Altitude.*

Figure 8 shows the velocity vs. altitude curves, which resemble the velocity vs. time curves. This is due to the fact that the altitude follows mostly a linear behavior over time. Furthermore, the acceleration vs. altitude curves (figure 9) resemble the acceleration vs. time curves due to the same reason.



Figures 9: *Acceleration vs. Altitude.*



Figures 10: *Absolute Acceleration vs. Speed.*

Figure 10 depicts the behavior of the acceleration relative to the speed. It can be noted that the acceleration is linearly proportional to the velocity except for when the terminal velocity is reached, in which case the curve is rounded. Moreover, the feature connecting the two straight lines on the right part of the figure corresponds to the critical change in the angle of attack, also reflected in figure 6 by the crossing of the horizontal and vertical components of the acceleration vs. time curves.

4 Conclusions

The simulation code was developed and run successfully. The results obtained by considering an idealized skydiver matched the expected results and the values found in the literature quite well. It was shown that the point of maximum acceleration, and therefore maximum stress in the gear, occurs with the deployment of the parachute. The horizontal motion is stopped a few seconds after leaving the airplane. The vertical velocity reaches a terminal velocity in essentially less than 20 seconds. Furthermore, examination of the results shows the importance of a good design of the skydiving gear in order to prevent fatalities.

5 Further work

Even though the idealized simulation can be considered a fairly accurate and precise representation of the real case, enhancements could be performed in order to add features to the simulation. For instance, a more elaborate program could consider winds, semi-smooth deployment of the parachute and the behavior of a modern canopy.

6 References

[1] Marion, J. B., Thornton, S. T, *Classical Dynamics of Particles and Systems (4th Ed.)*, Harcourt College Publishing, 1995.

[2] Cameron, J., *Numerical Simulation of an Idealized Skydiver in Freefall*, Illinois Institute of Technology (MMAE 505), 2001.

[3] Halliday, D., Resnick, R., Walker, J., *Fundamentals of Physics (6th Ed. Extended)*, John Wiley & Sons, Inc., 2001.

[4] Kallend, J., *Notes on the Freefall Trajectory Simulation*, Illinois Institute of Technology (online): http://www.iit.edu/~kallend/skydive/freefall.html [05.05.06]

[5] Barton, S., *Equations of the movement - modeling of the physical process* (online): http://adept.maplesoft.com/categories/science/astrophysics/html/eqofmov.html [05.05.06].

[6] Fabulousrocketeers.com, Skydiving FAQ (online): http://www.fabulousrocketeers.com/Photo_See_Ya.htm [05.05.06]

[7] Hypertextbook.com, *The Speed of a Skydiver (Terminal Velocity)*, online: http://hypertextbook.com/facts/JianHuang.shtml [05.05.06].

Appendix: Program Code

Main script:

```
%------------------------------------------------------------------------%
% --------------------------**-SKYDIVER-**-------------------------------%
% Program: Skydiver                                                      %
% Developer: Nil Valls                                                   %
% Institution: Illinois Institute of Technology                          %
% Date: May 2006                                                         %
% Course: Physics 309                                                    %
% Instructor: Prof. Johnson                                              %
%------------------------------------------------------------------------%
% This program simulates an idealized skydiver in free fall. It can      %
% calculate the components of the postion, velocity acceleration and     %
% drag force at different times by iteration. The following functions    %
% are called in this code: "position","velocity_xy","accel_xy".          %
%------------------------------------------------------------------------%



% Clear desktop, figures and close all figures (plots).
clc;
clear all;
close all;

% Define physical parameters of the skydiver and medium (air) in SI units.
drag_coeff=0.7;
cross_section=0.7;
density=1.1;
mass=80;
gamma=0.5*drag_coeff*cross_section*density/mass; % just for convenience.

% Declare and fill the data matrix with zeros. The first index corresponds
% to the number of iterations (allowed). The second index is associated
% with the different physical quantities, in order: x-position, y-position,
% x-velocity, y-velocity, x-acceleration, y-acceleration, time.
motion=zeros(600,7);

% Define initial conditions and simulation parameters in SI units.
dt=0.1;           % Time step in seconds.
x_i=0;            % Origin x-coordinate.
y_i=4300;         % Origin y-coordinate.
vel_x_i=60;       % Initial x-velocity (airplane speed).
vel_y_i=0;        % Initial y-velocity.
t=0;              % Original time.
deploy_alt=1200;  % Canopy deployment altitude.
chute_surf=30;    % Canopy surface area.
new_density=1.25; % Density of air at deployment altitude.
gamma_deploy=0.5*drag_coeff*chute_surf*new_density/mass;

i=1;             % First iteration assigment.
```

(Continues on the next page).

(Continued from previous page).

```
% Start loop, stop when the skydiver reaches the ground (y~0m).
while (y_i>0)

    if(y_i<deploy_alt)  % Deployment of the chute at "deploy_alt".
        gamma=gamma_deploy;
    end

    motion(i,7)=t;   % Store time.
    motion(i,1)=x_i; % Store x coordinate.
    motion(i,2)=y_i; % Store y coordinate.
     % Calculate new coordinates (after dt).
    [x,y]= position (x_i,y_i,vel_x_i,vel_y_i, dt, gamma);
    x_i=x;
    y_i=y;
    vel_y_i=vel_y;


    motion(i,3)=vel_x_i; % Store x-velocity.
    motion(i,4)=vel_y_i; % Store y-velocity.
     % Calculate new velocity and acceleration (after dt).
    [vel_x,vel_y, accel_x, accel_y]=velocity_xy(vel_x_i,vel_y_i,dt,gamma);
    motion(i,5)=accel_x; % Store x-acceleration.
    motion(i,6)=accel_y; % Store y-acceleration.

    t=t+dt; % Increment time by dt.
    vel_x_i=vel_x;

    % Plot datapoints.
     plot (motion(i,7),motion(i,1),'.c'); % "x" vs. time.
     plot (motion(i,7),motion(i,2),'.k'); % "y" vs. time.
     plot (motion(i,7),motion(i,3),'.b'); % "x-vel" vs. time.
     plot (motion(i,7),motion(i,4),'.r'); % "y-vel" vs. time.
     plot (motion(i,7),motion(i,5),'.g'); % "x-accel" vs. time.
     plot (motion(i,7),motion(i,6),'.m'); % "y-accel" vs. time.

     hold on; % Plot all datapoints in the same figure
     xlabel (' time [s]'); % Label ordinates axis.
     i=i+1; % Iteration counter


end

i  % Print out total number of iterations.
```

Position function:

```
function [x,y] = position (x_i, y_i, vel_x_i, vel_y_i, dt, gamma)
% -----------------------------------------------------------------%
% Program: Skydiver                                                %
% Developer: Nil Valls                                             %
% Institution: Illinois Institute of Technology                    %
% Date: May 2006                                                   %
% Course: Physics 309                                              %
% Instructor: Prof. Johnson                                        %
%------------------------------------------------------------------%
% This function computes and returns the coordinates of the        %
% skydiver at particular instants in time. The "velocity_xy"       %
% function is called in this code.                                 %
%------------------------------------------------------------------%

% Calculate initial velocity and acceleration components.
[vel_x,vel_y,accel_x,accel_y]=velocity_xy(vel_x_i,vel_y_i,dt,gamma);

% Compute new coordinates given previous position "x_i", "y_i"and
% time step "dt".
x = x_i + vel_x_i*dt + 0.5*accel_x*(dt^2) ;
y = y_i + vel_y_i*dt + 0.5*accel_y*(dt^2) ;

end
```

Velocity function:

```
function [vel_x,vel_y,accel_x,accel_y] =
velocity_xy(vel_x_i,vel_y_i,dt,gamma)
% -------------------------------------------------------------%
% Program: Skydiver                                            %
% Developer: Nil Valls                                         %
% Institution: Illinois Institute of Technology                %
% Date: May 2006                                               %
% Course: Physics 309                                          %
% Instructor: Prof. Johnson                                    %
%--------------------------------------------------------------%
% Function for the instantaneous velocity vector given the     %
% velocity (vel_x_i, vel_y_i) in the previous instant. The     %
% time increment is given by "dt", i.e. the iteration step     %
% size.                                                        %
%--------------------------------------------------------------%

% Since the instantaneous velocity is dependent on the
% instantaneous acceleration, call the acel_xy function and
% obtain the acceleration components.
[accel_x , accel_y ] = accel_xy (vel_x_i,vel_y_i,gamma ) ;

% Compute velocity components considering a small time
% increment "dt".
vel_x = vel_x_i + accel_x * dt ;
vel_y = vel_y_i + accel_y * dt ;

end
```

Acceleration function:

```
function [accel_x, accel_y] = accel_xy (vel_x_i,vel_y_i,gamma)

% -----------------------------------------------------------%
% Program: Skydiver                                          %
% Developer: Nil Valls                                       %
% Institution: Illinois Institute of Technology              %
% Date: May 2006                                             %
% Course: Physics 309                                        %
% Instructor: Prof. Johnson                                  %
%------------------------------------------------------------%
% This function computes and returns the acceleration vector %
% given the instantaneous velocity components and the        %
% gamma parameter, which includes the mass and cross-        %
% sectional area of the skydiver, the density of air and     %
% drag coefficient.                                          %
%------------------------------------------------------------%

% Declaration of the acceleration due to gravity (m/s/s).
grav = 9.81 ;

% Compute magnitude of velocity vector.
mag_vel = sqrt(vel_x_i^2 + vel_y_i^2) ;

% Compute trigonometric paramenters. Theta is the angle of
% the velocity vector with the horizontal.
cos_theta = abs(vel_x_i / mag_vel) ;
sin_theta = abs(vel_y_i / mag_vel) ;

% Finally compute the components of the acceleration vector,
% which will be returned by this function.
accel_x = -gamma * ( mag_vel^2 ) * cos_theta ;
accel_y = gamma * ( mag_vel^2 ) * sin_theta - grav ;

end
```

All code written in MatLab® 7.0.5 for Linux.