```
% ### EXtwobodyB.m ###       11.18.16 C. Bergevin

% Example code to simulate two body motion via numeric integration.
% Modified from original source code by K. Berwick (re Giordano's
book; see
% his original comments at bottom of code)
% NOTE: This version of the code simply plots everything at the end

% ---
% As per Giordano sec.4.1, assumptions made are:
% o there are two masses (m_S and m_E)
% o m_S >> m_E such that its motion can be neglected
% o by virtue of expressing things w/ units of AU, the consts. G and
m_S
% (which always appears together as a product) can appear simply as
4*pi^2
% when using this choice of units. Proof:
%   - m_S= 1.989 ◊ 10^30 kg
%   - G= 6.67408 ◊ 10-11 m^3 kg^-1 s^-2
%   --> G*m_S = 1.3275 x 10^20 m^3 s^-2
%   - 1 AU = 1.5 x 10^11 m
%   - 1 yr ~ 3.15 x 10^7 s
%   --> 1 m^3 s^-2 = (2.94 x 10^-19) AU^3 yr^-2
%   ** --> G*m_S = 1.3275 x 10^20 m^3 s^-2 ~ 39 AU^3 yr^-2 ~ 4*pi^2
AU^3 yr^-2
% o code uses a 2nd order Runge-Kutta routine (as opposed to the
simpler
% Euler method)

clear
% ===============================================

Np= 10000;    % # of time steps to compute {5000}
dt = 0.001;     % time step [yrs] {0.002}
x0=1;            % initial x-position of planet [AU] {1}
y0=0;            %    "    y-position     "   [AU] {0}
vx0= 0;          %    "    x velocity     "   [AU/yr] {0}
vy0= 1.3*pi;          %    "    y velocity     "   [AU/yr] {2*pi}

% ===============================================

% initialize variables
t=0;
x(1)= x0;   y(1)= y0;
vx(1)= vx0; vy(1)= vy0;

% +++++++++++
% loop over the timesteps
for nn = 1:Np-1;
```

```
        r= sqrt(x(nn)^2+ y(nn)^2);    % determine radius
        y_dash= y(nn) +0.5* vy(nn)* dt;
        vy_dash= vy(nn)- 0.5* (4* pi^2* y(nn)* dt)/(r^3);
        % update positions and new y velocity
        y_new=y(nn)+ vy_dash* dt;
        vy_new=vy(nn)-(4* pi^2* y_dash* dt)/(r^3);
        % Compute Runge Kutta values for the x equations
        x_dash= x(nn)+ 0.5* vx(nn)* dt;
        vx_dash= vx(nn) - 0.5* (4*pi^2* x(nn)* dt)/(r^3);
        % update positions using newly calculated velocity
        x_new= x(nn)+ vx_dash* dt;
        vx_new= vx(nn)-(4*pi^2* x_dash* dt)/(r^3);

        % Update x and y velocities with new velocities
        vx(nn+1)= vx_new;
        vy(nn+1)= vy_new;
        % Update x and y with new positions
        x(nn+1)= x_new;
        y(nn+1)= y_new;


end;

% +++++++++++
% Plot the Sun at the origin
figure(1); clf;
plot(0,0,'oy','MarkerSize',30, 'MarkerFaceColor','yellow');
xlim(1.2*[min(x) max(x)]);
ylim(1.2*[min(y) max(y)]);
xlabel('x_E (AU)'); ylabel('y_E (AU)'); hold on; grid on;
title('Two-body solutions (m_S and m_E; assumes m_S>>m_E and thus S
movement is negligible')
plot(x,y,'k.','MarkerSize',8);
h1=plot(x(1),y(1),'gs','LineWidth',2,'MarkerSize',8);
h2=plot(x(end),y(end),'ro','LineWidth',2,'MarkerSize',8);
legend([h1 h2],'start point','end point');


% ==============================================
% [original Berwick comments below]
% Planetary orbit using second order Runge-Kutta method.
% by Kevin Berwick,
% based on 'Computational Physics' book by N Giordano and H Nakanishi
% Section 4.2 %
```