CS201 Assignment 6

Lab PC no : 3

Name : Omkar Damle

ID: 201401114

Attachments with pdf file :

1. **ballMotion1.avi**
2. **ballMotion2_projectile.avi**
3. **BouncingBallMotion_projectile.avi**

Question 1. To visualize the solution of a second-order differential equation –

Solution –

In a certain region, wind is blowing with velocity that is constant in time, but varies spatially according to –

$dx/dt = v_x = 0.2 * x * x + 0.5 * y * y + 20$

$dy/dt = v_y = - 0.1 * y * y * y + 0.5 * x * x – 10$

Physical Insight –

Using this graph we can see both the speed and direction of wind (in this case), variations in wind speed along the streamlines, or the convergence or divergence of the streamlines. Depending on the velocity vector at various co-ordinates we can analyse the motion of a particle if it starts at a particular point then how will it move with the wind.

From the graph we can observe that the streamlines are converging as we are moving from left to right and that indicates that the wind velocity is increasing (as density of lines is more) in that direction. Streamlines can also be interpreted as the paths traced out by the individual wind particles.

Quiver plot –

Quiver displays various arrows at each data point in x and y such that the arrow direction and length represent the corresponding values in u and v.

A quiver plot displays velocity vectors as arrows with components (u,v) at the points (x,y). It is used to see the velocity at each point which gives us an idea of the velocity vectors existing in the space which helps up decide what to do depending on the situation like in fluid dynamics in helps us in knowing the intensity with which it is flowing at each point.

Streamlines –

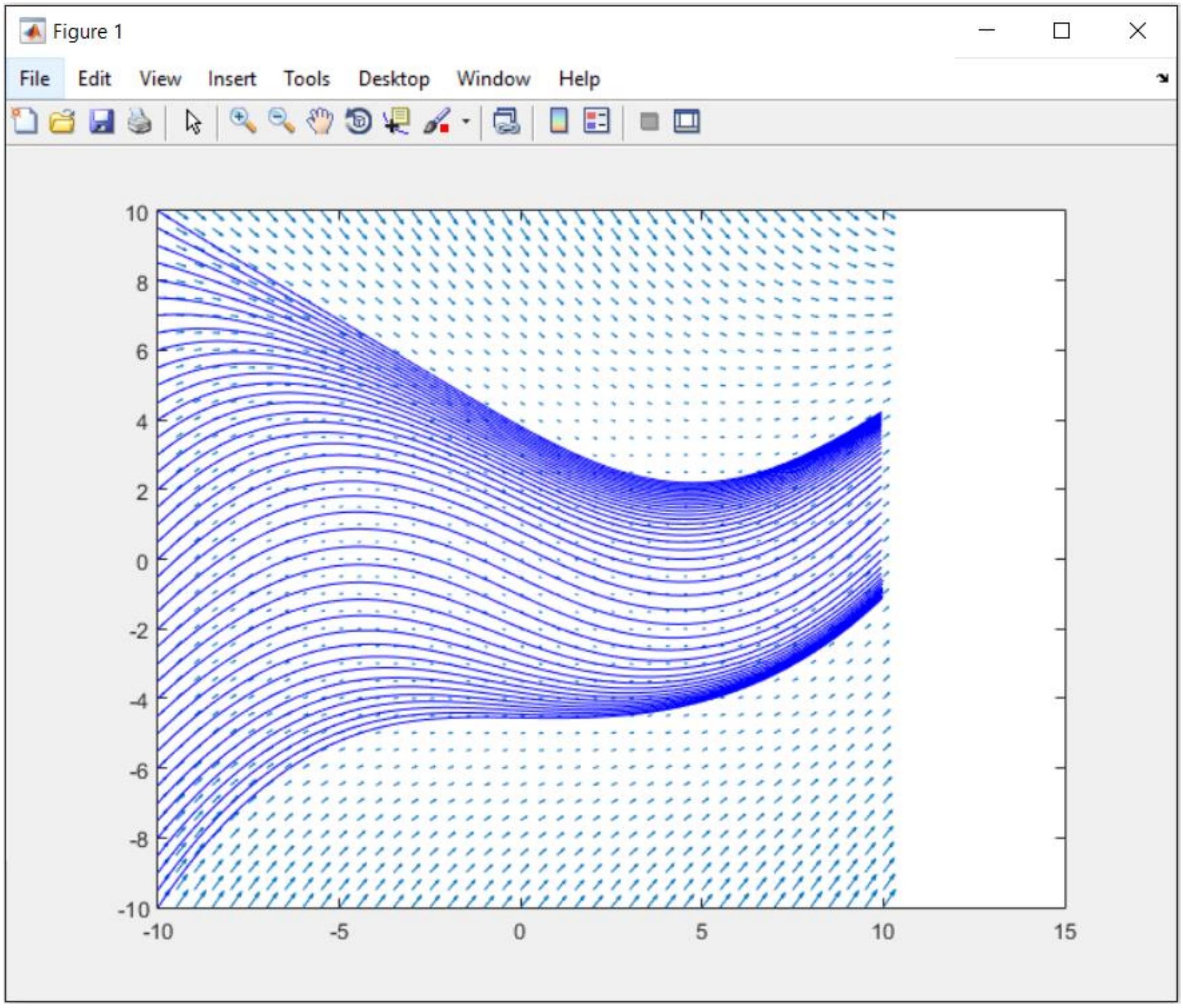
A streamline can be thought of as the path a massless particle takes flowing through a velocity field (i.e. vector field). Streamlines can be used to convey the structure of a vector field by providing a snapshot of the flow at a given instant in time. Multiple streamlines can be created to explore interesting features in the field. Streamlines are computed via numerical integration (integrating the product of velocity times delta T).

Streamlines can be diverging or converging which has different interpretation. It tells us the path that a particle follows if it is left at any point in space. In this case it is the wind velocity streams that is being showed in the graph. This can be used to study the wind patterns and this helps the pilots to decide which direction to go and with what velocity. Can be even used to study the weather conditions.

The flow plot (quiver and streamlines) –

Horizontal axis – x

Vertical axis – y



MATLAB CODE –

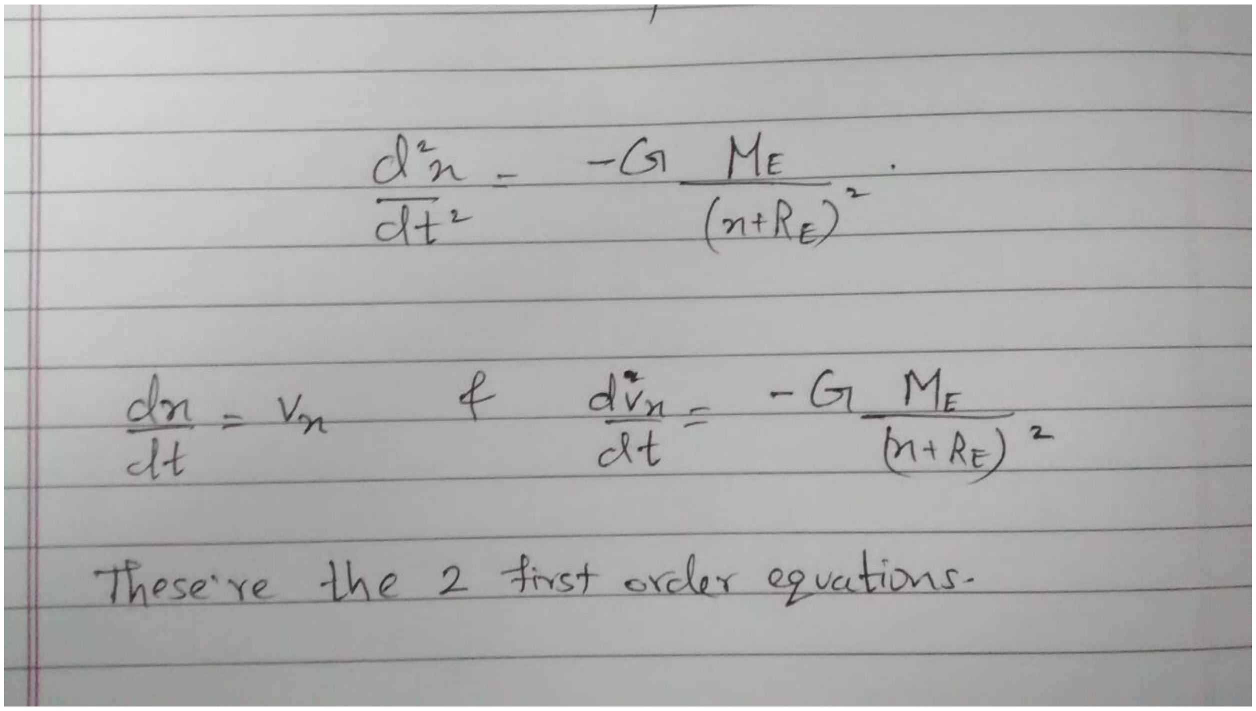
```
% first we make meshgrid of co-ordinates



[x,y] = meshgrid(-10 : 0.5 : 10 , -10 : 0.5 : 10);

vx = 0.2.*x.*x + 0.5.*y.*y + 20;

vy = -0.1.*power(y,3) + 0.5.*x.*x - 10;
```

```
figure
% note x,y,vx,vy are 2D arrays containing values of respective quantities
% at those points.

quiver(x,y,vx,vy)

% startx and starty will give the set of points from where we want the
% streamlines to start.

starty = [-10 : 0.5 : 10];
startx = ones(size(starty)).*-10 ;

streamline(x , y , vx , vy , startx , starty)
```

Question 2: Throwing a ball up in the air.

Summary : In this program we simulate the motion of a ball as it is thrown upwards in the air. We first experiment with low speeds of 1m/s, 10m/s, 40m/s and then with larger speeds of 1000m/s, 5000m/s and 10000m/s where the change in gravitational constant needs to be taken into account.
Also we observe the concept of escape velocity by simulation.
Finally we make a movie of a bouncing ball and convert it into .avi file

The two first order equations for simulating the motion of ball in x-direction are :



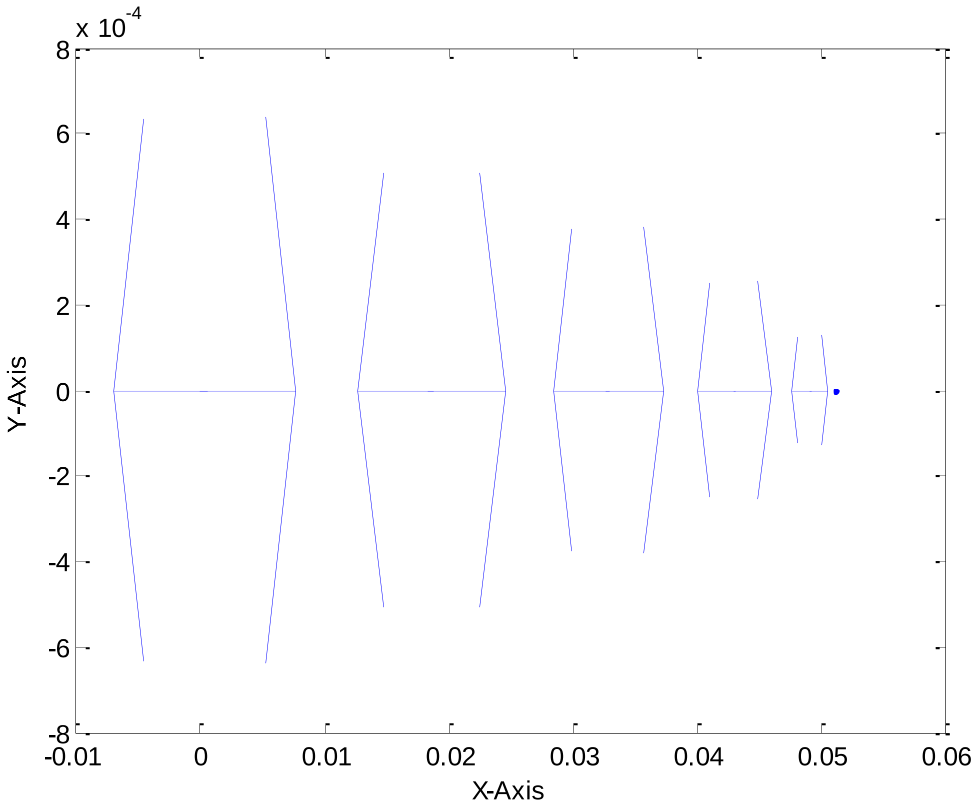
$$\frac{d^2x}{dt^2} = -G\frac{M_E}{(x+R_E)^2}.$$

$$\frac{dx}{dt} = v_x \quad \& \quad \frac{dv_x}{dt} = -G\frac{M_E}{(x+R_E)^2}$$

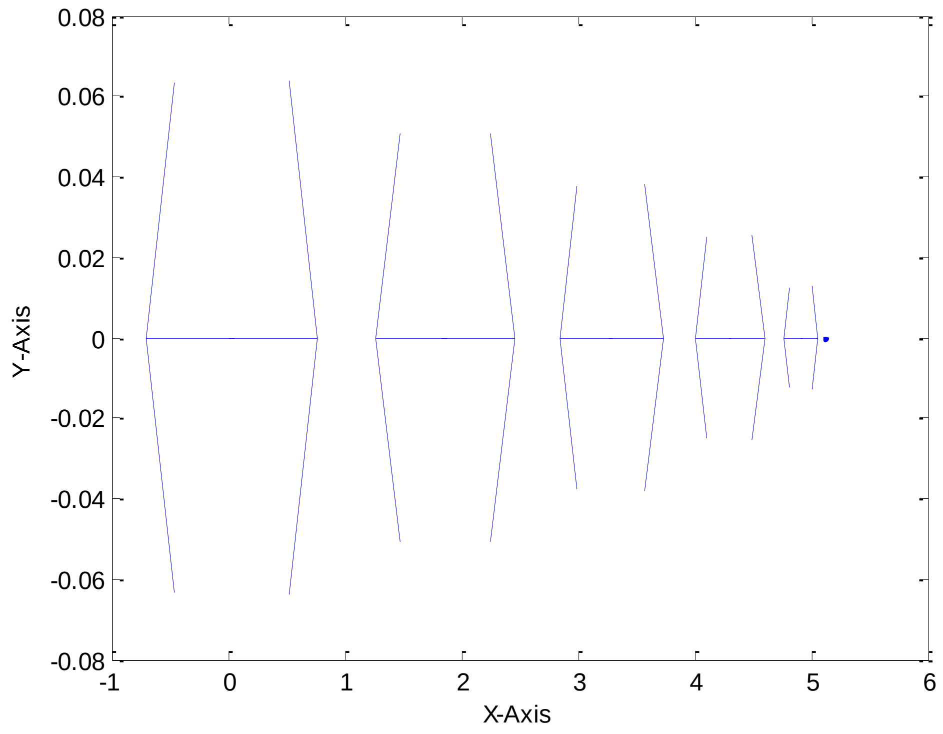
These're the 2 first order equations.

a)

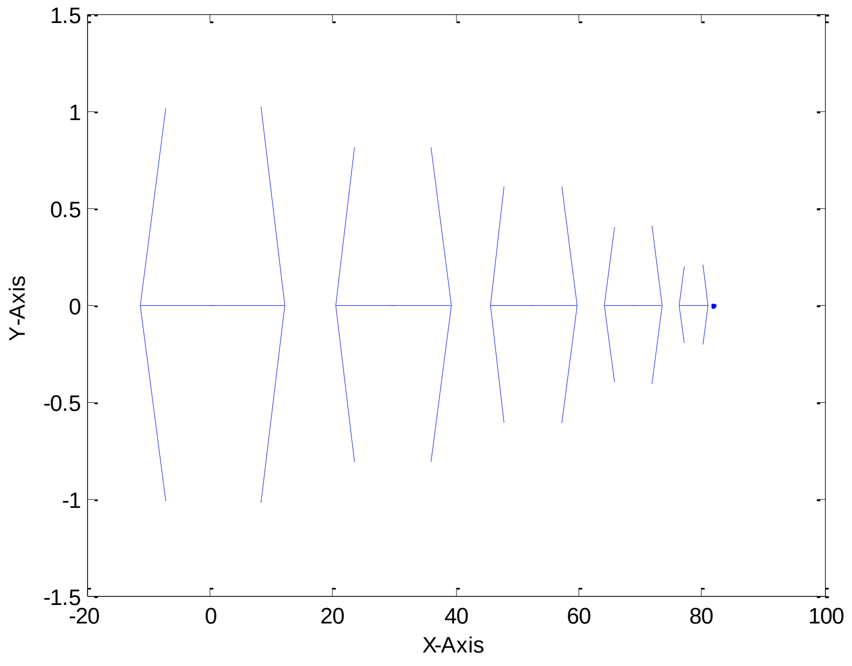
1. Quiver plot for initial_vel_x = 1m/s



2.

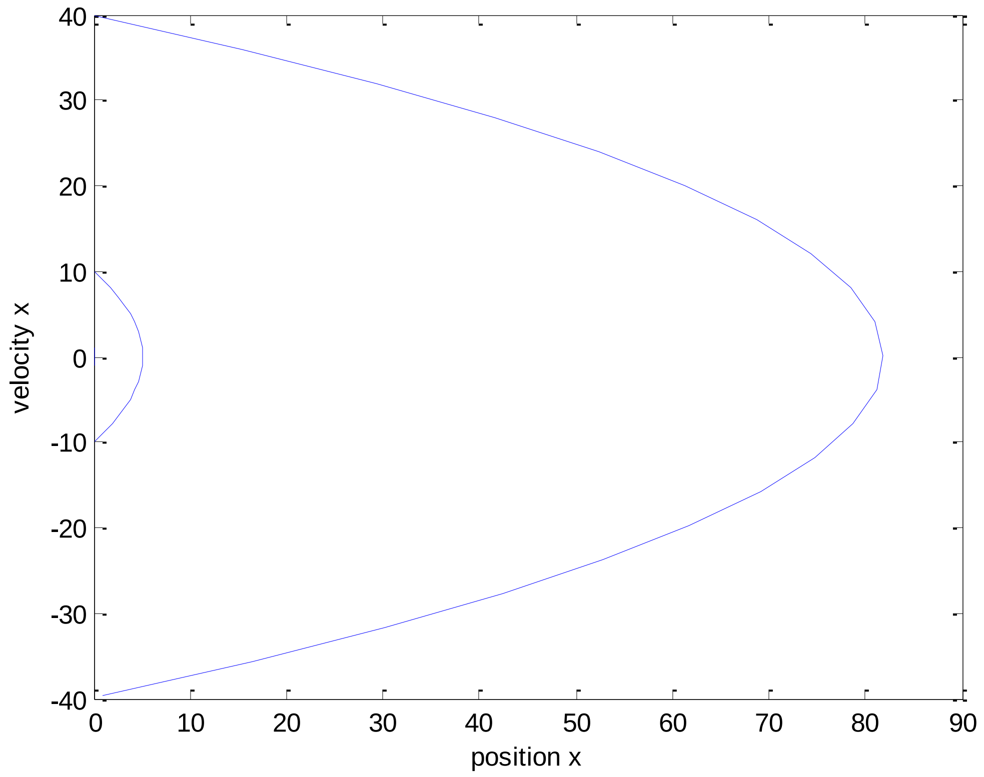

0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
Y-Axis
-1
0
1
2
3
4
5
6
X-Axis

3.

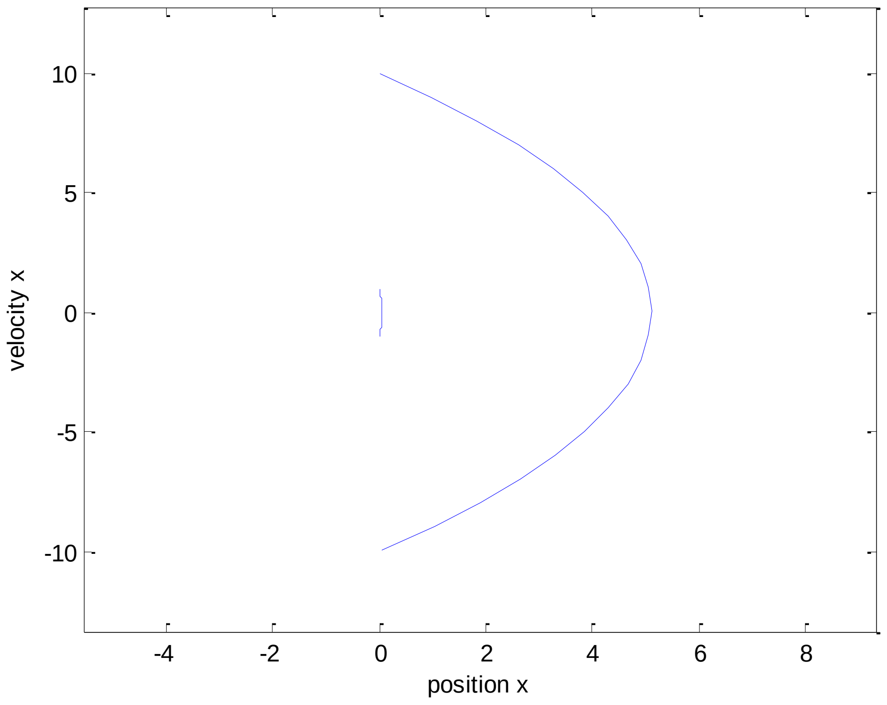

1.5
1
0.5
0
-0.5
-1
-1.5
Y-Axis
-20
0
20
40
60
80
100
X-Axis

The phase space trajectories for the 3 velocities

1. 1m/s
2. 10m/s
3. 40m/s

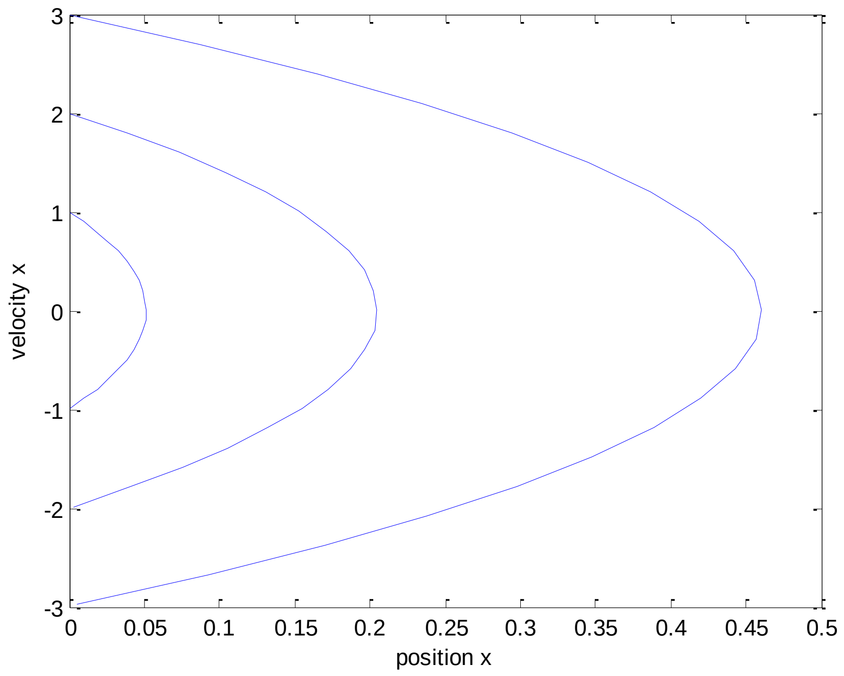


Here we are unable to see the phase space of vel = 1m/s. However if we zoom in,



We can see it.

Instead in order to compare it would be better to plot the phase space trajectories of velocities close to each other.
Eg: velocities 1m/s, 2m/s , 3ms give the following phase space plot :

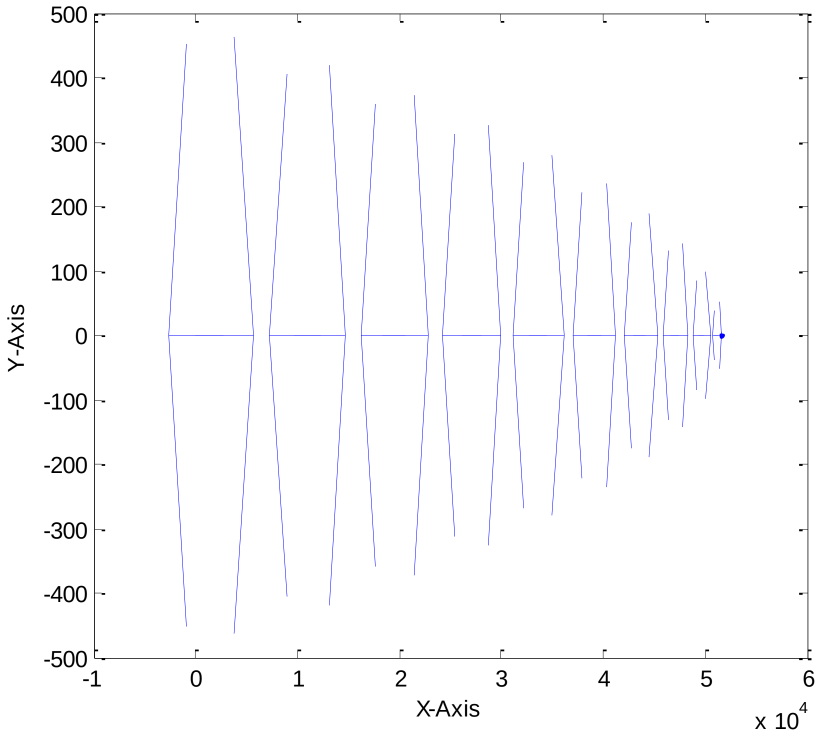


The phase space plot will be a parabola because the velocity of the ball and its position x are related by the following equation :

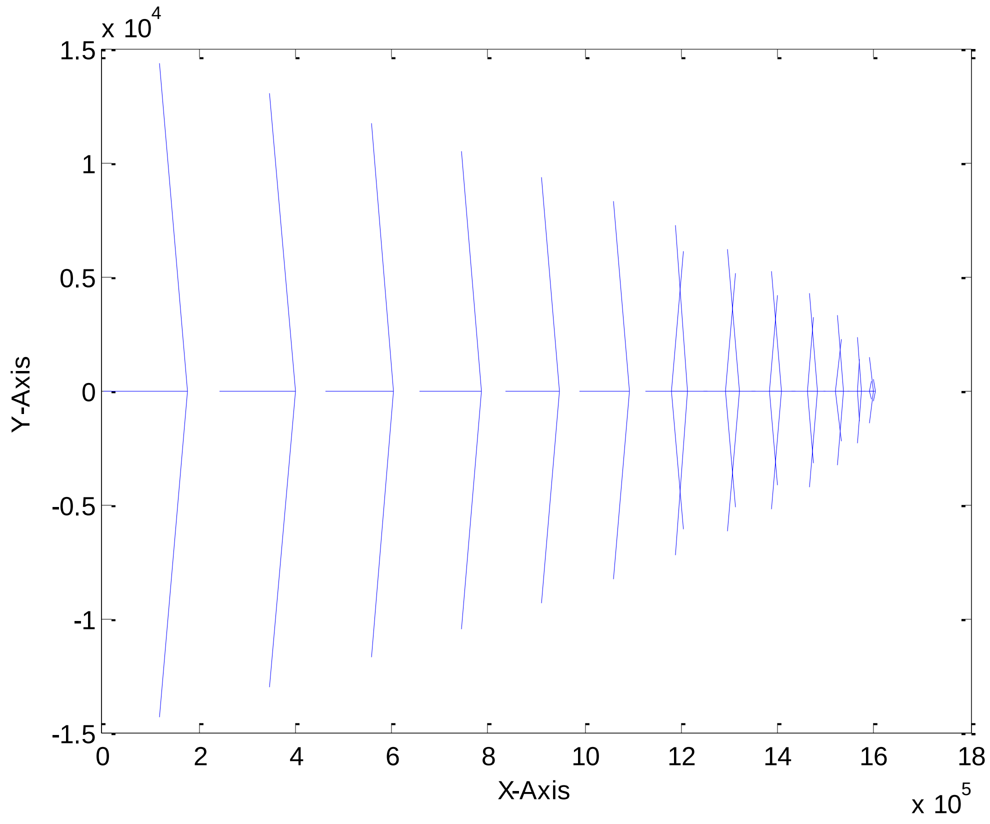
$$x = \frac{u_{x,\text{initial}}^2 - u_x^2}{2g}$$

b)
Quiver plots for the 3 velocities :

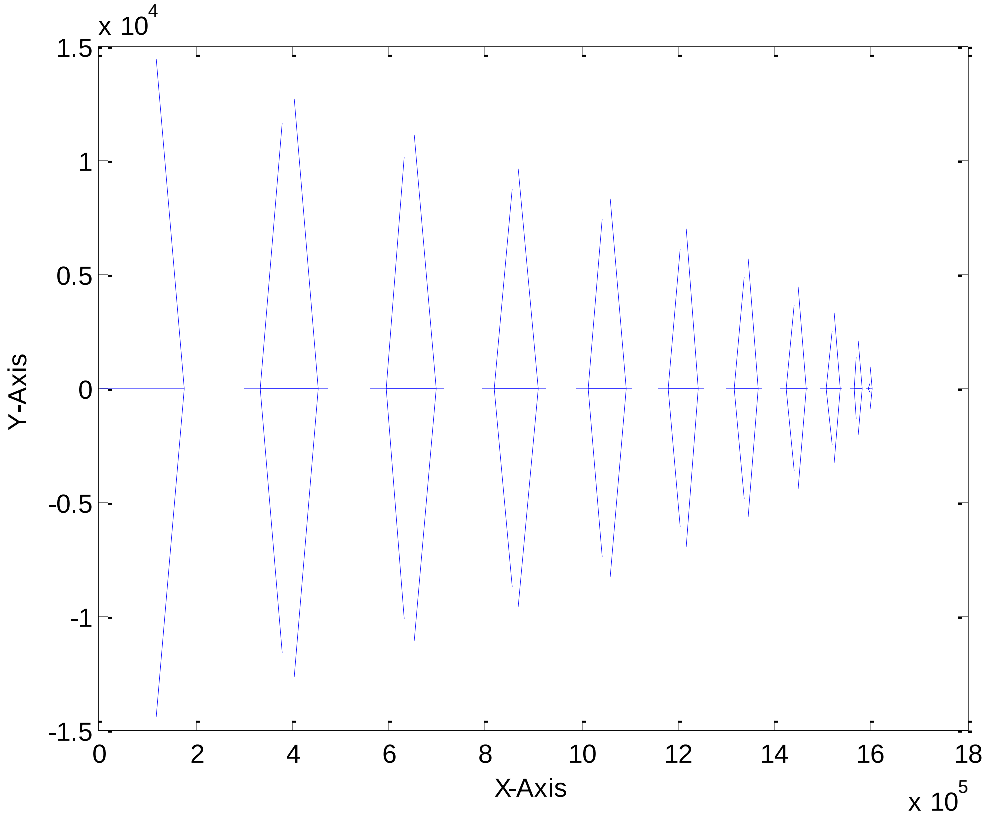
1. Initial_vel_x = 1000m/s



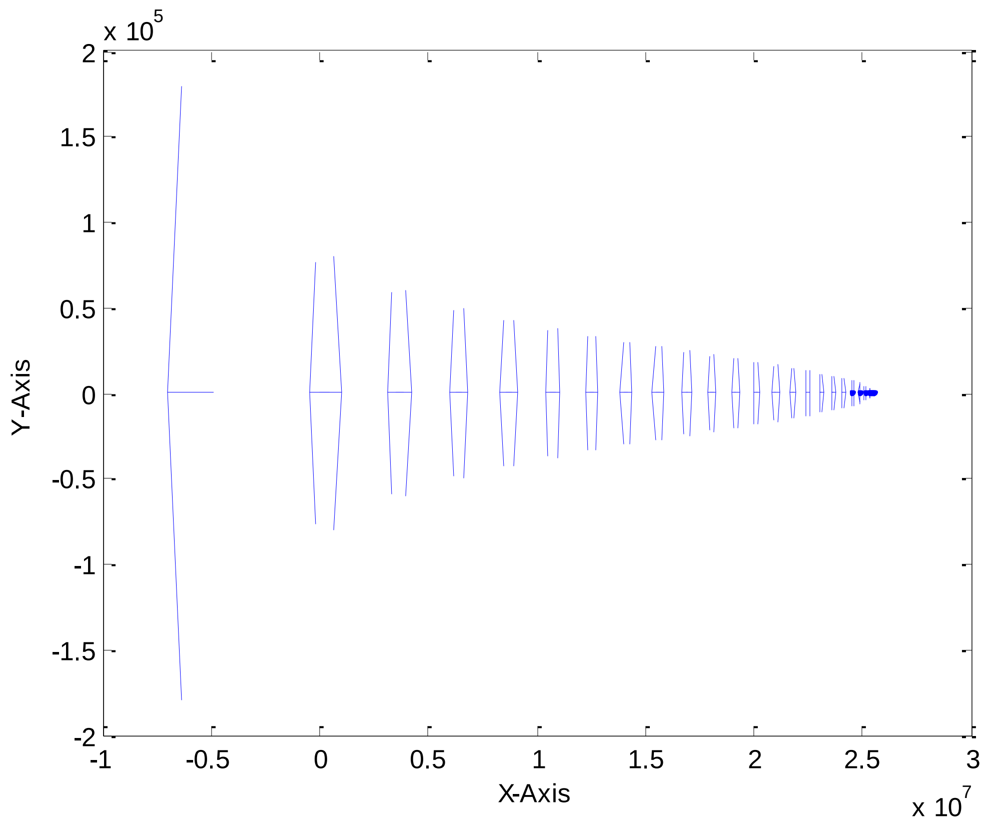
2. Initial_vel_x = 5000m/s



Now this graph shows an interesting observation. We had approximated the time of the simulation to be 2*ux/g (which is the formula for time period assuming constant gravitational acceleration). However, the effect of change is gravitational acceleration is clearly visible here because the height reached is comparable to the radius of the earth(64e5 m). Hence we need to increase the time of observation. The time of observation was increased to 2.5*ux/g. Then we get the following graph



3. Initial_vel_x = 10000m/s



In this case, the gravitational acceleration reduces to 0.406m/s^2 !. This is because of the large initial velocity given.

Time of simulation used is 20*ux/g.

Now we come to an interesting physical phenomena of ESCAPE VELOCITY.
If an object is released with the escape velocity from the surface of the earth then if can escape the gravitational field of the earth and never return back !
The escape velocity is calculated by assuming that at infinity the object has a gravitational potential energy of 0 and the kinetic energy is also 0(for just escaping the field of earth).
Hence by conservation of energy the gravitational PE + KE on surface of earth must be 0.

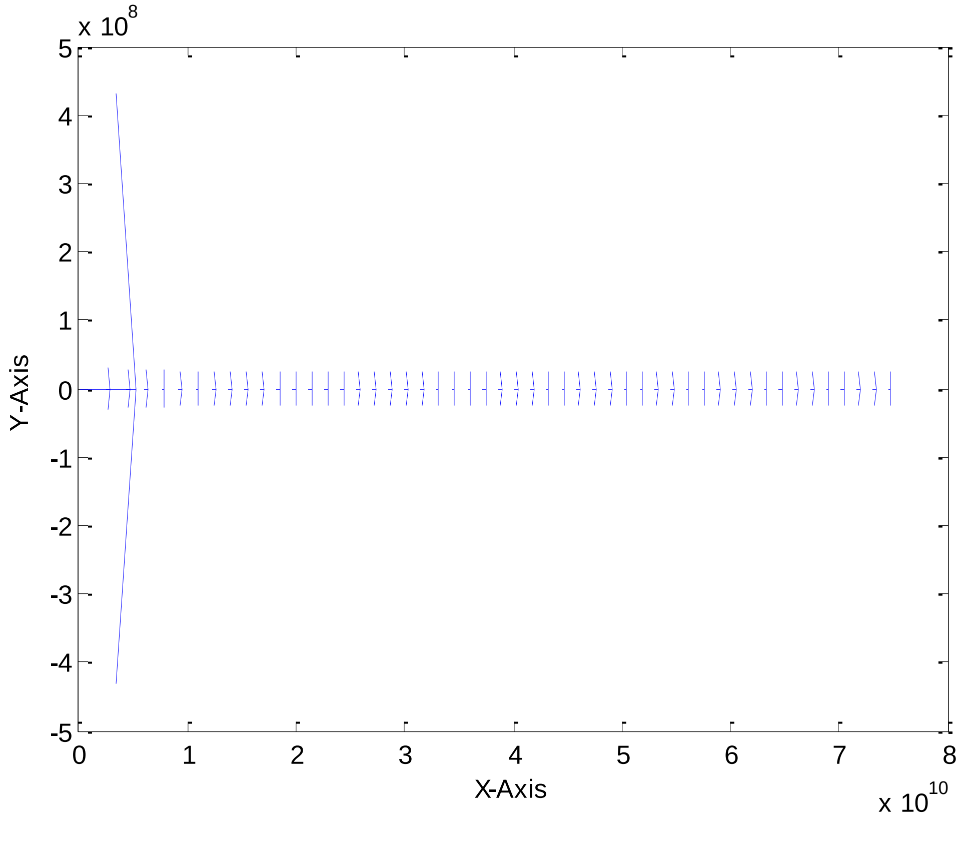
The formula for escape velocity is given below :

$$\frac{G m M_E}{R_E} = \frac{1}{2} m v^2$$

$$V_{escape} = \sqrt{\frac{2 G M_E}{R_E}}$$

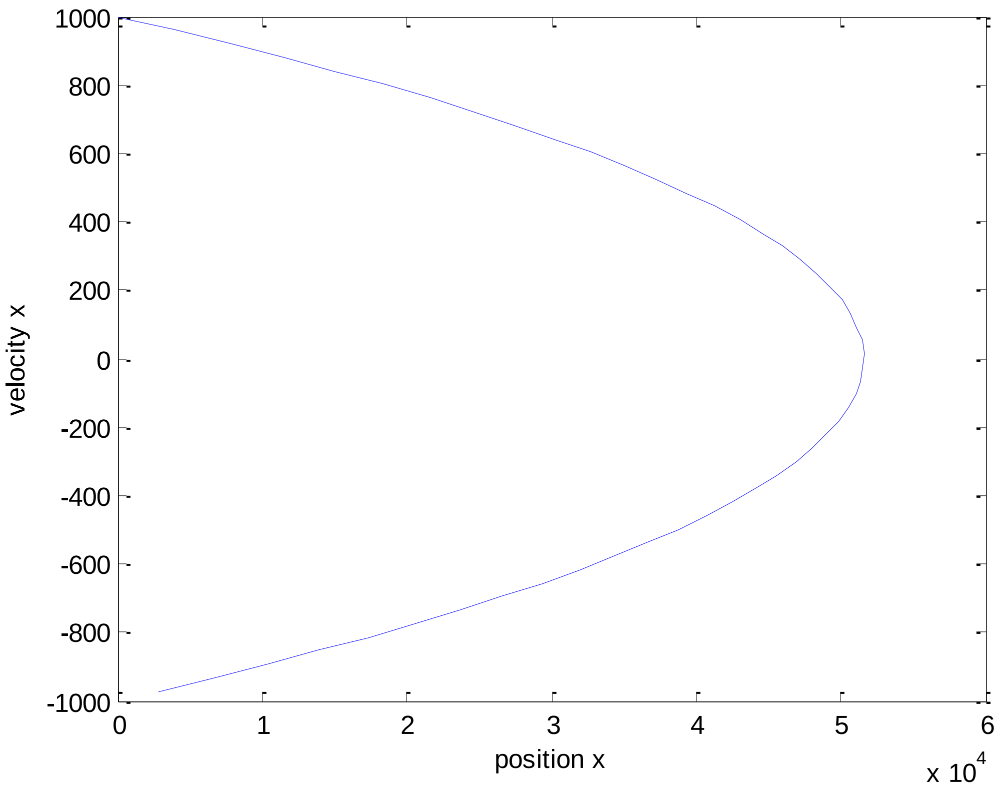
In case of Earth, the escape velocity is around 11.2 km/s. If a rocket is given this much velocity initially, it can escape the earth's gravitational field without the need for any further propellant. Note that here air drag is not considered.

Indeed if we simulate the situation , the ball never returns !!!

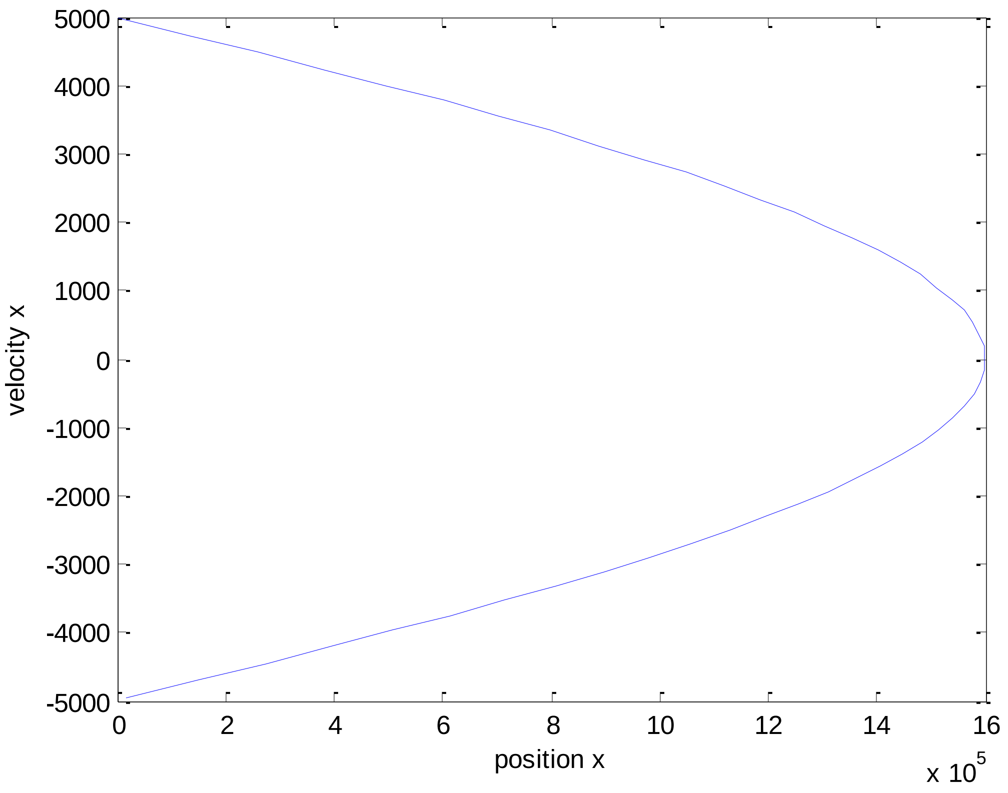


Phase space plots :

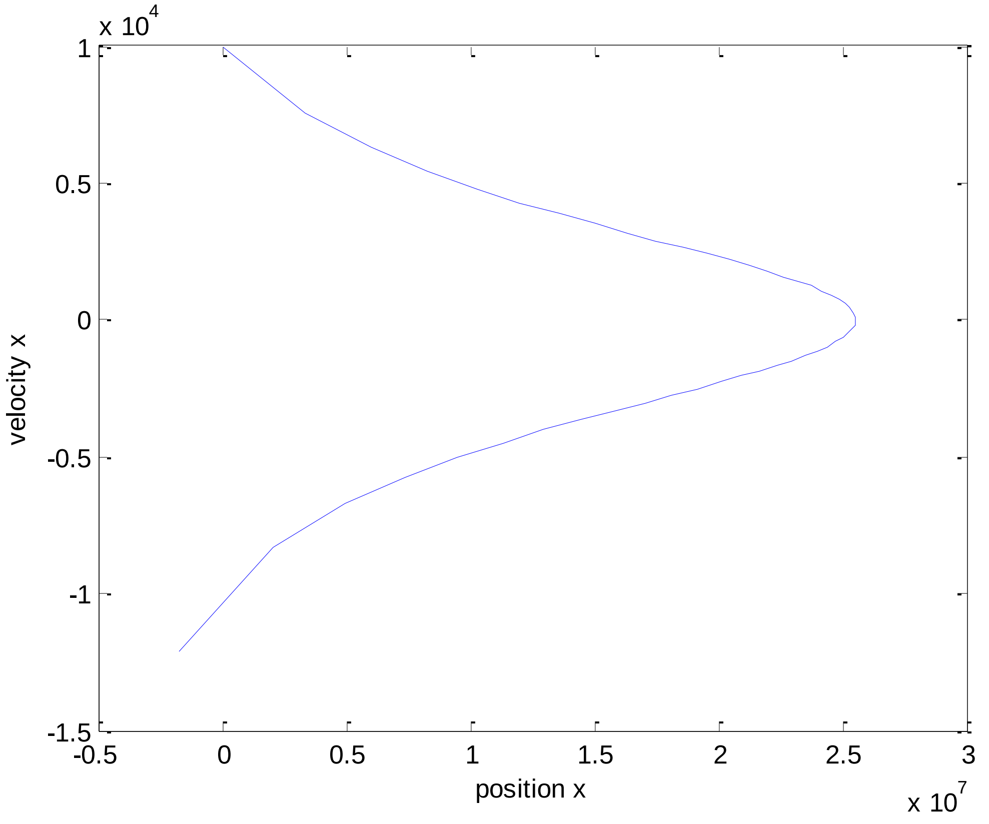
1. Initial_vel_x = 1000m/s



2. Initial_vel_x = 5000m/s

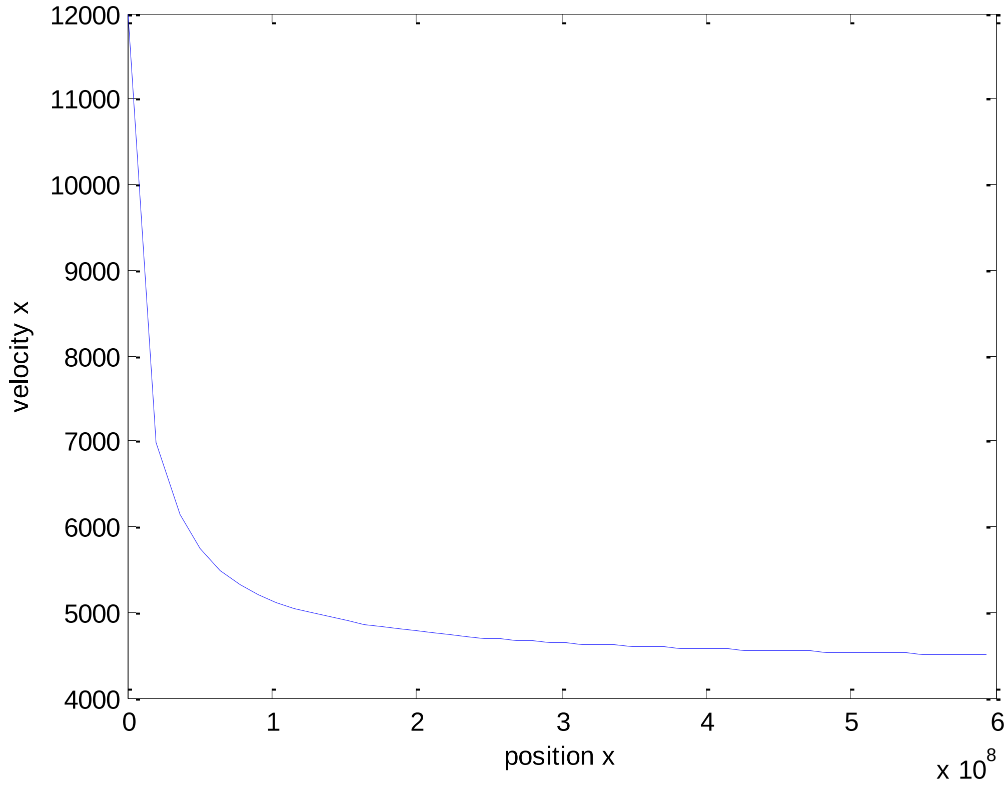


3. Initial_vel_x = 10000m/s



Note that the phase space plot is no longer a parabola because gravitational acceleration depends on x for higher altitudes.

Now if we simulate the case where the speed is slightly greater than the **escape velocity,** phase space diagram looks something like :



The time for which it is simulated is 100*ux/g
Even if we increase the time further we find that the ball never returns to the earth.

Code:

1. Main function

```
% this program simulates the motion of a ball as it is throen upwards in
% the air. We first experiment with low speeds of 1m/s, 10m/s, 40m/s and
% then with larger speeds of 1000m/s, 5000m/s and 10000m/s where the change
% in gravitational constant needs to be taken into account.
% Also we observe the concept of escape velocity by simulation.
% finnaly we make a movie of a bouncing ball
```

```
clear all;
close all;
global G M R ;
G = 6.67e-11;     %gravitational constant
M = 6e24;         % mass of earth
R = 6.4e6;        % radius of earth

initial_pos_x = 0;
initial_pos_y = 0;
initial_vel_x = 1;
initial_vel_y = 0;


start_time = 0;
end_time = 2*initial_vel_x/9.8;      % approximately
dt = (end_time - start_time) /40;

u0 = zeros(4 , 1);

u0(1) = initial_pos_x;
u0(2) = initial_vel_x;
u0(3) = initial_pos_y;
u0(4) = initial_vel_y;

figure
[ t, u ] = ode45(@rhs , start_time : dt : end_time, u0);


%{
% quiver plot
figure
quiver(u(:, 1) , u(: , 3) , u(: , 2) , u(: , 4) , 0.5)
xlabel('X-Axis')
ylabel('Y-Axis')
%}

%plot phase space trajectories.

%{
figure
plot(u(:, 1) , u(: , 2) )
xlabel('position x')
ylabel('velocity x')
%}
```

2. rhs.m

```
function  F = rhs( t ,u )
% This function is used to solve the equation given.

% u(1) = x
% u(2) = vx
% u(3) = y
% u(4) = vy
% F(1) = dx/dt
% F(2) = dvx/dt
% F(3) = dy/dt
% F(4) = dvy/dt

global G M R;

F = zeros(4 , 1);

 F(1) = u(2);
F(2) = -G*M/(power(u(1) + R , 2)) ;
F(3) = u(4);
F(4) = 0;

% this is for making a movie
%hold on
plot(u(1) , u(3) , 'o')
axis([0 0.06 -0.2 10])
pause(0.02);

end
```

MOVIE MAKING:

We tried to make a movie by using **movie2avi()** function. The code used was :

```
mov(1:nframe)=struct('cdata',[],'colormap',[]);
```

In the loop,

```
mov(i) = getframe(gcf);
```

After loop is finished,

```
movie2avi(mov, 'moviename1.avi', 'compression', 'None');
set(gca,'nextplot','replacechildren');
```

However we faced the problem of .avi file size. The size was 689MB !

Hence instead we **used VideoWriter object**.

The code for that is :

```
vidObj = VideoWriter('BouncingballMotion3_projectile.avi');
open(vidObj);

set(gca,'nextplot','replacechildren'); % to remove flickering in video
```

In the loop,

```
   currFrame = getframe;
   writeVideo(vidObj,currFrame);
```

After the loop is finished,

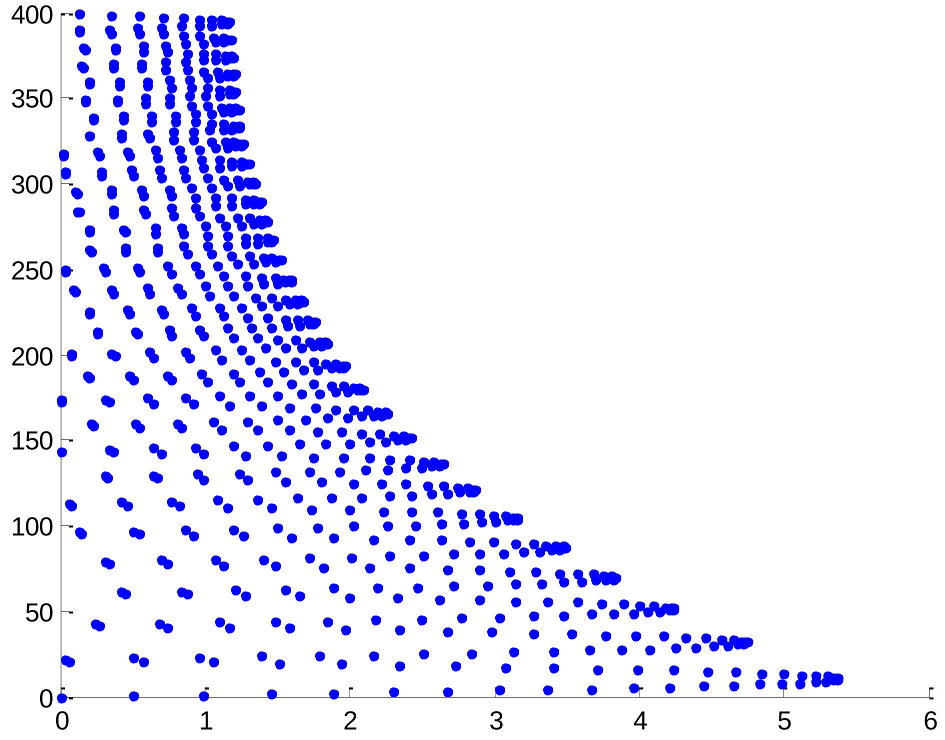
```
close(vidObj);
```

With the videowriter object, the size of the movie is around 6-7 MB.

BOUNCING CASE :

In order to simulate a bouncing ball, we did a small change in the code.
We used the following code :

```
for step=1 : niter-1
    position_x(step + 1 ) = position_x(step) + dt*velocity_x(step);
    velocity_x(step + 1 ) = velocity_x(step) + dt*(-G*M/(power(position_x(step) + R,2)));

    %this if statement handles the case when the ball hits the ground.

    if(position_x(step + 1) < 0 && velocity_x(step + 1) < 0)
       % velocity has been decreased by 0.9 in magnitude.
        velocity_x(step + 1) = velocity_x(step + 1)* -0.9 ;
    end

    position_y(step + 1 ) = position_y(step) + dt*velocity_y(step);
    velocity_y(step + 1 ) = velocity_y(step) + dt*0;
    plot(position_x(step) , position_y(step) , 'O')
       currFrame = getframe;
    writeVideo(vidObj,currFrame);
    axis([0 6 0 400])
    pause(0.01);
end
```



This graph shows the motion of a bouncing ball and how its height decreases gradually in magnitude.

The movie for this has been attached along with the pdf.