```
% orbit - Program to compute the orbit of a comet.
clear all;

type grav_L.m

%* Set initial position and velocity of the comet.
r0 = 1; % AU
v0 = 2*pi; % AU/yr
r = [r0 0];  v = [0 v0];
state = [ r(1) r(2) v(1) v(2) ];   % Used by R-K routines

%* Set physical parameters (mass, G*M)
GM = 4*pi^2;      % Grav. const. * Mass of Sun (au^3/yr^2)
alpha = 0.02;
mass = 1.;        % Mass of comet
Fp = 0.01 * (GM * mass)/(r0 ^ 2);
adaptErr = 1.e-3; % Error parameter used by adaptive Runge-Kutta
time = 0;

%* Loop over desired number of steps using specified
%  numerical method.
nStep = 2000;
tau = 0.005;
NumericalMethod = 3;
for iStep=1:nStep

  %* Record position and energy for plotting.
  rplot(iStep) = norm(r);           % Record position for polar plot
  thplot(iStep) = atan2(r(2),r(1));
  tplot(iStep) = time;
  kinetic(iStep) = .5*mass*norm(v)^2;   % Record energies
  potential(iStep) = - GM*mass/norm(r);

  %* Calculate new position and velocity using desired method.
  if( NumericalMethod == 1 )
    accel = -GM*r/norm(r)^3;
    r = r + tau*v;             % Euler step
    v = v + tau*accel;
    time = time + tau;
  elseif( NumericalMethod == 2 )
    accel = -GM*r/norm(r)^3;
    v = v + tau*accel;
    r = r + tau*v;             % Euler-Cromer step
    time = time + tau;
  elseif( NumericalMethod == 3 )
    param = [GM, Fp];
    state = rk4(state,time,tau,'grav_L',param);
    r = [state(1) state(2)];   % 4th order Runge-Kutta
    v = [state(3) state(4)];
    time = time + tau;
  else
    [state time tau] = rka(state,time,tau,adaptErr,'grav_L',param);
    r = [state(1) state(2)];   % Adaptive Runge-Kutta
```

```
    v = [state(3) state(4)];
  end
  % update L
  L(iStep) = norm(cross([r, 0], [v, 0]));
end

%* Graph the trajectory of the comet.
figure(1); clf;  % Clear figure 1 window and bring forward
polar(thplot,rplot,'+');  % Use polar plot for graphing orbit
xlabel('Distance (AU)');  grid;
pause(1)   % Pause for 1 second before drawing next plot

figure(2)
plot(L, tplot)
```