

Question 1:

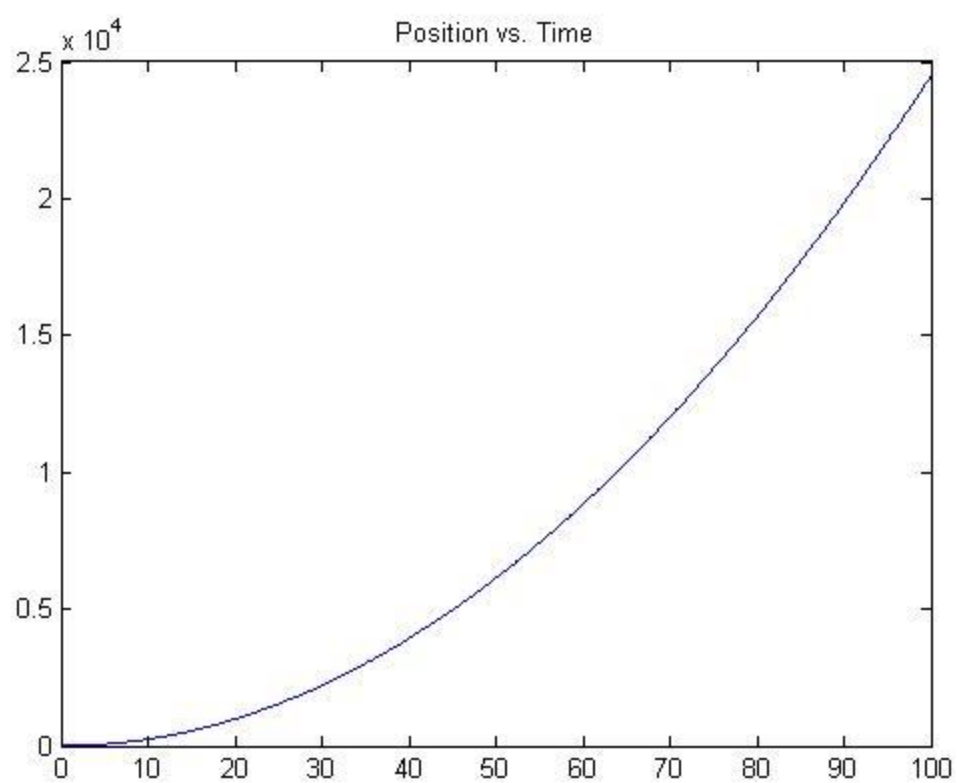
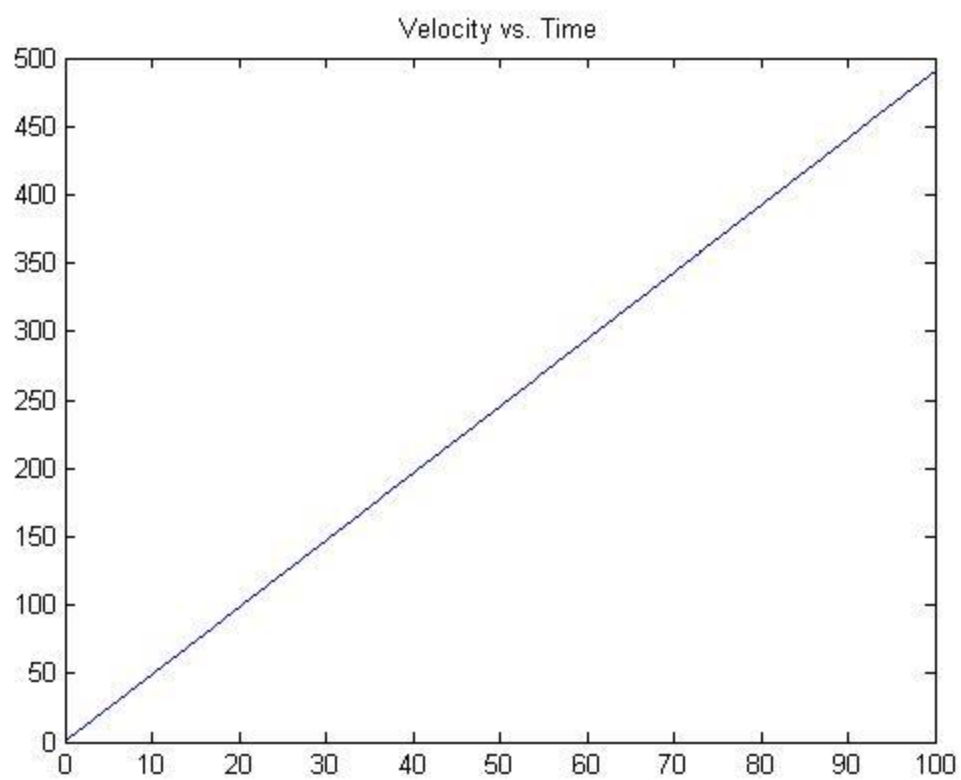
code:

```
clear;close all;
global a;
timescale=10;
dt=timescale/100;
tstart=0;
tfinal=10*timescale;
theta=30*pi/180;
g=9.8;
a=g*sin(theta);
u0=zeros(2,1);
u0(1)=0; % initial position;
u0(2)=0; % initial velocity % when we change it We get different graph.
[t,u]=ode45(@rhsq1,tstart:dt:tfinal,u0);
x=u(:,1); % radian-->degree
v=u(:,2);
plot(t,x)
title('Position vs. Time')
figure
plot(t,v)
title('Velocity vs. Time')
figure
plot(t,a)
title('acceleration vs. Time')
```

RHS:

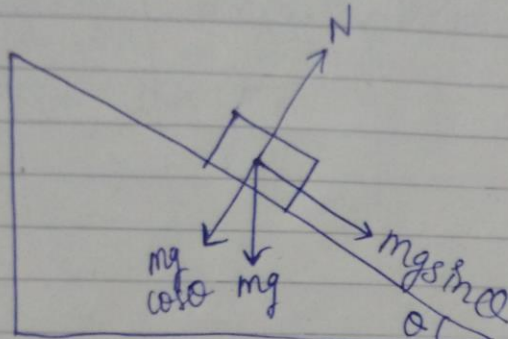
```
function F=rhsq1(t,u)
global a;
F=zeros(length(u),1);
F(1)=u(2);
F(2)=a;
```

graph:



Analytically :

1)



$$mg \cos \theta = N$$

$$F = mg \sin \theta$$

$$a = g \sin \theta$$

$$\frac{dv}{dt} = g \sin \theta$$

$$V_0 = \text{initial velocity}$$

$$s \rightarrow \text{displacement}$$

Here $\theta = 30^\circ$

$$\therefore V = gt \sin \theta + V_0$$

$$s = \frac{1}{2} gt^2 \sin \theta + V_0 t$$

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

→ Hence nature of the graph of $v \rightarrow t$ is straight line with slope $g \sin \theta$.

→ Nature of the graph of $s \rightarrow t$ is parabola.

Here , by changing initial parameters i.e. 1] changing angle of inclination , its velocity will increase. So in its graph slope would increase and graph of displacement vs time shrinks. 2] changing initial velocity , graph of velocity vs time shifts upward and graph of displacement vs time shrinks.

Hence from the above equations we can see that graph of velocity vs time would be a straight line , and graph of displacement vs time would be parabola. If initial conditions change, the nature of graph remains same but graph would shift.

Question :2)

code:

```
clear;
init_vel=0;
init_dis=0;
mass=10;
theta=(10*pi)/180;
g=9.8;
t=10;
dt=0.01;
nitr=t/dt;
us=0.4;
uk=0.3;
time=zeros(nitr,1);
vel=zeros(nitr,1);
dis=zeros(nitr,1);
dis(1)=init_dis;
vel(1)=init_vel;

if tan(theta)>=us
    acc=g*sin(theta)-g*uk*cos(theta);
    for step=1 :nitr-1

        vel(step+1)=vel(step)+acc*dt;
        dis(step+1)=dis(step)+vel(step)*dt;
        time(step+1)=time(step)+dt;
    end
end
plot(time,vel);
hold on;
```

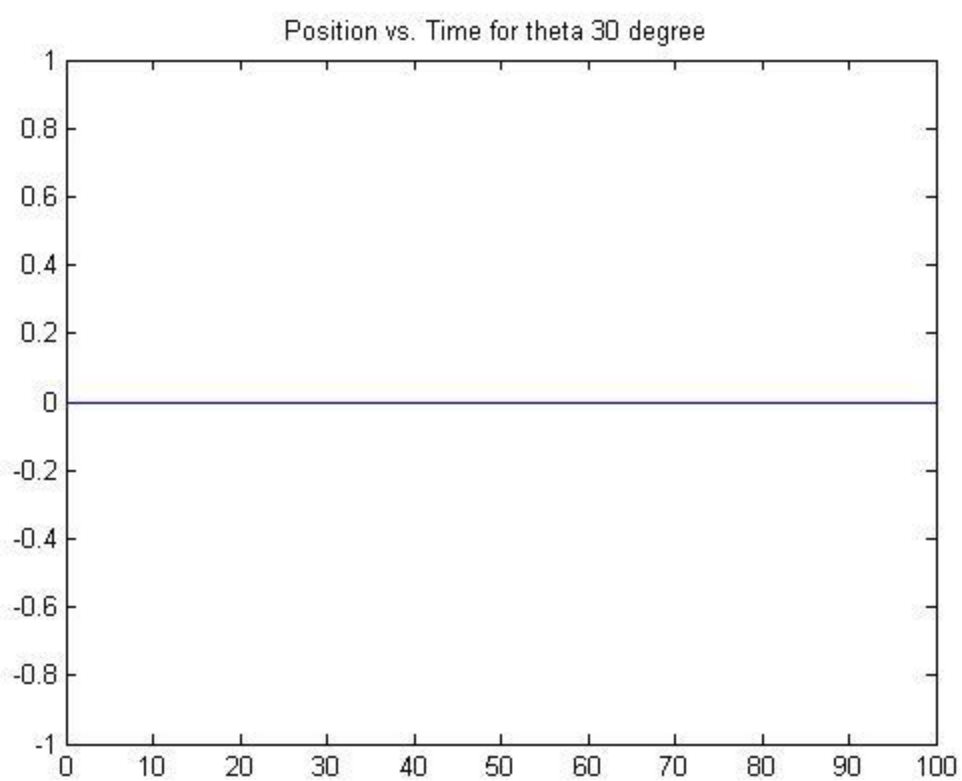
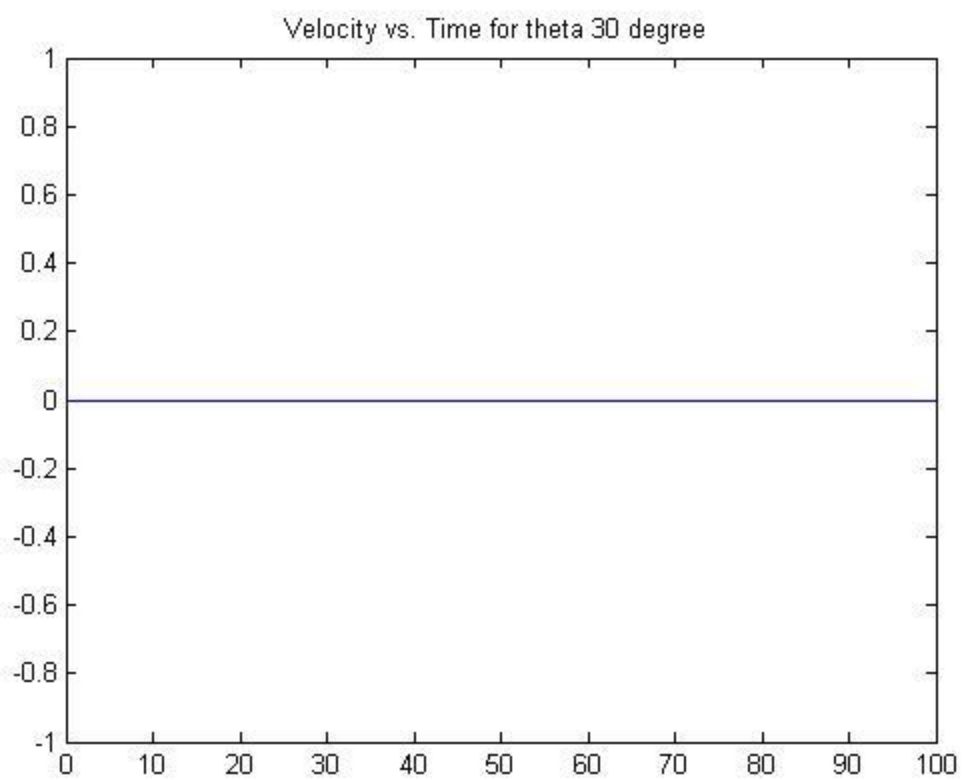
RHS:

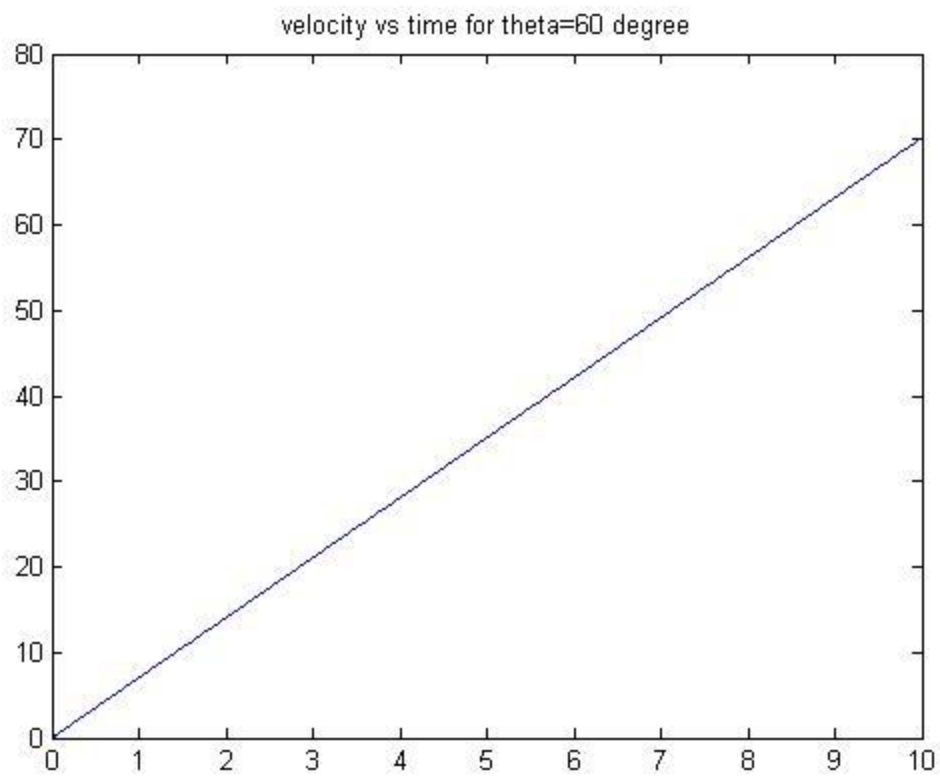
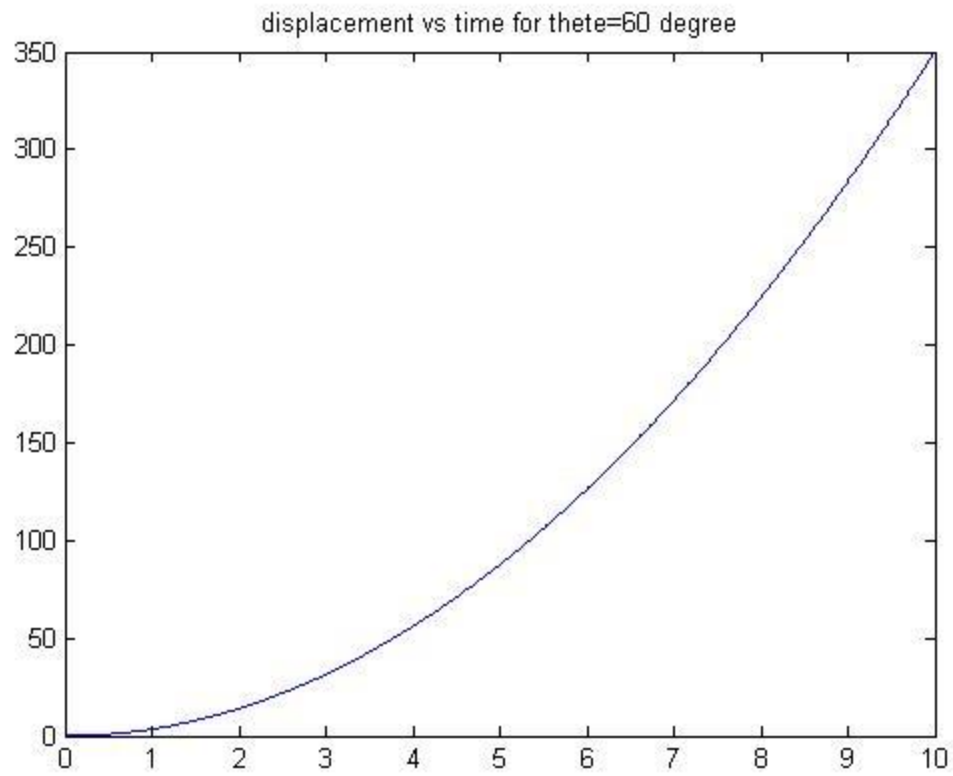
```
function F=rhsq2(t,u)
global g;
global theta;
global us;
global uk;
global a;
F=zeros(length(u),1);
F(1)=u(2);
if tan(theta)>=us/uk
    F(2)=g*sin(theta)-uk*g*cos(theta);
else
    F(2)=0;
```

end

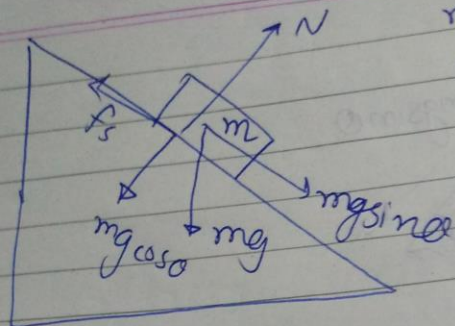
a=F(2);

Graphs:





(2)



maximum θ for which block starts to slide

$$F = mg \sin \theta - f_s$$

maximum value of f_s

$$f_{s \max} = \mu_s N$$

$$= \cancel{0.4} N = \mu mg \cos \theta$$

$$= 0$$

$$F = mg \sin \theta - \mu mg \cos \theta$$

$$\tan \theta \geq \mu$$

then

$$\therefore \theta \geq \tan^{-1}(\mu)$$

then block will start to slide. now kinetic friction will apply.

$$\therefore F = ma = mg \sin \theta - \mu_k N$$

$$F(1) = \checkmark \quad a = g \sin \theta - \mu_k g \cos \theta$$

$$F(2) = a = g \sin \theta - \mu_k g \cos \theta$$

Here angle of inclined plane plays a very important role in motion of the block. If angle is not more than a certain angle (depending on static friction) the block won't move on its own (with initial velocity 0)

$\tan(\theta) \geq \mu_s$ (static friction coefficient)

If this relation is satisfied block would start moving on its own.

Question 3:

Code Part: A:

```
clear; close all;

cases=2;
global g;
global theta;
global step;
for step=1:2
    timescale=10;
    dt=.01;
    tstart=0;
    tfinal=100*timescale;
    theta=30*pi/180;
    g=9.8;
    u0=zeros(4,1);
    init_vel=2000;
    u0(1)=0; % initial position in x
    u0(2)=init_vel*cos(theta); % initial velocity in x
    u0(3)=0; % initial position in y
    u0(4)=init_vel*sin(theta); % initial velocity in y
    [t,u]=ode45(@rhsq3a,tstart:dt:tfinal,u0);
    x=u(:,1);
    vx=u(:,2);
    y=u(:,3);
    vy=u(:,4);
    if step==1
        plot(x,y,'r')
    else
        plot(x,y,'g')
    end
end
title('red for ideal case, green for altitude varying g')
hold on
end
```

RHS for code 3A:

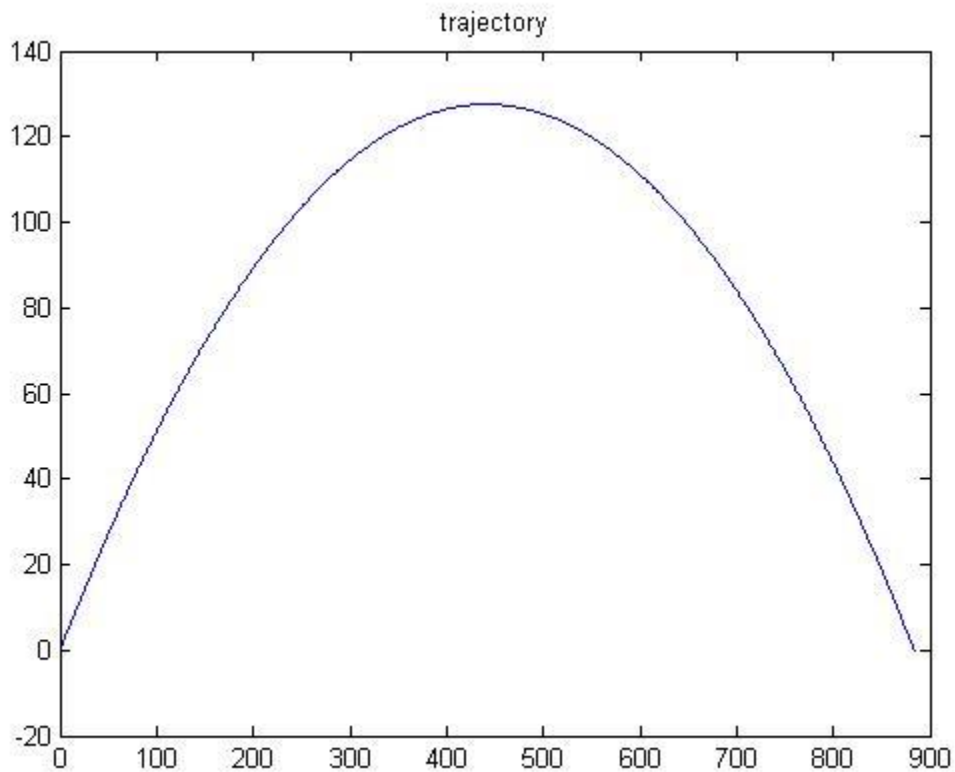
```
function F=rhsq3a(t,u)
```

```

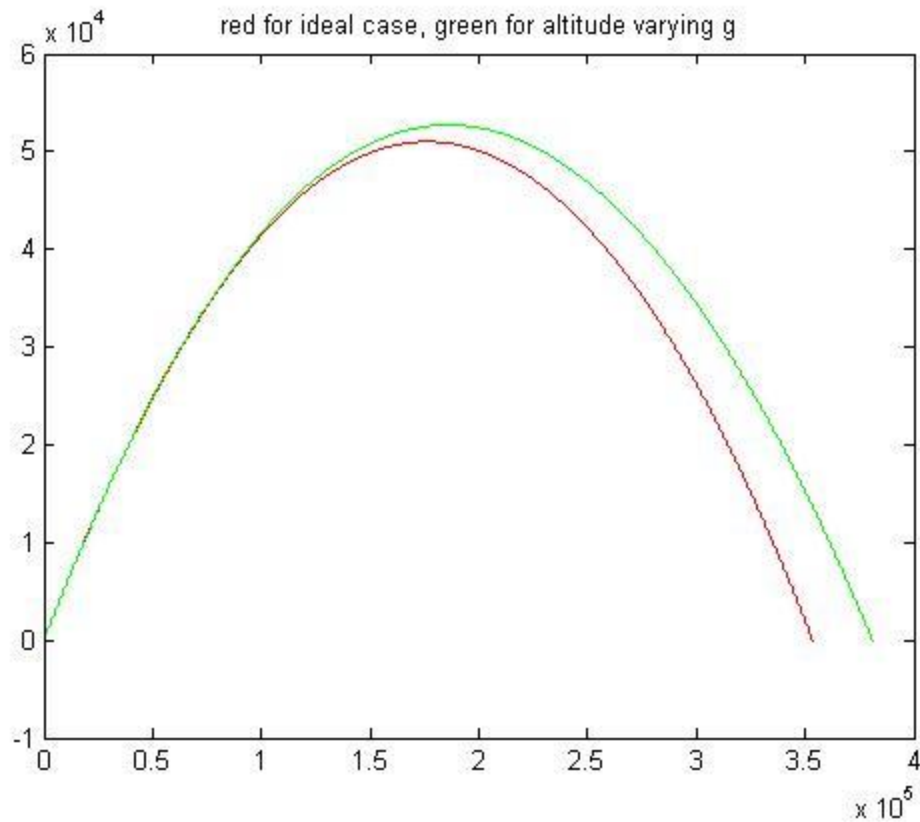
global g;
global step;
if step==2
g=g-g*u(3)/6400000;
end
F=zeros(length(u),1);
F(1)=u(2);
F(2)=0;
F(3)=u(4);
F(4)=-1*g;
if u(3)<0
    F(1)=0;
    F(2)=0;
    F(3)=0;
    F(4)=0;
end

```

Graph for code 3A:



But when we vary g according to the altitude we get the following graph.
 Here as altitude increases value of g decreases. Hence the value of opposing force decreases so range of projectile increases. Both the plots are shown on a same graph to recognize the difference.



Code for 3B:

```
clear;close all;

init_theta=40;
thetamax=50;
dtheta=.1;
ncases=(thetamax-init_theta)/dtheta;
global g;
global theta;
global step;
global const;
global arr;
arr=zeros(ncases,1);
global arrv;
arrv=zeros(ncases,1);
theta=init_theta*pi/180;
dtheta=dtheta*pi/180;

for step=1:ncases
arr(step)=theta;
timescale=10;
dt=1;
```

```

% set the initial and final times
tstart=0;
tfinal=100*timescale;
g=9.8;
% set the initial conditions in the u0 column vector
u0=zeros(4,1);
%initial velocity at angle theta
init_vel=750;
%constant b/m
const=4e-5;
u0(1)=0; % initial position in x
u0(2)=init_vel*cos(theta); % initial velocity in x
u0(3)=0; % initial position in y
u0(4)=init_vel*sin(theta); % initial velocity in y

% set the solve options
[t,u]=ode45(@rhsq3b,tstart:dt:tfinal,u0);

% store the solution that comes back into x and v arrays
x=u(:,1);
vx=u(:,2);
y=u(:,3);
vy=u(:,4);
%plot x vs y trajectory
theta=theta+dtheta;

plot(x,y)
hold on

end

maxr=-1;
maxrtheta=arr(1);
for step2=1:ncases
    if arrv(step2)>maxr
        maxr=arrv(step2);
        maxrtheta=arr(step2);
    end
end

maxr
maxrtheta*180/pi

```

RHS for code 3B:
function F=rhsq3b(t,u)

```

% function output =name(input)
% right-hand side function for Matlab's ODE solver,

% declare the globals so its value
% set in the main script can be used here
global g;
global const;
global arrv;
global step;
F=zeros(length(u),1);
y0=1000;
drag=const*sqrt(u(2)*u(2)+u(4)*u(4))*exp(-1*u(3)/y0);
% Now build the elements of F
F(1)=u(2);
F(2)=-1*drag*u(2);
F(3)=u(4);
F(4)=-1*g-(1*drag*u(4));
if u(3)<0
    arrv(step)=u(1);
    F(1)=0;
    F(2)=0;
    F(3)=0;
    F(4)=0;
end
end

```

$$g = g \left(1 - \frac{y}{R_e} \right)$$

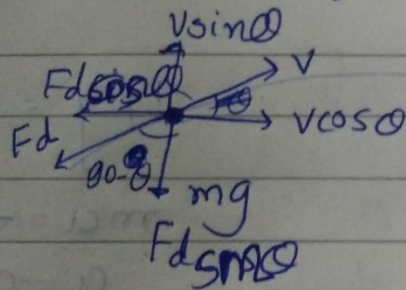
$$R_e = 6400,000 ;$$

(b) air drag and reduced air density:

$$F_{\text{drag}} = -Bv^2$$

$$\rho = \rho_0 \exp(-y/y_0)$$

$$F^*_{\text{drag}} = \frac{\rho}{\rho_0} F_{\text{drag}}$$



$$\cancel{F_x = -F_d \sin \theta}$$

$$F_x = -F_d \cos \theta$$

$$F_y = -mg - F_d \sin \theta$$

$$a_x = - \frac{F_d \cos \theta}{m} = - \frac{\rho}{\rho_0} \times \frac{B}{m} \times v \times \theta \cos \theta$$

$$a_y = -g - \frac{F_d \sin \theta}{m}$$

$$= -g - \frac{\rho}{\rho_0} \times \frac{B}{m} \times v \times \theta \sin \theta$$

$$\cancel{V_x = a_x t} \quad V_x = v_0 \cos \theta - \frac{F_d \cos \theta}{m}$$

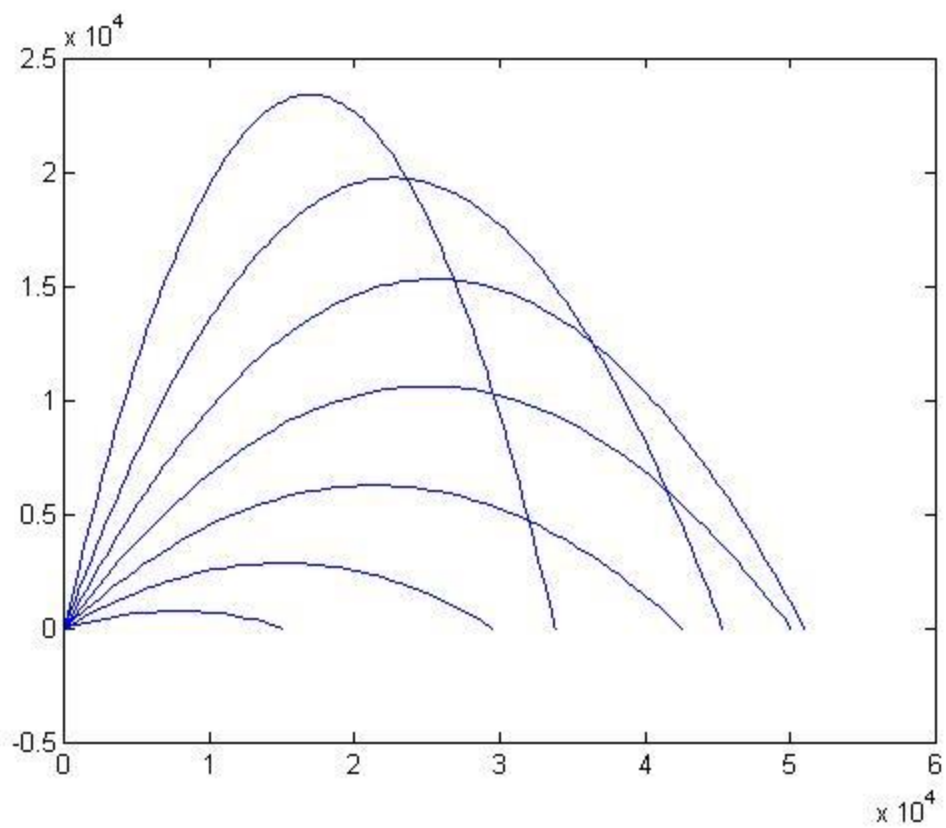
$$\frac{dV_y}{dt} = \cancel{g} - a_y$$

$$v_x = \frac{dx}{dt} \quad v_y = \frac{dy}{dt}$$

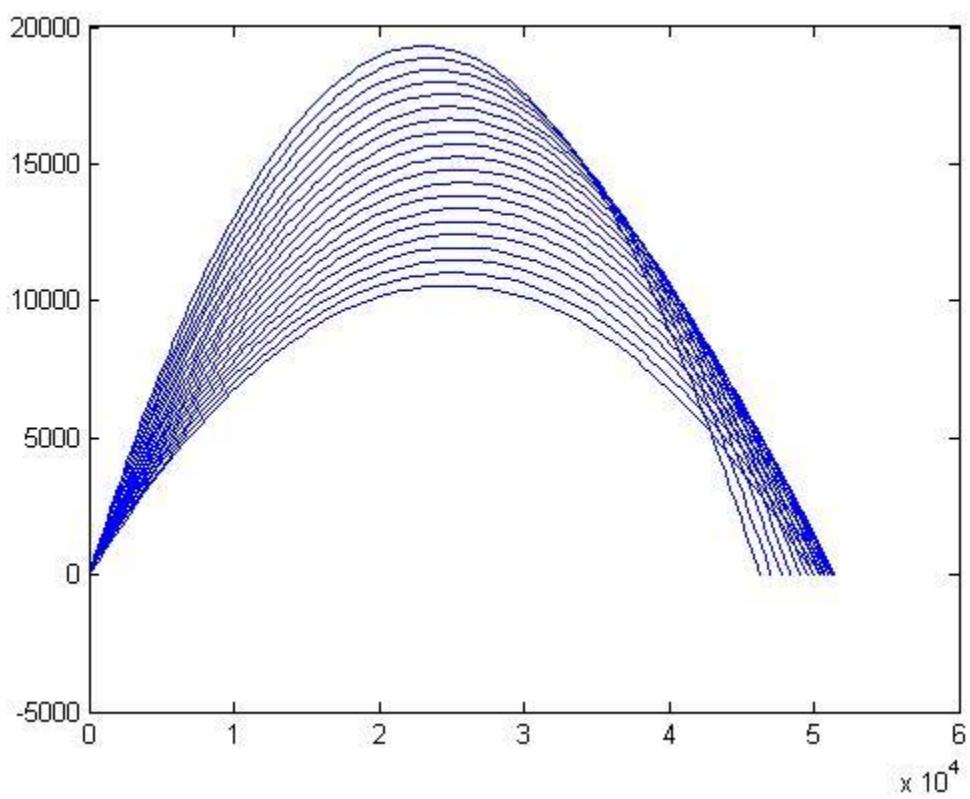
When $y=0$ ^(again), x is Range. \Rightarrow

Graph for problem 3B:

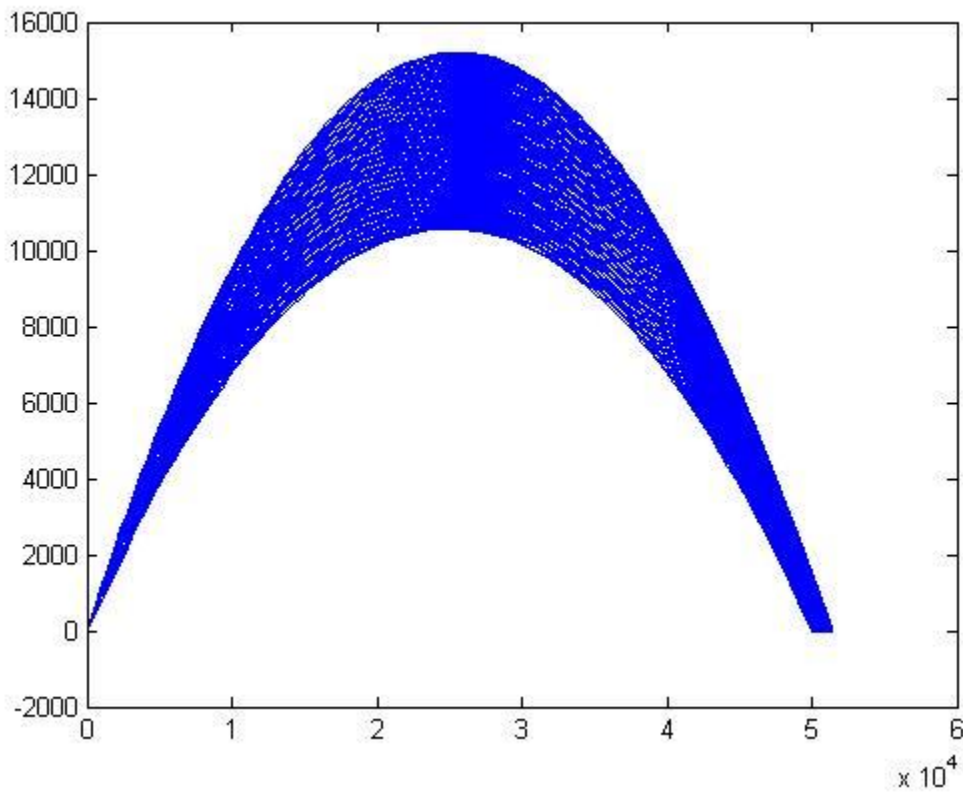
This is a graph for 10,20,30,...,70 degree of initial angle and their trajectories. From here we can see that maximum would occur between 40 to 60, so for more precise answer and less computation we would plot trajectories for 40 to 60 degree in next graph.



40 to 60 degrees :dtheta =1



40 to 50 degrees : $d\theta = 0.1$



Hence from these computations, by decreasing our target range and increasing computational speed we calculated maximum range = 51379 m at initial angle 46.5.

Code : 3C

$$(c) \quad y = x \tan^2 \theta - \frac{gx^2}{2v^2 \cos^2 \theta}$$

$$\frac{1}{v^2} = \frac{(x \tan^2 \theta - y) 2 \cos^2 \theta}{gx^2}$$

$$v = \sqrt{\frac{gx^2}{2 \cos^2 \theta (x \tan \theta - y)}}$$

$$v = \sqrt{\frac{g}{2}} \frac{x}{\sqrt{\cos^2 \theta (x \tan \theta - y)}}$$

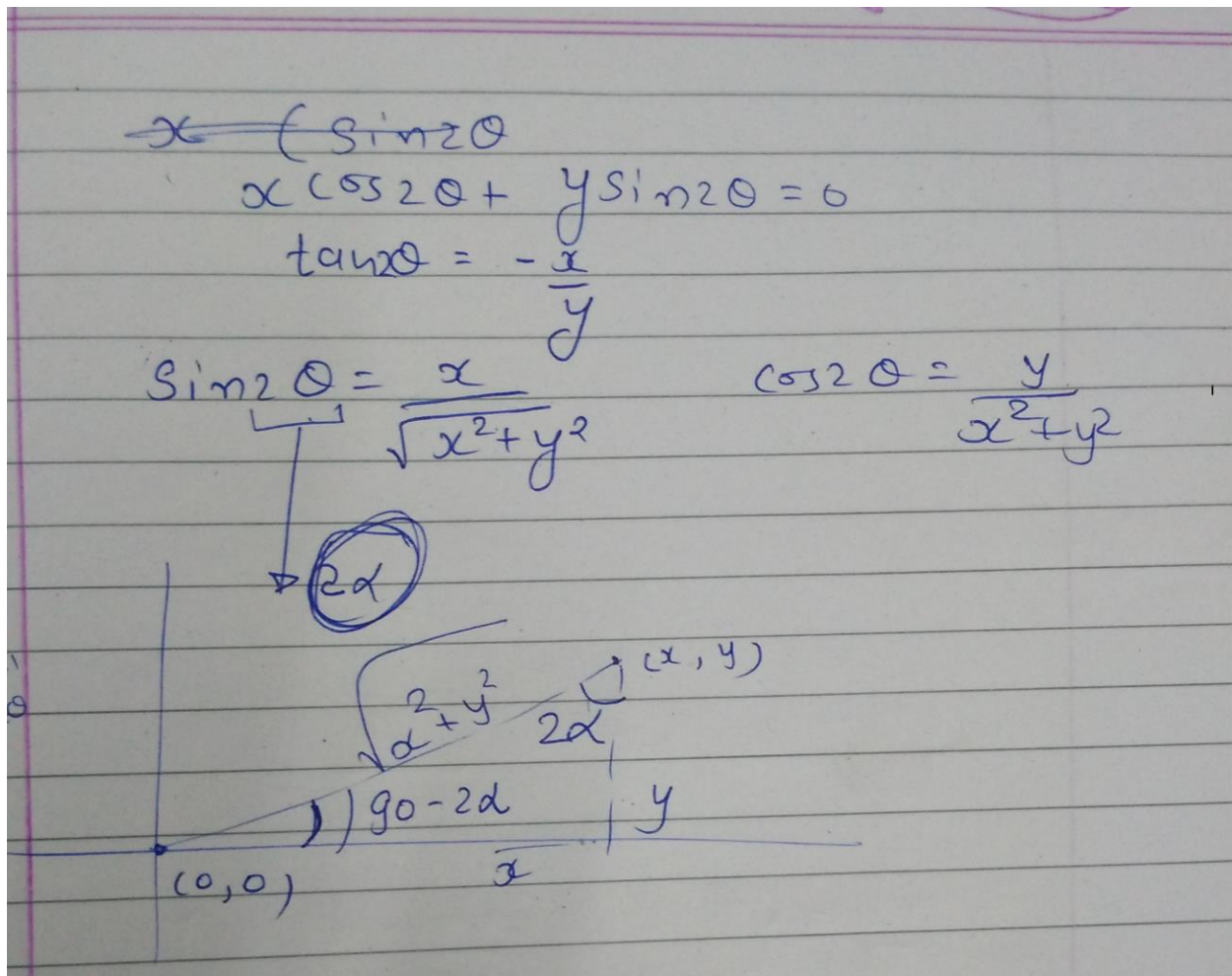
$$\cos^2 \theta (x \tan \theta - y) \rightarrow \max$$

$$P_y = \cos^2 \theta (x \tan \theta - y)$$

$$\frac{dP_y}{d\theta} = \cos^2 \theta (x \sec^2 \theta) + (x \tan \theta - y)(-2 \sin \theta \cos \theta)$$

$$= x + x \frac{\sin \theta}{\cos \theta} (-2 \sin \theta \cos \theta)$$

$$x (1 - 2 \sin^2 \theta) + y \sin 2\theta = 0$$



```

clear;
close all;
xf=500;
g=9.8;
start=-500;
finish=500;
varr=zeros(2*finish+1,1);
for iter=1:2*finish+1
    varr(iter,1)=99999;
end;
iter=1;
for yf=start:finish
    for theta=-89:0.1:89
        denomi=xf*tand(theta)-yf;
        if(denomi<=0)
            continue;
        end
    end
end

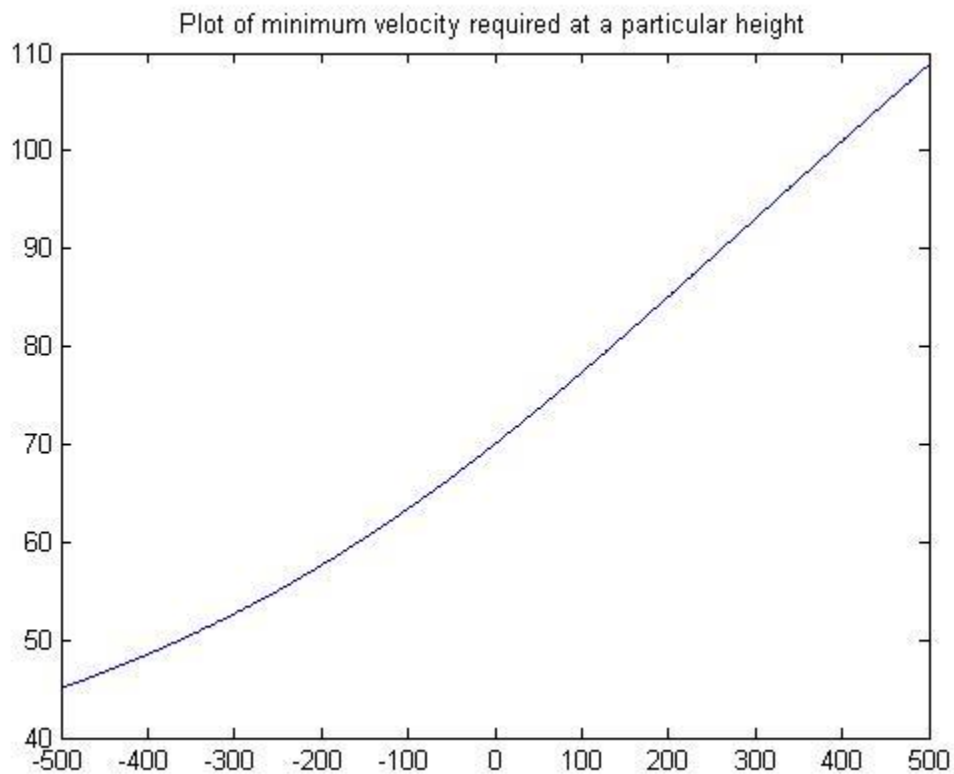
```

```

        v=sqrt((g*xf*xf)/(2*cosd(theta)*cosd(theta)*denomi));
        varr(iter,1)=min(v,varr(iter,1));
    end
    iter=iter+1;
end
yf=-500:500;
plot(yf,varr);
title('minimum velocity vs vertical displacement')

```

Graph for code 3C:



Code 3D:

```

clear;close all;

cases=2;
global g;
global theta;
global step;
for step=1:2

```

```

timescale=10;
dt=.01;

% set the initial and final times
tstart=0;
tfinal=100*timescale;
theta=30*pi/180;
g=9.8;
% set the initial conditions in the u0 column vector
u0=zeros(5,1);
%initial velocity at angle theta
init_vel=2000;
u0(1)=0; % initial position in x
u0(2)=init_vel*cos(theta); % initial velocity in x
u0(3)=0; % initial position in y
u0(4)=init_vel*sin(theta); % initial velocity in y
u0(5)=0; % initial position in z
u0(6)=0; % initial velocity in z

% set the solve options
[t,u]=ode45(@rhsq3a,tstart:dt:tfinal,u0);

% store the solution that comes back into x and v arrays
x=u(:,1);
vx=u(:,2);
y=u(:,3);
vy=u(:,4);
z=u(:,5);
vz=u(:,6);
%plot x vs y vs z trajectory

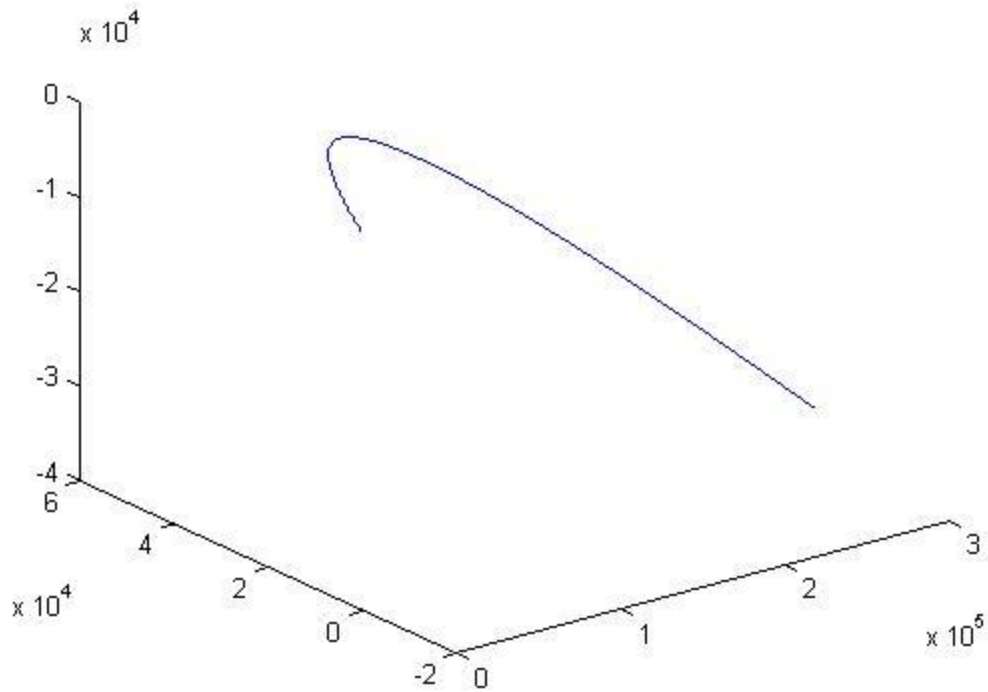
    plot3(x,y,z);

end
title('red for ideal case, green for altitude varying g')
hold on

```

Graph:

red for ideal case, green for altitude varying g



4)

For pendulum:
code:

```
clear;close all;  
% declare the pendulum variables to be global and set it  
global cnst;  
global beta;
```

```
g=9.8;  
l=1;  
cnst=g/l;
```

```
mass=2;  
b=sqrt(4*mass*mass*cnst);
```

```
%beta=b/(2*mass); for damped system  
beta=0;  
timescale=2*pi*sqrt(l/g);  
dt=timescale/100;
```

```
% set the initial and final times
```

```

tstart=0;
tfinal=10*timescale;

% set the initial conditions in the y0 column vector
u0=zeros(2,1);
u0(1)=.2; % initial position; %theta(1)=.2;
u0(2)=0; % initial velocity
%[t,u]=ode45(@rhs,[tstart,tfinal],u0,options);
[t,u]=ode45(@rhs34,[tstart:dt:tfinal],u0);

% store the solution that comes back into x and v arrays
x=57.5*u(:,1); % radian-->degree
v=u(:,2);
% plot the position vs. time
plot(t,x)
title('Position vs. Time')
% make a "phase-space" plot of v vs. x
figure
plot(x,v)

```

RHS for code 4:

```

function F=rhs34(t,u)
% function output =name(input)
% right-hand side function for Matlab's ODE solver,

```

% In our case we will use:

```

% u(1) -> x
% u(2) -> v

```

```

% declare the globals so its value
% set in the main script can be used here
global cnst;

```

```

global beta;

```

```

% make the column vector F filled with zeros
F=zeros(length(u),1);

```

```

% Now build the elements of F
%
% so the equation  $dx/dt=v$  means that  $F(1)=u(2)$ 
F(1)=u(2);
% Again, in our original ODEs we have:

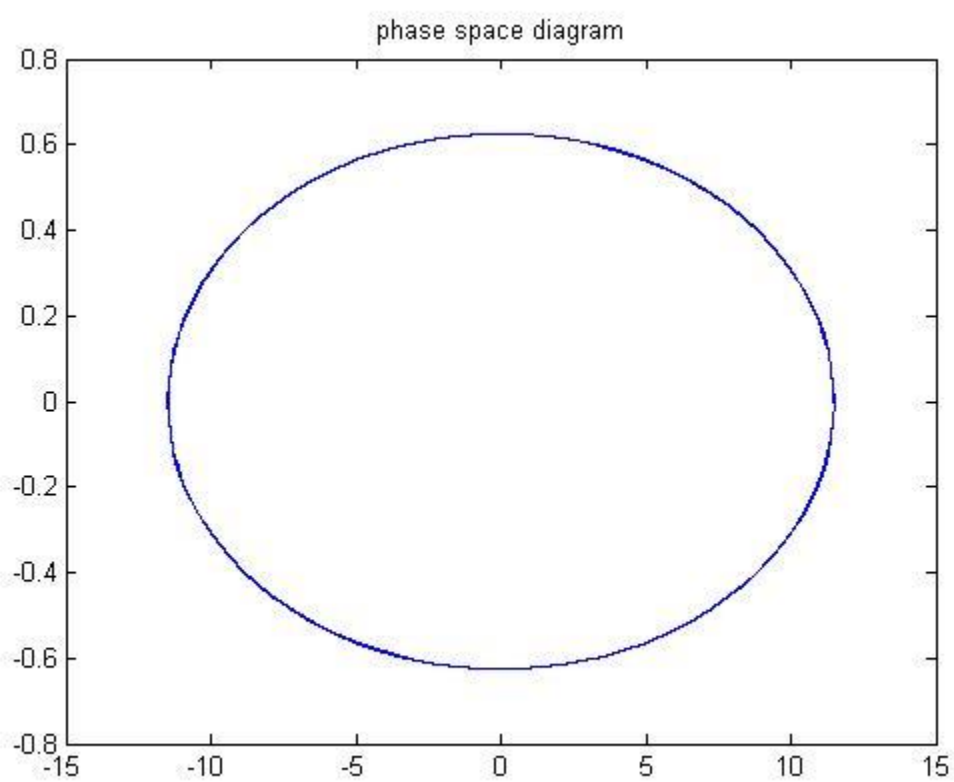
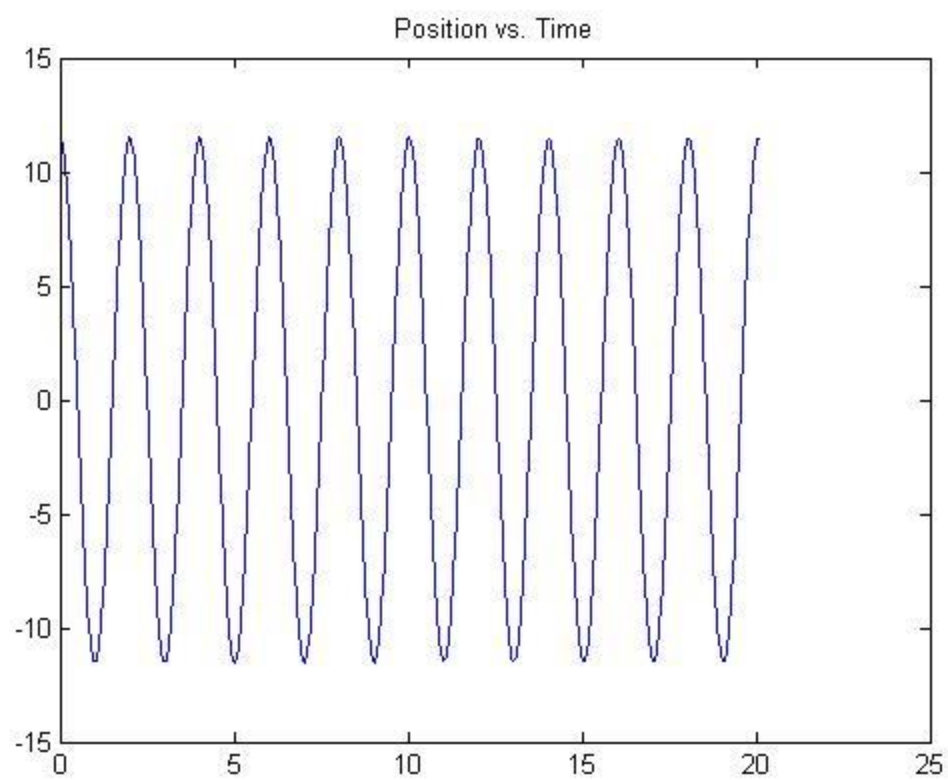
```

```

% so the equation  $dv/dt=-....$ 
F(2)=-1*(2*beta*u(2)+cnst*u(1));

```

Graph:



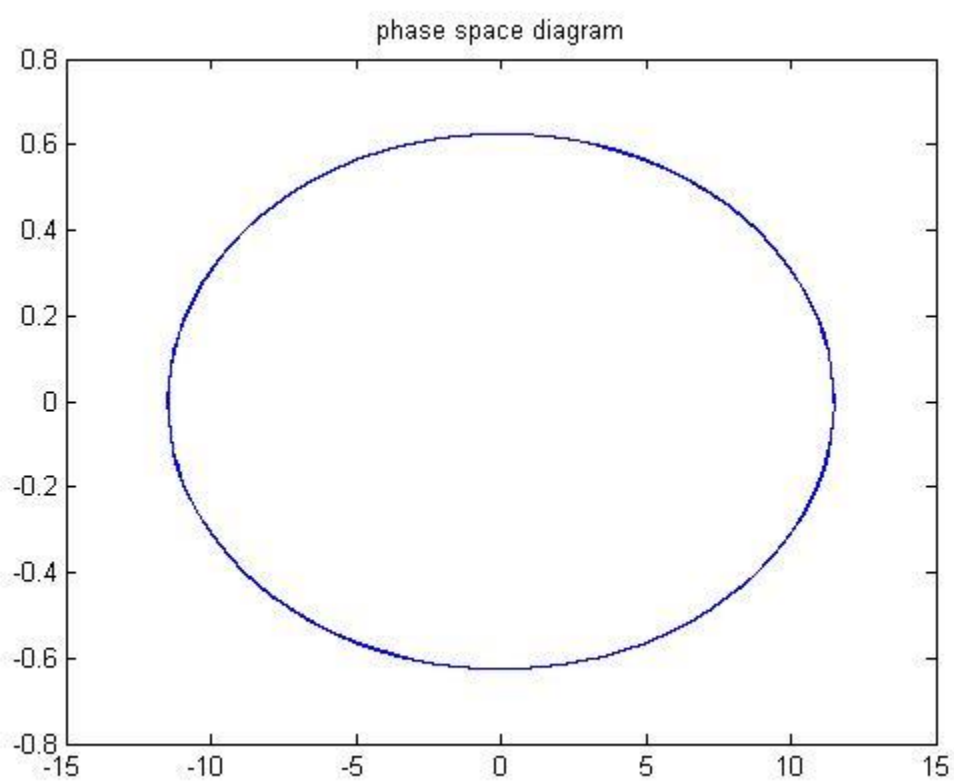
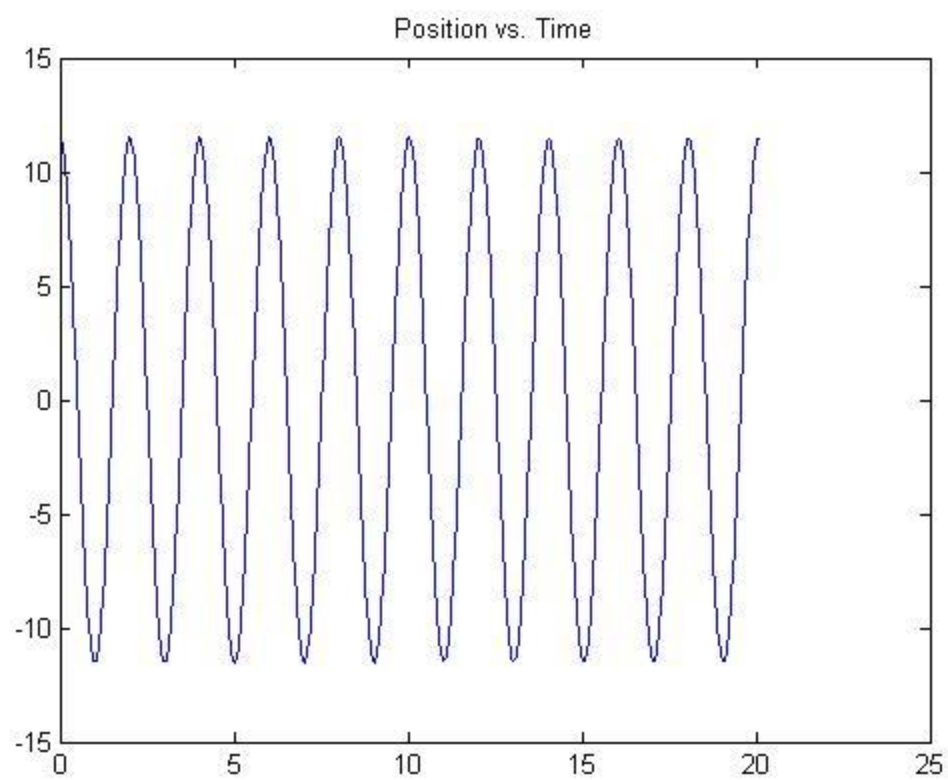
For Spring mass system:**code:**

```
clear;close all;
global cnst;
global beta;
k=1000;
mass=5;
cnst=k/m;
b=sqrt(4*mass*mass*cnst);
%beta=b/(2*mass); for damped system
beta=0;
timescale=2*pi*sqrt(l/g);
dt=timescale/100;
tstart=0;
tfinal=10*timescale;
u0=zeros(2,1);
u0(1)=.2; % initial position; %theta(1)=.2;
u0(2)=0; % initial velocity
[t,u]=ode45(@rhs34,[tstart:dt:tfinal],u0);
x=57.5*u(:,1); % radian-->degree
v=u(:,2);
plot(t,x)
title('Position vs. Time')
figure
plot(x,v)
```

RHS for code 4:

```
function F=rhs34(t,u)
global cnst;
global beta;
F=zeros(length(u),1);
F(1)=u(2);
F(2)=-1*(2*beta*u(2)+cnst*u(1));
```

Graph:



Graph of both pendulum and spring mass system remains same.

The only difference in both the system is of ω (omega). In pendulum $\omega = \sqrt{g/l}$, hence it depends on length of the string in this case and not on mass while in spring mass system $\omega = \sqrt{k/m}$, it depends on mass.

The phase space diagram of both would remain same. Both have restoring force. In both cases when velocity is maximum displacement is minimum and vice versa.

Both have sinusoidal trajectories.