```
% Celestial Mechanics:  Three-Body Problem Plot (Earth-Sun-Mars)
%                             *** "Restricted" Three-Body Problem
% Inputs (Function Arguments):
%   1.  Time Interval
%   2.  Initial Conditions
%       ic = [ x1 y1 vx1 vy1 x2 y2 vx2 vy2 x3 y3 vx3 vy3]
%              x position, y position, x velocity, y velocty,   Body #1
%              x position, y position, x velocity, y velocty    Body #2
%              x position, y position, x velocity, y velocty    Body #3
%   3.  Number of Steps: n
%   4.  Steps per point plotted: p
%   5.  Differential Equation "Solver"
%           a.  1 = RK4
%           b.  2 = Trapezoid
% Calls:
%       1.  Runge-Kutta (RK4)
%       2.  Trapezoid Method (trapstep.m:   one-step)
% Invocation: "Command Line" -
%   1.  Clear workspace variables:
%           clear all; clearvars; close all; clc;
%   2.  Example Usage:
% Prob.   Rearrange             |---Earth---||----Sun-----| |---Mars----|
%  #      Function Arguments:  x1 y1 vx1 vy1 x2 y2 vx2 vy2 x3 y3 vx3 vy3
% #12     orbit3body([0 100], [2  2 .2 -.2  0  0  0   0   -2 -2 -0.2 0.2], 10000, ↙
10,1)
%                   *** Orbit3Body_fig00.fig
%                   *** Orbit3Body_RK4_00.mp4

% Prospective Upgrades:
%   1.  Add dynamic "Center-of-Mass" (COM) "X" position
function z=orbit3body(inter,ic,n,p,solver)

% plot n points over interval
h=(inter(2)-inter(1))/n;

% Incorporate Initial Conditons
x1=ic(1);y1=ic(2);vx1=ic(3);vy1=ic(4);
x2=ic(5);y2=ic(6);vx2=ic(7);vy2=ic(8);
x3=ic(9);y3=ic(10);vx3=ic(11);vy3=ic(12);

y(1,:)=[x1 vx1 y1 vy1 x2 vx2 y2 vy2 x3 vx3 y3 vy3];
%y(1,:)=[x1 y1 vx1 vy1 x2 y2 vx2 vy2]
%vector_position_velocity = [x1 y1 vx1 vy1 x2 y2 vx2 vy2 x3 y3 vx3 vy3]
t(1)=inter(1);

% Construct Animation Objects for Graph
set(gca,'XLim',[-5 5],'YLim',[-5 5],...
    'XTick',[-5 0 5],'YTick',[-5 0 5]);
earth=animatedline('color','r','Marker','.','markersize',35);
```

```
earthtail=animatedline('color','b','LineStyle','-');
sun=animatedline('color','y','Marker','.','markersize',50);
suntail=animatedline('color', 'b', 'LineStyle', '-');
mars=animatedline('color','m','Marker','.','markersize',35);
marstail=animatedline('color','g','LineStyle','-');

grid on;

title({'Celestial Mechanics:  Three (3) Body Problem';
       'Problem 12a (RK4):  m1=m3=0.03, m2=0.3    (Orbit3Body_fig00.fig)' });
%title('Celestial Mechanics:  Three(3) Body Problem');
%title({'Celestial Mechanics:  Two(2) Body Problem';'IC:  x1 = ',num2str(x(1))});

%[px,py]=ginput(1);              % Include these three lines
%[px1,py1]=ginput(1);            %   to enable mouse support
% y(1,:)=[px px1-px py py1-py]; %   2 clicks set direction

%v=VideoWriter('num9','MPEG-4');
v=VideoWriter('Orbit3Body_RK4_00','MPEG-4');
open(v)

for k=1:n/p
    for i=1:p
        t(i+1)=t(i)+h;

        switch(solver)
            case 1
%               vector_position_velocity(i+1,:)=rk4step(t(i),vector_position_velocity
(i,:),h);
                y(i+1,:)=rk4step(t(i),y(i,:),h);
                fprintf('===> RK4 <===\n' );
            case 2
%               vector_position_velocity(i+1,:)=trapstep(t(i),
vector_position_velocity(i,:),h);
                y(i+1,:)=trapstep(t(i),y(i,:),h);
                fprintf('===> Trapezoid Method <===\n' );
            otherwise
                fprintf('*** "Scope" Parameter Invalid!!!\n' );
        end
    end
    y(1,:)=y(p+1,:);
    t(1)=t(p+1);
    clearpoints(sun);
    clearpoints(earth);
    clearpoints(mars);

    addpoints(earth,y(1,1),y(1,3))
    addpoints(earthtail,y(1,1),y(1,3))
    addpoints(sun,y(1,5),y(1,7))
```

```
    addpoints(suntail,y(1,5),y(1,7))
    addpoints(mars,y(1,9),y(1,11))
    addpoints(marstail,y(1,9),y(1,11))

    drawnow;
    frame=getframe;
    writeVideo(v,frame);
end
close(v)

%function y = trapstep(t,x,h) % one step of the Trapezoid Method
function y = trapstep(t,vec_pv,h) % one step of the Trapezoid Method
    % Evaluating each endpoint; taking average y value

    z1=ydot(t,vec_pv);
    g=vec_pv+h*z1;
    z2=ydot(t+h,g);
    y=vec_pv+h*(z1+z2)/2;

function y=rk4step(t,vpv,h)
    s1 = ydot(t,vpv);
    s2 = ydot(t+h/2,vpv+h*s1/2);
    s3 = ydot(t+h/2,vpv+h*s2/2);
    s4 = ydot(t+h,vpv+h*s3);
    y=vpv+h*(s1+2*s2+2*s3+s4)/6;

% Differential Equations
function z = ydot(t,x)
m1=0.03;     % Earth
m2=0.3;      % Sun
m3=0.03;    % Mars

g=1;
mg1=m1*g;
mg2=m2*g;
mg3=m3*g;

z=zeros(1,12);

px1=x(1);py1=x(3);vx1=x(2);vy1=x(4);
px2=x(5);py2=x(7);vx2=x(6);vy2=x(8);
px3=x(9);py3=x(11);vx3=x(10);vy3=x(12);

dist12=sqrt((px1-px2)^2+(py1-py2)^2);
dist13=sqrt((px1-px3)^2+(py1-py3)^2);
dist21=sqrt((px2-px1)^2+(py2-py1)^2);
dist23=sqrt((px2-px3)^2+(py2-py3)^2);
dist31=sqrt((px3-px1)^2+(py3-py1)^2);
dist32=sqrt((px3-px2)^2+(py3-py2)^2);
```

```
z(1)=vx1;
z(2)=(mg2*(px2-px1))/(dist21^3) + (mg3*(px3-px1))/(dist31^3);
z(3)=vy1;
z(4)=(mg2*(py2-py1))/(dist21^3) + (mg3*(py3-py1))/(dist31^3);
z(5)=vx2;
z(6)=(mg1*(px1-px2))/(dist12^3) + (mg3*(px3-px2))/(dist32^3);
z(7)=vy2;
z(8)=(mg1*(py1-py2))/(dist12^3) + (mg3*(py3-py2))/(dist32^3);
z(9)=vx3;
z(10)=(mg1*(px1-px3))/(dist13^3) + (mg2*(px2-px3))/(dist23^3);
z(11)=vy3;
z(12)=(mg1*(py1-py3))/(dist13^3) + (mg2*(py2-py3))/(dist23^3);

z_temp = z

%******************** BELOW THE LINE ************************************
%{
    addpoints(earth,y(1,1),y(1,2))
    addpoints(earthtail,y(1,1),y(1,2))
    addpoints(sun,y(1,5),y(1,6))
    addpoints(suntail,y(1,5),y(1,6))
%}

% build y vector

        %y(i+1,:)=trapstep(t(i),y(i,:),h);

%{
figure
h = animatedline;
axis([0 4*pi -1 1])

for x = linspace(0,4*pi,10000)
    y = sin(x);
    addpoints(h,x,y)
    drawnow limitrate
end
%}
%{
    addpoints(head,y(1,1),y(1,3))
    addpoints(sun,y(1,5),y(1,7))
    addpoints(tail,y(1,1),y(1,3))
    addpoints(suntail,y(1,5),y(1,7))
%}
%{
    temp00 = vector_position_velocity(1,1)
    temp01 = vector_position_velocity(1,2)
    temp02 = vector_position_velocity(1,3)
```

```
    temp03 = vector_position_velocity(1,4)
    temp04 = vector_position_velocity(1,5)
    temp05 = vector_position_velocity(1,6)
    temp06 = vector_position_velocity(1,7)
    temp07 = vector_position_velocity(1,8)
%}
%{
    addpoints(earth,y(1,1),y(1,2))
    addpoints(earthtail,y(1,1),y(1,2))
    addpoints(sun,y(1,5),y(1,6))
    addpoints(suntail,y(1,5),y(1,6))
%}
%head=animatedline('color','r','Marker','.','markersize',35);


%dist=sqrt((px2-px1)^2+(py2-py1)^2);
%z=zeros(1,8);

%{
z(1)=vx1;
z(2)=vy1;
z(3)=(mg2*(px2-px1))/(dist^3);
z(4)=(mg2*(py2-py1))/(dist^3);
z(5)=vx2;
z(6)=vy2;
z(7)=(mg1*(px1-px2))/(dist^3);
z(8)=(mg1*(py1-py2))/(dist^3);
%}
%z(7)=(mg1*(px1-px2))/(dist21^3) + (;
%z(8)=(mg1*(py1-py2))/(dist^3);
%z(11)=(mg3*(px1-px2))/(dist^3);
%z(12)=(mg3*(py1-py2))/(dist^3);

%{
    temp_x1  = vector_position_velocity(1,1)
    temp_y1  = vector_position_velocity(1,2)
    temp_vx1 = vector_position_velocity(1,3)
    temp_vy1 = vector_position_velocity(1,4)
    temp_x2  = vector_position_velocity(1,5)
    temp_y2  = vector_position_velocity(1,6)
    temp_vx2 = vector_position_velocity(1,7)
    temp_vy2 = vector_position_velocity(1,8)
    temp_x3  = vector_position_velocity(1,9)
    temp_y3  = vector_position_velocity(1,10)
    temp_vx3 = vector_position_velocity(1,11)
    temp_vy3 = vector_position_velocity(1,12)
%}
%{
    addpoints(earth,vector_position_velocity(1,1),vector_position_velocity(1,2))
```

```
    addpoints(earthtail,vector_position_velocity(1,1),vector_position_velocity(1,2))
    addpoints(sun,vector_position_velocity(1,5),vector_position_velocity(1,6))
    addpoints(suntail,vector_position_velocity(1,5),vector_position_velocity(1,6))
    addpoints(mars,vector_position_velocity(1,9),vector_position_velocity(1,10))
    addpoints(marstail,vector_position_velocity(1,9),vector_position_velocity(1,10))
%}
% #9   (a)  orbit2btest03([0 100], [2  2 .2 -.2 0  0  -.02 .02], 10000, 5,1)
% #9   (b)  orbit2btest03([0 100], [2  2 .2 -.2 0  0  -.02 .02], 10000, 10,1)
% #10  (c)  orbit2btest03([0 100], [0  1 .1 -.1 -2 -1 -.01 .01], 10000, 10k,1)
% #10  (c)  orbit2btest03([0 100], [0  1 .1 -.1 -2 -1 -.01 .01], 10000, 10,1)
% #10  (c)  orbit2btest03([0 100], [0  1 .1 -.1 -2 -1 -.01 .01], 10000, 10,2)
%
%      (d)  orbit2btest03([0 100], [2  2 .25 -.25 0  0  -.02 .02], 10000, 10)
%      (e)  orbit2btest03([0 100], [2.1 2.1 .25 -.25 0  0  -.02 .02], 10000, 5)
%      (f)  orbit2btest03([0 100], [2 .2  2 -.2 0 -.02  0  .02], 10000, 5)
%                                  x1 vx1 y1 vy1 x2 vx2 y2 vy2
%                                  |-------- Original -------|
%{
    z1=ydot(t,x);
    g=x+h*z1;
    z2=ydot(t+h,g);
    y=x+h*(z1+z2)/2;
%}
    %y(1,:)=y(p+1,:);
%   vector_position_velocity(1,:)=vector_position_velocity(p+1,:);
%{
px1=x(1);py1=x(2);vx1=x(3);vy1=x(4);
px2=x(5);py2=x(6);vx2=x(7);vy2=x(8);
px3=x(9);py3=x(10);vx3=x(11);vy3=x(12);
dist12=sqrt((px1-px2)^2+(py1-py2)^2);
dist13=sqrt((px1-px3)^2+(py1-py3)^2);
dist21=sqrt((px2-px1)^2+(py2-py1)^2);
dist23=sqrt((px2-px3)^2+(py2-py3)^2);
dist31=sqrt((px3-px1)^2+(py3-py1)^2);
dist32=sqrt((px3-px2)^2+(py3-py2)^2);
%}
%{
z(1)=px1;
z(2)=py1;
z(3)=(mg2*(px2-px1))/(dist21^3) + (mg3*(px3-px1))/(dist31^3);
z(4)=(mg2*(py2-py1))/(dist21^3) + (mg3*(py3-py1))/(dist31^3);
z(5)=px2;
z(6)=py2;
z(7)=(mg1*(px1-px2))/(dist12^3) + (mg3*(px3-px2))/(dist32^3);
z(8)=(mg1*(py1-py2))/(dist12^3) + (mg3*(py3-py2))/(dist32^3);
z(9)=px3;
z(10)=py3;
z(11)=(mg1*(px1-px3))/(dist13^3) + (mg2*(px2-px3))/(dist23^3);
z(12)=(mg1*(py1-py3))/(dist13^3) + (mg2*(py2-py3))/(dist23^3);
```

```
%}
%{
z(3)=vx1;
z(2)=(mg2*(px2-px1))/(dist21^3) + (mg3*(px3-px1))/(dist31^3);
z(3)=vy1;
z(4)=(mg2*(py2-py1))/(dist21^3) + (mg3*(py3-py1))/(dist31^3);
z(5)=vx2;
z(6)=(mg1*(px1-px2))/(dist12^3) + (mg3*(px3-px2))/(dist32^3);
z(7)=vy2;
z(8)=(mg1*(py1-py2))/(dist12^3) + (mg3*(py3-py2))/(dist32^3);
z(9)=vx3;
z(10)=(mg1*(px1-px3))/(dist13^3) + (mg2*(px2-px3))/(dist23^3);
z(11)=vy3;
z(12)=(mg1*(py1-py3))/(dist13^3) + (mg2*(py2-py3))/(dist23^3);
%}
%pause
```