Numerical Methods in Astrophysics

Project 3

Two Dimensional Random Walk, Circular Binary and Hypervelocity Stars

Saksham Kaushal

Contents

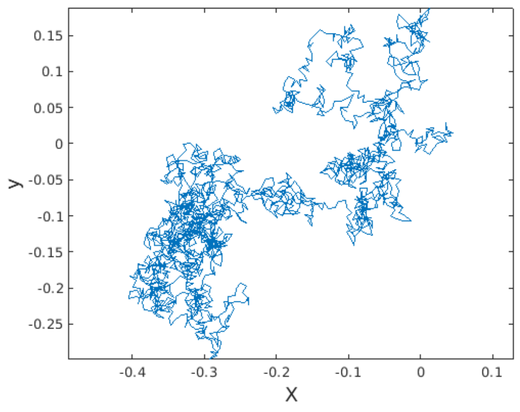
Part I

Two Dimensional Random Walk

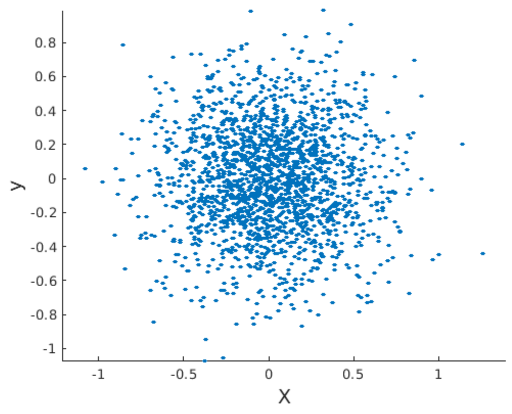
1 Introduction

A random walk is a process that describes the path of an object in a two dimensional plane consisting of successive random steps. In physics, these processes play an important role in study of polymers, Brownian motion, diffusion, etc.

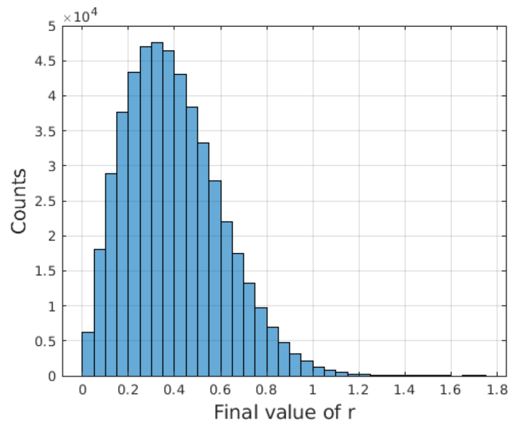
2 Methods, Results and Discussions



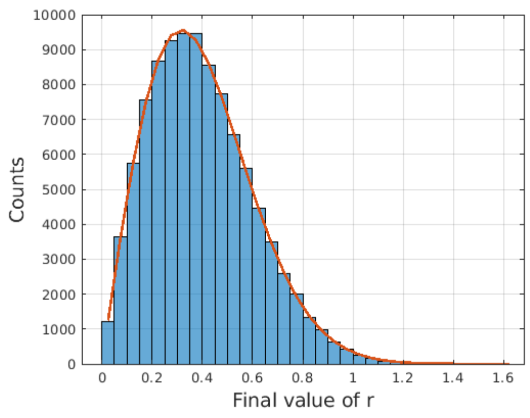
(a) Two dimensional random walk of a particle over 2000 time steps



(b) Distribution of final coordinates of 2000 random walk particles



(c) Number distribution with distance of 5×10^5 random walk particles



(d) Theoretical estimates and observed distribution of 10^5 random walk points. The blue histogram represents the observed number distribution , while the red line is a plot of analytical estimates.

Figure 1: Problem 1.1

(a) **Methods** – Random walk of a particle which is initially located at origin (0,0), is computed for 2000 time steps. At each time step, the object moves a distance, $d = 0.01$ units in a random direction which is mathematically represented using angle, $\theta \in [0, 2\pi]$, computed in the Matlab program with the help of inbuilt function, `rand()`. This function generates random number uniformly in the range $[0, 1]$, therefore, multiplying the obtained random number by 2π give us the random angle θ, in line 6 of the code 1 Successive movements in the two dimensional plane are computed in lines 7 and 8 of code 1, which are enclosed in a *for loop* defined in line 5, which repeats the process for each of the 2000 time steps.

```
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
```

Code 1: prob1aa.m - 2D random walk of a particle

Results – The plot of two dimensional random walk of a particle obtained using Matlab code given in code 1 is shown in figure 1a.

(b) **Methods** – Random walk for 2000 time steps, like the one performed in previous part is performed for 2000 different particles. Accordingly, the crux of code 1 is executed in a *for loop*, 2000 times, i.e. once for each particle, as shown in line 4 of code 2. Instead of visualizing the complete path of random walk, this time only the final coordinates of the particle after finishing the random walk are considered and stored separately in lines 12 and 13 of code 2.

```
 1 clear;
 2 d   = 0.01;
 3
 4 for i = 1:2000                            % for loop with 2000 steps
 5     x = 0;                                % initial x at origin
 6     y = 0;                                % initial y at origin
 7     for j = 1:2000
 8         theta   = 2*pi*rand();            % random theta between zero and 2pi
 9         x       = x+d*cos(theta);         % next value of x
10         y       = y+d*sin(theta);         % next value of y
11     end
12     xfinal(i)   = x;
13     yfinal(i)   = y;
14 end
15 scatter(xfinal,yfinal,10,"filled") ;      % plot the random walk
16 axis equal;                               % equal dimensions for axes
17 xlabel('x','FontSize',14);                % x-axis label
18 ylabel('y','FontSize',14);                % y-axis label
```

Code 2: prob1ab.m - Scatter of final positions of 2000 2D random walks

Results – A scatter plot of final positions after 2000 time steps, of 2000 particles undergoing two dimensional random walk is shown in figure 1b.

(c) **Methods** – Using elements from code 2, final positions of 5×10^5 particles are computed. The functions `tic` and `toc` are used to compute the time elapsed in execution of the code. With the help of these functions, the total number of particles is chosen, based purely on estimate of the maximum computations that can be performed in feasible time. So, The initial 18 lines of code 3 are essentially adopted from code 2, with an exception of line 17, where the modulus of displacement of a particle from origin is calculated. The following lines evaluate and plot the histogram of number distribution of particles with distance travelled. Using a bin width of $dr = 0.05$, binedges are computed and the histogram is plotted on line 24.
An alternate method in which counts in each bin are calculated to produce the histogram, was used to confirm results, and is not shown in code 3.

Results – The histogram showing particle number distribution, $N(r, r + \Delta r)$, of 500000 particles for $\Delta r = 0.05$ is given in figure 1c.

(d) **Methods** – Initial lines of code 4, till line 27, are adopted from code 3 and the value of `np` is set to 10^5, i.e. histogram is generated for 10^5 particles. Theoretically expected number distribution

```
clear;                                      % clear variables and functions
tic;                                        % start clock
d       = 0.01;
np      = 5.e5;                             % 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)
ylabel('Counts','FontSize',14)
toc                                         % stop clock
```

Code 3: prob1ac.m - Number distribution with distance of 5×10^5 particles

is then computed for each bin in lines 30-33, and the obtained curve is overlayed on the histogram. Analytically calculated number distribution of particles with distance is given by equation (3) of project notes as,

$$N(r, r + \Delta r) = N_p \exp\left(-\frac{r^2}{nd^2}\right)\left\{1 - \exp\left(-\frac{\Delta r(2r + \Delta r)}{nd^2}\right)\right\}, \tag{1}$$

where, $N(r, r + \Delta r)$ is the number of particles that have eventually travelled an absolute value of displacement in the range $[r, r + \Delta r]$, N_p is the total number of particles, n is the number of time steps and d is the jump size at each time step. This equation 1 is encoded in code 4 in lines 31 and 32.

```
28 hold on;
29 midpoints       = binedges+dr/2;         % midpoints of bins, for plotting
30 for i = 1:length(binedges)
31     n(i) = np * exp(-(binedges(i)^2)/(tstep*d*d))...
32            * (1-exp((-(2*binedges(i)*dr+dr*dr))/(tstep*d*d)));
33 end
34
35 plot(midpoints,n,'LineWidth',2);
```

Code 4: (Part of) prob1ad.m - Analytical estimates and observed number distributions with distance of 10^5 particles

Results – The histogram for 10^5 particles is shown in figure 1d. The analytical estimates is overlayed on the same plot.

Discussion – For a large number of particles, the observed number distribution seems to coincide well with the theoretical estimates, as seen in figure 1d.

(e) **Methods** – The number distribution of photons in this case is given by,

$$N(r)dr = 2\pi r \rho(r) dr, \tag{2}$$

where, $\rho(r)$ is the particle density distribution, given by,

$$\rho(r) = \frac{N_p}{\pi n d^2} \exp\left(-\frac{r^2}{nd^2}\right) \tag{3}$$

Substituting equation 3 in equation 2, we obtain,

$$N(r)dr = \frac{2rN_p}{nd^2} \exp\left(-\frac{r^2}{nd^2}\right) dr.$$

At peak number density, derivative of n with respect to displacement r equals zero. From this we obtain,

$$\frac{dN}{dr} = \frac{2N_p}{nd^2}\frac{d}{dr}\left[r \exp\left(-\frac{r^2}{nd^2}\right)\right] = 0$$

$$\Rightarrow \frac{2N_p}{nd^2} \exp\left(-\frac{r^2}{nd^2}\right)\left[1 - \frac{2r^2}{nd^2}\right] = 0$$

$$\Rightarrow n = \frac{2r^2}{d^2} \tag{4}$$

Using value of $r = R_\odot = 7 \times 10^8$ m and $d = 1$ mm $= 10^{-3}$ m in equation 4, we get, number of time steps,

$$n = 9.8 \times 10^{23}. \tag{5}$$

Total distance travelled by the photon before reaching the surface is, $s =$ number of time steps $\times$ distance travelled in each time step, i.e.,

$$s = 9.8 \times 10^{23} \times 10^{-3} = 9.8 \times 10^{20}\text{m}. \tag{6}$$

With photon travelling at speed of light, c, the time elapsed is given by,

$$t = \frac{s}{c} \approx 3.267 \times 10^{12}\text{sec} \approx 10^5\text{years} \tag{7}$$

Discussion – The rough estimate of time taken by photon considers an ideal case with several approximations. Despite that, the calculation gives us a reasonable order of magnitude estimate in equation 7, which is quite close to the value of a few million years, usually obtained for a realistic case.

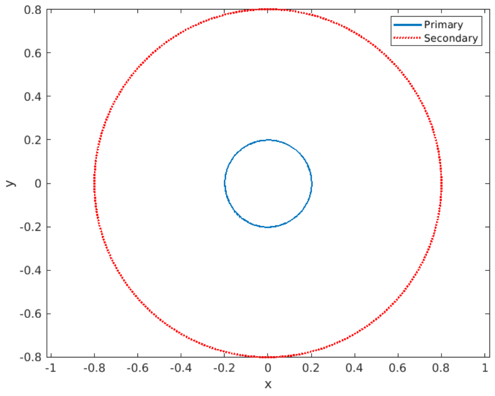
3 Conclusions

Part II

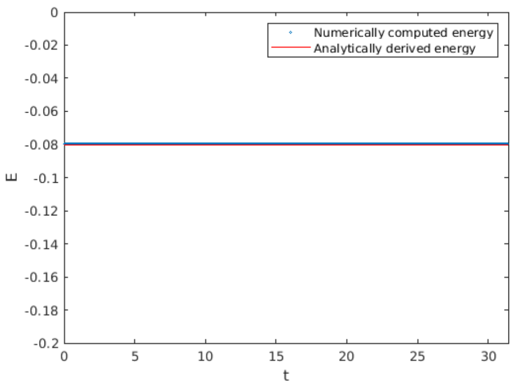
Circular Binary

1 Introduction

2 Methods, Results and Discussions



(a) 2c



(b) 2d

Figure 2: Problem 2

(a) **Methods** –
Results –
Discussions –

(b) **Methods** –
Results –
Discussions –

(c) **Methods** –
Results –
Discussions –

(d) **Methods** –
Results –
Discussions –

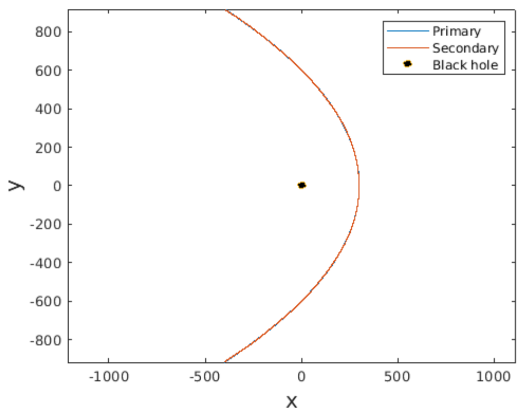
```
h      = 1.d-2;       % time-step size
Ns     = 1000 ;       % sampling
mp     = 0.8  ;       % primary mass
ms     = 0.2  ;       % secondary mass
x(1)   = -ms  ;       % primary x
x(2)   = 0    ;       % primary y
x(3)   = 0    ;       % primary vx
x(4)   = -ms  ;       % primary vy
x(5)   = mp   ;       % secondary x
x(6)   = 0    ;       % secondary y
x(7)   = 0    ;       % secondary vx
x(8)   = mp   ;       % secondary vy
tmax   = 10*pi;       % final time
```

Code 5: first code

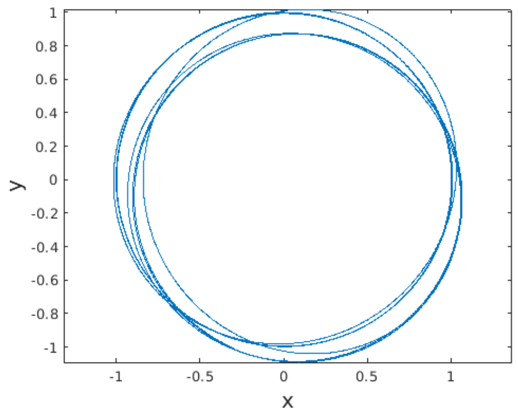
```
mp       = 0.8    ;
ms       = 0.2    ;
tmax     = 10*pi ;
load out                             % load the data file out
t        = out(:,1)  ;               % time
E        = out(:,6)  ;               % energy
exact    = -mp*ms/2  ;               % analytic estimate

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
```

Code 6: second code

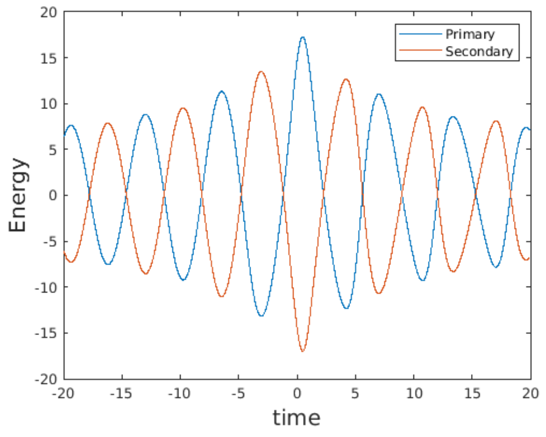
3 Conclusions



(a) 3a



(b) 3b



(c) 3c

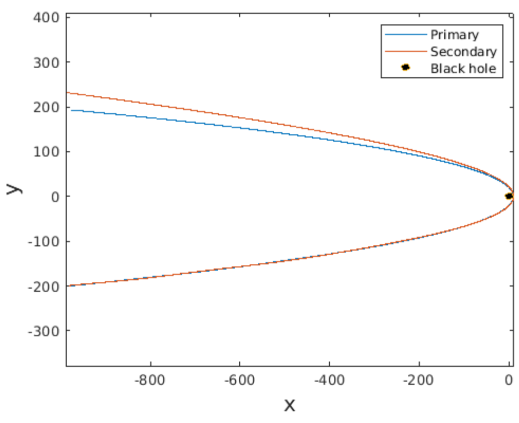
Figure 3: Problem 3 - fig 1

Part III
Hypervelocity Stars

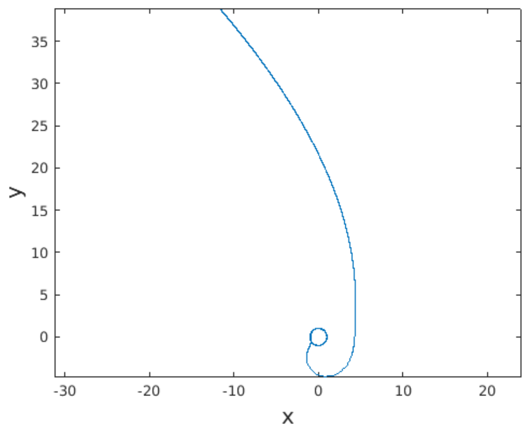
1 Introduction

2 Methods, Results and Discussions

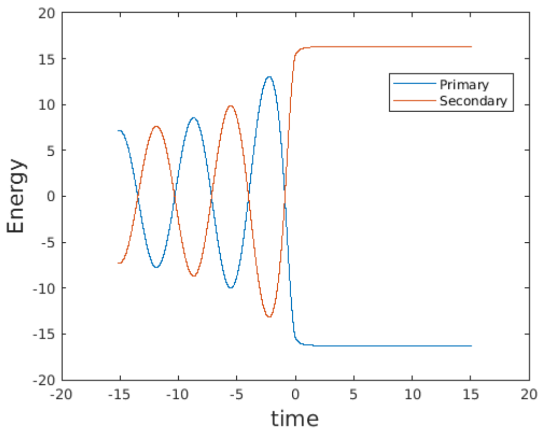
```
Rt       = (mb)^(1/3)                            ;   % tidal radius
R0       = 10 * Rt                               ;   % initial distance between BH and binary
Rp       = 3 * Rt                                ;   % periastron radius
f0       = -acos(-1+(D/5))                       ;   % initial 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
```



(a) 3da



(b) 3db



(c) 3dc

Figure 4: Problem 3 - fig 2

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

```
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
```

Code 7: code 1

```
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
```

Code 8: code 2

(a) **Methods** –
Results –
Discussions –

(b) **Methods** –
Results –
Discussions –

(c) **Methods** –
Results –
Discussions –

(d) **Methods** –
Results –
Discussions –

3 Conclusions