Dynamic Soaring: Assignment 5

Izza (AE18B006) &

RV Ramkumar (AE20D025)

The Assignment is to find the trajectory of a sphere ($C_D = 1.0$, $m = 0.160kg$, $r = 0.225m$) which is falling from rest under gravity ($g = 0.376 \times 9.81$, $\rho = 1.225/50\ kg/m^3$) from a height of 100 m with zero wind and with varying wind using pseudo-spectral method.

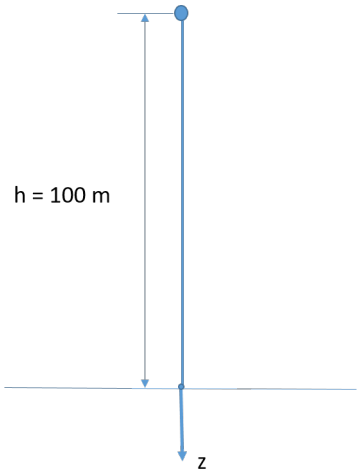

Figure 1: Indicative Geometry of the Problem

Equations of Motion

The speed of the sphere with respect to air is given by $V = \dot{z} + w_z$ where V is the air speed, w_z is the vertical wind velocity. Therefore the equation of motion is given by:

$$m\frac{d^2z}{dt^2} = m\frac{d(V - w_z)}{dt} = mg - C_D\frac{1}{2}\rho V^2 S \tag{1}$$

Above equation can be written as

$$\begin{aligned}\frac{dV}{dt} &= g - C_D\frac{1}{2}\rho V^2\frac{S}{m} + \frac{dw_z}{dt} \\ &= g - C_D\frac{1}{2}\rho V^2\frac{S}{m} + \frac{dw_z}{dz}\dot{z}\end{aligned}$$

In state space form, the equation of motion can be written as:

$$\begin{bmatrix} \dot{z} \\ \dot{V} \end{bmatrix} = \begin{bmatrix} V - w_z \\ g - C_D\frac{1}{2}\rho V^2\frac{S}{m} + \frac{dw_z}{dz}\frac{dz}{dt} \end{bmatrix} = \begin{bmatrix} V - w_z \\ g - C_D\frac{1}{2}\rho V^2\frac{S}{m} + \frac{dw_z}{dz}(V - w_z) \end{bmatrix} \tag{2}$$

The above equations to be solved using the pseudo spectral method for the different conditions of the parameters (viz., without drag, with drag and varying wind velocities). A brief review on solving the above differential equation using the pseudo-spectral method is explained here under.

Pseudo-Spectral Method

The basic idea of pseudo-spectral method which is also called as spectral collocation method for solving differential equations is to approximate the time varying function as a weighted sum of the polynomials (basis

function) at the collocation points. Supposing there are a total of N+1 collocation points in the interval $[-1, 1]$ and the function is also defined in the interval $[-1, 1]$, then the function can be approximated by a polynomial of degree N, i.e.,

$$f(t) \approx P_N(t) = \sum_{j=1}^{N+1} \phi_j(t) f(t_j) \tag{3}$$

where $\phi_j(t)$ the Lagrange interpolation Polynomial which is expressed as

$$\phi_j(t) = \prod_{i=1 i \neq j}^{N+1} \frac{t - t_i}{t_j - t_i} \tag{4}$$

$$\phi_j(t_i) = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{if } j \neq i \end{cases} \tag{5}$$

Further, from the above expressions, it is also clear that the polynomial approximation satisfies the function at every point, i.e., $f(t_j) = P_N(t_j)$. The derivative of the function at every node can be represented by the expression

$$f'(t_i) = \sum_{j=1}^{N+1} \left[\frac{d}{dt} \phi_j(t) \right]_{t=t_i} f(t_j) \tag{6}$$

In view of the above, the numerical differentiation can be performed by means of matrix vector product, i.e.,

$$f'(t_i) = \sum_{j=1}^{N+1} D_{ij} f(t_j) \tag{7}$$

For the equation (3), the N+1 collocation points in the interval $[-1, 1]$ can be obtained by using Legendre polynomial. Supposing the Legendre polynomial of order N is denoted as $L_N(t)$ and the zeros of the first derivative of the Legendre polynomial i.e., $(\dot{L}_N(t) = 0)$ are considered as the nodes $(t_i, i = 2, 3, ...N)$ including the end points of $t_1 = -1$ and $t_{N+1} = 1$, these nodes or points are known as Legendre-Gauss-Lobatto (LGL) collocation points. The recursion formula of the Legendre Polynomial is given as

$$(N+1) L_{N+1}(t) = (2N+1) t L_N(t) - N L_{N-1}(t) \tag{8}$$

With $P_0(t) = 1$ and $P_1(t) = t$, the next higher order polynomials can be obtained by using the recursive relation given in equation (8). The differentiation matrix D_{ij} of size (N+1) x (N+1), if L-G-L collocation points are used is given by

$$D_{ij} = \begin{cases} \frac{L_N(t_i)}{L_N(t_j)} \frac{1}{(t_i - t_j)} & i \neq j \\ \frac{-(N)(N+1)}{4} & i = j = 0 \\ \frac{(N)(N+1)}{4} & i = j = N \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

To start the solving of the differential equations, the time scaling to be used in order convert the domain $[t_0, t_f]$ to the interval $[-1, 1]$,

$$t = \frac{t_f - t_0}{2} \tau + \frac{t_f + t_0}{2}$$

and

$$dt = \frac{t_f - t_0}{2} d\tau$$

Therefore, the equations of motion can be written in the transformed time τ,

$$\begin{bmatrix} \dot{z} \\ \dot{V} \end{bmatrix} = \begin{bmatrix} \frac{dz}{d\tau}\frac{d\tau}{dt} \\ \frac{dV}{d\tau}\frac{d\tau}{dt} \end{bmatrix} = \begin{bmatrix} V - w_z \\ g - C_D\frac{1}{2}\rho V^2\frac{S}{m} + \frac{dw_z}{dz}\frac{dz}{dt} \end{bmatrix} = \begin{bmatrix} V - w_z \\ g - C_D\frac{1}{2}\rho V^2\frac{S}{m} + \frac{dw_z}{dz}(V - w_z) \end{bmatrix}$$

The above equation for the τ interval between $[-1, 1]$ is given by

$$\frac{dX}{d\tau} = f(X,\tau) = \begin{bmatrix} \frac{dz}{d\tau} \\ \frac{dV}{d\tau} \end{bmatrix} = \begin{bmatrix} \frac{(t_f - t_0)}{2}(V - w_z) \\ \frac{(t_f - t_0)}{2}\left[g - C_D\frac{1}{2}\rho V^2\frac{S}{m} + \frac{dw_z}{dz}(V - w_z)\right] \end{bmatrix} \tag{10}$$

It is pertinent to note that the value of z and V need to be obtained at each of the time instant (at time $(N+1)$ collocation time instants). If it is considered that both the z and V are discretized and formed as a single vector i.e., $f = (z_1, z_2, ..., z_{N+1}, v_1, v_2, ..., v_{N+1})$, then the equation using the differentiation matrix becomes,

$$X_i = \sum_{j=1}^{2(N+1)} D_{ij} f_j \tag{11}$$

It is also important that the equation also satisfies the rhs of the equation 10 at every time instant. Considering a constant (may be unity) as an objective function with the equations of motion as nonlinear constraint, it is possible to identify the value of the X_i at every node as a dynamic optimization problem. i.e.,

$$\text{Min} \quad J = c \tag{12}$$

$$\text{subject to} \frac{dX}{d\tau} = f(X,\tau)$$

with known initial conditions.

From the above, it can be noticed that the final time which is an unknown (t_f) also need to be estimated. Knowing the physics of the problem, it is possible to define an upper limit for t_f so that optimization of the unknown function may be initialized with the known initial conditions for the vector. The "*fmincon*" function available in Matlab can be utilized to solve this problem. The upper and lower bound for each of the variables also to be defined to use the *fmincon* function.

Organization of the Code

The Matlab code to solve the problem is organized as follows:

1. Main function (main.m)

   This function sets up the problem, define the initial condition, define the bounds of the variables, guess values, etc., and takes the output from the matlab "fmincon" and plots the results

2. Equations of Motion function (eqnmotion.m)

   This function takes the values of τ and X_i, and outputs the rhs of the derivative of the function. Further in this function the various cases are simulated.

3. Constraints function (odeconstraints.m)

   This constraint function evaluates the difference between the $\frac{dX}{d\tau} - f(X,\tau)$ considering the vector z which is from 1 to $(N+1)$ and V from $(N+2)$ to $(2N+3)$. In case if N is the no. of collocation points, then the polynomial is of the order N-1 and the z or V will be from 1 to N and V or z will be from $N+1$ to $2N$. The t_f which is the time of flight is also appended below of the same vector effectively making the length of f as $2N+1$.

4. Objective function (costfun.m)

   This function is the cost / objective function which to be evaluated.

5. Differentiation Matrix function (legDc.m)

   This function takes in the degree of the Legendre polynomial (N) as input. Evaluates the collocation points (time instants) between the interval $[1, -1]$ by finding the zeros of the Legendre polynomial and finds the $(N+1) \times (N+1)$ Differentiation Matrix.

Matlab Code

main.m

```
clear all; close all; clc;
global Dmat t0 hfinal N g tau hstart

%-------------------------------------------------------------
% tau is the legendre knots / nodes in the interval 1 to -1
% Dmat is the Differentiation matrix
% No. of collocation points is N => degree of polynomial = N-1;
%-------------------------------------------------------------

g=0.376*9.81;      %acceleration due to gravity
N=30;              % No of collocation points

[tau,Dmat] = legDc(N-1);    % N-1 is the degree of polynomial
sum(Dmat(1,:));      % just to check the sum of all the columns in a row is zero;

t0 = 0;
hfinal = 100;   % changed because of tau from 1 to -1 instead of -1 to 1;
hstart = 0;     % changed because of tau from 1 to -1 instead of -1 to 1;

ALE = [];
bLE = [];
ALIE = [];
bLIE = [];

%decision variable Cost = [V(1 to N) Z(N+1 to 2N) tf(2N+1)]

limits=[0 1000;  %w
    -100 1000; %z
    0 50]; %tf
for i=1:2
    Lb(N*(i-1)+1:i*N) = limits(i,1)*ones(1,N);  % lower bound at every node
    Ub(N*(i-1)+1:i*N) = limits(i,2)*ones(1,N);  % upper bound at every node
end
Lb(2*N+1) = limits(3,1);
Ub(2*N+1) = limits(3,2);

% Total number of variables to be optimized.
% velocity at each N nodes; height at each N node and 1 for tfinal
% N+N+1 = 2*N+1;

totalvars=2*N+1;     % no of varibales which is 2*N+1;

tfg = sqrt(2*abs(hfinal)/g);   % initial guess of the time of flight;
tvec = (0+tfg)/2 + tfg/2*tau;  % mapping from 0 to tf to -1 to 1
z_guess = 0+1/2*g*tvec.^2;     % guess of the vertical height z at each node
```

```
V_guess = g*tvec;              % velocity guess at each node

X_init = [V_guess;z_guess;tfg];

Aeq=zeros(totalvars);
Beq=zeros(totalvars,1);
Aeq(N,N)=1;     Aeq(N+1,N+1)=1;         Aeq(2*N,2*N)=1;
Beq(N)=0;       Beq(N+1)= hfinal;       Beq(2*N) = hstart;

% costfun=@(x)(1);

[Z,costval,exitflag,output] = fmincon(@costfun,X_init,[],[],Aeq,Beq,Lb,Ub,@odeConstraints);
Tf = Z(end)
optk = Z(2*N+1:2*N)

tvecnew = (0+Tf)/2 + (Tf)/2*tau;    % reconversion to the physical time domain

% Solution through the Ode-45 for comparison between the time interval

[tsol,xsol] = ode45(@eqnmotion,flip(tvecnew),[0;0]);

%Plotting the output

figure(1);
plot (tvecnew,Z(1:N),'r+-',tsol,xsol(:,1),'k-');
fs='FontSize'; set (gca,fs,15);
xlabel('Time(sec)',fs,15); ylabel('Rel. Velocity (m/s)',fs,15);
legend ('PS method','ODE-45','Location','SouthEast'); grid on;
xlim ([0 10]); ylim ([0 30]);
title('Case 1 - Without Drag');
print -depsc 'vel_case1.eps'

figure(2)
plot (tvecnew,hfinal-Z(N+1:2*N),'r+-',tsol,hfinal-xsol(:,2),'k-');
fs='FontSize'; set (gca,fs,15);
xlabel('Time(sec)',fs,15); ylabel('Height (m)',fs,15)
legend ('PS method','ODE-45'); grid on;
xlim ([0 10]); ylim ([-20 120]);
title ('Case 1 - Without Drag');
print -depsc 'z_case1.eps'
```

eqnmotion.m

```
function xdot = eqnmotion(t,x)
global g hfinal hstart

%-----------------------------------------------
% Case 1: Cd = 0.0;
% Case 2: Cd = 1.0;
% Case 3: Cd = 1.0 with variation of vertical wind profile
% Case 4: To find the optimal wind to maximize tf
%-----------------------------------------------
Cd = 0.0;       % Case 1;
wz = 0.0;   dwdz = 0.0;       % Case 1: wind speed;
% Cd = 1.0;       % Case 2;
% wz = 0.0;   dwdz = 0.0;       % Case 2: wind speed;

% Cd = 1.0;          % Case 3;
% if ((hfinal-x(2)) < 30)
%     dwdz = -0.1;
```

```
%     wz = dwdz*(hfinal-x(2));
% else
%     dwdz = 0.0;
%     wz = 3.0;
% end

m=0.160;
S=pi*0.225^2;
rho = 1.225/50;
xdot = zeros(2,1);
xdot(1) = g - 0.5*rho*S/m*Cd*(x(1)).^2 + dwdz*(x(1)-wz);
xdot(2) = x(1)-wz;
end
```

odeconstraints.m

```
function [c,ceq] = odeConstraints(Z)
global Dmat t0 N tau k

%indexing as a single column vector for V, h and tf;
% 1 to N corresponds to V
% N+1 to 2*N corresponds to Z
% 2*N+1 is the tf;


V = Z(1:N);
h = Z(N+1:2*N);
tf =Z(2*N+1);
k = Z(2*N+1:2*N);

ceq = zeros(2*N,1);

tvec = (t0+tf)/2 + (tf-t0)/2*tau;
VRHS = zeros(N,1);
hRHS = zeros(N,1);
for i=1:N
    dx = eqnmotion(tvec(i),[V(i);h(i)]);
    VRHS(i) = dx(1);
    hRHS(i) = dx(2);
end

ceq(1:N) = Dmat*V - 0.5*(tf-t0)*VRHS;
ceq(N+1:2*N) = Dmat*h - 0.5*(tf-t0)*hRHS;

c=[];
end
```

costfun.m

```
function [cost] = costfun(Z)
cost = 1;
end
```

legDc.m

```
function [x,D]=legDc(N);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
```

```
% legDc.m
%
% Computes the Legendre differentiation matrix with collocation at the
% Legendre-Gauss-Lobatto nodes.
%
% Reference:
%   C. Canuto, M. Y. Hussaini, A. Quarteroni, T. A. Tang, "Spectral Methods
%   in Fluid Dynamics," Section 2.3. Springer-Verlag 1987
%
% Written by Greg von Winckel - 05/26/2004
% Contact: gregvw@chtm.unm.edu
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Truncation + 1
N1=N+1;

% CGL nodes
xc=cos(pi*(0:N)/N)';

% Uniform nodes
xu=linspace(-1,1,N1)';

% Make a close first guess to reduce iterations
if N<3
    x=xc;
else
    x=xc+sin(pi*xu)./(4*N);
end

% The Legendre Vandermonde Matrix
P=zeros(N1,N1);

% Compute P_(N) using the recursion relation
% Compute its first and second derivatives and
% update x using the Newton-Raphson method.

xold=2;
while max(abs(x-xold))>eps
    xold=x;
    P(:,1)=1;    P(:,2)=x;
    for k=2:N
        P(:,k+1)=( (2*k-1)*x.*P(:,k)-(k-1)*P(:,k-1) )/k;
    end
    x=xold-( x.*P(:,N1)-P(:,N) )./( N1*P(:,N1) );   % Newton Raphson Method
end

X=repmat(x,1,N1);
Xdiff=X-X'+eye(N1);

L=repmat(P(:,N1),1,N1);
L(1:(N1+1):N1*N1)=1;
D=(L./(Xdiff.*L'));
D(1:(N1+1):N1*N1)=0;
D(1)=(N1*N)/4;
D(N1*N1)=-(N1*N)/4;
```

Simulation Results

The simulation is performed for various cases as per the assignment. Further, the results obtained by using the pseudo spectral method is compared with those obtained by using the ODE-45 (RK-4) solver of Matlab.

1. Sphere under free fall without Drag.
2. Sphere under free fall with Drag
3. Sphere under free fall with Drag and Varying Updraft

Relative Velocity Vs Time for various Cases

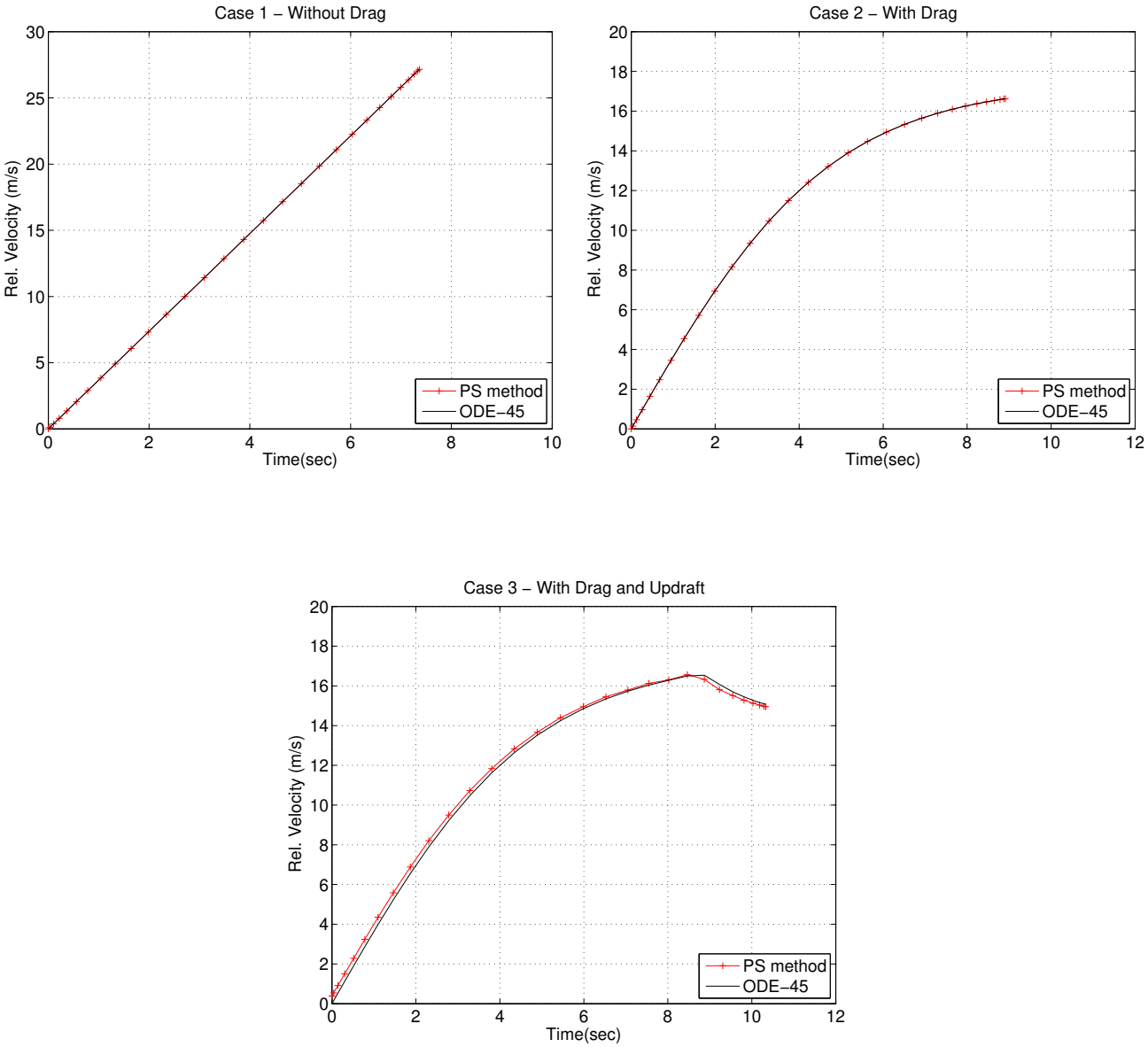

Figure 2: Relative Velocity Vs Time for Various Cases

Height Vs Time for Various Cases

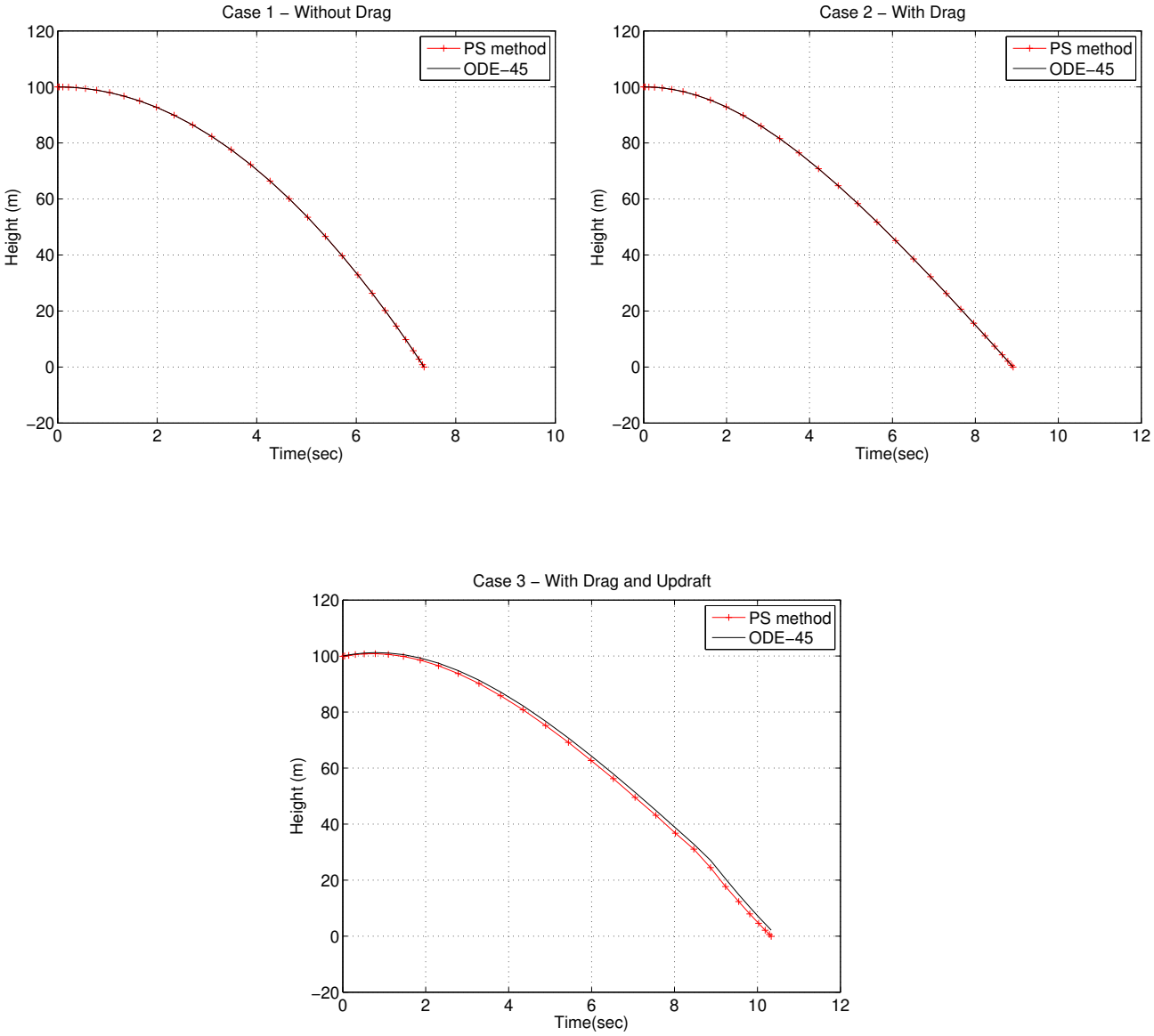

Figure 3: Height Vs Time for Various Cases

The total time of flight for the three cases are summarized and indicated in the table 1 below.

Table 1: Time of Flight for Various cases

Cases	Time of Flight (sec)
Case 1	7.37
Case 2	8.91
Case 3	10.33