Physics Computer Modelling

Portfolio - Spring 2018

Moritz M. Konarski

American University of Central Asia

# Table of Contents

This portfolio includes the program code for the mentioned programs. It is advised to view the code in the Octave program because the formatting is made for that editor. Also, Octave provides color codes for different elements in the code, improving the readability greatly.

# Simple Programs

These programs were the first ones created, they are simple and don't do much useful stuff. They were an introduction to octave and meant to become familiar with the program.

## House

In these two programs houses are drawn using different easy methods to do so.

### Simple House

This program draws a simple house from one perspective using lines that are pre-set.

```
%creating Window and Axis
figure('units','norm','position',[.1 .1 .8 .7],'color',[1 1 1],'name',
      "House") %Creating Window
axes('position',[.1 .1 .8 .8],'color',[.9 .9 .9]) %Creating Axis
view(315,135) %Tilting coordinate system
grid on %Turning on Grid
a = axis('square', [-2 12 -2 8 -2 10]) %Squaring it up and setting limits
a = xlabel ("X") %labeling axis
a = ylabel ("Y") %labeling axis
a = zlabel ("Z") %labeling axis
a = title ("House") %labeling diagram

%drawing lines to create house

line([0 10],[6 6],[0 0],'linewidth',[2]) %Drawing Foundation
line([10 10],[0 6],[0 0],'linewidth',[2]) %Drawing Foundation
line([0 10],[0 0],[0 0],'linewidth',[2]) %Drawing Foundation
line([0 0],[0 6],[0 0],'linewidth',[2]) %Drawing Foundation
drawnow
line([10 10],[0 0],[0 5],'linewidth',[2]) %Drawing Walls
line([10 10],[6 6],[0 5],'linewidth',[2]) %Drawing Walls
line([0 0],[6 6],[0 5],'linewidth',[2]) %Drawing Walls
line([0 0],[0 0],[0 5],'linewidth',[2]) %Drawing Walls
drawnow
line([10 10],[.7 .7],[0 4],'linewidth',[2]) %Drawing Door
line([10 10],[2.7 2.7],[0 4],'linewidth',[2]) %Drawing Door
line([10 10],[.7 2.7],[4 4],'linewidth',[2]) %Drawing Door
drawnow
line([10 10],[5.5 5.5],[2 4],'linewidth',[2]) %Drawing Window by Door
line([10 10],[3.5 5.5],[2 2],'linewidth',[2]) %Drawing Window by Door
line([10 10],[3.5 5.5],[4 4],'linewidth',[2]) %Drawing Window by Door
```

```
line([10 10],[3.5 3.5],[2 4],'linewidth',[2]) %Drawing Window by Door
drawnow
line([0 0],[0 6],[5 5],'linewidth',[2]) %Drawing Ceiling
line([10 10],[0 6],[5 5],'linewidth',[2]) %Drawing Ceiling
drawnow
line([10 0],[3 3],[8 8],'linewidth',[2]) %Drawing roof
line([10 10],[-1 3],[4 8],'linewidth',[2]) %Drawing roof
line([10 10],[7 3],[4 8],'linewidth',[2]) %Drawing roof
line([0 0],[7 3],[4 8],'linewidth',[2]) %Drawing roof
line([0 0],[-1 3],[4 8],'linewidth',[2]) %Drawing roof
line([10 0],[-1 -1],[4 4],'linewidth',[2]) %Drawing roof
line([10 0],[7 7],[4 4],'linewidth',[2]) %Drawing roof
drawnow
line([8.5 1.5],[6 6],[4 4],'linewidth',[2]) %Drawing Window by Door
line([8.5 8.5],[6 6],[2 4],'linewidth',[2]) %Drawing Window by Door
line([1.5 1.5],[6 6],[2 4],'linewidth',[2]) %Drawing Window by Door
line([1.5 8.5],[6 6],[2 2],'linewidth',[2]) %Drawing Window by Door
drawnow
```

## Block House

This program creates a house out of blocks that are stacked on top of each other using for-loops to create the individual blocks.

```
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],'name',
      "Cube of Cubes") %Creating Window
axes('position',[.1 .1 .8 .8],'color',[1 1 1]) %Creating Axis
view(325,135)       %Tilting coordinate system
grid on             %turning on the grid in coordinate system
xlabel ("X-Axis")   %labeling the axis
ylabel ("Y-Axis")   %labeling the axis
zlabel ("Z-Axis")   %labeling the axis
title ("Block House")

%---------------changeable variables---------------------

l=1;                %length of side of one cube
n=4;                %number of cubes per side of big cube (whole numbers
      only)
t=1;                %line thickness of cubes

%---------------------------------------------------------

f=(n*l)-l;          %limit for loop repetition
S=0;                %for stacking the cubes
w=0;                %for stacking the cubes
h=0;                %for stacking the cubes

axis('equal',[-1 n*l+1 -1 n*l+1 -1 n*l+1]) %preparing coordinate system
      limits
```

To Table of Contents

```
for h=0:l:f         %shifting cubes in +z direction


    for S=0:l:f     %shifting cubes in +x direction

    %--------------drawing one cube----------------------

    line([l+S l+S],[0+w l+w],[0+h 0+h], 'linewidth', [t])
    line([0+S 0+S],[0+w l+w],[0+h 0+h], 'linewidth', [t])
    line([0+S l+S],[0+w 0+w],[0+h 0+h], 'linewidth', [t])
    line([0+S l+S],[l+w l+w],[0+h 0+h], 'linewidth', [t])
    line([l+S l+S],[0+w l+w],[l+h l+h], 'linewidth', [t])
    line([0+S 0+S],[0+w l+w],[l+h l+h], 'linewidth', [t])
    line([0+S l+S],[0+w 0+w],[l+h l+h], 'linewidth', [t])
    line([0+S l+S],[l+w l+w],[l+h l+h], 'linewidth', [t])
    line([0+S 0+S],[0+w 0+w],[0+h l+h], 'linewidth', [t])
    line([0+S 0+S],[l+w l+w],[0+h l+h], 'linewidth', [t])
    line([l+S l+S],[0+w 0+w],[0+h l+h], 'linewidth', [t])
    line([l+S l+S],[l+w l+w],[0+h l+h], 'linewidth', [t])

    line([0+S+.25 0+S+.25],[l+w l+w],[-.25+l+h -.75+l+h], 'linewidth', [t])
    line([0+S+.75 0+S+.75],[l+w l+w],[-.25+l+h -.75+l+h], 'linewidth', [t])
    line([0+S+.25 0+S+.75],[l+w l+w],[-.25+l+h -.25+l+h], 'linewidth', [t])
    line([0+S+.25 0+S+.75],[l+w l+w],[-.75+l+h -.75+l+h], 'linewidth', [t])

    drawnow %drawing the cube that was just created into coordinate system

    endfor

  endfor
```

## Plotted Functions

This program plots five functions that were given in individual windows using the `subplot` command to create smaller windows inside the main window.

```
%------------preparation-----------------------------------------------------
      ------------------------------
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],'name',
      "Drawing Functions") %Creating Window
axes('position',[.1 .1 .8 .8],'color',[1 1 1]) %Creating Axis
grid on                   %turning on grid in coordinate system
xlabel ("X-Axis")         %labeling the axis
ylabel ("Y-Axis")         %labeling the axis
zlabel ("Z-Axis")         %labeling the axis

%---Sinus
      function---------------------------------------------------------------
      --------------------------
```

```
subplot (2,3,1);          %create plot
drawnow;                  %draw this plot
x = -2*pi:0.01:2*pi;      %set x limits
plot (x, sin (5*x));      %plot the function
grid on;                  %turn on the grid
xlabel ("t");             %label x axis
ylabel ("sin(5t)");       %label y axis
title ("y=sin(5t)");      %label plot
axis('equal');            %set axis to equal proportions
drawnow;
%---e-function-----------------------------------------------------------------------------------------
subplot (2,3,2);          %See Sinus Function for more info
drawnow;
x = -2:0.01:2;
plot (x, e.^(-4*x.^(2)));
grid on;
xlabel ("t");
ylabel ("e(-4t^2)");
title ("y=e(-4t^2)");
axis('equal');
drawnow;

%---Root
      function---------------------------------------------------------------------------------
subplot (2,3,3);          %See Sinus Function for more info
drawnow;
x = -1:0.01:1;
plot (x, 1./(sqrt(1-x.^(2))));
grid on;
xlabel ("t");
ylabel ("1/(sqrt(1-t^2))");
title ("y=1/(sqrt(1-t^2))");
axis('equal');
drawnow;

%---Arctan
      function-------------------------------------------------------------------------------
subplot (2,3,4);          %See Sinus Function for more info
drawnow;
x = -7:0.01:7;
plot (x, atan(x));
grid on;
xlabel ("t");
ylabel ("arctan(x)");
title ("y=arctan(x)");
axis('equal');
drawnow;
```

To Table of Contents

```
%---Curve of second
      order-----------------------------------------------------------------
      -----------------
subplot (2,3,5);          %See Sinus Function for more info
drawnow;
x = -7:0.5:7;
plot (x, 3.*x.^2+2.*x+1);
grid on;
xlabel ("t");
ylabel ("3t^2+2t+1");
title ("y=3t^2+2t+1");
ylimit ([-7 50]);
axis('equal');
drawnow;
```

## Stacking Cubes

A program that draws a big cube that is built of smaller cubes that get stacked. This is done using for-loops to draw the cubes. The number of cubes in the big cube as well as the length of the sides of the small cubes can be changed inside the code.

```
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],'name',
      "Cube of Cubes") %Creating Window
axes('position',[.1 .1 .8 .8],'color',[1 1 1]) %Creating Axis
view(325,135)       %Tilting coordinate system
grid on             %turning on the grid in coordinate system
xlabel ("X-Axis")   %labeling the axis
ylabel ("Y-Axis")   %labeling the axis
zlabel ("Z-Axis")   %labeling the axis

%---------------changeable variables---------------------

l=1;                %length of side of one cube
n=4;                %number of cubes per side of big cube (whole numbers
      only)
t=1;                %line thickness of cubes

%---------------------------------------------------------

f=(n*l)-l;          %limit for loop repetition
S=0;                %for stacking the cubes
w=0;                %for stacking the cubes
h=0;                %for stacking the cubes

axis('equal',[-1 n*l+1 -1 n*l+1 -1 n*l+1]) %preparing coordinate system
      limits

for h=0:l:f         %shifting cubes in +z direction

  for w=0:l:f       %shifting cubes in +y direction
```

```
    for S=0:l:f     %shifting cubes in +x direction

    %--------------drawing one cube----------------------

    line([l+S l+S],[0+w l+w],[0+h 0+h], 'linewidth', [t])
    line([0+S 0+S],[0+w l+w],[0+h 0+h], 'linewidth', [t])
    line([0+S l+S],[0+w 0+w],[0+h 0+h], 'linewidth', [t])
    line([0+S l+S],[l+w l+w],[0+h 0+h], 'linewidth', [t])
    line([l+S l+S],[0+w l+w],[l+h l+h], 'linewidth', [t])
    line([0+S 0+S],[0+w l+w],[l+h l+h], 'linewidth', [t])
    line([0+S l+S],[0+w 0+w],[l+h l+h], 'linewidth', [t])
    line([0+S l+S],[l+w l+w],[l+h l+h], 'linewidth', [t])
    line([0+S 0+S],[0+w 0+w],[0+h l+h], 'linewidth', [t])
    line([0+S 0+S],[l+w l+w],[0+h l+h], 'linewidth', [t])
    line([l+S l+S],[0+w 0+w],[0+h l+h], 'linewidth', [t])
    line([l+S l+S],[l+w l+w],[0+h l+h], 'linewidth', [t])

    drawnow %drawing the cube that was just created into coordinate system

    endfor

  endfor

endfor
```

# Solar System

This group of programs simulates the solar system we live in. The programs use for-loops, the angular velocities and radii of the planets to calculate their position, and then draws them. The programs simulate the Sun (yellow), Venus (dark red), Mercury (green), Earth (blue), the Moon (grey), Mars (red), a moon of Mars (grey), Jupiter (orange) and four of Jupiter's moons (grey). The radii and the angular velocities were given for each of the planets.
The code is basically the same in all programs, it is written in such a way that the user can enter the velocity and radius of any planet as the data of the sun, and then the planet whose data is now the sun's is at the center of the system. This works because all the planets are displayed relative to the sun. If the user changes the sun, it is in a different position, while the planets are still in the same relative motion to it.

## Heliocentric

This program simulates the heliocentric solar system: it has the sun at its center.

```
%------------preparation-----------------------------------------------------
      ------------------------------------
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],'name',
      "Heliocentric"); %Creating Window
```

```
axes('position',[.1 .1 .8 .8],'color',[1 1 1], 'xlim',[-13 13],'ylim',[-13
      13]);  %Creating Axis
grid off;                                        %turning on grid in
      coordinate system
xlabel ("X-Axis");                               %labeling the axis
ylabel ("Y-Axis");                               %labeling the axis
zlabel ("Z-Axis");                               %labeling the axis
axis ('square');                                 %squaring axis
drawnow;

%----------------Changeable
      Variables-------------------------------------------------------------
      ---------------
%---------------------------------------------------------------------------
      -------------------------------------
n=1;            %Number of Revolutions of earth/number of years shown
i=.05;          %Interval during for-loop
g=0;            %Set to >1 to deactivate lines of planets

%--------------pre-calculations-------------------------------------------
      -------------------------------------
l=n.*2.*pi;     %limit for for-loop

%----------------Preparing the
      Planets---------------------------------------------------------------
      ------------
%------------w[x] is the angular speed; r[x] is the radius relative to the
      sun-----------------------------------
%----To adapt this system to center around other planets, set ws and rs for
      the sun to the ones of the desired planet
%------------Drawing
      Sun-------------------------------------------------------------------
      ---------------------
h_Sun=line(0,0,0,'marker','o','markersize',[17],'markerfacecolor','yellow','c
      olor','yellow'); %creating the ball
ws=0;              %angular speed
rs=0;              %orbit radius

%------------Drawing
      Mercure---------------------------------------------------------------
      ------------------------
h_Mercure=line(0,0,0,'marker','o','markersize',[7],'color',[.7 0
      0],'markerfacecolor',[.7 0 0]); %creating the ball
wm=2;              %angular speed mercure
rm=3;              %orbit radius mercure

%------------Drawing
      Venus-----------------------------------------------------------------
      ---------------------
h_Venus=line(0,0,0,'marker','o','markersize',[10],'color',[0 .7
      0],'markerfacecolor',[0 .7 0]); %creating the ball
```

```
wv=1.5;             %angular speed venus
rv=4;               %orbit radius venus

%------------Drawing
      Earth---------------------------------------------------------------
      ----------------------
h_Earth=line(0,0,0,'marker','o','markersize',[10],'color',[0 0
      1],'markerfacecolor',[0 0 1]); %creating the ball
we=1;               %angular speed earth
re=5;               %orbit radius earth

%------------Drawing
      Moon----------------------------------------------------------------
      -----------------------
h_Moon=line(0,0,0,'marker','o','markersize',[5],'color',[.3 .3
      .3],'markerfacecolor',[.3 .3 .3]); %creating the ball
wem=12;             %angular speed Moon
rem=.7;             %orbit radius moon

%------------Drawing
      Mars----------------------------------------------------------------
      -----------------------
h_Mars=line(0,0,0,'marker','o','markersize',[9],'color',[1 0
      0],'markerfacecolor',[1 0 0]); %creating the ball
wma=.8;             %angular speed Mars
rma=7;              %orbit radius Mars

%------------Drawing
      Marsmoom------------------------------------------------------------
      ---------------------------
h_Marsmoon=line(0,0,0,'marker','o','markersize',[4.5],'color',[0 0
      0],'markerfacecolor',[0 0 0]); %creating the ball
wmam=10;             %angular speed Marsmoon
rmam=.5;             %orbit radius Marsmoon

%------------Drawing
      Jupiter-------------------------------------------------------------
      --------------------------
h_Jupiter=line(0,0,'marker','o','markersize',[14],'color',[1 .5
      .3],'markerfacecolor',[1 .5 .3]);
wj=.5;              %angular speed jupiter
rj=11;              %orbit radius jupiter

%------------Drawing
      MoonJ1--------------------------------------------------------------
      -------------------------
h_MoonJ1=line(0,0,'marker','o','markersize',[5],'color',[.3 .3
      .3],'markerfacecolor',[.3 .3 .3]);
wmj1=10;             %angular speed Moon
rmj1=.6;             %orbit radius moon
```

```
%------------Drawing
      MoonJ2-----------------------------------------------------------------
      -------------------------
h_MoonJ2=line(0,0,'marker','o','markersize',[5],'color',[0 .3
      .3],'markerfacecolor',[0 .3 .3]);
wmj2=13;             %angular speed Moon
rmj2=1;               %orbit radius moon

%------------Drawing
      MoonJ3-----------------------------------------------------------------
      -------------------------
h_MoonJ3=line(0,0,'marker','o','markersize',[5],'color',[.3 0
      .3],'markerfacecolor',[.3 0 .3]);
wmj3=12;             %angular speed Moon
rmj3=.9;             %orbit radius moon

%------------Drawing
      MoonJ4-----------------------------------------------------------------
      -------------------------
h_MoonJ4=line(0,0,'marker','o','markersize',[5],'color',[.3 0
      .6],'markerfacecolor',[.3 0 .6]);
wmj4=9;               %angular speed Moon
rmj4=1.2;             %orbit radius moon

drawnow
%------------Making planets
      move-------------------------------------------------------------------
      -----------------
%----------------------------------------------------------------------------
      --------------------------------------
for t=0:i:l                                 %setting up loop for one
      rotation

  %----Sun----
  xs=rs*cos(ws*t);                                %getting x for sun
  ys=rs*sin(ws*t);                                %getting y for sun
  set(h_Sun,'xdata',xs,'ydata',ys);               %putting sun there

  %----Mercure----
  xm=rm*cos(wm*t);                                %getting x for mercure
  ym=rm*sin(wm*t);                                %getting y for mercure
  set(h_Mercure,'xdata',xm-xs,'ydata',ym-ys);     %putting mercure there

  %----Venus----
  xv=rv*cos(wv*t);                                %getting x for venus
  yv=rv*sin(wv*t);                                %getting y for venus
  set(h_Venus,'xdata',xv-xs,'ydata',yv-ys);       %putting venus there

  %----Earth----
  xe=re*cos(we*t);                                %getting x for earth
  ye=re*sin(we*t);                                %getting y for earth
```

```
set(h_Earth,'xdata',xe-xs,'ydata',ye-ys);          %putting earth there

%----Moon----
xem=rem*cos(wem*t)+xe;                             %getting x for moon
yem=rem*sin(wem*t)+ye;                             %getting y for moon
set(h_Moon,'xdata',xem-xs,'ydata',yem-ys);         %putting moon there

%----Mars----
xma=rma*cos(wma*t);                                %getting x for mars
yma=rma*sin(wma*t);                                %getting y for mars
set(h_Mars,'xdata',xma-xs,'ydata',yma-ys);         %putting mars there

%----Marsmoon----
xmam=rmam*cos(wmam*t)+xma;                         %getting x for marsmoon
ymam=rmam*sin(wmam*t)+yma;                         %getting y for marsmoon
set(h_Marsmoon,'xdata',xmam-xs,'ydata',ymam-ys);   %putting marsmoon there

%----Jupiter----
xj=rj*cos(wj*t);                                   %getting x for jupiter
yj=rj*sin(wj*t);                                   %getting y for jupiter
set(h_Jupiter,'xdata',xj-xs,'ydata',yj-ys);        %putting jupiter there

%----MoonJ1----
xmj1=rmj1*cos(wmj1*t)+xj;                          %getting x for
    jupitermoon
ymj1=rmj1*sin(wmj1*t)+yj;                          %getting y for
    jupitermoon
set(h_MoonJ1,'xdata',xmj1-xs,'ydata',ymj1-ys);     %putting jupitermoon
    there

%----MoonJ2----
xmj2=rmj2*cos(wmj2*t)+xj;                          %getting x for
    jupitermoon
ymj2=rmj2*sin(wmj2*t)+yj;                          %getting x for
    jupitermoon
set(h_MoonJ2,'xdata',xmj2-xs,'ydata',ymj2-ys);     %putting jupitermoon
    there

%----MoonJ3----
xmj3=rmj3*cos(wmj3*t)+xj;                          %getting x for
    jupitermoon
ymj3=rmj3*sin(wmj3*t)+yj;                          %getting x for
    jupitermoon
set(h_MoonJ3,'xdata',xmj3-xs,'ydata',ymj3-ys);     %putting jupitermoon
    there

%----MoonJ4----
xmj4=rmj4*cos(wmj4*t)+xj;                          %getting x for
    jupitermoon
ymj4=rmj4*sin(wmj4*t)+yj;                          %getting x for
    jupitermoon
```

```
  set(h_MoonJ4,'xdata',xmj4-xs,'ydata',ymj4-ys);     %putting jupitermoon
      there

  drawnow;

  if (mod(t,0.5.*i)==g)  %Trails of the planets in an interval

    line(xs,ys,'marker','.','markersize',[7]);    %Sun
    line(xm-xs,ym-ys,'marker','.','markersize',[7]);  %Mercure
    line(xv-xs,yv-ys,'marker','.','markersize',[7]);  %Venus
    line(xe-xs,ye-ys,'marker','.','markersize',[7]);  %Earth
    line(xma-xs,yma-ys,'marker','.','markersize',[7]);%Mars
    line(xj-xs,yj-ys,'marker','.','markersize',[9]);  %Jupiter

  endif

endfor
```

## Heliocentric With Realistic Distances

This program simulates the solar system from the heliocentric perspective, but with distances and speeds that are in a realistic ratio to each other.

```
%------------preparation-----------------------------------------------------
      ------------------------------------
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],'name',
      "Realistic"); %Creating Window
axes('position',[.1 .1 .8 .8],'color',[1 1 1], 'xlim',[-18 18],'ylim',[-18
      18]);  %Creating Axis
grid off;                                        %turning on grid in
      coordinate system
xlabel ("X-Axis");                               %labeling the axis
ylabel ("Y-Axis");                               %labeling the axis
zlabel ("Z-Axis");                               %labeling the axis
axis ('square');                                 %squaring axis
drawnow

%----------------Changeable
      Variables-----------------------------------------------------------------
      ---------------
%---------------------------------------------------------------------------
      ------------------------------------
n=1;             %Number of Revolutions of earth/number of years shown
i=.05;           %Interval during for-loop
g=0;            %Set to >1 to deactivate lines

%---------------pre-calculations---------------------------------------------
      ------------------------------------
l=n.*2.*pi;     %limit for for-loop
```

To Table of Contents

```
%-----------------Preparing the
      Planets--------------------------------------------------------------------
      ------------
%------------w[x] is the angular speed; r[x] is the radius relative to the
      sun-----------------------------------
%----To adapt this system, set ws and rs for the sun to the ones of the
      desired planet
%------------Drawing
      Sun------------------------------------------------------------------------
      ----------------------
h_Sun=line(0,0,0,'marker','o','markersize',[15],'markerfacecolor','yellow','c
      olor','yellow'); %creating the ball
ws=0;               %angular speed
rs=0;               %orbit radius

%------------Drawing
      Mercure--------------------------------------------------------------------
      ------------------------
h_Mercure=line(0,0,0,'marker','o','markersize',[5],'color',[.7 0
      0],'markerfacecolor',[.7 0 0]); %creating the ball
wm=4.1;               %angular speed mercure
rm=1.2;               %orbit radius mercure

%------------Drawing
      Venus----------------------------------------------------------------------
      ----------------------
h_Venus=line(0,0,0,'marker','o','markersize',[6],'color',[0 .7
      0],'markerfacecolor',[0 .7 0]); %creating the ball
wv=1.6;             %angular speed venus
rv=2.1;               %orbit radius venus

%------------Drawing
      Earth----------------------------------------------------------------------
      ----------------------
h_Earth=line(0,0,0,'marker','o','markersize',[8],'color',[0 0
      1],'markerfacecolor',[0 0 1]); %creating the ball
we=1;               %angular speed earth
re=3;               %orbit radius earth

%------------Drawing
      Moon-----------------------------------------------------------------------
      -----------------------
h_Moon=line(0,0,0,'marker','o','markersize',[3],'color',[.3 .3
      .3],'markerfacecolor',[.3 .3 .3]); %creating the ball
wem=12;             %angular speed Moon
rem=.7;             %orbit radius moon

%------------Drawing
      Mars-----------------------------------------------------------------------
      -----------------------
```

```
h_Mars=line(0,0,0,'marker','o','markersize',[7],'color',[1 0
      0],'markerfacecolor',[1 0 0]); %creating the ball
wma=.53;             %angular speed Mars
rma=4.5;              %orbit radius Mars

%------------Drawing
      Marsmoom-------------------------------------------------------------------
      ---------------------------
h_Marsmoon=line(0,0,0,'marker','o','markersize',[2.5],'color',[0 0
      0],'markerfacecolor',[0 0 0]); %creating the ball
wmam=10;             %angular speed Marsmoon
rmam=.5;             %orbit radius Marsmoon

%------------Drawing
      Jupiter--------------------------------------------------------------------
      --------------------------
h_Jupiter=line(0,0,'marker','o','markersize',[12],'color',[1 .5
      .3],'markerfacecolor',[1 .5 .3]);
wj=.09;              %angular speed jupiter
rj=15.6;              %orbit radius jupiter

%------------Drawing
      MoonJ1---------------------------------------------------------------------
      -------------------------
h_MoonJ1=line(0,0,'marker','o','markersize',[3],'color',[.3 .3
      .3],'markerfacecolor',[.3 .3 .3]);
wmj1=10;             %angular speed Moon
rmj1=.6;             %orbit radius moon

%------------Drawing
      MoonJ2---------------------------------------------------------------------
      -------------------------
h_MoonJ2=line(0,0,'marker','o','markersize',[3],'color',[0 .3
      .3],'markerfacecolor',[0 .3 .3]);
wmj2=13;             %angular speed Moon
rmj2=1;               %orbit radius moon

%------------Drawing
      MoonJ3---------------------------------------------------------------------
      -------------------------
h_MoonJ3=line(0,0,'marker','o','markersize',[3],'color',[.3 0
      .3],'markerfacecolor',[.3 0 .3]);
wmj3=12;             %angular speed Moon
rmj3=.9;             %orbit radius moon

%------------Drawing
      MoonJ4---------------------------------------------------------------------
      -------------------------
h_MoonJ4=line(0,0,'marker','o','markersize',[3],'color',[.3 0
      .6],'markerfacecolor',[.3 0 .6]);
wmj4=9;               %angular speed Moon
```

```
rmj4=1.2;            %orbit radius moon

drawnow
%------------Making planets
      move----------------------------------------------------------------
      -----------------
%----------------------------------------------------------------------------
      ---------------------------------------
for t=0:i:l                                    %setting up loop for one
      rotation

  %----Sun----
  xs=rs*cos(ws*t);                             %getting x for sun
  ys=rs*sin(ws*t);                             %getting y for sun
  set(h_Sun,'xdata',xs,'ydata',ys);            %putting sun there

  %----Mercure----
  xm=rm*cos(wm*t);                             %getting x for mercure
  ym=rm*sin(wm*t);                             %getting y for mercure
  set(h_Mercure,'xdata',xm-xs,'ydata',ym-ys);        %putting mercure
      there

  %----Venus----
  xv=rv*cos(wv*t);                             %getting x for venus
  yv=rv*sin(wv*t);                             %getting y for venus
  set(h_Venus,'xdata',xv-xs,'ydata',yv-ys);          %putting venus
      there

  %----Earth----
  xe=re*cos(we*t);                             %getting x for earth
  ye=re*sin(we*t);                             %getting y for earth
  set(h_Earth,'xdata',xe-xs,'ydata',ye-ys);          %putting earth
      there

  %----Moon----
  xem=rem*cos(wem*t)+xe;                          %getting x for moon
  yem=rem*sin(wem*t)+ye;                          %getting y for moon
  set(h_Moon,'xdata',xem-xs,'ydata',yem-ys);            %putting moon
      there

  %----Mars----
  xma=rma*cos(wma*t);                          %getting x for mars
  yma=rma*sin(wma*t);                          %getting y for mars
  set(h_Mars,'xdata',xma-xs,'ydata',yma-ys);         %putting mars
      there

  %----Marsmoon----
  xmam=rmam*cos(wmam*t)+xma;                          %getting x for
      marsmoon
  ymam=rmam*sin(wmam*t)+yma;                          %getting y for
      marsmoon
```

```
set(h_Marsmoon,'xdata',xmam-xs,'ydata',ymam-ys);               %putting
    marsmoon there

%----Jupiter----
xj=rj*cos(wj*t);                                          %getting x for jupiter
yj=rj*sin(wj*t);                                          %getting y for jupiter
set(h_Jupiter,'xdata',xj-xs,'ydata',yj-ys);                    %putting jupiter
    there

%----MoonJ1----
xmj1=rmj1*cos(wmj1*t)+xj;                                      %getting x for
    jupitermoon
ymj1=rmj1*sin(wmj1*t)+yj;                                      %getting y for
    jupitermoon
set(h_MoonJ1,'xdata',xmj1-xs,'ydata',ymj1-ys);                       %putting
    jupitermoon there

%----MoonJ2----
xmj2=rmj2*cos(wmj2*t)+xj;                                      %getting x for
    jupitermoon
ymj2=rmj2*sin(wmj2*t)+yj;                                      %getting x for
    jupitermoon
set(h_MoonJ2,'xdata',xmj2-xs,'ydata',ymj2-ys);                       %putting
    jupitermoon there

%----MoonJ3----
xmj3=rmj3*cos(wmj3*t)+xj;                                      %getting x for
    jupitermoon
ymj3=rmj3*sin(wmj3*t)+yj;                                      %getting x for
    jupitermoon
set(h_MoonJ3,'xdata',xmj3-xs,'ydata',ymj3-ys);                       %putting
    jupitermoon there

%----MoonJ4----
xmj4=rmj4*cos(wmj4*t)+xj;                                      %getting x for
    jupitermoon
ymj4=rmj4*sin(wmj4*t)+yj;                                      %getting x for
    jupitermoon
set(h_MoonJ4,'xdata',xmj4-xs,'ydata',ymj4-ys);        %putting jupitermoon
    there

drawnow;

if (mod(t,0.5.*i)==g)     %Trails of Planets

  line(xs,ys,'marker','.','markersize',[7]);   %Sun
  line(xm-xs,ym-ys,'marker','.','markersize',[7]);   %Mercure
  line(xv-xs,yv-ys,'marker','.','markersize',[7]);   %Venus
  line(xe-xs,ye-ys,'marker','.','markersize',[7]);   %Earth
  line(xma-xs,yma-ys,'marker','.','markersize',[7]);   %Mars
  line(xj-xs,yj-ys,'marker','.','markersize',[7]);   %Jupiter
```

To Table of Contents

```
  endif

endfor
```

## Geocentric

This program simulates the solar system having the earth at its center. You notice that the planets have really strange paths now. The programs is the same as the first heliocentric one, the only difference is that the speed and radius of the sun are set to those of earth.

```
%------------preparation-----------------------------------------------------
      ------------------------------------
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],'name',
      "Geocentric"); %Creating Window
axes('position',[.1 .1 .8 .8],'color',[1 1 1], 'xlim',[-18 18],'ylim',[-18
      18]);  %Creating Axis
grid off;                                          %turning on grid in
      coordinate system
xlabel ("X-Axis");                                 %labeling the axis
ylabel ("Y-Axis");                                 %labeling the axis
zlabel ("Z-Axis");                                 %labeling the axis
axis ('square');                                   %squaring axis
drawnow                                  %Tilting coordinate system


%----------------Changeable
      Variables-----------------------------------------------------------------
      ---------------
%---------------------------------------------------------------------------
      -------------------------------------
n=1;              %Number of Revolutions of earth/number of years shown
i=.05;            %Interval during for-loop
g=0;              %Set to >1 to deactivate lines

%---------------pre-calculations---------------------------------------------
      -------------------------------------
l=n.*2.*pi;     %limit for for-loop

%----------------Preparing the
      Planets-------------------------------------------------------------------
      ------------
%------------w[x] is the angular speed; r[x] is the radius relative to the
      sun-----------------------------------
%----To adapt this system, set ws and rs for the sun to the ones of the
      desired planet
%------------Drawing
      Sun-----------------------------------------------------------------------
      ----------------------
```

```
h_Sun=line(0,0,0,'marker','o','markersize',[18],'markerfacecolor','yellow','c
      olor','yellow'); %creating the ball
ws=1;              %angular speed
rs=5;              %orbit radius

%------------Drawing
      Mercure-------------------------------------------------------------------
      ------------------------
h_Mercure=line(0,0,0,'marker','o','markersize',[7],'color',[.7 0
      0],'markerfacecolor',[.7 0 0]); %creating the ball
wm=2;              %angular speed mercure
rm=3;              %orbit radius mecure

%------------Drawing
      Venus---------------------------------------------------------------------
      ----------------------
h_Venus=line(0,0,0,'marker','o','markersize',[8],'color',[0 .7
      0],'markerfacecolor',[0 .7 0]); %creating the ball
wv=1.5;            %angular speed venus
rv=4;              %orbit radius venus

%------------Drawing
      Earth---------------------------------------------------------------------
      ----------------------
h_Earth=line(0,0,0,'marker','o','markersize',[10],'color',[0 0
      1],'markerfacecolor',[0 0 1]); %creating the ball
we=1;              %angular speed earth
re=5;              %orbit radius earth

%------------Drawing
      Moon----------------------------------------------------------------------
      -----------------------
h_Moon=line(0,0,0,'marker','o','markersize',[5],'color',[.3 .3
      .3],'markerfacecolor',[.3 .3 .3]); %creating the ball
wem=12;            %angular speed Moon
rem=.7;            %orbit radius moon

%------------Drawing
      Mars----------------------------------------------------------------------
      -----------------------
h_Mars=line(0,0,0,'marker','o','markersize',[9],'color',[1 0
      0],'markerfacecolor',[1 0 0]); %creating the ball
wma=.8;            %angular speed Mars
rma=7;             %orbit radius Mars

%------------Drawing
      Marsmoom------------------------------------------------------------------
      ---------------------------
h_Marsmoon=line(0,0,0,'marker','o','markersize',[4.5],'color',[0 0
      0],'markerfacecolor',[0 0 0]); %creating the ball
wmam=10;            %angular speed Marsmoon
```

```
rmam=.5;              %orbit radius Marsmoon

%------------Drawing
      Jupiter------------------------------------------------------------------
      --------------------------
h_Jupiter=line(0,0,'marker','o','markersize',[14],'color',[1 .5
      .3],'markerfacecolor',[1 .5 .3]);
wj=.5;               %angular speed jupiter
rj=11;               %orbit radius jupiter

%------------Drawing
      MoonJ1-------------------------------------------------------------------
      -------------------------
h_MoonJ1=line(0,0,'marker','o','markersize',[5],'color',[.3 .3
      .3],'markerfacecolor',[.3 .3 .3]);
wmj1=10;              %angular speed Moon
rmj1=.6;              %orbit radius moon

%------------Drawing
      MoonJ2-------------------------------------------------------------------
      -------------------------
h_MoonJ2=line(0,0,'marker','o','markersize',[5],'color',[0 .3
      .3],'markerfacecolor',[0 .3 .3]);
wmj2=13;              %angular speed Moon
rmj2=1;                %orbit radius moon

%------------Drawing
      MoonJ3-------------------------------------------------------------------
      -------------------------
h_MoonJ3=line(0,0,'marker','o','markersize',[5],'color',[.3 0
      .3],'markerfacecolor',[.3 0 .3]);
wmj3=12;              %angular speed Moon
rmj3=.9;              %orbit radius moon

%------------Drawing
      MoonJ4-------------------------------------------------------------------
      -------------------------
h_MoonJ4=line(0,0,'marker','o','markersize',[5],'color',[.3 0
      .6],'markerfacecolor',[.3 0 .6]);
wmj4=9;                %angular speed Moon
rmj4=1.2;              %orbit radius moon

drawnow
%------------Making planets
      move---------------------------------------------------------------------
      -----------------
%--------------------------------------------------------------------------
      --------------------------------------
for t=0:i:l                                    %setting up loop for one
      rotation
```

```
%----Sun----
xs=rs*cos(ws*t);                                    %getting x for sun
ys=rs*sin(ws*t);                                    %getting y for sun
set(h_Sun,'xdata',xs,'ydata',ys);                   %putting sun there

%----Mercure----
xm=rm*cos(wm*t);                                    %getting x for mercure
ym=rm*sin(wm*t);                                    %getting y for mercure
set(h_Mercure,'xdata',xm-xs,'ydata',ym-ys);               %putting mercure
    there

%----Venus----
xv=rv*cos(wv*t);                                    %getting x for venus
yv=rv*sin(wv*t);                                    %getting y for venus
set(h_Venus,'xdata',xv-xs,'ydata',yv-ys);                 %putting venus
    there

%----Earth----
xe=re*cos(we*t);                                    %getting x for earth
ye=re*sin(we*t);                                    %getting y for earth
set(h_Earth,'xdata',xe-xs,'ydata',ye-ys);                 %putting earth
    there

%----Moon----
xem=rem*cos(wem*t)+xe;                                 %getting x for moon
yem=rem*sin(wem*t)+ye;                                 %getting y for moon
set(h_Moon,'xdata',xem-xs,'ydata',yem-ys);                   %putting moon
    there

%----Mars----
xma=rma*cos(wma*t);                                 %getting x for mars
yma=rma*sin(wma*t);                                 %getting y for mars
set(h_Mars,'xdata',xma-xs,'ydata',yma-ys);                %putting mars
    there

%----Marsmoon----
xmam=rmam*cos(wmam*t)+xma;                                 %getting x for
    marsmoon
ymam=rmam*sin(wmam*t)+yma;                                 %getting y for
    marsmoon
set(h_Marsmoon,'xdata',xmam-xs,'ydata',ymam-ys);                 %putting
    marsmoon there

%----Jupiter----
xj=rj*cos(wj*t);                                    %getting x for jupiter
yj=rj*sin(wj*t);                                    %getting y for jupiter
set(h_Jupiter,'xdata',xj-xs,'ydata',yj-ys);               %putting jupiter
    there

%----MoonJ1----
```

To Table of Contents

```
  xmj1=rmj1*cos(wmj1*t)+xj;                              %getting x for
      jupitermoon
  ymj1=rmj1*sin(wmj1*t)+yj;                              %getting y for
      jupitermoon
  set(h_MoonJ1,'xdata',xmj1-xs,'ydata',ymj1-ys);               %putting
      jupitermoon there

  %----MoonJ2----
  xmj2=rmj2*cos(wmj2*t)+xj;                              %getting x for
      jupitermoon
  ymj2=rmj2*sin(wmj2*t)+yj;                              %getting x for
      jupitermoon
  set(h_MoonJ2,'xdata',xmj2-xs,'ydata',ymj2-ys);               %putting
      jupitermoon there

  %----MoonJ3----
  xmj3=rmj3*cos(wmj3*t)+xj;                              %getting x for
      jupitermoon
  ymj3=rmj3*sin(wmj3*t)+yj;                              %getting x for
      jupitermoon
  set(h_MoonJ3,'xdata',xmj3-xs,'ydata',ymj3-ys);               %putting
      jupitermoon there

  %----MoonJ4----
  xmj4=rmj4*cos(wmj4*t)+xj;                              %getting x for
      jupitermoon
  ymj4=rmj4*sin(wmj4*t)+yj;                              %getting x for
      jupitermoon
  set(h_MoonJ4,'xdata',xmj4-xs,'ydata',ymj4-ys);        %putting jupitermoon
      there

  drawnow;

  if (mod(t,0.5.*i)==g)     %Trails of Planets

    line(xs,ys,'marker','.','markersize',[7]);   %Sun
    line(xm-xs,ym-ys,'marker','.','markersize',[7]);   %Mercure
    line(xv-xs,yv-ys,'marker','.','markersize',[7]);   %Venus
    line(xe-xs,ye-ys,'marker','.','markersize',[7]);   %Earth
    line(xma-xs,yma-ys,'marker','.','markersize',[7]);   %Mars
    line(xj-xs,yj-ys,'marker','.','markersize',[7]);   %Jupiter

  endif

endfor
```

## Jupiter Centric

This program simulates the solar system with jupiter at its center. You notice that the planets have really strange paths now. The programs is the same as the first heliocentric one, the only difference is that the speed and radius of the sun are set to those of jupiter.

```
%------------preparation-----------------------------------------------------
      -----------------------------------
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],'name',
      "Jupiter-Centric"); %Creating Window
axes('position',[.1 .1 .8 .8],'color',[1 1 1], 'xlim',[-20 20],'ylim',[-20
      20]);  %Creating Axis
grid off;                                        %turning on grid in
      coordinate system
xlabel ("X-Axis");                               %labeling the axis
ylabel ("Y-Axis");                               %labeling the axis
zlabel ("Z-Axis");                               %labeling the axis
axis ('square');                                 %squaring axis
drawnow

%----------------Changeable
      Variables-----------------------------------------------------------
      ---------------
%---------------------------------------------------------------------------
      -------------------------------------
n=1;             %Number of Revolutions of earth/number of years shown
i=.05;           %Interval during for-loop
g=0;            %Set to >1 to deactivate lines

%---------------pre-calculations--------------------------------------------
      -------------------------------------
l=n.*2.*pi;     %limit for for-loop

%----------------Preparing the
      Planets-------------------------------------------------------------
      ------------
%------------w[x] is the angular speed; r[x] is the radius relative to the
      sun-----------------------------------
%----To adapt this system, set ws and rs for the sun to the ones of the
      desired planet
%------------Drawing
      Sun-----------------------------------------------------------------
      ---------------------
h_Sun=line(0,0,0,'marker','o','markersize',[17],'markerfacecolor','yellow','c
      olor','yellow'); %creating the ball
ws=.5;              %angular speed
rs=11;              %orbit radius
```

To Table of Contents

```
%------------Drawing
      Mercure-------------------------------------------------------------------
      ------------------------
h_Mercure=line(0,0,0,'marker','o','markersize',[7],'color',[.7 0
      0],'markerfacecolor',[.7 0 0]); %creating the ball
wm=2;               %angular speed mercure
rm=3;               %orbit radius mecure

%------------Drawing
      Venus---------------------------------------------------------------------
      ----------------------
h_Venus=line(0,0,0,'marker','o','markersize',[8],'color',[0 .7
      0],'markerfacecolor',[0 .7 0]); %creating the ball
wv=1.5;             %angular speed venus
rv=4;               %orbit radius venus

%------------Drawing
      Earth---------------------------------------------------------------------
      ----------------------
h_Earth=line(0,0,0,'marker','o','markersize',[10],'color',[0 0
      1],'markerfacecolor',[0 0 1]); %creating the ball
we=1;               %angular speed earth
re=5;               %orbit radius earth

%------------Drawing
      Moon----------------------------------------------------------------------
      -----------------------
h_Moon=line(0,0,0,'marker','o','markersize',[5],'color',[.3 .3
      .3],'markerfacecolor',[.3 .3 .3]); %creating the ball
wem=12;             %angular speed Moon
rem=.7;             %orbit radius moon

%------------Drawing
      Mars----------------------------------------------------------------------
      -----------------------
h_Mars=line(0,0,0,'marker','o','markersize',[9],'color',[1 0
      0],'markerfacecolor',[1 0 0]); %creating the ball
wma=.8;             %angular speed Mars
rma=7;              %orbit radius Mars

%------------Drawing
      Marsmoom------------------------------------------------------------------
      ---------------------------
h_Marsmoon=line(0,0,0,'marker','o','markersize',[4.5],'color',[0 0
      0],'markerfacecolor',[0 0 0]); %creating the ball
wmam=10;             %angular speed Marsmoon
rmam=.5;             %orbit radius Marsmoon

%------------Drawing
      Jupiter-------------------------------------------------------------------
      -------------------------
```

```
h_Jupiter=line(0,0,'marker','o','markersize',[14],'color',[1 .5
      .3],'markerfacecolor',[1 .5 .3]);
wj=.5;              %angular speed jupiter
rj=11;              %orbit radius jupiter

%------------Drawing
      MoonJ1-----------------------------------------------------------------
      -------------------------
h_MoonJ1=line(0,0,'marker','o','markersize',[5],'color',[.3 .3
      .3],'markerfacecolor',[.3 .3 .3]);
wmj1=10;             %angular speed Moon
rmj1=.6;             %orbit radius moon

%------------Drawing
      MoonJ2-----------------------------------------------------------------
      -------------------------
h_MoonJ2=line(0,0,'marker','o','markersize',[5],'color',[0 .3
      .3],'markerfacecolor',[0 .3 .3]);
wmj2=13;             %angular speed Moon
rmj2=1;               %orbit radius moon

%------------Drawing
      MoonJ3-----------------------------------------------------------------
      -------------------------
h_MoonJ3=line(0,0,'marker','o','markersize',[5],'color',[.3 0
      .3],'markerfacecolor',[.3 0 .3]);
wmj3=12;             %angular speed Moon
rmj3=.9;             %orbit radius moon

%------------Drawing
      MoonJ4-----------------------------------------------------------------
      -------------------------
h_MoonJ4=line(0,0,'marker','o','markersize',[5],'color',[.3 0
      .6],'markerfacecolor',[.3 0 .6]);
wmj4=9;               %angular speed Moon
rmj4=1.2;             %orbit radius moon

drawnow
%------------Making planets
      move-------------------------------------------------------------------
      -----------------
%----------------------------------------------------------------------------
      ---------------------------------------
for t=0:i:l                                       %setting up loop for one
      rotation

  %----Sun----
  xs=rs*cos(ws*t);                                %getting x for sun
  ys=rs*sin(ws*t);                                %getting y for sun
  set(h_Sun,'xdata',xs,'ydata',ys);               %putting sun there
```

```
%----Mercure----
xm=rm*cos(wm*t);                                    %getting x for mercure
ym=rm*sin(wm*t);                                    %getting y for mercure
set(h_Mercure,'xdata',xm-xs,'ydata',ym-ys);               %putting mercure
    there

%----Venus----
xv=rv*cos(wv*t);                                    %getting x for venus
yv=rv*sin(wv*t);                                    %getting y for venus
set(h_Venus,'xdata',xv-xs,'ydata',yv-ys);                 %putting venus
    there

%----Earth----
xe=re*cos(we*t);                                    %getting x for earth
ye=re*sin(we*t);                                    %getting y for earth
set(h_Earth,'xdata',xe-xs,'ydata',ye-ys);                 %putting earth
    there

%----Moon----
xem=rem*cos(wem*t)+xe;                                 %getting x for moon
yem=rem*sin(wem*t)+ye;                                 %getting y for moon
set(h_Moon,'xdata',xem-xs,'ydata',yem-ys);                   %putting moon
    there

%----Mars----
xma=rma*cos(wma*t);                                 %getting x for mars
yma=rma*sin(wma*t);                                 %getting y for mars
set(h_Mars,'xdata',xma-xs,'ydata',yma-ys);                %putting mars
    there

%----Marsmoon----
xmam=rmam*cos(wmam*t)+xma;                                 %getting x for
    marsmoon
ymam=rmam*sin(wmam*t)+yma;                                 %getting y for
    marsmoon
set(h_Marsmoon,'xdata',xmam-xs,'ydata',ymam-ys);                 %putting
    marsmoon there

%----Jupiter----
xj=rj*cos(wj*t);                                    %getting x for jupiter
yj=rj*sin(wj*t);                                    %getting y for jupiter
set(h_Jupiter,'xdata',xj-xs,'ydata',yj-ys);               %putting jupiter
    there

%----MoonJ1----
xmj1=rmj1*cos(wmj1*t)+xj;                                 %getting x for
    jupitermoon
ymj1=rmj1*sin(wmj1*t)+yj;                                 %getting y for
    jupitermoon
set(h_MoonJ1,'xdata',xmj1-xs,'ydata',ymj1-ys);                  %putting
    jupitermoon there
```

To Table of Contents

```
  %----MoonJ2----
  xmj2=rmj2*cos(wmj2*t)+xj;                                %getting x for
      jupitermoon
  ymj2=rmj2*sin(wmj2*t)+yj;                                %getting x for
      jupitermoon
  set(h_MoonJ2,'xdata',xmj2-xs,'ydata',ymj2-ys);                 %putting
      jupitermoon there

  %----MoonJ3----
  xmj3=rmj3*cos(wmj3*t)+xj;                                %getting x for
      jupitermoon
  ymj3=rmj3*sin(wmj3*t)+yj;                                %getting x for
      jupitermoon
  set(h_MoonJ3,'xdata',xmj3-xs,'ydata',ymj3-ys);                 %putting
      jupitermoon there

  %----MoonJ4----
  xmj4=rmj4*cos(wmj4*t)+xj;                                %getting x for
      jupitermoon
  ymj4=rmj4*sin(wmj4*t)+yj;                                %getting x for
      jupitermoon
  set(h_MoonJ4,'xdata',xmj4-xs,'ydata',ymj4-ys);        %putting jupitermoon
      there

  drawnow;

   if (mod(t,0.5.*i)==g)     %Trails of Planets

    line(xs,ys,'marker','.','markersize',[7]);   %Sun
    line(xm-xs,ym-ys,'marker','.','markersize',[7]);   %Mercure
    line(xv-xs,yv-ys,'marker','.','markersize',[7]);   %Venus
    line(xe-xs,ye-ys,'marker','.','markersize',[7]);   %Earth
    line(xma-xs,yma-ys,'marker','.','markersize',[7]);   %Mars
    line(xj-xs,yj-ys,'marker','.','markersize',[7]);   %Jupiter

  endif

endfor
```

## Mercury Centric

This program simulates the solar system with mercury at its center. You notice that the planets have really strange paths now. The programs is the same as the first heliocentric one, the only difference is that the speed and radius of the sun are set to those of mercury.

```
%------------preparation-------------------------------------------------------
      ------------------------------------
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],'name',
      "Mercure-Centric"); %Creating Window
```

```
axes('position',[.1 .1 .8 .8],'color',[1 1 1], 'xlim',[-18 18],'ylim',[-18
      18]);  %Creating Axis
grid off;                                        %turning on grid in
      coordinate system
xlabel ("X-Axis");                               %labeling the axis
ylabel ("Y-Axis");                               %labeling the axis
zlabel ("Z-Axis");                               %labeling the axis
axis ('square');                                 %squaring axis
drawnow

%-----------------Changeable
      Variables-------------------------------------------------------------
      ---------------
%---------------------------------------------------------------------------
      -------------------------------------
n=1;             %Number of Revolutions of earth/number of years shown
i=.05;           %Interval during for-loop
g=0;            %Set to >1 to deactivate lines

%---------------pre-calculations--------------------------------------------
      -------------------------------------
l=n.*2.*pi;     %limit for for-loop

%-----------------Preparing the
      Planets---------------------------------------------------------------
      ------------
%------------w[x] is the angular speed; r[x] is the radius relative to the
      sun-------------------------------------
%----To adapt this system, set ws and rs for the sun to the ones of the
      desired planet
%------------Drawing
      Sun-------------------------------------------------------------------
      ----------------------
h_Sun=line(0,0,0,'marker','o','markersize',[15],'markerfacecolor','yellow','c
      olor','yellow'); %creating the ball
ws=2;              %angular speed
rs=3;              %orbit radius

%------------Drawing
      Mercure---------------------------------------------------------------
      ------------------------
h_Mercure=line(0,0,0,'marker','o','markersize',[5],'color',[.7 0
      0],'markerfacecolor',[.7 0 0]); %creating the ball
wm=2;              %angular speed mercure
rm=3;              %orbit radius mecure

%------------Drawing
      Venus-----------------------------------------------------------------
      ----------------------
h_Venus=line(0,0,0,'marker','o','markersize',[6],'color',[0 .7
      0],'markerfacecolor',[0 .7 0]); %creating the ball
```

```
wv=1.5;             %angular speed venus
rv=4;               %orbit radius venus

%------------Drawing
      Earth----------------------------------------------------------------------
      ----------------------
h_Earth=line(0,0,0,'marker','o','markersize',[8],'color',[0 0
      1],'markerfacecolor',[0 0 1]); %creating the ball
we=1;               %angular speed earth
re=5;               %orbit radius earth

%------------Drawing
      Moon-----------------------------------------------------------------------
      -----------------------
h_Moon=line(0,0,0,'marker','o','markersize',[3],'color',[.3 .3
      .3],'markerfacecolor',[.3 .3 .3]); %creating the ball
wem=12;             %angular speed Moon
rem=.7;             %orbit radius moon

%------------Drawing
      Mars-----------------------------------------------------------------------
      -----------------------
h_Mars=line(0,0,0,'marker','o','markersize',[7],'color',[1 0
      0],'markerfacecolor',[1 0 0]); %creating the ball
wma=.8;             %angular speed Mars
rma=7;              %orbit radius Mars

%------------Drawing
      Marsmoom-------------------------------------------------------------------
      ---------------------------
h_Marsmoon=line(0,0,0,'marker','o','markersize',[2.5],'color',[0 0
      0],'markerfacecolor',[0 0 0]); %creating the ball
wmam=10;             %angular speed Marsmoon
rmam=.5;             %orbit radius Marsmoon

%------------Drawing
      Jupiter--------------------------------------------------------------------
      --------------------------
h_Jupiter=line(0,0,'marker','o','markersize',[12],'color',[1 .5
      .3],'markerfacecolor',[1 .5 .3]);
wj=.5;              %angular speed jupiter
rj=11;              %orbit radius jupiter

%------------Drawing
      MoonJ1---------------------------------------------------------------------
      -------------------------
h_MoonJ1=line(0,0,'marker','o','markersize',[3],'color',[.3 .3
      .3],'markerfacecolor',[.3 .3 .3]);
wmj1=10;             %angular speed Moon
rmj1=.6;             %orbit radius moon
```

```
%------------Drawing
      MoonJ2-----------------------------------------------------------------------
      -------------------------
h_MoonJ2=line(0,0,'marker','o','markersize',[3],'color',[0 .3
      .3],'markerfacecolor',[0 .3 .3]);
wmj2=13;             %angular speed Moon
rmj2=1;               %orbit radius moon

%------------Drawing
      MoonJ3-----------------------------------------------------------------------
      -------------------------
h_MoonJ3=line(0,0,'marker','o','markersize',[3],'color',[.3 0
      .3],'markerfacecolor',[.3 0 .3]);
wmj3=12;             %angular speed Moon
rmj3=.9;             %orbit radius moon

%------------Drawing
      MoonJ4-----------------------------------------------------------------------
      -------------------------
h_MoonJ4=line(0,0,'marker','o','markersize',[3],'color',[.3 0
      .6],'markerfacecolor',[.3 0 .6]);
wmj4=9;               %angular speed Moon
rmj4=1.2;             %orbit radius moon

drawnow
%------------Making planets
      move-------------------------------------------------------------------------
      -----------------
%-----------------------------------------------------------------------------
      ---------------------------------------
for t=0:i:l                                          %setting up loop for one
      rotation

  %----Sun----
  xs=rs*cos(ws*t);                                   %getting x for sun
  ys=rs*sin(ws*t);                                   %getting y for sun
  set(h_Sun,'xdata',xs,'ydata',ys);                  %putting sun there

  %----Mercure----
  xm=rm*cos(wm*t);                                   %getting x for mercure
  ym=rm*sin(wm*t);                                   %getting y for mercure
  set(h_Mercure,'xdata',xm-xs,'ydata',ym-ys);              %putting mercure
      there

  %----Venus----
  xv=rv*cos(wv*t);                                   %getting x for venus
  yv=rv*sin(wv*t);                                   %getting y for venus
  set(h_Venus,'xdata',xv-xs,'ydata',yv-ys);                %putting venus
      there

  %----Earth----
```

```
xe=re*cos(we*t);                                %getting x for earth
ye=re*sin(we*t);                                %getting y for earth
set(h_Earth,'xdata',xe-xs,'ydata',ye-ys);             %putting earth
    there

%----Moon----
xem=rem*cos(wem*t)+xe;                            %getting x for moon
yem=rem*sin(wem*t)+ye;                            %getting y for moon
set(h_Moon,'xdata',xem-xs,'ydata',yem-ys);              %putting moon
    there

%----Mars----
xma=rma*cos(wma*t);                             %getting x for mars
yma=rma*sin(wma*t);                             %getting y for mars
set(h_Mars,'xdata',xma-xs,'ydata',yma-ys);            %putting mars
    there

%----Marsmoon----
xmam=rmam*cos(wmam*t)+xma;                            %getting x for
    marsmoon
ymam=rmam*sin(wmam*t)+yma;                            %getting y for
    marsmoon
set(h_Marsmoon,'xdata',xmam-xs,'ydata',ymam-ys);            %putting
    marsmoon there

%----Jupiter----
xj=rj*cos(wj*t);                                %getting x for jupiter
yj=rj*sin(wj*t);                                %getting y for jupiter
set(h_Jupiter,'xdata',xj-xs,'ydata',yj-ys);           %putting jupiter
    there

%----MoonJ1----
xmj1=rmj1*cos(wmj1*t)+xj;                            %getting x for
    jupitermoon
ymj1=rmj1*sin(wmj1*t)+yj;                            %getting y for
    jupitermoon
set(h_MoonJ1,'xdata',xmj1-xs,'ydata',ymj1-ys);            %putting
    jupitermoon there

%----MoonJ2----
xmj2=rmj2*cos(wmj2*t)+xj;                            %getting x for
    jupitermoon
ymj2=rmj2*sin(wmj2*t)+yj;                            %getting x for
    jupitermoon
set(h_MoonJ2,'xdata',xmj2-xs,'ydata',ymj2-ys);            %putting
    jupitermoon there

%----MoonJ3----
xmj3=rmj3*cos(wmj3*t)+xj;                            %getting x for
    jupitermoon
```

To Table of Contents

```
  ymj3=rmj3*sin(wmj3*t)+yj;                                   %getting x for
      jupitermoon
  set(h_MoonJ3,'xdata',xmj3-xs,'ydata',ymj3-ys);                    %putting
      jupitermoon there

  %----MoonJ4----
  xmj4=rmj4*cos(wmj4*t)+xj;                                   %getting x for
      jupitermoon
  ymj4=rmj4*sin(wmj4*t)+yj;                                   %getting x for
      jupitermoon
  set(h_MoonJ4,'xdata',xmj4-xs,'ydata',ymj4-ys);        %putting jupitermoon
      there

  drawnow;

 if (mod(t,0.5.*i)==g)     %Trails of Planets

    line(xs,ys,'marker','.','markersize',[7]);   %Sun
    line(xm-xs,ym-ys,'marker','.','markersize',[7]);   %Mercure
    line(xv-xs,yv-ys,'marker','.','markersize',[7]);   %Venus
    line(xe-xs,ye-ys,'marker','.','markersize',[7]);   %Earth
    line(xma-xs,yma-ys,'marker','.','markersize',[7]);   %Mars
    line(xj-xs,yj-ys,'marker','.','markersize',[7]);   %Jupiter

  endif

endfor
```

## Spacetime

This program is the same as the heliocentric one, but now, to simulate time (spacetime), the whole system moves upwards (positive z) while the planets revolve around the sun.

```
%------------preparation------------------------------------------------------
      ------------------------------------
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],'name',
      "Solar System"); %Creating Window
axes('position',[.1 .1 .8 .8],'color',[1 1 1], 'xlim',[-13 13],'ylim',[-13
      13],'zlim',[-1 4]);  %Creating Axis
grid on;         %turning on grid in coordinate system
xlabel ("X-Axis");                               %labeling the axis
ylabel ("Y-Axis");                               %labeling the axis
zlabel ("Z-Axis");                               %labeling the axis
axis ('square');                                 %squaring axis

view(325,160)                                   %Tilting coordinate system
      Drawnow
%----------------Changeable
      Variables-------------------------------------------------------------
      ---------------
```

```
%-----------------------------------------------------------------------
------------------------------------------
n=3;            %Number of Revolutions of earth/number of years shown
(>=3)
i=.05;          %Interval during for-loop
%----To adapt this system, set angular speed and radius for the sun to
the ones of the desired planet


%---------------pre-calculations---------------------------------------
------------------------------------------
l=n.*2.*pi;     %limit for for-loop


%----------------Preparing the
Planets----------------------------------------------------------------
------------
%------------w[x] is the angular speed; r[x] is the radius relative to
the sun------------------------------------
%------------Drawing
Sun--------------------------------------------------------------------
----------------------
h_Sun=line(0,0,0,'marker','o','markersize',[15],'markerfacecolor','yell
ow','color','yellow'); %creating the ball
ws=0;              %angular speed
rs=0;              %orbit radius

%------------Drawing
Mercure----------------------------------------------------------------
------------------------
h_Mercure=line(0,0,0,'marker','o','markersize',[5],'color',[.7 0
0],'markerfacecolor',[.7 0 0]); %creating the ball
wm=2;              %angular speed mercure
rm=3;              %orbit radius mecure

%------------Drawing
Venus------------------------------------------------------------------
----------------------
h_Venus=line(0,0,0,'marker','o','markersize',[6],'color',[0 .7
0],'markerfacecolor',[0 .7 0]); %creating the ball
wv=1.5;            %angular speed venus
rv=4;              %orbit radius venus

%------------Drawing
Earth------------------------------------------------------------------
----------------------
h_Earth=line(0,0,0,'marker','o','markersize',[8],'color',[0 0
1],'markerfacecolor',[0 0 1]); %creating the ball
we=1;              %angular speed earth
re=5;              %orbit radius earth
```

```
%------------Drawing
Moon-------------------------------------------------------------------
-----------------------
h_Moon=line(0,0,0,'marker','o','markersize',[3],'color',[.3 .3
.3],'markerfacecolor',[.3 .3 .3]); %creating the ball
wem=12;             %angular speed Moon
rem=.7;             %orbit radius moon

%------------Drawing
Mars-------------------------------------------------------------------
-----------------------
h_Mars=line(0,0,0,'marker','o','markersize',[7],'color',[1 0
0],'markerfacecolor',[1 0 0]); %creating the ball
wma=.8;             %angular speed Mars
rma=7;              %orbit radius Mars

%------------Drawing
Marsmoom---------------------------------------------------------------
---------------------------
h_Marsmoon=line(0,0,0,'marker','o','markersize',[2.5],'color',[0 0
0],'markerfacecolor',[0 0 0]); %creating the ball
wmam=10;             %angular speed Marsmoon
rmam=.5;             %orbit radius Marsmoon

%------------Drawing
Jupiter----------------------------------------------------------------
--------------------------
h_Jupiter=line(0,0,'marker','o','markersize',[12],'color',[1 .5
.3],'markerfacecolor',[1 .5 .3]);
wj=.5;              %angular speed jupiter
rj=11;              %orbit radius jupiter

%------------Drawing
MoonJ1-----------------------------------------------------------------
-------------------------
h_MoonJ1=line(0,0,'marker','o','markersize',[3],'color',[.3 .3
.3],'markerfacecolor',[.3 .3 .3]);
wmj1=10;             %angular speed Moon
rmj1=.6;             %orbit radius moon

%------------Drawing
MoonJ2-----------------------------------------------------------------
-------------------------
h_MoonJ2=line(0,0,'marker','o','markersize',[3],'color',[0 .3
.3],'markerfacecolor',[0 .3 .3]);
wmj2=13;             %angular speed Moon
rmj2=1;               %orbit radius moon

%------------Drawing
MoonJ3-----------------------------------------------------------------
-------------------------
```

```
h_MoonJ3=line(0,0,'marker','o','markersize',[3],'color',[.3 0
.3],'markerfacecolor',[.3 0 .3]);
wmj3=12;             %angular speed Moon
rmj3=.9;             %orbit radius moon

%------------Drawing
MoonJ4------------------------------------------------------------------
-------------------------
h_MoonJ4=line(0,0,'marker','o','markersize',[3],'color',[.3 0
.6],'markerfacecolor',[.3 0 .6]);
wmj4=9;               %angular speed Moon
rmj4=1.2;             %orbit radius moon

drawnow
%------------Making planets
move--------------------------------------------------------------------
-----------------
%-----------------------------------------------------------------------
---------------------------------------------
for t=0:i:l                                         %setting up loop
for one rotation

  z=t./(2*pi);

  %----Sun----
  xs=rs*cos(ws*t);                                  %getting x for sun
  ys=rs*sin(ws*t);                                  %getting y for sun
  set(h_Sun,'xdata',xs,'ydata',ys,'zdata', z);
%putting sun there

  %----Mercure----
  xm=rm*cos(wm*t);                                  %getting x for
mercure
  ym=rm*sin(wm*t);                                  %getting y for
mercure
  set(h_Mercure,'xdata',xm-xs,'ydata',ym-ys,'zdata', z);
%putting mercure there

  %----Venus----
  xv=rv*cos(wv*t);                                  %getting x for
venus
  yv=rv*sin(wv*t);                                  %getting y for
venus
  set(h_Venus,'xdata',xv-xs,'ydata',yv-ys,'zdata', z);
%putting venus there

  %----Earth----
  xe=re*cos(we*t);                                  %getting x for
earth
  ye=re*sin(we*t);                                  %getting y for
earth
```

```
  set(h_Earth,'xdata',xe-xs,'ydata',ye-ys,'zdata', z);
%putting earth there

  %----Moon----
  xem=rem*cos(wem*t)+xe;                                  %getting x for
moon
  yem=rem*sin(wem*t)+ye;                                  %getting y for
moon
  set(h_Moon,'xdata',xem-xs,'ydata',yem-ys,'zdata', z);
%putting moon there

  %----Mars----
  xma=rma*cos(wma*t);                                  %getting x for
mars
  yma=rma*sin(wma*t);                                  %getting y for
mars
  set(h_Mars,'xdata',xma-xs,'ydata',yma-ys,'zdata', z);
%putting mars there

  %----Marsmoon----
  xmam=rmam*cos(wmam*t)+xma;                                  %getting x
for marsmoon
  ymam=rmam*sin(wmam*t)+yma;                                  %getting y
for marsmoon
  set(h_Marsmoon,'xdata',xmam-xs,'ydata',ymam-ys,'zdata', z);
%putting marsmoon there

  %----Jupiter----
  xj=rj*cos(wj*t);                                  %getting x for
jupiter
  yj=rj*sin(wj*t);                                  %getting y for
jupiter
  set(h_Jupiter,'xdata',xj-xs,'ydata',yj-ys,'zdata', z);
%putting jupiter there

  %----MoonJ1----
  xmj1=rmj1*cos(wmj1*t)+xj;                                  %getting x
for jupitermoon
  ymj1=rmj1*sin(wmj1*t)+yj;                                  %getting y
for jupitermoon
  set(h_MoonJ1,'xdata',xmj1-xs,'ydata',ymj1-ys,'zdata', z);
%putting jupitermoon there

  %----MoonJ2----
  xmj2=rmj2*cos(wmj2*t)+xj;                                  %getting x
for jupitermoon
  ymj2=rmj2*sin(wmj2*t)+yj;                                  %getting x
for jupitermoon
  set(h_MoonJ2,'xdata',xmj2-xs,'ydata',ymj2-ys,'zdata', z);
%putting jupitermoon there
```

```
  %----MoonJ3----
  xmj3=rmj3*cos(wmj3*t)+xj;                                 %getting x
for jupitermoon
  ymj3=rmj3*sin(wmj3*t)+yj;                                 %getting x
for jupitermoon
  set(h_MoonJ3,'xdata',xmj3-xs,'ydata',ymj3-ys,'zdata', z);
%putting jupitermoon there

  %----MoonJ4----
  xmj4=rmj4*cos(wmj4*t)+xj;                                 %getting x
for jupitermoon
  ymj4=rmj4*sin(wmj4*t)+yj;                                 %getting x
for jupitermoon
  set(h_MoonJ4,'xdata',xmj4-xs,'ydata',ymj4-ys,'zdata', z);
%putting jupitermoon there

  drawnow;

  if (mod(t,0.5.*i)==0)      %Trails of Planets

    line(xs,ys,z,'marker','.','markersize',[7]);   %Sun
    line(xm-xs,ym-ys,z,'marker','.','markersize',[7]);   %Mercure
    line(xv-xs,yv-ys,z,'marker','.','markersize',[7]);   %Venus
    line(xe-xs,ye-ys,z,'marker','.','markersize',[7]);   %Earth
    line(xma-xs,yma-ys,z,'marker','.','markersize',[7]);   %Mars
    line(xj-xs,yj-ys,z,'marker','.','markersize',[7]);   %Jupiter

  endif

endfor
```

# Aristotle's Wheel

This program simulates a wheel rolling on a surface and draws the trail of one point on the circumference of the circle (r=1). The user is asked to input another radius, which is also displayed and has a trail too. This second radius represents another wheel. Both wheels have the same speed and their centers are at the same points.
If both wheels have the same speed and the same distance is travelled, they should have the same radius. But here the wheels have different radii but still move at the same speed. This suggests a paradox.
If we look at the trails of the radii though, we see that if the user entered radius is smaller than one, the smaller radius slips and does not roll correctly because it would normally roll a shorter distance. If the radius is larger than one, the trail of the bigger radius goes in loops (it slips, too) because the wheel would normally roll a greater distance.

```
%-------dialog box asking user for radius-----------
input = inputdlg ({"Radius"},"Cycloid",[1,20],{"1"});   %getting user input
```

```
Ru=str2double(input{1});                                %radius converted
      from user input

%-----------preparing window---------------------------
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],...
       'name', "Aristotle's Wheel");
      %Creating Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]);
      %Creating Axis
txt = ["Aristotle's Wheel, R=" num2str(Ru) ];
      %preparing title string
a : title (txt,'fontsize',[17]);
      %titling diagram
a : ylabel ("Y",'fontsize',[17]);
      %labeling y axis
a : xlabel ("X",'fontsize',[17]);
      %labeling x axis
axis('equal',[-2 4*pi+2 -Ru Ru+2]);
      %setting axis limits
grid on;
      %toggling the grid in coordinate system
drawnow;
      %drawing it now

%===Drawing the Wheel===
X_wheel=0;      %setting initial values
Y_wheel=0;      %setting initial values
R=1;            %setting initial values
k=1;            %setting initial values
for fi=0:.01:2*pi;         %this calculates the points of the circle that
      will be used a the wheel
X_wheel(k)=R*cos(fi);
Y_wheel(k)=R+R*sin(fi);
k=k+1;
endfor;
h_wheel=line(X_wheel,Y_wheel,'linewidth',[5]);  %this line draws the wheel
      from the calculated data

X_R1=0;   %setting initial values
Y_R1=0;   %setting initial values
h_rad=line(0,0,'marker','o'); %initializing the line that represents the
      radius of the wheel
w=1;    %setting initial values for angular velocity
v=w*R;    %setting initial values for velocity
R1=R;   %setting initial values
x0=R;   %setting initial values
y0=1;   %setting initial values
xu0=Ru;   %setting initial values
yu0=1;    %setting initial values
```

```
for t=0:.04:4*pi;   %loop that runs for two rotations of the wheel, t
      represents time
  set(h_wheel,'xdata',X_wheel+v*t,'ydata',Y_wheel); %setting the wheel to a
      place

  %setting the coordinates that are used to draw the standard radius and its
      trail in relation to the passing time
  x=R1*cos(-w*t);
  y=R+R1*sin(-w*t);
  X_rad=[v*t, v*t+x];
  Y_rad=[R, y];
  X_R1=[v*t+x];
  Y_R1=[y];

  %setting the coordinates that are used to draw the user provided radius and
      its trail in relation to the passing time
  xu=Ru*cos(-w*t);
  yu=R+Ru*sin(-w*t);
  Xu_rad=[v*t, v*t+xu];
  Yu_rad=[R, yu];
  Xu_R1=[v*t+xu];
  Yu_R1=[yu];
  set(h_rad,'xdata',Xu_rad,'ydata',Yu_rad);   %setting the radius to a
      specific place

  line([x0 X_R1],[y0 Y_R1],'color',[0 0 1]);      %drawing line of r=1
  line([xu0 Xu_R1],[yu0 Yu_R1],'color',[1 0 0]);  %drawing line of user input

  %===storing variable for use in the next drawing of the lines
  x0=X_R1;
  y0=Y_R1;
  xu0=Xu_R1;
  yu0=Yu_R1;
  drawnow;    %drawing the things now
endfor;
```

# Projectile Movement

This set of programs simulates the movement of projectiles in various environments.

## Fireworks

This program is meant to show fireworks that look good.

```
%-----------preparing window----------------------------
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],...
       'name', "Fireworks");
      %Creating Window
```

```
a = axes('position',[.1 .1 .8 .8],'color',[.2 .3 .3],'fontsize',[12]);
      %Creating Axis
txt = ["Fireworks"];     %preparing amplitude string
a : title (txt,'fontsize',[17]);  %titling diagram
grid on;                          %turning on the grid in coordinate system
view(3);                          %setting view to 3d
axis ('equal',[-5 5 -5 5 0 12]);       %setting aspect ratio and limits
drawnow;                          %drawing this

%---initial variables---
x0=0;
z0=0;

%---first section of fireworks---
%---drawing projectile---
h_proj=line(0,0,0,'marker','o','color',[1 0 0],'markerfacecolor',[1 0 0],
      'markersize',[13]); %creating the projectile
drawnow;                          %drawing this
%---loop---
for z=0:.05:6;    %moving firework upwards
  set(h_proj,'zdata',z);            %setting the projectile to the position
  line(0,0,[z0 z],'color','red','linewidth',[3]);   %drawing the trail
  drawnow;                          %drawing this
  z0=z;                             %storing z for next line
endfor;
set(h_proj,'markersize',[0]); %making first projectile disappear

%---second section of fireworks---
%---drawing projectiles---
h_proj1=line(0,0,6,'marker','o','color',[1 1 0],'markerfacecolor',[1 1 0],
      'markersize',[9]); %creating the projectile
h_proj2=line(0,0,6,'marker','o','color',[1 0 1],'markerfacecolor',[1 0 1],
      'markersize',[9]); %creating the projectile
h_proj3=line(0,0,6,'marker','o','color',[0 0 1],'markerfacecolor',[0 0 1],
      'markersize',[9]); %creating the projectile
h_proj4=line(0,0,6,'marker','o','color',[1 1 1],'markerfacecolor',[1 1 1],
      'markersize',[9]); %creating the projectile
h_proj5=line(0,0,6,'marker','o','color',[1 .5 1],'markerfacecolor',[1 .5 1],
      'markersize',[9]); %creating the projectile
drawnow;                          %drawing this
%---loop---
for x=0:.07:6;    %moving fireworks like a throw in different directions
  %calculating z value for throw
  z=10*x-.5*9.81*x^2+6;
  %projectile 1
  set(h_proj1,'xdata',x,'ydata',x,'zdata',z);    %setting the projectile to
      the position
  line([x0 x],[x0 x],[z0 z],'color',[1 1 0],'linewidth',[2]);   %drawing the
      trail
  %projectile 2
```

```
  set(h_proj2,'xdata',x,'ydata',0,'zdata',z);   %setting the projectile to
      the position
  line([x0 x],0,[z0 z],'color',[1 0 1],'linewidth',[2]);   %drawing the trail
  %projectile 3
  set(h_proj3,'xdata',-x,'ydata',-.9*x,'zdata',z);   %setting the projectile
      to the position
  line([-x0 -x],[-.9*x0 -.9*x],[z0 z],'color',[0 0 1],'linewidth',[2]);
      %drawing the trail
  %projectile 4
  set(h_proj4,'xdata',x,'ydata',-.5*x,'zdata',z);   %setting the projectile
      to the position
  line([x0 x],[-.5*x0 -.5*x],[z0 z],'color',[1 1 1],'linewidth',[2]);
      %drawing the trail
  %projectile 5
  set(h_proj5,'xdata',-x,'ydata',x,'zdata',z);   %setting the projectile to
      the position
  line([-x0 -x],[x0 x],[z0 z],'color',[1 .5 1],'linewidth',[2]);   %drawing
      the trail
  %drawing
  drawnow;                          %drawing this
  x0=x;                             %storing x for next line
  y0=y;                             %storing y for next line
  z0=z;                             %storing z for next line
  if z<0;
    break;
  endif;
endfor;
```

## Projectile Motion Without Bounce

This program simulates the movement of a projectile based on the velocity and angle that the user enters into the dialog box at the start of the program. The angle should be 0°<alpha<90°. For comparison, the program will always show a trajectory with v0=10 and alpha=45°.

```
%-------dialog box asking user for input-----------
input = inputdlg ({"Starting Velocity v0 (m/s)", "Angle fi (deg.)"},...
"Projectile Motion",[1,20;1,20],{"10", "45"});      %getting user input
v0=str2double(input{1});         %velocity converted from user input
fi0=str2double(input{2});        %angle converted from user input


%-----------preparing window----------------------------
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],...
       'name', "Projectile Motion");
      %Creating Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]);
      %Creating Axis
txt = [ "Projectile Motion\nv0=" num2str(v0) "m/s ; fi=" num2str(fi0) "°"];
      %preparing amplitude string
a : title (txt,'fontsize',[17]);        %titling diagram
```

```
a : ylabel ("Y in m",'fontsize',[17]);  %labeling y axis
a : xlabel ("X in m",'fontsize',[17]);  %labeling x axis
grid on;                               %turning on the grid in coordinate system
drawnow;                                 %drawing the system

%--------------constants------------------------
g=9.81;                                                   %setting g
x0=y0=xs0=ys0=0;                                          %initializing

%---------------limit calculation-----------------
fi=fi0/(360/(2*pi));                  %converting degrees to radian
lxp=((v0.^2.*sin(2.*fi))./g).*1.15;           %calculating x limit
if lxp<11.8;%setting limit such that standard graph is always fully visible
  lxp=11.9;
endif;

lyp=((v0.^2.*sin(fi).^2)/(2.*g)).*1.15;     %calculating y limit
if lyp<2.6;%setting limit such that standard graph is always fully visible
  lyp=2.7;
endif;

axis('equal',[0 lxp 0 lyp]);        %setting axis limits
drawnow;                                                 %Drawing it now
h_proj=line(0,0,'marker','o','color',[1 0 0],'markerfacecolor',[1 0 0],
      'markersize',[10]); %creating the projectile
drawnow;                                    %Drawing the ball now

for t=0:.02:1000;                      %loop for drawing the throw

  %----modulated function------
  x=v0*cos(fi)*t;                      %calculating x value
  y=v0*sin(fi)*t-.5*g*t^2;             %calculating y value
  set(h_proj,'xdata',x,'ydata',y);     %setting projectile there
  line([x0 x],[y0 y],'color',[1 0 0]);%drawing trail of projectile
  x0=x;                              %setting the values for next line draw
  y0=y;                              %setting the values for next line draw

  %----standard function----------
  xs=10*cos(pi/4)*t;                   %calculating x value
  ys=10*sin(pi/4)*t-.5*g*t^2;          %calculating y value
  line([xs0 xs],[ys0 ys]);             %drawing trail of projectile
  xs0=xs;                            %setting the values for next line draw
  ys0=ys;                            %setting the values for next line draw

  if y<0 & ys<0;        %checking to see if y<0 and loop should be stopped
    drawnow;
    break;                               %---
  endif;                                 %---
  drawnow;                     %drawing what the loop created
endfor;
```

To Table of Contents

# Projectile Motion With Bounce

This program simulates the movement of a projectile based on the velocity and angle that the user enters into the dialog box at the start of the program. Additionally, the projectile now bounces if it hits the ground and loses 20% of its velocity each time. The number of bounces is also set by the user. The initial velocity should be >=2, otherwise the program will not work.

```
%-------dialog box asking user for input-----------
input = inputdlg ({"Starting Velocity v0 (>=2m/s)", "Angle fi (deg.)",
      "Number of Bounces"},...
"Projectile Motion",[1,20;1,20;1,20],{"10", "45","4"}); %getting user input
v01=str2double(input{1});    %velocity converted from user input
fi0=str2double(input{2});    %angle converted from user input
limn=str2double(input{3});   %getting number of bounces from user input


%-----------preparing window---------------------------
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],...
       'name', "Projectile Motion");
      %Creating Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]);
      %Creating Axis
txt = [ "Projectile Motion\nv0=" num2str(v01) "m/s ; fi=" num2str(fi0) "° ; "
      num2str(limn) " Bounces"];     %preparing amplitude string
a : title (txt,'fontsize',[17]);         %titling diagram
a : ylabel ("Y in m",'fontsize',[17]);    %labeling y axis
a : xlabel ("X in m",'fontsize',[17]);    %labeling x axis
grid on;                                 %turning on the grid in coordinate
      system
drawnow;                                  %drawing the system

%--------------constants------------------------
g=9.81;                        %setting g
x0=y0=xs0=ys0=xm=0;            %initializing
v0=v01*(1/.8);             %preparing v0 for the loop

%---------------limit calculation-----------------
fi=deg2rad(fi0);                 %converting angle from degrees to rad
lxp1=((v01.^2.*sin(2.*fi))./g);  %x-limit of first bounce
lxp=lxp1;                    %duplicating lxp for next loop
for nn=1:1:limn;           %calculating x-limit for n bounces
lxp=lxp+lxp1*.8^(2*nn);
endfor;
lxp=lxp*1.02;                          %adding a bit of room
lyp=((v01.^2.*sin(fi).^2)/(2.*g)).*1.15;     %calculating y limit
if v01<3;                      %correcting x-limit for small values
  lxp=lxp*1.1;
```

To Table of Contents

```
endif;
axis('equal',[0 lxp 0 lyp]);         %setting axis limits
drawnow;                            %Drawing it now
h_proj=line(0,0,'marker','o','color',[1 0 0],'markerfacecolor',[1 0 0],
      'markersize',[10]); %creating the projectile
drawnow;                                 %Drawing the ball now

%--------the throw-------------------
for n=0:1:limn;                         %loop for the number of bounces
v0=.8*v0;                               %reducing v0 each bounce

for t=0:.02:1000;                       %loop for drawing the throw

  %----modulated function------
  x=v0*cos(fi)*t+xm;                    %calculating x value
  y=v0*sin(fi)*t-.5*g*t^2;              %calculating y value
  set(h_proj,'xdata',x,'ydata',y);    %setting projectile there
  line([x0 x],[y0 y],'color',[1 0 0]);%drawing trail of projectile
  x0=x;                             %setting the values for next line draw
  y0=y;                             %setting the values for next line draw
  if y<0;                  %checking to see if y<0 and loop should be stopped
    drawnow;
    xm=x;
    break;                                %---
  endif;                                  %---
  drawnow;                                %drawing what the loop created
  endfor;
endfor;
```

# Oscillations

This group of programs simulates oscillations in the form of sine and cosine waves using the formula y=A*sin(w*x+fi). These waves are modulated based on user input. The oscillations range from x=-pi to x=3*pi and for comparison to the modulated wave the standard wave of y=sin(x) is drawn.

## Modulated Oscillation

In this program the user can enter the amplitude, frequency and phase of the oscillation and then see how it looks in comparison to y=sin(x).

```
%-------dialog box asking user for input-----------
input = inputdlg ({"Amplitude", "Frequency", "Phase"},...
"Harmonic Oscillation",[1,20;1,20;1,20],{"1", "1", "0"});%getting user input
A=str2double(input{1});             %amplitude converted from user input
w=str2double(input{2});             %frequency converted from user input
fi=str2double(input{3});            %phase converted from user input
```

```
%-----------preparing window----------------------------
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],...
       'name', "Harmonic Oscillation");
      %Creating Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]);
      %Creating Axis
txt = [ "Modulation\ny=" num2str(A) "*sin(" num2str(w) "*t+" num2str(fi)
      ")"];  %preparing title string
a : title (txt,'fontsize',[17]);    %titling diagram
a : ylabel ("y",'fontsize',[17]);   %labeling y axis
a : xlabel ("x",'fontsize',[17]);   %labeling x axis
grid on;                            %turning on the grid in coordinate system
drawnow;                            %drawing the system

%---------------changeable variables---------------------
i=.05;                                   %interval of loop
nplus=1.5;                               %number of oscillations in positive x
      direction
nneg=.5;                                 %number of oscillations in negative x
      direction

%----------calculating limits of coordinate system--------
lxp=nplus*2*pi;                          %x-limit positive
lxn=-nneg*2*pi;                          %x-limit negative
lyp=abs(A)+.5;                           %y-limit positive
lyn=-abs(A)-.5;                          %y-limit negative
axis('equal',[lxn-.5 lxp+.5 lyn lyp]) %preparing coordinate system limits
drawnow;                                 %drawing the prepared system

%---------------initializing variables-------------------
t=0;                                     %time (==x)
y=0;                                     %y, the elongation
t1=lxn;                                  %additional t variable
y1=0;                                    %additional y variable
n=0;                                     %initializing n

%-------------creating ball for function-----------------
h_Ball=line(lxn,0,'marker','o','markersize',[10],'color',...
            [1 0 0],'markerfacecolor',[1 0 0]);   %creating the ball
drawnow;       %drawing the ball initially

%--------drawing lines indicating segments of length pi--------
for limit=lxn:pi:lxp;

  line([limit limit],[lyn lyp],'linewidth', [.7],'linestyle','--');

endfor;

line([0 0],[lyn lyp],'linewidth', [1.5]); %drawing a line indicating x=0

%-----------------function-------------------------------
```

```
y1=A*sin(w*lxn+fi);      %creating first point used to draw trail user
yf1=sin(t);              %creating first point used to draw trail standard
for t=lxn:i:lxp+.02                         %loop to draw the function

    yf=sin(t);                % standard function
    y=A*sin(w*t+fi);          %user function
    set(h_Ball,'xdata',t,'ydata',y); %drawing ball at point specified by user
      function
    line([t1 t],[y1 y],'linewidth',[2],'linestyle','--') %drawing trail of
      the ball (user func)
    line([t1 t],[yf1 yf],'linewidth',[.5]) %drawing trail of standard func
    drawnow;             %drawing trail of functions
    t1=t;                %saving old variables for next turn to draw trail
    y1=y;                 %saving old variables for next turn to draw trail
    yf1=yf;              %saving old variables for next turn to draw trail
endfor;
```

## Dampened Oscillation

In this program the user can enter the amplitude, frequency, phase and damping coefficient of the oscillation and then see how it looks in comparison to y=sin(x). This is the same as the first one, but the amplitude now decreases with time based on the coefficient of damping.

```
%-------dialog box asking user for input-----------
input = inputdlg ({"Amplitude", "Frequency", "Phase","Damping Coefficient
      (best if <0.5)"},...
"Non-Harmonic Oscillation",[1,25;1,25;1,25;1,25],{"1", "1", "0","0"});
      %getting user input
A=str2double(input{1});          %amplitude converted from user input
w=str2double(input{2});        %frequency converted from user input
fi=str2double(input{3});       %phase converted from user input
d=str2double(input{4});        %coefficient converted from user input

%-----------preparing window----------------------------
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],...
       'name', "Harmonic Oscillation");
      %Creating Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]);
      %Creating Axis
txt = [ "Modulation\ny=" num2str(A) "*e^^(" num2str(d) "*t)*sin("...
         num2str(w) "*t+" num2str(fi) ")"];  %preparing title string
a : title (txt,'fontsize',[17]);  %titling diagram
a : ylabel ("y",'fontsize',[17]); %labeling y axis
a : xlabel ("x",'fontsize',[17]); %labeling x axis
grid on;       %turning on the grid in coordinate system
drawnow;       %drawing the system

%---------------changeable variables---------------------
i=.05;                 %interval of loop
nplus=1.5;             %number of oscillations in positive x direction
```

```
nneg=.5;                %number of oscillations in negative x direction

%----------calculating limits of coordinate system--------
lxp=nplus*2*pi;                      %x-limit positive
lxn=-nneg*2*pi;                      %x-limit negative
lyp=A*e^(-d*lxn)+.5;                 %y-limit positive
lyn=-A*e^(-d*lxn)-.5;                %y-limit negative
axis('equal',[lxn-.5 lxp+.5 lyn lyp]) %preparing coordinate system limits
drawnow;                             %drawing the prepared system

%---------------initializing variables-------------------
t=0;                                 %time (==x)
y=0;                                 %y, the elongation
t1=lxn;                              %additional t variable
y1=0;                                %additional y variable
n=0;                                 %initializing n

%-------------creating ball for function-----------------
h_Ball=line(lxn,0,'marker','o','markersize',[10],'color',...
            [1 0 0],'markerfacecolor',[1 0 0]);   %creating the ball
drawnow;       %drawing the ball initially

%--------drawing lines indicating segments of length pi--------
for limit=lxn:pi:lxp;

  line([limit limit],[lyn lyp],'linewidth', [.7],'linestyle','--');

endfor;

line([0 0],[lyn lyp],'linewidth', [1.5]); %drawing a line indicating x=0

%-----------------function--------------------------------
y1=A*e^(-d*lxn)*sin(w*lxn+fi);%creating first point used to draw trail user
yf1=sin(lxn);           %creating first point used to draw trail standard
for t=lxn:i:lxp+.02                  %loop to draw the function

    yf=sin(t);                   % standard function
    y=A*e^(-d*t)*sin(w*t+fi);   %user function
    set(h_Ball,'xdata',t,'ydata',y); %drawing ball at point specified by user
      function
    line([t1 t],[y1 y],'linewidth',[2],'linestyle','--') %drawing trail of
      the ball (user func)
    line([t1 t],[yf1 yf],'linewidth',[.5]) %drawing trail of the standard
      func
    drawnow;          %drawing trail of functions
    t1=t;             %saving old variables for next turn to draw trail
    y1=y;             %saving old variables for next turn to draw trail
    yf1=yf;           %saving old variables for next turn to draw trail
endfor;
```

## Nice Picture

This program just displays a nice looking picture that is created if y=sin(5*t) and x=cos(7*t). This is then run in a loop until the picture is completed. If the user wants, the values can be changed inside the code. Different combinations produce different pictures.

```
%-----------preparing window----------------------------
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],...
       'name', "Nice Picture");                          %Creating Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]);
      %Creating Axis
a : title ("Nice Picture",'fontsize',[17]);  %titling diagram
a : ylabel ("y",'fontsize',[17]);   %labeling y axis
a : xlabel ("x",'fontsize',[17]);   %labeling x axis
grid on;     %turning on the grid in coordinate system
drawnow;     %drawing the coordinate system

%---------------changeable variables---------------------
w1=5;                             %frequency for y
w2=7;                             %frequency for x
i=.02;                            %interval of loop

%----------calculating limits of coordinate system--------
axis('equal',[-1.5 1.5 -1.5 1.5]) %preparing coordinate system limits
drawnow;                          %drawing the prepared system

%---------------initializing variables-------------------
t=0;                              %time
y=0;                              %y, the elongation
x1=0;                             %additional t variable
y1=0;                             %additional y variable

%-------------creating ball for function-----------------
h_Ball=line(0,0,'marker','o','markersize',[10],'color',[1 0
      0],'markerfacecolor',[1 0 0]);    %creating the ball
drawnow;                            %drawing the ball initially

%-----------------function-------------------------------
y1=sin(w1*0);             %creating first point used to draw trail
x1=cos(w2*0);              %creating first point used to draw trail

for t=0:i:(2*pi)+.01;                %loop to draw the function
    y=sin(w1*t);                     %function y
    x=cos(w2*t);                     %function x
    set(h_Ball,'xdata',x,'ydata',y);%drawing ball at point specified by
      function
    line([x1 x],[y1 y],'linewidth',[2]) %drawing trail of the ball
    drawnow;                            %drawing trail of function
    x1=x;               %saving old variables for next turn to draw trail
```

```
    y1=y;         %saving old variables for next turn to draw trail

endfor;
```

## Electromagnetic Wave

This program simulates an electromagnetic wave by showing both wave components, the electric and the magnetic fields. The electric field is shown in red and the magnetic field in blue.

```
%------creating figure-------------------------
figure('units','norm','position',[.1 .1 .8 .7],...
       'name', "Electromagnetic Oscillation");   %Creating Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]);
      %Creating Axis
txt = [ "Electromagnetic Wave"];     %labeling diagram
a : title (txt,'fontsize',[17]);     %titling diagram
a : ylabel ("y",'fontsize',[17]);    %labeling y axis
a : xlabel ("t",'fontsize',[17]);    %labeling x axis
a : zlabel ("z",'fontsize',[17]);    %labeling z axis
view(325,-35);                      %setting view to 3d
grid on;         %turning on the grid in coordinate system
drawnow;

%-------variables----------------------------
c=1;                         %speed of light (no need to change)
T=3;                         %number of periods
i=.1;                        %interval of loop

%--------calculations------------------------
xlim=2*pi*T;                                   %calculating x limits
axis('equal',[0 xlim -1.5 1.5 -1.5 1.5]);       %setting limits

for t=0:i:xlim;

  x=c*t;

%----in y-plane--------
  e_f=sin(x);
  Xe=[x x];
  Ye=[0 e_f];
  Ze=[0 0];
  line(Xe,Ye,Ze,'marker','d','color',[1 0 0]); %drawing the line of the
      electric field

%----in z-plane--------
  m_f=sin(x);
  Xm=[x x];
  Ym=[0 0];
  Zm=[0 m_f];
```

```
  line(Xm,Ym,Zm,'marker','d','color',[0 0 1]);%drawing the line of the
      magnetic field

  drawnow;

endfor;
```

## Polarized Electromagnetic Wave

This program shows an electromagnetic wave like the one just before it, this time the wave is polarized in a clockwise motion. This happens if the phases of the two waves are not equal. In this case the magnetic wave is pi/2 (one quarter of a full oscillation) behind the electric wave.

```
%------creating figure-------------------------
figure('units','norm','position',[.1 .1 .8 .7],...
       'name', "Electromagnetic Oscillation");
      %Creating Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]);
      %Creating Axis
txt = [ "Electromagnetic Wave-Polarized"];    %labeling diagram
a : title (txt,'fontsize',[17]);              %titling diagram
a : ylabel ("E",'fontsize',[17]);             %labeling y axis
a : xlabel ("t",'fontsize',[17]);             %labeling x axis
a : zlabel ("B",'fontsize',[17]);             %labeling z axis
view(325,-35);                                %setting view for 3d aspect
grid on;                        %turning on the grid in coordinate system
drawnow;                                      %drawing this now

%-------variables----------------------------
c=1;                        %speed of light (no need to change)
T=3;                        %number of periods
i=.1;                       %interval of loop
x=0;                        %initializing x

%--------calculations------------------------
xlim=2*pi*T+.1;                              %calculating x limit
axis('equal',[0 xlim -1.5 1.5 -1.5 1.5]);    %setting limits
m_fe=-1;           %setting magnetic field
e_fe=0;            %setting electric field
Xe=0;              %initializing
n=0;               %initializing
line([0 xlim],[0 0]); %drawing a line where the center of the wave is

for t=0:i:xlim;      %loop for drawing the function

  x=c*t;

%----in y-plane--------
  e_f=sin(x);
```

```
  X=[x x];
  Ye=[0 e_f];
  Ze=[0 0];
%----in z-plane--------
  m_f=sin(x-(pi/2));
  X=[x x];
  Ym=[0 0];
  Zm=[0 m_f];
%-----line of polarization---------
  Xl=[Xe x];
  Yl=[e_fe e_f];
  Zl=[m_fe m_f];
  line(Xl,Yl,Ze,'color',[1 0 0],'linewidth',[3]); %electric field (E)
  line(Xl,Ym,Zl,'color',[0 0 1],'linewidth',[3]); %magnetic field (M)
  line(Xl,Yl,Zl,'linewidth',[3]);                %polarization

%drawing lines from waves to the central line
  line(x,[0 0],[0 m_f],'linewidth',[1],'color',[0 0 1]);
  line(x,[0 e_f],[0 0],'linewidth',[1],'color',[1 0 0]);
  drawnow;

  Xe=x;       %storing variable for next line
  e_fe=e_f;   %storing variable for next line
  m_fe=m_f;   %storing variable for next line

endfor;
```

# Runge-Kutta Method

This group of programs is based on the Runge-Kutta Method. This method is used to approximate differential equations. It works by calculating four different data points that each are slightly different (later in time). For this the interval dt is used. The final value is then calculated through the weighted average of the four data points.
This produces an accurate value representative of the differential equation. Additionally, most programs in this group are based on Newton's Second Law: F=m*a. By rearranging this formula and all its more specific versions into the form a=F/m, the acceleration can be calculated. Based on this, the speed and position of objects is calculated and they are then shown in the program.

## Cathode Ray Tube (CRT)

This program simulates a CRT, which is commonly found in old TVs and monitors. CRTs work by accelerating electrons using a magnetic field and the voltage Ua. These electrons are then shot through capacitors. These capacitors have a certain voltage and charge and they influence the movement of the electron. The electron gets pulled to one of the plates of the capacitor. This gives the electron movement in two directions, other than forward. With this, the electron can be precisely shot to one spot on the screen, where it lights it up. We can then

see the electron as light, which gets emitted. By changing the voltage on the two capacitors, the electron can be shot to the corners of the screen, the middle or any other place.
The user can enter the voltage used for accelerating the electron, and the two voltages of the capacitors. The program tells the user how much capacitor voltage can be used and for which values the corners of the screen are hit. Then the path of the electron is drawn to illustrate the process. The program also shows the speed at which the electron would travel in real life.

```
clear;    %clearing variables for good fresh start
%=====dialog box asking user for input========
input1 = inputdlg ({"Enter the values for the simulation\nUa (voltage for
      acceleration)"},...
                  "CRT",[1,10],{"1"}); %getting user input
Ua=str2num(input1{1});     %voltage for acceleration of the electron

%=====initializing variables==================
x=x0=0;              %initializing
y=y0=vy=0;           %initializing
z=z0=vz=0;           %initializing
e=1.602176*10^(-19); %charge of an electron
me=9.109381*10^(-31); %mass of an electron
%---must be l<2*d---
d=.75;               %setting space between the plates of the capacitor
l=1.2;               %setting length of the capacitor

%===calculating variables===
vx=sqrt((2*Ua*e)/(me));               %calculating vx
U_max=(d^2*me*vx^2)/(e*l^2);  %calculating maximal voltage to not hit the
      capacitors
U_max_hit=.98*(10*d*me*vx^2)/(2*l*e*(10-l)+e*l^2); %calculating the U max
      while hitting the screen
U_max_str=[ "(U<=|" num2str(round(100*U_max)/100) ...
          "|)\nTo hit the screen U<=|" ...
          num2str(round(10000*U_max_hit)/10000) ...
          "|\nTo hit the corner, leave the set values"]; %preparing string
      stating the limits
dt=10/(vx*400);   %interval so that one full loop has 400 steps
limt=11/vx;       %setting limit of loop

%===second user input===
input2 = inputdlg ({["Enter the values for the simulation\nUy " U_max_str
      ""],...
          ["\nUz " U_max_str ""]},"CRT",...
          [1,10;1,10],{ [ U_max_hit ] , [ U_max_hit ] }); %getting user
      input2
Uy=str2num(input2{1});          %Voltage for y-capacitor
Uz=str2num(input2{2});          %Voltage for z-capacitor

%=====preparing window=======================
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],...
```

```
        'name', "Newtons 2nd Law");
       %Creating Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]);
       %Creating Axis
txt = ["Newtons 2nd Law - Cathode Ray Tube\nUa=" num2str(Ua) " V ; Uy="
       num2str(Uy) ...
         "V ; Uz=" num2str(Uz) " V\nvx=" num2str(round(vx)) " m/s"];
       %preparing  title, stating variables
a : title (txt,'fontsize',[17]);     %titleing diagram
a : xlabel ("X",'fontsize',[17]);    %labeling y axis
a : ylabel ("Y",'fontsize',[17]);    %labeling x axis
a : zlabel ("Z",'fontsize',[17]);    %labeling z axis
grid on;                             %turing on the grid
axis('equal',[-1 10.1 -6 6 -6 6]);   %setting axis limits
view(3);                             %setting the view to 3d
drawnow;       %drawing the system
%===drawing screen===
line([10 10],[-5 5],[5 5],'linewidth',[3]);
line([10 10],[-5 5],[-5 -5],'linewidth',[3]);
line([10 10],[5 5],[-5 5],'linewidth',[3]);
line([10 10],[-5 -5],[-5 5],'linewidth',[3]);
line([10 10],[0 0],[-5 5],'linewidth',[1]);
line([10 10],[-5 5],[0 0],'linewidth',[1]);
%===drawing the two capacitors===
sl=d/2;     %getting lenght of sides from space between plates
line([0 0],[-sl sl],[sl sl],'linewidth',[2]);
line([0 0],[-sl sl],[-sl -sl],'linewidth',[2]);
line([0 0],[sl sl],[-sl sl],'linewidth',[2]);
line([0 0],[-sl -sl],[-sl sl],'linewidth',[2]);
line([l l],[-sl sl],[sl sl],'linewidth',[2]);
line([l l],[-sl sl],[-sl -sl],'linewidth',[2]);
line([l l],[sl sl],[-sl sl],'linewidth',[2]);
line([l l],[-sl -sl],[-sl sl],'linewidth',[2]);
line([0 l],[-sl -sl],[sl sl],'linewidth',[2]);
line([0 l],[-sl -sl],[-sl -sl],'linewidth',[2]);
line([0 l],[sl sl],[-sl -sl],'linewidth',[2]);
line([0 l],[sl sl],[sl sl],'linewidth',[2]);
%===drawing the electron===
h_electron=line(0,0,'marker','o', 'markersize',[5],'color','blue',...
'markerfacecolor','blue');    %creating an electron
drawnow;       %drawing the system

%=====declaring function for acceleration x===
function ax=funVX(Uy,Uz,e,me,d,lock);
  ax=0;       %acceleration for x
endfunction;

%=====declaring function for velocity x=======
function Vx=funX(vx);
  Vx=vx;     %speed x
endfunction;
```

```
%=====declaring function for acceleration y===
function ay=funVY(Uy,Uz,e,me,d);
  ay=(Uy*e)/(d*me);    %acceleration for y
endfunction;

%=====declaring function for velocity y=======
function Vy=funY(vy);
  Vy=vy;       %speed y
endfunction;

%=====declaring function for acceleration z===
function az=funVZ(Uy,Uz,e,me,d);
  az=(Uz*e)/(d*me);    %acceleration for z
endfunction;

%=====declaring function for velocity z=======
function Vz=funZ(vz);
  Vz=vz;     %speed z
endfunction;

%=====loop for time progression and graphing==
for t=0:dt:limt;        %loop from 0 to limt with interval dt
  %=====X + Y variable=======================
  %---calculating k1---
  k1x=funX(vx);
  k1y=funY(vy);
  k1z=funZ(vz);
  k1Vx=funVX(Uy,Uz,e,me,d);
  k1Vy=funVY(Uy,Uz,e,me,d);
  k1Vz=funVZ(Uy,Uz,e,me,d);

  %---calculating k2---
  k2x=funX(vx+k1Vx*dt/2);
  k2y=funY(vy+k1Vy*dt/2);
  k2z=funZ(vz+k1Vz*dt/2);
  k2Vx=funVX(Uy,Uz,e,me,d);
  k2Vy=funVY(Uy,Uz,e,me,d);
  k2Vz=funVZ(Uy,Uz,e,me,d);

  %---calculating k3---
  k3x=funX(vx+k2Vx*dt/2);
  k3y=funY(vy+k2Vy*dt/2);
  k3z=funZ(vz+k2Vz*dt/2);
  k3Vx=funVX(Uy,Uz,e,me,d);
  k3Vy=funVY(Uy,Uz,e,me,d);
  k3Vz=funVZ(Uy,Uz,e,me,d);

  %---calculating k4---
  k4x=funX(vx+k3Vx*dt);
  k4y=funY(vy+k3Vy*dt);
```

```
  k4z=funZ(vz+k3Vz*dt);
  k4Vx=funVX(Uy,Uz,e,me,d);
  k4Vy=funVY(Uy,Uz,e,me,d);
  k4Vz=funVZ(Uy,Uz,e,me,d);

  %---calculating new x & y---
  x=x+(k1x+2*k2x+2*k3x+k4x)*dt/6;
  y=y+(k1y+2*k2y+2*k3y+k4y)*dt/6;
  z=z+(k1z+2*k2z+2*k3z+k4z)*dt/6;

  %---calculating new vx & vy---
  vx=vx+(k1Vx+2*k2Vx+2*k3Vx+k4Vx)*dt/6;
  vy=vy+(k1Vy+2*k2Vy+2*k3Vy+k4Vy)*dt/6;
  vz=vz+(k1Vz+2*k2Vz+2*k3Vz+k4Vz)*dt/6;

  %===creating line of y and x================
  line([x0  x],[y0 y],[z0 z],'linewidth',[2],'color',"blue");%line of x and y
  set(h_electron,'xdata',x,'ydata',y,'zdata',z);  %setting the electron to
      its position
  x0=x;                                    %storing x for next line
  y0=y;                                    %storing y for next line
  z0=z;                                    %storing z for next line
  drawnow;                                 %drawing line and electron

  if x>l;    %setting voltages to 0 when electron leaves the capacitor
    Uy=Uz=0;
  endif;
  if x>10;   %stopping loop when screen is hit
    break;
    set(h_electron,'xdata',x,'ydata',y,'zdata',z);  %setting the electron to
      the position
  endif;
endfor;
```

## Chaos Theory - Lorenz Attractor

This system of equations describes atmospheric convection in a simple way. This system of equations exhibits chaotic behavior and the resulting graph resembles a butterfly or a figure eight ‘8’. The initial given values show these figures.
The program first lets the user choose between three different settings. The first one is user input, where the user can input the variables according to their liking. The second one is a preset that shows the standard figure, only larger, and the third setting shows a spiral. Finally, the program always shows the standard version of the equations on the left side of the window for comparison and the changed equations on the right side.

```
clear;             %resetting everything for good new start
%=====menu that lets user choose the mode of the
      programm=====================
CHOICE = menu ('Sigma, Rho, Beta', 'User Input', '20, 40, 16/3',...
```

```
               '5, 45, 8');    %creating choices on menu

if CHOICE==1;      %showing input dialog only if user wants
%=====dialog box asking user for
      input==========================================
input = inputdlg ({"Enter the values for the simulation\nx", "\ny","\nz",...
                   "\nsigma","\nrho","\nbeta"},"Chaos
      Theory",[1,10;1,10;1,...
                   10;1,10;1,10;1,10],{"1","1","1","10","28","8/3"});
      %getting user input
endif;

%=====setting initial
      variables==================================================
dt=.005;           %interval of loop
limt=10000*dt;           %setting limit of loop

%=====creating
      figure=========================================================
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],...
       'name', "Newtons 2nd Law");   %Creating Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]);
      %Creating Axis

%=====declaring
      functions======================================================
%=============================================================================
      ===

%=====declaring functions for velocity
      x==========================================
function Vx=funX(x,y,z,sigma,rho,beta);
  Vx=sigma*(y-x);                             %speed x
endfunction;

%=====declaring function for velocity
      y==========================================
function Vy=funY(x,y,z,sigma,rho,beta);
  Vy=x*(rho-z)-y;                             %speed y
endfunction;

%=====declaring function for velocity
      z==========================================
function Vz=funZ(x,y,z,sigma,rho,beta);
  Vz=x*y-beta*z;                              %speed z
endfunction;

%===creating the left plot
      window==================================================
%=============================================================================
      ===
```

```
subplot(1,2,1);

%===setting variables===
k=1;
x=1;
y=1;
z=1;
sigma=10;
rho=28;
beta=8/3;

%===setting up the coordinate
      system============================================
txt = ["Newton's 2nd Law - Chaos Theory - Standard\nx=" num2str(x) " ; y="
      num2str(y) ...
       " ; z=" num2str(z) "\nsigma=" num2str(sigma) " ; rho=" num2str(rho) "
      ; beta=" ...
       num2str(beta) ];  %preparing  title, stating variables
a : title (txt,'fontsize',[17]);     %titling diagram
a : xlabel ("X",'fontsize',[17]);    %labeling y axis
a : ylabel ("Y",'fontsize',[17]);    %labeling x axis
a : zlabel ("Z",'fontsize',[17]);    %labeling z axis
axis('equal',[-40 40 -40 40 -10 60]);      %setting axis limits
view(3);           %setting viewing angle
grid on;           %turning on the grid
drawnow;           %drawing the system

%=====loop for time progression and
      graphing====================================
for t=0:dt:limt;
      %loop from 0 to limit with interval dt

  %=====X + Y + Z variable====================
  %---calculating k1---
  k1x=funX(x,y,z,sigma,rho,beta);
  k1y=funY(x,y,z,sigma,rho,beta);
  k1z=funZ(x,y,z,sigma,rho,beta);

  %---calculating k2---
  k2x=funX(x+k1x*dt/2,y+k1y*dt/2,z+k1z*dt/2,sigma,rho,beta);
  k2y=funY(x+k1x*dt/2,y+k1y*dt/2,z+k1z*dt/2,sigma,rho,beta);
  k2z=funZ(x+k1x*dt/2,y+k1y*dt/2,z+k1z*dt/2,sigma,rho,beta);

  %---calculating k3---
  k3x=funX(x+k2x*dt/2,y+k2y*dt/2,z+k2z*dt/2,sigma,rho,beta);
  k3y=funY(x+k2x*dt/2,y+k2y*dt/2,z+k2z*dt/2,sigma,rho,beta);
  k3z=funZ(x+k2x*dt/2,y+k2y*dt/2,z+k2z*dt/2,sigma,rho,beta);

  %---calculating k4---
  k4x=funX(x+k3x*dt,y+k3y*dt,z+k3z*dt,sigma,rho,beta);
  k4y=funY(x+k3x*dt,y+k3y*dt,z+k3z*dt,sigma,rho,beta);
```

```
  k4z=funZ(x+k3x*dt,y+k3y*dt,z+k3z*dt,sigma,rho,beta);

  %---calculating new x & y & z---
  x=x+(k1x+2*k2x+2*k3x+k4x)*dt/6;
  y=y+(k1y+2*k2y+2*k3y+k4y)*dt/6;
  z=z+(k1z+2*k2z+2*k3z+k4z)*dt/6;

  %==creating variable for line of y, x and z=
  X_path1(k)=x;
  Y_path1(k)=y;
  Z_path1(k)=z;
  k=k+1;
endfor;
h_path=line(X_path1,Y_path1,Z_path1,'linewidth',[1]);   %drawing path
drawnow;        %really drawing path

%===creating the right plot
      window=============================================
%=========================================================================
      ===
subplot(1,2,2);
%===setting variables===
k=1;

if CHOICE==1;                        %if user chooses input, use it
  x=str2num(input{1});               % x converted from user input
  y=str2num(input{2});               % y converted from user input
  z=str2num(input{3});              % z converted from user input
  sigma=str2num(input{4});           %sigma converted from user input
  rho=str2num(input{5});      %rho converted from user input
  beta=str2num(input{6});            %beta converted from user input

elseif CHOICE==2;         %if user chooses 1 preset, use it
  x=1;
  y=1;
  z=1;
  sigma=20;
  rho=40;
  beta=16/3;

elseif CHOICE==3;       %if user chooses 2 preset, use it
  x=1;
  y=1;
  z=1;
  sigma=5;
  rho=45;
  beta=8,
endif;

%===setting up the coordinate
      system==========================================
```

```
txt = ["Newtons 2nd Law - Chaos Theory - User Input\nx=" num2str(x) " ; y="
      num2str(y) ...
        " ; z=" num2str(z) "\nsigma=" num2str(sigma) " ; rho=" num2str(rho) "
      ; beta=" ...
        num2str(beta) ];              %preparing  title, stating variables
a : title (txt,'fontsize',[17]);         %titling diagram
a : xlabel ("X",'fontsize',[17]);         %labeling y axis
a : ylabel ("Y",'fontsize',[17]);         %labeling x axis
a : zlabel ("Z",'fontsize',[17]);       %labeling z axis
axis('equal',[-40 40 -40 40 -10 60]);   %setting axis limits
view(3);                                      %setting viewing angle
grid on;                                %turning on the grid
drawnow;                           %drawing the system

%=====loop for time progression and
      graphing====================================
for t=0:dt:limt;                %loop from 0 to limit with interval dt

  %=====X + Y + Z variable====================
  %---calculating k1---
  k1x=funX(x,y,z,sigma,rho,beta);
  k1y=funY(x,y,z,sigma,rho,beta);
  k1z=funZ(x,y,z,sigma,rho,beta);

  %---calculating k2---
  k2x=funX(x+k1x*dt/2,y+k1y*dt/2,z+k1z*dt/2,sigma,rho,beta);
  k2y=funY(x+k1x*dt/2,y+k1y*dt/2,z+k1z*dt/2,sigma,rho,beta);
  k2z=funZ(x+k1x*dt/2,y+k1y*dt/2,z+k1z*dt/2,sigma,rho,beta);

  %---calculating k3---
  k3x=funX(x+k2x*dt/2,y+k2y*dt/2,z+k2z*dt/2,sigma,rho,beta);
  k3y=funY(x+k2x*dt/2,y+k2y*dt/2,z+k2z*dt/2,sigma,rho,beta);
  k3z=funZ(x+k2x*dt/2,y+k2y*dt/2,z+k2z*dt/2,sigma,rho,beta);

  %---calculating k4---
  k4x=funX(x+k3x*dt,y+k3y*dt,z+k3z*dt,sigma,rho,beta);
  k4y=funY(x+k3x*dt,y+k3y*dt,z+k3z*dt,sigma,rho,beta);
  k4z=funZ(x+k3x*dt,y+k3y*dt,z+k3z*dt,sigma,rho,beta);

  %---calculating new x & y & z---
  x=x+(k1x+2*k2x+2*k3x+k4x)*dt/6;
  y=y+(k1y+2*k2y+2*k3y+k4y)*dt/6;
  z=z+(k1z+2*k2z+2*k3z+k4z)*dt/6;

  %==creating variable for line of y, x and z=
  X_path2(k)=x;
  Y_path2(k)=y;
  Z_path2(k)=z;
  k=k+1;
endfor;
```

To Table of Contents

```
h_path=line(X_path2,Y_path2,Z_path2,'linewidth',[1], 'color', 'blue');
      %drawing path
drawnow;                                            %really drawing path
```

## Electromagnetic Field (EMF)

These programs simulate the movement of charged objects in an electromagnetic field.

### Orbit in EMF

This program simulates the movement of a charged object in an electromagnetic field. The charge moves in a circle around the center it is attracted to. The user can input the velocities in x, z and y direction, the mass of the object, the strength of the magnetic field, the strength of the charge of the object as well as w, which, together with B=B0*sin(w*t) changes the strength of the magnetic field, leading to interesting orbit patterns.

```
%=====dialog box asking user for input========
input = inputdlg ({"Enter the values for the simulation\nvx",
      "\nvy","\nvz",...
                  "\nB","\nm","\nQ","\nw"},"Object in Magnetic
      Field",[1,10;1,10;1,...
                  10;1,10;1,10;1,10;1,10;1,10],{"0","1","0","1","1","1","0"});
      %getting user input
vx1=str2num(input{1});        %velocity x converted from user input
vy1=str2num(input{2});        %velocity y converted from user input
vz=str2num(input{3});         %velocity z converted from user input
B0=str2num(input{4});                  %B converted from user input
m=str2num(input{5});                   %m converted from user input
Q=str2num(input{6});                   %Q converted from user input
w=str2num(input{7});                   %w converted from user input

%=====calculating variables==================
x=x0=1;                                       %initializing
y=y0=z0=z=vx=0;          %initializing
vy=sqrt(vx1^2+vy1^2);  %calculating vy from both vx1 and vy1 for simplicity
r=abs((m*vy)/(Q*B0));      %calculating radius of circle
dt=2*pi*r/500;    %interval so that one full circle has 500 steps
limt=2*pi*r;   %setting limit of loop to stop after one full radius
tz=limt/vy;   %calculating time the movement takes
limz=vz*tz;       %calculating limit for z coordinate axis

%=====preparing window=======================
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],...
       'name', "Newtons 2nd Law");   %Creating Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]); %Creating
      Axis
txt = ["Newtons 2nd Law - Magnetic Field\nv0x=" num2str(vx1) "m/s ; v0y="
      num2str(vy1) ...
```

```
        "m/s ; v0z=" num2str(vz) "\nm=" num2str(m) "kg ; Q=" num2str(Q) "C ;
      B=" ...
        num2str(B0) "T ; w=" num2str(w)];  %preparing  title, stating
      variables
a : title (txt,'fontsize',[17]);           %titling diagram
a : xlabel ("X in m",'fontsize',[17]);     %labeling y axis
a : ylabel ("Y in m",'fontsize',[17]);    %labeling x axis
grid on;              %turning on the grid in coordinate system
axis('equal',[x-1 x+2*r+1 -r-1 r+1 0 2*limz+1]); %setting axis limits based
      on radius of circle
h_center=line(r+1,0,'marker','o', 'markersize',[15],'color','red',...
'markerfacecolor','red');  %setting a ball to  center circle
if vz!=0;         %in case there is a z-velocity
  view(3);        %setting view to also see z movement
  limt=limt*2;    %doubling length of loop
  a : zlabel ("Z in m",'fontsize',[17]);                    %labeling z axis
endif;
drawnow;                                  %drawing the system

%=====declaring function for acceleration x===
function ax=funVX(x,y,vx,vy,B,Q,m);
  ax=(Q*vy*B)/(m);                             %acceleration for x
endfunction;

%=====declaring function for velocity x=======
function Vx=funX(vx);
  Vx=vx;                                       %speed x
endfunction;

%=====declaring function for acceleration y===
function ay=funVY(x,y,vx,vy,B,Q,m);
  ay=-(Q*vx*B)/(m);                            %%acceleration for y
endfunction;

%=====declaring function for velocity y=======
function Vy=funY(vy);
  Vy=vy;                                       %speed y
endfunction;

%=====declaring function for acceleration z===
function az=funVZ(vz);
  az=0;                                        %acceleration for z
endfunction;

%=====declaring function for velocity z=======
function Vz=funZ(vz);
  Vz=vz;                                       %speed z
endfunction;

%=====loop for time progression and graphing==
for t=0:dt:limt;           %loop from 0 to limt with interval dt
```

```
B=B0+sin(w*t);
%=====X + Y variable========================
%---calculating k1---
k1x=funX(vx);
k1y=funY(vy);
k1z=funZ(vz);
k1Vx=funVX(x,y,vx,vy,B,Q,m);
k1Vy=funVY(x,y,vx,vy,B,Q,m);
k1Vz=funVZ(vz);

%---calculating k2---
k2x=funX(vx+k1Vx*dt/2);
k2y=funY(vy+k1Vy*dt/2);
k2z=funZ(vz+k1Vz*dt/2);
k2Vx=funVX(x+k1x*dt/2,y+k1y*dt/2,vx+k1Vx*dt/2,vy+k1Vy*dt/2,B,Q,m);
k2Vy=funVY(x+k1x*dt/2,y+k1y*dt/2,vx+k1Vx*dt/2,vy+k1Vy*dt/2,B,Q,m);
k2Vz=funVZ(vz+k1Vz*dt/2);

%---calculating k3---
k3x=funX(vx+k2Vx*dt/2);
k3y=funY(vy+k2Vy*dt/2);
k3z=funZ(vz+k2Vz*dt/2);
k3Vx=funVX(x+k2x*dt/2,y+k2y*dt/2,vx+k2Vx*dt/2,vy+k2Vy*dt/2,B,Q,m);
k3Vy=funVY(x+k2x*dt/2,y+k2y*dt/2,vx+k2Vx*dt/2,vy+k2Vy*dt/2,B,Q,m);
k3Vz=funVZ(vz+k2Vz*dt/2);

%---calculating k4---
k4x=funX(vx+k3Vx*dt);
k4y=funY(vy+k3Vy*dt);
k4z=funZ(vz+k3Vz*dt);
k4Vx=funVX(x+k3x*dt,y+k3y*dt,vx+k3Vx*dt,vy+k3Vy*dt,B,Q,m);
k4Vy=funVY(x+k3x*dt,y+k3y*dt,vx+k3Vx*dt,vy+k3Vy*dt,B,Q,m);
k4Vz=funVZ(vz+k3Vz*dt);

%---calculating new x & y---
x=x+(k1x+2*k2x+2*k3x+k4x)*dt/6;
y=y+(k1y+2*k2y+2*k3y+k4y)*dt/6;
z=z+(k1z+2*k2z+2*k3z+k4z)*dt/6;

%---calculating new vx & vy---
vx=vx+(k1Vx+2*k2Vx+2*k3Vx+k4Vx)*dt/6;
vy=vy+(k1Vy+2*k2Vy+2*k3Vy+k4Vy)*dt/6;
vz=vz+(k1Vz+2*k2Vz+2*k3Vz+k4Vz)*dt/6;

%===creating line of y and x================
line([x0  x],[y0 y],[z0 z],'linewidth',[2]);%line of x and y
set(h_center,'xdata',r+1,'ydata',0,'zdata',z);  %setting the ball to the
    center
x0=x;                                        %storing x for next line
y0=y;                                        %storing y for next line
z0=z;                                        %storing z for next line
```

```
  drawnow;                                    %drawing line and ball

endfor;
```

## Charge in Changing EMF

This program shows the movement of a charged object in a changing EM field. Both E and B get changed based on sin(t). This leads to interesting patterns. The user can input E, B, Q, m and vx into the program. This program invites the user to try and find interesting patterns.

```
%=====dialog box asking user for input========
input = inputdlg ({"Enter the values for the simulation\nvx", "\nB",...
                   "\nm","\nE","\nQ"},"Object in Magnetic Field",...
                   [1,10;1,10;1,10;1,10;1,10],{"2","1","1","1","2"});
%getting user input
vx=str2num(input{1});  %velocity x converted from user input
B0=str2num(input{2});  %B converted from user input
m=str2num(input{3});   %m converted from user input
E0=str2num(input{4});  %E converted from user input
Q=str2num(input{5});   %Q converted from user input

%=====calculating variables==================
x=x0=0;     %initializing
y=y0=z0=0;  %initializing
vz=vy=z=0;  %initializing
dt=.02;     %interval
limt=1000;  %setting limit of loop

%=====preparing window=======================
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],...
       'name', "Newtons 2nd Law");       %Creating Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]);
%Creating Axis
txt = ["Newtons 2nd Law - Magnetic Field\nvx=" num2str(vx) " ; B="
num2str(B0) ...
       " ; m=" num2str(m) "\nE=" num2str(E0) " ; Q=" num2str(Q) " ; Q/m="
num2str(Q/m) ];  %preparing  title, stating variables
a : title (txt,'fontsize',[17]);     %titleing diagram
a : xlabel ("X",'fontsize',[17]);    %labeling y axis
a : ylabel ("Y",'fontsize',[17]);    %labeling x axis
a : zlabel ("Z",'fontsize',[17]);    %labeling z axis
grid on;             %turning on the grid in coordinate system
axis('equal',[-1 26 -11 11 ]); %setting axis limits based on radius of circle
h_center1=line(0,0,'marker','o', 'markersize',[10],'color','red',...
              'markerfacecolor','red'); %setting a ball to  center circle
drawnow;                                %drawing the system

%=====declaring function for acceleration x===
function ax=funVX(x,y,vx,vy,vz,B,Q,m,E);
  ax=(Q*vy*B)/(m);   %acceleration for x
```

```
endfunction;

%=====declaring function for velocity x=======
function Vx=funX(vx);
  Vx=vx;             %speed x
endfunction;

%=====declaring function for acceleration y===
function ay=funVY(x,y,vx,vy,vz,B,Q,m,E);
  ay=Q*(E-B*vx);     %acceleration for y +vy*B
endfunction;

%=====declaring function for velocity y=======
function Vy=funY(vy);
  Vy=vy;             %speed y
endfunction;

%=====declaring function for acceleration z===
function az=funVZ(x,y,vx,vy,vz,B,Q,m,E);
  az=0;              %acceleration for z E+
endfunction;

%=====declaring function for velocity z=======
function Vz=funZ(vz);
  Vz=vz;             %speed z
endfunction;

%=====loop for time progression and graphing==
for t=0:dt:limt;       %loop from 0 to limt with interval dt
  %---changing B and E periodically---
  B=B0*sin(t);
  E=E0*sin(t);
  %=====X + Y variable========================
  %---calculating k1---
  k1x=funX(vx);
  k1y=funY(vy);
  k1z=funZ(vz);
  k1Vx=funVX(x,y,vx,vy,vz,B,Q,m,E);
  k1Vy=funVY(x,y,vx,vy,vz,B,Q,m,E);
  k1Vz=funVZ(x,y,vx,vy,vz,B,Q,m,E);

  %---calculating k2---
  k2x=funX(vx+k1Vx*dt/2);
  k2y=funY(vy+k1Vy*dt/2);
  k2z=funZ(vz+k1Vz*dt/2);

k2Vx=funVX(x+k1x*dt/2,y+k1y*dt/2,vx+k1Vx*dt/2,vy+k1Vy*dt/2,vz+k1Vy*dt/2,B,Q,m
,E);

k2Vy=funVY(x+k1x*dt/2,y+k1y*dt/2,vx+k1Vx*dt/2,vy+k1Vy*dt/2,vz+k1Vy*dt/2,B,Q,m
,E);
```

```
k2Vz=funVZ(x+k1x*dt/2,y+k1y*dt/2,vx+k1Vx*dt/2,vy+k1Vy*dt/2,vz+k1Vy*dt/2,B,Q,m
,E);

  %---calculating k3---
  k3x=funX(vx+k2Vx*dt/2);
  k3y=funY(vy+k2Vy*dt/2);
  k3z=funZ(vz+k2Vz*dt/2);

k3Vx=funVX(x+k2x*dt/2,y+k2y*dt/2,vx+k2Vx*dt/2,vy+k2Vy*dt/2,vz+k2Vy*dt/2,B,Q,m
,E);

k3Vy=funVY(x+k2x*dt/2,y+k2y*dt/2,vx+k2Vx*dt/2,vy+k2Vy*dt/2,vz+k2Vy*dt/2,B,Q,m
,E);

k3Vz=funVZ(x+k2x*dt/2,y+k2y*dt/2,vx+k2Vx*dt/2,vy+k2Vy*dt/2,vz+k2Vy*dt/2,B,Q,m
,E);

  %---calculating k4---
  k4x=funX(vx+k3Vx*dt);
  k4y=funY(vy+k3Vy*dt);
  k4z=funZ(vz+k3Vz*dt);
  k4Vx=funVX(x+k3x*dt,y+k3y*dt,vx+k3Vx*dt,vy+k3Vy*dt,vz+k3Vy*dt,B,Q,m,E);
  k4Vy=funVY(x+k3x*dt,y+k3y*dt,vx+k3Vx*dt,vy+k3Vy*dt,vz+k3Vy*dt,B,Q,m,E);
  k4Vz=funVZ(x+k3x*dt,y+k3y*dt,vx+k3Vx*dt,vy+k3Vy*dt,vz+k3Vy*dt,B,Q,m,E);

  %---calculating new x & y---
  x=x+(k1x+2*k2x+2*k3x+k4x)*dt/6;
  y=y+(k1y+2*k2y+2*k3y+k4y)*dt/6;
  z=z+(k1z+2*k2z+2*k3z+k4z)*dt/6;

  %---calculating new vx & vy---
  vx=vx+(k1Vx+2*k2Vx+2*k3Vx+k4Vx)*dt/6;
  vy=vy+(k1Vy+2*k2Vy+2*k3Vy+k4Vy)*dt/6;
  vz=vz+(k1Vz+2*k2Vz+2*k3Vz+k4Vz)*dt/6;

  %===creating line of y and x================
  line([x0  x],[y0 y],[z0 z],'linewidth',[2]);%line of x and y
  set(h_center1,'xdata',x,'ydata',y,'zdata',z);  %setting the ball to the
center
  x0=x;                                        %storing x for next line
  y0=y;                                        %storing y for next line
  z0=z;                                        %storing z for next line
  drawnow;

  if x>25 | y>10 | y<-10; %stops the loop if the ball leaves the coordinate
system
    break;
    set(h_center1,'xdata',x,'ydata',y,'zdata',z);  %setting the ball to the
center
  endif;
```

```
endfor;
```

## Satellite Orbit Around Earth

This program simulates the movement of a satellite around earth. The user can enter the radius at which the satellite moves and the speed at which it moves. It should be noted that the radius that is entered includes the radius of the earth, it is relative to the center of earth. If the user wants to see a perfectly round orbit, the speed should be set to 7294 m/s at 1000 km above earth (r=7400 km or $7.4*10^6$ m). The user can also enter a coefficient of drag that will slow down the satellite while it is in space. Now the satellite continually gets slower and ultimately hits earth.

```
%=====dialog box asking user for input========
input = inputdlg ({"Starting Velocity v0", "Coefficient of Drag", "Y0 in
      10^6m"},"Projectile Motion",[1,20;1,20;1,20],{"1000","0.00005","7.4"});
      %getting user input
vx=str2num(input{1});    %velocity converted from user input
alph=str2num(input{2});   %drag converted from user input
y=y0=10^6*str2num(input{3});  %drag converted from user input

%=====preparing window========================
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],...
       'name', "Newtons 2nd Law");                 %Creating Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]); %Creating
      Axis
txt = ["Newtons 2nd Law - Satellite\nv0=" num2str(vx) ...
        "m/s ; alpha=" num2str(alph) " ; y0=" num2str(y0) "m"];
      %preparing  title
a : title (txt,'fontsize',[17]);           %titling diagram
a : xlabel ("X in m",'fontsize',[17]);  %labeling y axis
a : ylabel ("Y in m",'fontsize',[17]);     %labeling x axis
grid on;                  %turning on the grid in coordinate system
axis('equal',[(-4/3)*y0 (4/3)*y0 (-4/3)*y0 (4/3)*y0]);  %setting axis limits
drawnow;                                 %drawing the system

%=====initializing variables==================
x=x0=vy=n=0;                                  %initializing
k=1;                                          %k for circle drawing
dt=10;                                        %interval

%=====drawing earth===========================
for fi=0:.01:2*pi;                            %calculating points of circle
X_earth(k)=6.4*10^6*cos(fi);
Y_earth(k)=6.4*10^6*sin(fi);
k=k+1;
endfor;
h_earth=line(X_earth,Y_earth,'linewidth',...
        [5],'color','blue');                  %drawing earth
```

```
%=====declaring function for acceleration x===
function ax=funVX(x,y,vx,vy,alph);
  Me=5.97.*10.^(24);                          %initializing mass of earth
  G=6.67.*10.^(-11);                          %initializing G
  Rs=sqrt(x.^2+y.^2);                         %calculating satellite radius
  f1x=G*Me./Rs.^2;                            %Gravity towards earth
  f2x=-x*f1x./Rs;                         %getting the x part of gravity
  ax=f2x-alph*vx;                       %calculating acceleration with drag
endfunction;

%=====declaring function for velocity x=======
function Vx=funX(vx);
  Vx=vx;
endfunction;

%=====declaring function for acceleration y===
function ay=funVY(x,y,vx,vy,alph);
  Me=5.97.*10.^(24);                          %initializing mass of earth
  G=6.67.*10.^(-11);                          %initializing G
  Rs=sqrt(x.^2+y.^2);                         %calculating satellite radius
  f1y=G*Me./Rs.^2;                            %Gravity towards earth
  f2y=-y*f1y./Rs;                           %getting the y part of gravity
  ay=f2y-alph*vy;                       %calculating acceleration with drag
endfunction;

%=====declaring function for velocity y=======
function Vy=funY(vy);
  Vy=vy;
endfunction;

%=====loop for time progression and graphing==
for t=0:dt:100000;

  %=====X + Y variable========================
  %---calculating k1---
  k1x=funX(vx);
  k1y=funY(vy);
  k1Vx=funVX(x,y,vx,vy,alph);
  k1Vy=funVY(x,y,vx,vy,alph);

  %---calculating k2---
  k2x=funX(vx+k1Vx*dt/2);
  k2y=funY(vy+k1Vy*dt/2);
  k2Vx=funVX(x+k1x*dt/2,y+k1y*dt/2,vx+k1Vx*dt/2,vy+k1Vy*dt/2,alph);
  k2Vy=funVY(x+k1x*dt/2,y+k1y*dt/2,vx+k1Vx*dt/2,vy+k1Vy*dt/2,alph);

  %---calculating k3---
  k3x=funX(vx+k2Vx*dt/2);
  k3y=funY(vy+k2Vy*dt/2);
  k3Vx=funVX(x+k2x*dt/2,y+k2y*dt/2,vx+k2Vx*dt/2,vy+k2Vy*dt/2,alph);
  k3Vy=funVY(x+k2x*dt/2,y+k2y*dt/2,vx+k2Vx*dt/2,vy+k2Vy*dt/2,alph);
```

```
  %---calculating k4---
  k4x=funX(vx+k3Vx*dt);
  k4y=funY(vy+k3Vy*dt);
  k4Vx=funVX(x+k3x*dt,y+k3y*dt,vx+k3Vx*dt,vy+k3Vy*dt,alph);
  k4Vy=funVY(x+k3x*dt,y+k3y*dt,vx+k3Vx*dt,vy+k3Vy*dt,alph);

  %---calculating new x & y---
  x=x+(k1x+2*k2x+2*k3x+k4x)*dt/6;
  y=y+(k1y+2*k2y+2*k3y+k4y)*dt/6;

  %---calculating new vx & vy---
  vx=vx+(k1Vx+2*k2Vx+2*k3Vx+k4Vx)*dt/6;
  vy=vy+(k1Vy+2*k2Vy+2*k3Vy+k4Vy)*dt/6;

  %===creating line of y and x================
  line([x0  x],[y0 y],'linewidth',[2]);        %line of x and y
  x0=x;
  y0=y;
  drawnow;

  %===Break if projectile hits the ground====
  if sqrt(x.^2+y.^2)<6.4*10^6;
    break;
  endif;

  if x<3*10^6 && x>0;
    n=n+1;
  endif;

  if alph==0 && n>60 && x>0 && y>0;
    break;
  endif;
endfor;
```

## Movement of Pendulum

This program simply simulates the movement of a spring pendulum. Additionally, the velocity of the pendulum is drawn in the same coordinate system.

```
%=====preparing window==========================
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],...
       'name', "Newton's 2nd Law");   %Creating Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]);
      %Creating Axis
txt = ["Newton's 2nd Law - Spring Pendulum\nBlack - Position ; Red -
      Velocity"];
      %preparing amplitude string
a : title (txt,'fontsize',[17]);      %titling diagram
a : xlabel ("T in seconds",'fontsize',[17]);  %labeling y axis
```

```
a : ylabel ("X in meters",'fontsize',[17]);   %labeling x axis
axis('equal',[0 10 -3 3]);   %setting the area of the coordinate system
grid on;       %turning on the grid in coordinate system
drawnow;       %drawing the system

%=====setting initial conditions==============
t0=0;                          %initializing time
x=x0=1;                        %setting x to starting position
vx=v0=0;                       %setting initial velocity
dt=.05;                        %setting interval

%=====declaring function for acceleration=========
function ax=funVX(t,x,vx);

  k=1;                                 %stiffness of spring
  m=1;                                 %mass of weight
  a=.5;                               %coefficient of drag
  ax=-k*x/m;                           %acceleration

endfunction;

%=====declaring function for velocity=========
function Vx=funX(t,x,vx);

  Vx=vx;                       %velocity put into another place

endfunction;

%=====loop for time progression and graphing====
for t=0:dt:10;

  %---calculating k1---
  k1x=funX(t,x,vx);
  k1Vx=funVX(t,x,vx);
  %---calculating k2---
  k2x=funX(t+dt/2,x+k1x*dt/2,vx+k1Vx*dt/2);
  k2Vx=funVX(t+dt/2,x+k1x*dt/2,vx+k1Vx*dt/2);
  %---calculating k3---
  k3x=funX(t+dt/2,x+k2x*dt/2,vx+k2Vx*dt/2);
  k3Vx=funVX(t+dt/2,x+k2x*dt/2,vx+k2Vx*dt/2);
  %---calculating k4---
  k4x=funX(t+dt,x+k3x*dt,vx+k3Vx*dt);
  k4Vx=funVX(t+dt,x+k3x*dt,vx+k3Vx*dt);
  %---calculating new x---
  x=x+(k1x+2*k2x+2*k3x+k4x)*dt/6;
  %---calculating new vx---
  vx=vx+(k1Vx+2*k2Vx+2*k3Vx+k4Vx)*dt/6;
  %---creating line of t and x---
  line([t0 t],[x0 x],'linewidth',[3]);                %line of x
  line([t0 t],[v0 vx],'linewidth',[3],'color',"red"); %line of vx
  %---storing variables for line drawing---
```

```
  t0=t;
  x0=x;
  v0=vx;

  %---drawing line---
  drawnow;

endfor;
```

## Projectile

This program simulates the movement of a projectile that is fired at an angle and moves a certain initial speed. The user can enter both values and the coefficient of drag, which slows down the projectile like air would in real life. The program also shows how far the projectile would have travelled without drag by drawing a red line there.

```
%-------dialog box asking user for input-----------
input = inputdlg ({"Starting Velocity v0", "Angle fi (deg.)", ...
"Coefficient of Drag"},"Projectile Motion",[1,20;1,20;1,20],{"10", "45",
      ".05"}); %getting user input
v0=str2num(input{1});          %velocity converted from user input
fi0=str2num(input{2});         %angle converted from user input
alpha=str2num(input{3});       %coef. of drag converted from user input

%=====preparing window==========================
figure('units','norm','position',[.1 .1 .8 .7],'color',[.9 .9 .9],...
       'name', "Newtons 2nd Law");                                 %Creating
      Window
a = axes('position',[.1 .1 .8 .8],'color',[1 1 1],'fontsize',[12]); %Creating
      Axis
txt = ["Newtons 2nd Law - Projectile Motion\nv0=" num2str(v0) ...
"m/s ; fi=" num2str(fi0) "° ; alpha=" num2str(alpha)];           %preparing
      amplitude string
a : title (txt,'fontsize',[17]);            %titleing diagram
a : xlabel ("X in meters",'fontsize',[17]); %labeling y axis
a : ylabel ("Y in meters",'fontsize',[17]); %labeling x axis
grid on;                                    %turing on the grid in coordinate
      system
drawnow;                                    %drawing the system

%=====setting initial conditions==============
x=x0=y0=y=0;        %setting x/y to starting position
g=9.81;             %setting g
dt=.05;             %setting interval
fi=deg2rad(fi0);    %converting angle from degrees to rad
vx=cos(fi)*v0;      %calculating x-component of velocity
vy=sin(fi)*v0;      %calculating y-component of velocity
lxp=((v0.^2.*sin(2.*fi))./g)*1.02;      %x-limit of first bounce
lyp=((v0.^2.*sin(fi).^2)/(2.*g)).*1.15; %calculating y limit
axis('equal',[0 lxp 0 lyp]);            %setting axis limits
```

```
line([lxp/1.02 lxp/1.02],[0 lyp],'color',"red"); %drawing line where
      projectile wihtout drag lands
drawnow;       %Drawing it now
limt=lxp/vx;   %calculating limit of loop
if limt<3;     %setting more accurate parameters for low v0's
  limt=3;
  dt=.01;
endif;

%=====declaring function for acceleration x=========
function ax=funVX(t,x,vx);
  ax=0;
endfunction;

%=====declaring function for velocity x=========
function Vx=funX(t,x,vx,alpha);
  Vx=vx;
  Vx=vx-alpha*Vx;           %adding drag
endfunction;

%=====declaring function for acceleration y=========
function ay=funVY(t,y,vy);
  ay=0;
endfunction;

%=====declaring function for velocity y=========
function Vy=funY(t,y,vy,alpha);
  g=9.81;
  Vy=vy-g*t;                  %vy changed by gravity
  Vy=Vy-alpha*Vy;             %additional drag
endfunction;

%=====loop for time progression and graphing====
for t=0:dt:limt;

  %=====X variable=====
  %---calculating k1---
  k1x=funX(t,x,vx,alpha);
  k1Vx=funVX(t,x,vx);
  %---calculating k2---
  k2x=funX(t+dt/2,x+k1x*dt/2,vx+k1Vx*dt/2,alpha);
  k2Vx=funVX(t+dt/2,x+k1x*dt/2,vx+k1Vx*dt/2);
  %---calculating k3---
  k3x=funX(t+dt/2,x+k2x*dt/2,vx+k2Vx*dt/2,alpha);
  k3Vx=funVX(t+dt/2,x+k2x*dt/2,vx+k2Vx*dt/2);
  %---calculating k4---
  k4x=funX(t+dt,x+k3x*dt,vx+k3Vx*dt,alpha);
  k4Vx=funVX(t+dt,x+k3x*dt,vx+k3Vx*dt);
  %---calculating new x---
  x=x+(k1x+2*k2x+2*k3x+k4x)*dt/6;
  %---calculating new vx---
```

```
  vx=vx+(k1Vx+2*k2Vx+2*k3Vx+k4Vx)*dt/6;

  %=====Y variable=====
  %---calculating k1---
  k1y=funY(t,y,vy,alpha);
  k1Vy=funVY(t,y,vy);
  %---calculating k2---
  k2y=funY(t+dt/2,y+k1y*dt/2,vy+k1Vy*dt/2,alpha);
  k2Vy=funVY(t+dt/2,y+k1y*dt/2,vy+k1Vy*dt/2);
  %---calculating k3---
  k3y=funY(t+dt/2,y+k2y*dt/2,vy+k2Vy*dt/2,alpha);
  k3Vy=funVY(t+dt/2,y+k2y*dt/2,vy+k2Vy*dt/2);
  %---calculating k4---
  k4y=funY(t+dt,y+k3y*dt,vy+k3Vy*dt,alpha);
  k4Vy=funVY(t+dt,y+k3y*dt,vy+k3Vy*dt);
  %---calculating new y---
  y=y+(k1y+2*k2y+2*k3y+k4y)*dt/6;
  %---calculating new vy---
  vy=vy+(k1Vy+2*k2Vy+2*k3Vy+k4Vy)*dt/6;

  %---creating line of y and x---
  line([x0 x],[y0 y],'linewidth',[3]);                %line of x and y
  %---storing variables for line drawing---
  x0=x;
  y0=y;

  %---drawing line---
  drawnow;

  %---Break if projectile hits the ground---
  if y<0;
    break;
  endif;

endfor;
```