Numerical Methods in Astrophysics

Project 3

Two Dimensional Random Walk, Circular Binary and Hypervelocity Stars

Saksham Kaushal

Contents

1 Problem1 - Two Dimensional Random Walk

1.1 Introduction

1.2 Methods

Listing 1: first code

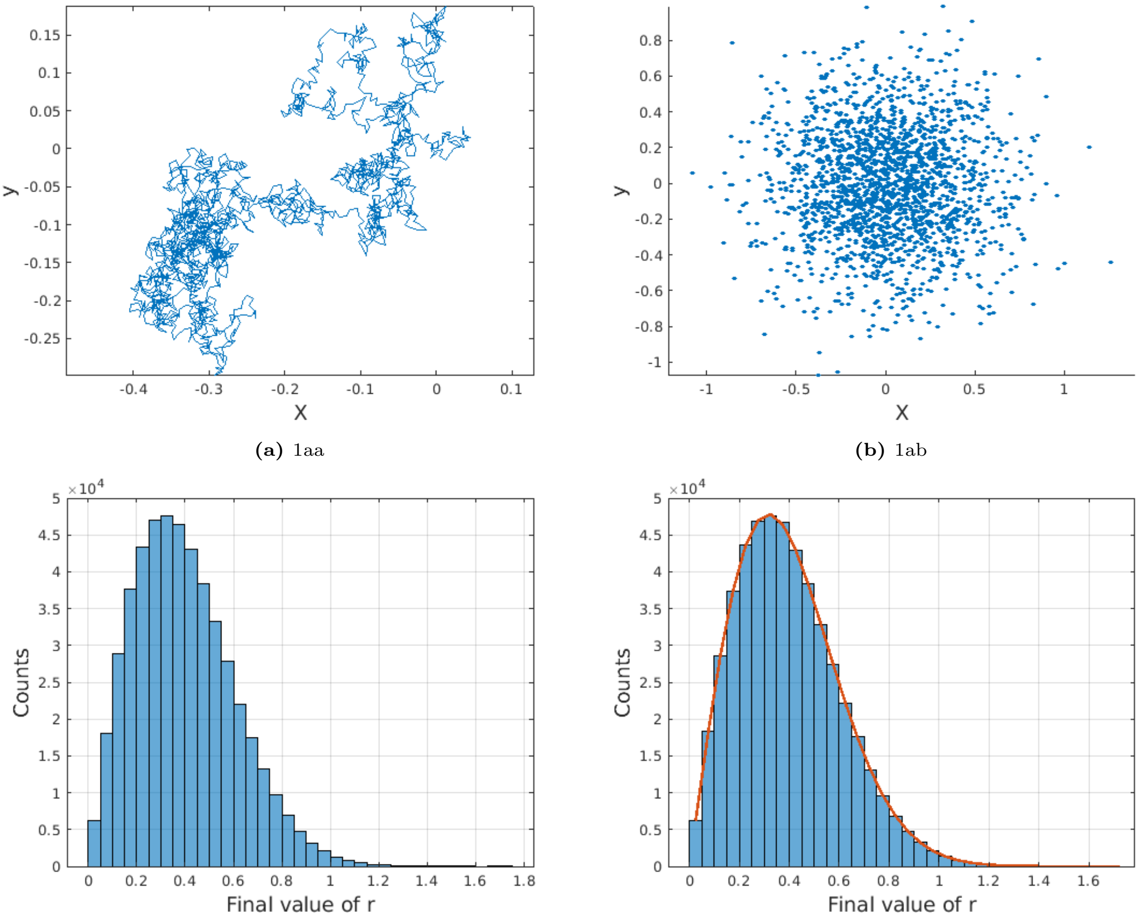
```
clear;
x(1)     = 0;                          % initial x at origin
y(1)     = 0;                          % initial y at origin
d        = 0.01;
for i = 1:2000                         % for loop with 2000 steps
    theta    = 2*pi*rand();            % random theta between zero and 2pi
    x(i+1)   = x(i)+d*cos(theta);      % next value of x
    y(i+1)   = y(i)+d*sin(theta);      % next value of y
end
plot(x,y);                             % plot the random walk
axis equal;                            % equal dimensions for axes
xlabel('X','FontSize',14);             % x-axis label
ylabel('y','FontSize',14);             % y-axis label
```

Listing 2: second code

```
clear;
d   = 0.01;

for i = 1:2000                         % for loop with 2000 steps
    x = 0;                             % initial x at origin
    y = 0;                             % initial y at origin
    for j = 1:2000
        theta    = 2*pi*rand();        % random theta between zero and 2pi
        x        = x+d*cos(theta);     % next value of x
        y        = y+d*sin(theta);     % next value of y
    end
    xfinal(i)    = x;
    yfinal(i)    = y;
    r(i)         = sqrt(x^2+y^2);
end
scatter(xfinal,yfinal,10,"filled") ;   % plot the random walk
axis equal;                            % equal dimensions for axes
xlabel('X','FontSize',14);             % x-axis label
ylabel('y','FontSize',14);             % y-axis label
```

Listing 3: third code

```
clear;                                 % clear variables and functions
tic;                                   % start clock
d        = 0.01;
np       = 500000;                     % number of particles
tstep    = 2000

for i = 1:np
    x = 0;                             % initial x at origin
    y = 0;                             % initial y at origin
    for j = 1:tstep
        theta    = 2*pi*rand();        % random theta between zero and 2pi
        x        = x+d*cos(theta);     % next value of x
        y        = y+d*sin(theta);     % next value of y
    end
    xfinal(i)    = x;
    yfinal(i)    = y;
    r(i)         = sqrt(x^2+y^2);
end

dr           = 0.05;                   % bin width
binedges     = 0:dr:max(r)+dr;         % bin edges. Starts at zero,
                                       % step size of binwidth,
                                       % ends at ceil of maximum value of r.
histogram(r,binedges);
grid on;
xlabel('Final value of r','FontSize',14)
```



(a) 1aa (b) 1ab (c) 1ac (d) 1ad

Figure 1: Problem 1.1

```
ylabel('Counts','FontSize',14)
toc                                            % stop clock
```

Listing 4: fourth code

```
hold on;
midpoints        = binedges+dr/2;        % midpoints of bins, for plotting
for i = 1:length(binedges)
    n(i) = np * exp(-(binedges(i)^2)/(tstep*d*d))...
           * (1-exp((-(2*binedges(i)*dr+dr*dr))/(tstep*d*d)));
end

plot(midpoints,n,'LineWidth',2);
```

1.3 Results

1.4 Discussions

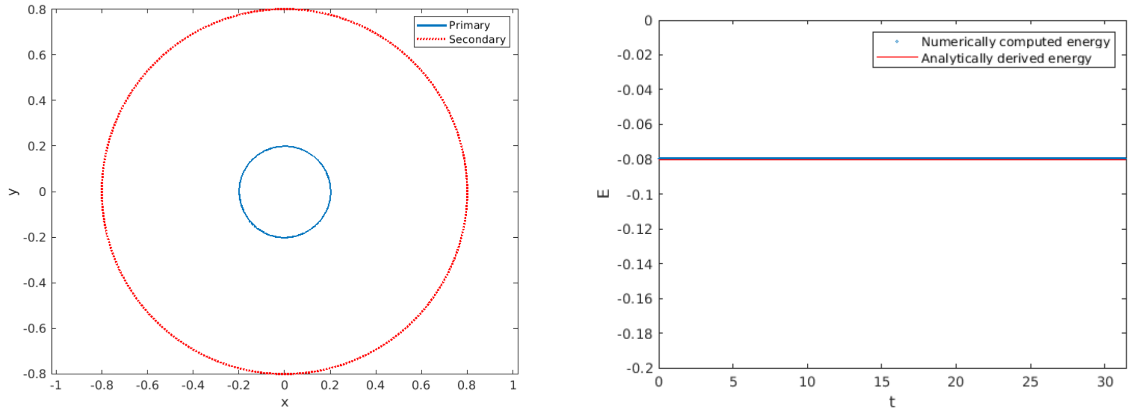
2 Problem2 - Circular Binary

2.1 Introduction

2.2 Methods

Listing 5: first code

```
h       = 1.d-2;        % time-step size
Ns      = 1000  ;       % sampling
mp      = 0.8   ;       % primary mass
```



(a) 2c

(b) 2d

Figure 2: Problem 2

```
ms     = 0.2  ;        % secondary mass
x(1)   = -ms  ;        % primary x
x(2)   = 0    ;        % primary y
x(3)   = 0    ;        % primary vx
x(4)   = -ms  ;        % primary vy
x(5)   = mp   ;        % secondary x
x(6)   = 0    ;        % secondary y
x(7)   = 0    ;        % secondary vx
x(8)   = mp   ;        % secondary vy
tmax   = 10*pi;        % final time
```

Listing 6: second code

```
mp       = 0.8    ;
ms       = 0.2    ;
tmax     = 10*pi  ;
load out                                % load the data file out
t        = out(:,1)  ;                  % time
E        = out(:,6)  ;                  % energy
exact    = -mp*ms/2  ;                  % analytic estimate

plot(t,E,'o','MarkerSize',0.9,...
    'DisplayName','Numerically computed energy')    % numerical energy
hold on
plot([0 tmax],[exact exact],'r',...
    'DisplayName','Analytically derived energy')    % exact line
xlabel('t','FontSize',12)
ylabel('E','FontSize',12)
xlim([0 tmax])
ylim([-0.2 0])
set(gca, 'Fontsize', 10)
legend
```

2.3 Results

2.4 Discussions

3 Problem3 - Hypervelocity Stars

3.1 Introduction

3.2 Methods

Listing 7: code 1

Quantity	Variable	D=3	D=0.1
t_0	t	-19.9555	-15.1290
x_p	x(1)	-400.0000	-980.0000
y_p	x(2)	-917.3151	-199.7975
v_{px}	x(3)	37.6166	44.6972
v_{py}	x(4)	24.4949	4.4721
x_s	x(5)	-400.0000	-980.0000
y_s	x(6)	-916.3151	-198.7975
v_{sx}	x(7)	36.6166	43.6972
v_{sy}	x(8)	24.4949	4.4721

Table 1: Table of variable values for two values of D

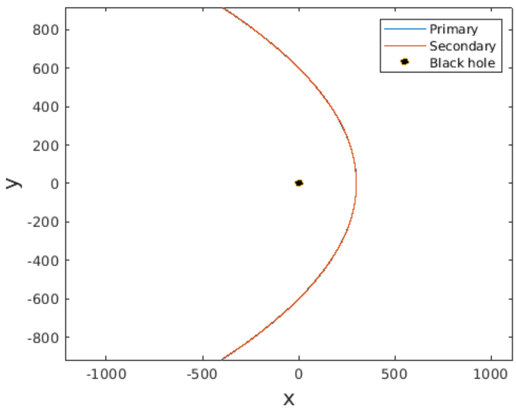
```
Rt       = (mb)^(1/3)                        ;   % tidal radius
R0       = 10 * Rt                           ;   % initial distance between BH and binary
Rp       = 3 * Rt                            ;   % periastron radius
f0       = -acos(-1+(D/5))                   ;   % initial true anomaly (eq 44)
Rdot     = sin(f0) * mb^(1/3) / (sqrt(2*D))  ;   % dR/dt (eq 49a)
Fdot     = (1+cos(f0))^2 * sqrt(2) / ...
           (4*D^(3/2))                       ;   % df/dt (eq 49b)

xcmxdot = Rdot*cos(f0) - R0*Fdot*sin(f0)     ;   % d(xcmx)/dt (using eq 41a)
xcmydot = Rdot*sin(f0) + R0*Fdot*cos(f0)     ;   % d(xcmy)/dt (using eq 41b)

phi      = pi/2                              ;   % binary phase
rpxdot   = -ms*sin(phi+pi)                   ;   % d(rpx)/dt (using problem 2)
rpydot   = mp*cos(phi+pi)                    ;   % d(rpy)/dt (using problem 2)
rsxdot   = -mp*sin(phi)                      ;   % d(rsx)/dt (using problem 2)
rsydot   = ms*cos(phi)                       ;   % d(rsy)/dt (using problem 2)

t        = (sqrt(2)/3) * (D^(3/2)) * ...
           (tan(f0/2))*(3+(tan(f0/2))^2)     ;   % initial time t0
x(1)     = (R0*cos(f0)) + (mp*cos(phi+pi))   ;   % x_p
x(2)     = (R0*sin(f0)) + (mp*sin(phi+pi))   ;   % y_p
x(3)     = xcmxdot+rpxdot                    ;   % v_xp
x(4)     = xcmydot+rpydot                    ;   % v_yp
x(5)     = (R0*cos(f0)) + (ms*cos(phi))      ;   % x_s
x(6)     = (R0*sin(f0)) + (ms*sin(phi))      ;   % y_s
x(7)     = xcmxdot+rsxdot                    ;   % v_xs
x(8)     = xcmydot+rsydot                    ;   % v_ys
```

Listing 8: code 2

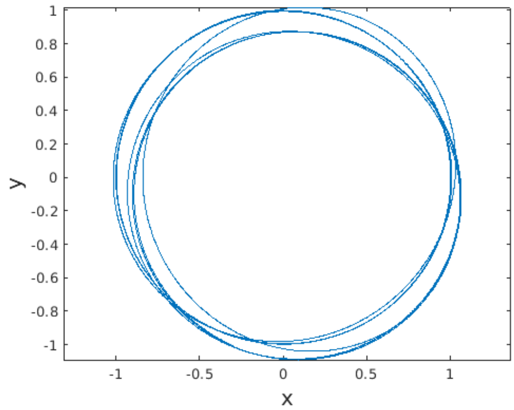
```
function dxdt = f(t,x,mb,mp,ms)
  r = sqrt((x(1)-x(5))^2+(x(2)-x(6))^2)            ;
  rp = sqrt(x(1)^2 + x(2)^2)                       ;
  rs = sqrt(x(5)^2 + x(6)^2)                       ;
  dxdt(1) = x(3)                                   ;   % v_px
  dxdt(2) = x(4)                                   ;   % v_py
  dxdt(3) = (ms*(x(5)-x(1))/r^3)+mb*(-x(1)/rp^3)   ;   % a_px
  dxdt(4) = (ms*(x(6)-x(2))/r^3)+mb*(-x(2)/rp^3)   ;   % a_py
  dxdt(5) = x(7)                                   ;   % v_sx
  dxdt(6) = x(8)                                   ;   % v_sy
  dxdt(7) = (mp*(x(1)-x(5))/r^3)+mb*(-x(5)/rs^3)   ;   % a_sx
  dxdt(8) = (mp*(x(2)-x(6))/r^3)+mb*(-x(6)/rs^3)   ;   % a_sy
```

3.3 Results

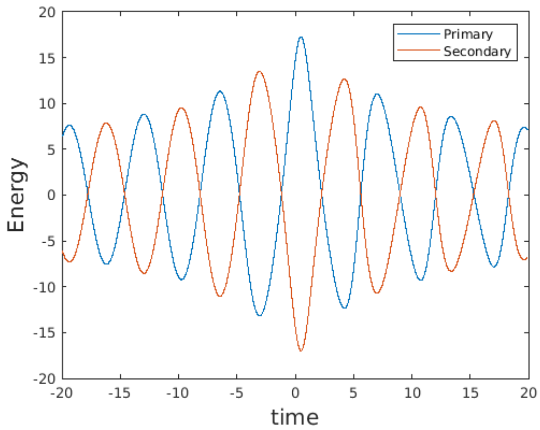
3.4 Discussions



(a) 3a

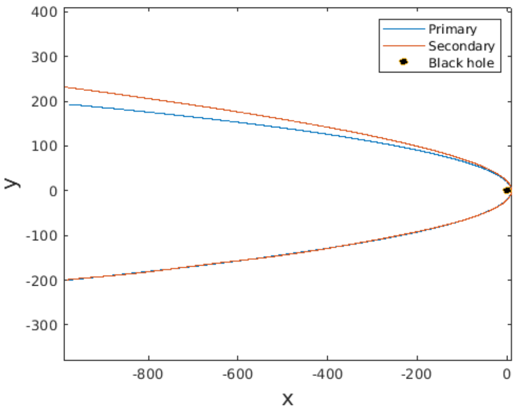


(b) 3b

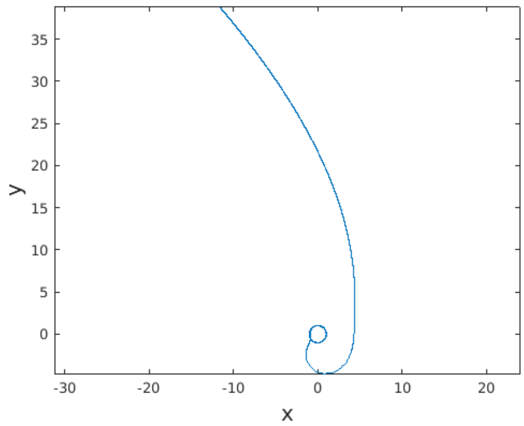


(c) 3c

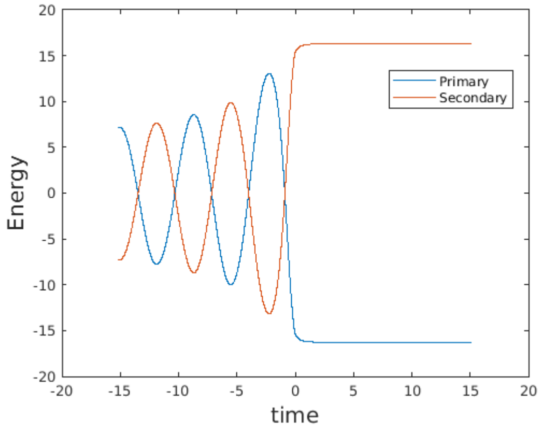
Figure 3: Problem 3 - fig 1



(a) 3da



(b) 3db



(c) 3dc

Figure 4: Problem 3 - fig 2