Solutions to Problem Set 4

Chris H. Rycroft[*]

November 10, 2006

1 First passage for biased diffusion

1.1 The first passage time to the origin

The PDF $\rho(x,t)$ of a continuous diffusion process with drift velocity v and diffusivity D satisfies a Fokker-Planck equation

$$\rho_t + v\rho_x = D\rho_{xx}.$$

For this problem, we are interested in solving in the domain $x > 0$. Walkers which reach $x = 0$ achieve first passage and are removed, so we make use of the boundary condition $\rho(0,t) = 0$. Since the walker starts at $x = x_0$, our initial condition is $\rho(x,0) = \delta(x - x_0)$. Without the boundary, we would just get a solution of the form

$$\frac{1}{\sqrt{4\pi Dt}} e^{-(x-x_0-vt)^2/4Dt}.$$

For this problem, we make use of the image method, introducing another term starting at $x = -x_0$ with magnitude A. Our PDF is therefore

$$\rho(x,t) = \frac{1}{\sqrt{4\pi Dt}} \left(e^{-(x-x_0-vt)^2/4Dt} + Ae^{-(x+x_0-vt)^2/4Dt} \right).$$

This trivially satisfies the Fokker-Planck equation, and we wish to choose A so that our boundary condition is satisfied. Setting $x = 0$ gives

$$\begin{aligned}\rho(0,t) &= \frac{1}{\sqrt{4\pi Dt}} \left(e^{(-x_0-vt)^2/4Dt} + Ae^{-(x_0-vt)^2/4Dt} \right) \\ &= \frac{e^{(x_0^2+v^2t^2)/4Dt}}{\sqrt{4\pi Dt}} \left(e^{x_0 v/2D} + Ae^{-x_0 v/2D} \right).\end{aligned}$$

If $A = -e^{x_0 v/D}$ then our boundary condition is satisfied, and hence

$$\rho(x,t) = \frac{1}{\sqrt{4\pi Dt}} \left(e^{-(x-x_0-vt)^2/4Dt} - e^{-x_0 v/D} e^{-(x+x_0-vt)^2/4Dt} \right).$$

[*]Solutions to problems 1 and 3 based on sections of *A Guide to-First Passage Processes* by Sidney Redner (2001).

By evaluating the probability current at $x = 0$, we find that the first passage probability density is given by

$$\begin{aligned}
f(t) &= -v\rho + D\rho_x|_{x=0} \\
&= \frac{D}{\sqrt{4\pi Dt}}\left(-\frac{x-x_0-vt}{2Dt}e^{-(x-x_0-vt)^2/4Dt} + \frac{x+x_0-vt}{2Dt}e^{x_0v/D}e^{-(x+x_0-vt)^2/4Dt}\right)\Bigg|_{x=0} \\
&= \frac{1}{4\pi t}\left(\frac{x_0}{2DT}e^{-(x_0+vt)^2/4Dt} + \frac{x_0}{2DT}e^{x_0v/D}e^{(x_0+vt)^2/4Dt}\right) \\
&= \frac{x_0}{\sqrt{4\pi Dt^3}}e^{-(x_0+vt)^2/4Dt}.
\end{aligned}$$

1.2 The survival probability

By integrating the $f(t)$, we find that the survival probability is

$$\begin{aligned}
S(t) &= 1 - \int_0^t f(q)\,dq \\
&= 1 - e^{-vx_0/2D}\int_0^t \frac{x_0}{\sqrt{4\pi Dq^3}}e^{-x_0^2/4Dq}e^{-v^2q/4D}\,dq.
\end{aligned}$$

Using the substitution $u^2 = x^2/4Dt$ and the Péclet number $Pe = vx_0/2D$, we find

$$\begin{aligned}
S(t) &= 1 - \frac{2}{\sqrt{\pi}}e^{-Pe}\int_{x/\sqrt{4Dt}}^{\infty} e^{-u^2 - Pe^2/4u^2}du \\
&= 1 - \frac{1}{2}\left[1 - \operatorname{erf}\left(\frac{x_0}{\sqrt{4Dt}} + \frac{Pe}{2}\frac{\sqrt{4Dt}}{x_0}\right)\right] \\
&\quad + \frac{e^{-2Pe}}{2}\left[1 - \operatorname{erf}\left(\frac{x_0}{\sqrt{4Dt}} - \frac{Pe}{2}\frac{\sqrt{4Dt}}{x_0}\right)\right]. \\
&\sim 1 - \frac{1}{2}e^{-Pe-|Pe|}\left[2 - \operatorname{erfc}\left(\frac{|Pe|}{2}\frac{\sqrt{4Dt}}{x_0}\right)\right].
\end{aligned}$$

As $t \to \infty$, we get two different behaviors for $S(t)$, depending on the sign of v:

$$\begin{aligned}
S(t) &\sim \begin{cases} 1 - e^{-2Pe} & \text{for } Pe > 0 \\ \frac{x_0}{Pe\sqrt{\pi Dt}}e^{-Pe^2Dt/x_0^2} & \text{for } Pe \le 0 \end{cases} \\
&\sim \begin{cases} 1 - e^{-vx_0/D} & \text{for } v > 0 \\ \sqrt{\frac{4D}{\pi v^2 t}}e^{-v^2t/4D} & \text{for } v \le 0. \end{cases}
\end{aligned}$$

From these expressions, we see that if $v > 0$ then there is a probability of $e^{-vx_0/D}$ of eventual first passage.

1.3 Minimum first passage time

Let the random variables for the first passage times be $T_1, T_2, \ldots, T_N$. Since the walkers are independent, we know that

$$\begin{aligned} P(\min\{T_1, T_2, \ldots, T_N\} > t) &= P(T_1 > t, T_2 > t, \ldots, T_N > t) \\ &= P(T_1 > t)P(T_2 > t)\ldots P(T_N > t) \\ &= S(t)^N \end{aligned}$$

and hence the PDF of the minimum first passage time is given by

$$p_n(t) = -\frac{d}{dt}S(t)^N = f(t)NS(t)^{N-1}$$

where $f(t)$ and $S(t)$ are explicitly given in the previous sections.

2 First passage for anomalous walks

2.1 Unbiased Cauchy walk

Appendix A provides a simple C++ code to simulate first passage times for the Cauchy walk. For a large number of trials, it was found that the standard C++ math `rand()` function was inadequate, and that slight biases in the probabilities around $n = 30$ could be seen. A second code, listed in appendix B, was therefore written, making use of the more advanced random number generation routines found in the GNU Scientific Library (GSL) [1].

The GSL code was run with 2×10^{10} trials for the case of $d = 0.0$. Walks that did not achieve first passage in 10^5 steps were prematurely terminated. Figure 1 shows the computed values of $f(n)$ for low values of n, while figure 2 shows a logarithmic plot highlighting the asymptotic behavior. For large n, the curve becomes almost linear, and by applying regression over the range $10^3 \leq n \leq 10^5$ we find that $f(n) \propto n^{-1.50061}$, which appears to match the theoretical result of $f(n) \propto n^{-3/2}$ for the Bernoulli walk.

Figure 3 shows a plot of the survival probability $S(n)$. Again, this curve appears to become linear for large n, and by applying regression we find $S(n) \propto n^{-0.500147}$. Since we have a negative exponent, we see that $S(n) \to 0$ as $n \to \infty$, and thus our expected probability of return is 1.

2.2 Biased Cauchy walk

The GSL code was also run for $d = 1.0$ and $d = -1.0$. The same number of trials were used for $d = -1.0$ as for the unbiased case, but 2×10^8 trials were used for $d = 1.0$, since many of these walks took a great number of steps to complete, thus creating a larger computational overhead. The computed $f(n)$ for low values of n is shown in figure 1, while a log plot showing the asymptotic behavior is shown in figure 2. We see that for large n, the curves in this figure become almost linear. Applying linear regression over the range $10^3 \leq n \leq 10^5$ shows that $f(n) \propto n^{-1.75027}$ for $d = -1.0$, and $f(n) \propto n^{-1.25015}$ for $d = 1.0$.

The survival probability function $S(n)$ for these cases is shown in figure 3. Again, these curves appear linear as n increases, and by applying linear regression we find that $S(n) \propto n^{-0.750051}$ for $d = -1.0$ and $S(n) \propto n^{-0.250044}$ for $d = 1.0$. Thus we expect that the probability of return is always 1, even for the case of positive drift, although some of these walks may take a very long time to return. Nevertheless, this fits with our intuition, since Cauchy walkers are capable of taking extremely large steps, on a scale larger than the drift.

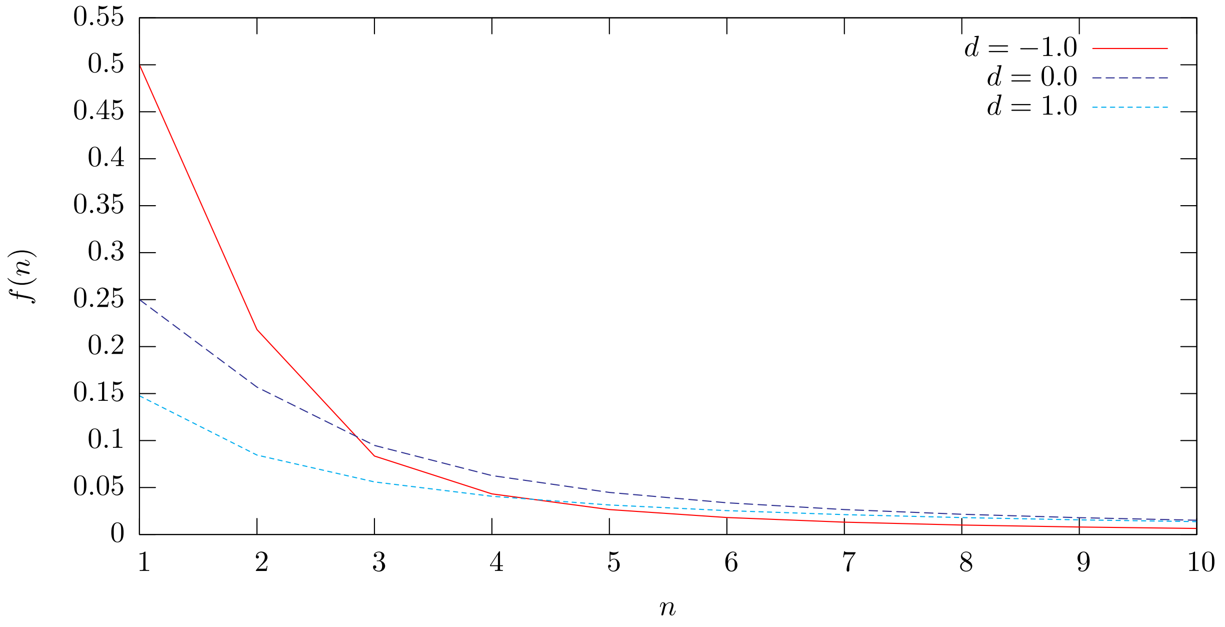

Figure 1: Plots of the first passage probability functions $f(n)$ for the unbiased and biased Cauchy walks.

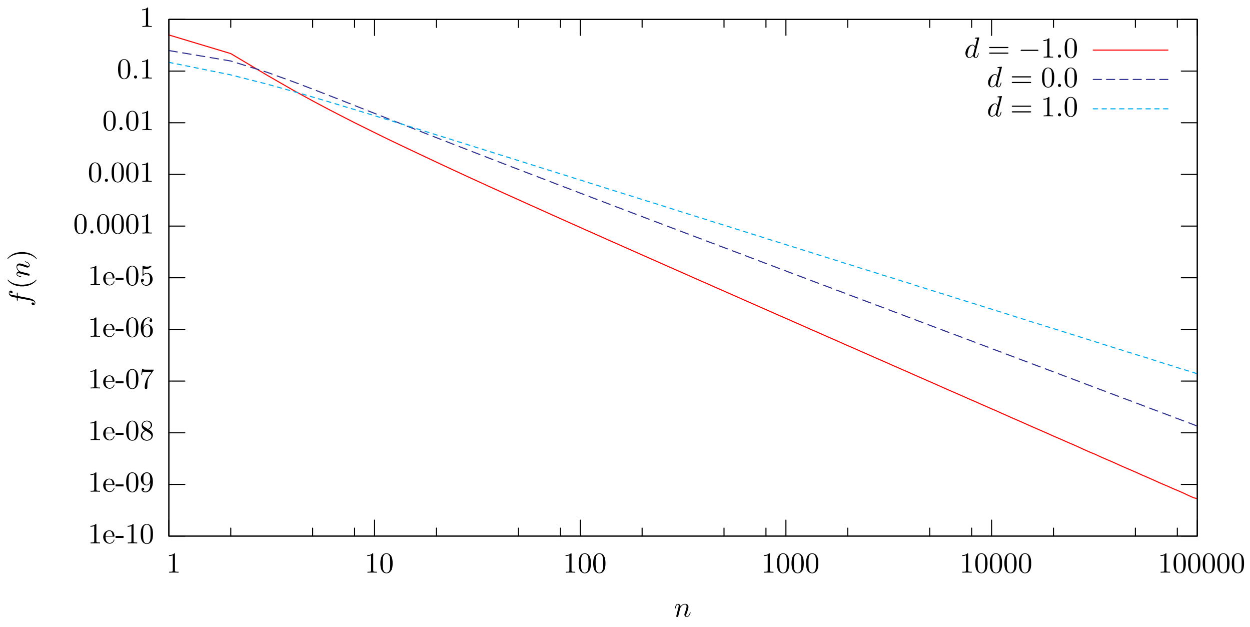

Figure 2: Log plots of the first passage probability functions $f(n)$ for the unbiased and biased Cauchy walks.

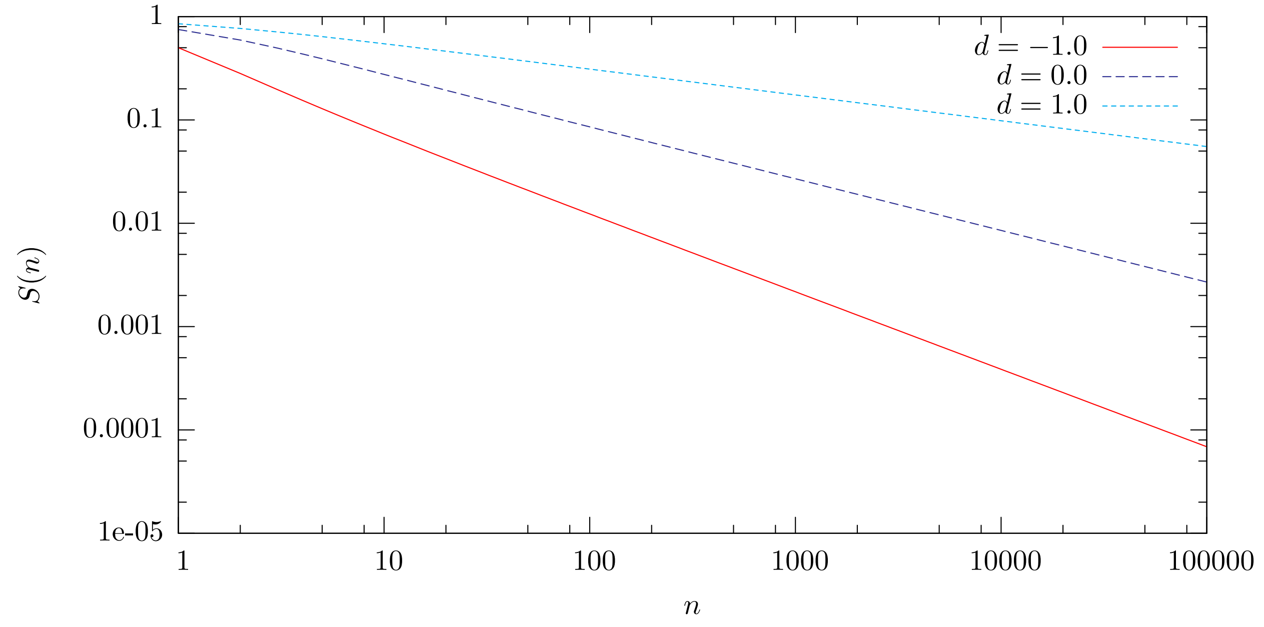

Figure 3: Log plots of the survival probability functions $S(n)$ for the unbiased and biased Cauchy walks.

3 First passage to a sphere

To calculate the probability of first passage to the sphere, we make use of the electrostatic analogy. We consider the corresponding problem of a point charge of magnitude $q = 1/4\pi R^2 D$ located at a distance r_0 from the sphere, with the sphere's surface is kept at zero potential. The probability of absorption at a point on the sphere's surface will be given by the magnitude of the electric field there. Let $\vec{r}_0$ be the the position of the walker, and by symmetry, consider pointing this in the positive z-direction. In the absence of the sphere, the electric potential is given by

$$\Phi(\vec{r}) = \frac{q}{|\vec{r} - \vec{r}_0|},$$

which can be rewritten in terms of spherical coordinates (r, θ, ϕ) as

$$\Phi(\vec{r}) = \frac{q}{\sqrt{r^2 \sin^2\theta + (r_0 - r\cos\theta)^2}}.$$

To solve for the electric potential in the presence of the sphere, we make use of the image method, introducing a charge of magnitude v at a location $(x, y, z) = (0, 0, s)$, to give a solution of the form

$$\Phi(\vec{r}) = \frac{q}{\sqrt{r^2 \sin^2\theta - (r_0 - r\cos\theta)^2}} + \frac{v}{\sqrt{r^2 \sin^2\theta - (r\cos\theta - s)^2}}.$$

In order to set the electric potential to zero on the sphere at $r = R$, we must have

$$\begin{aligned} \frac{q}{\sqrt{R^2 \sin^2\theta - (r_0 - R\cos\theta)^2}} &= -\frac{v}{\sqrt{R^2 \sin^2\theta - (R\cos\theta - s)^2}} \\ q^2(R^2 \sin^2\theta - (R\cos\theta - s)^2) &= v^2(R^2 \sin^2\theta - (r_0 - R\cos\theta)^2) \\ q^2(R^2 - s^2 - 2Rs\cos\theta) &= v^2(R^2 - r_0^2 - 2Rr_0\cos\theta). \end{aligned}$$

To be valid for all θ, we must have $q^2 s = v^2 r_0$, and

$$\begin{aligned} q^2(R^2 - s^2) &= v^2(R^2 - r_0^2) \\ \frac{R}{s}(R^2 - s^2) &= (R^2 - r_0^2) \\ (s - r_0)\left(s\frac{R^2}{r_0}\right) &= 0. \end{aligned}$$

Thus $s = R^2/r_0$, which is inside of the sphere, since $r_0 > R$. The magnitude of the charge is given by

$$\begin{aligned} v^2 r_0 &= q^2\left(\frac{R^2}{r_0}\right) \\ v &= \frac{-qR}{r_0}. \end{aligned}$$

Thus the electric potential is

$$\Phi(\vec{r}) = \frac{q}{\sqrt{r^2\sin^2\theta - (r_0 - r\cos\theta)^2}} - \frac{qR}{r_0\sqrt{r^2\sin^2\theta - (r\cos\theta - \frac{R^2}{r_0})^2}}.$$

Taking the normal derivative and multiplying by $-D$, we find that the PDF of absorption at a position (R, θ) on the sphere is

$$P(R,\theta) = \frac{1}{4\pi R r}\frac{1 - \frac{R^2}{r_0^2}}{\left(1 - \frac{2R}{r_0}\cos\theta + \frac{R^2}{r_0^2}\right)^{3/2}}.$$

The ratio between the probability of hitting at the nearest point on the sphere and the farthest is

$$\frac{P(R,0)}{P(R,\pi)} = \left(\frac{1 + 2R/r_0 + R^2/r_0^2}{1 - 2R/r_0 + R^2/r_0^2}\right)^{d/2} = \left(\frac{1 + R/r_0}{1 - R/r_0}\right)^d.$$

4 The Ballot Problem

We define P_i and Q_i be the partial scores for the two candidates after i votes have been counted, and let $R_i = P_i - Q_i$ be the difference between the two. At each step, R_i can either increase or decrease by one, and it is therefore a Bernoulli pathway on the integers, as discussed in lecture 14.

We know that $P_{p+q} = p$ and $Q_{p+q} = q$, so $R_{p+q} = p - q$. In terms of the quantities introduced in lecture, we know that the number of possible ways to count the votes is therefore $N(p-q, p+q)$, and each of these paths is equally likely.

If the first candidate always has more votes than the second, we know that the R_i trace out a non-returning path to $(p-q, p+q)$, and as shown in the lecture there are $(p-q)N(p-q, p+q)/(p+q)$ of these. To obtain the probability, we just need to divide by the total number of paths, $N(p-q, p+q)$, to obtain $(p-q)/(p+q)$.

p	q	r	P	p	q	r	P	p	q	r	P	p	q	r	P	p	q	r	P
1	0	0	1.0000	3	1	1	0.3000	4	1	1	0.4000	5	0	1	0.6667	5	1	4	0.0889
2	0	0	1.0000	3	1	2	0.1333	4	1	2	0.2381	5	0	2	0.4285	5	2	2	0.2302
2	0	1	0.3333	3	2	2	0.0762	4	1	3	0.1071	5	0	3	0.2500	5	2	3	0.1461
2	1	1	0.1667	4	0	0	1.0000	4	2	2	0.1571	5	0	4	0.1111	5	2	4	0.0693
3	0	0	1.0000	4	0	1	0.6000	4	2	3	0.0762	5	1	1	0.4761	5	3	3	0.1004
3	0	1	0.5000	4	0	2	0.3333	4	3	3	0.0457	5	1	2	0.3214	5	3	4	0.0508
3	0	2	0.2000	4	0	3	0.1428	5	0	0	1.0000	5	1	3	0.1944	5	4	4	0.0313

Table 1: Computed probabilities for the three-person ballot problem for $p \leq 5$.

	$r=0$	$r=1$	$r=2$	$r=3$	$r=4$	$r=5$
$q=0$	1.0000	0.7142	0.5000	0.3333	0.2000	0.0909
$q=1$	0.7142	0.5357	0.3890	0.2667	0.1636	0.0758
$q=2$	0.5000	0.3890	0.2937	0.2082	0.1313	0.0622
$q=3$	0.3333	0.2667	0.2082	0.1543	0.1013	0.0495
$q=4$	0.2000	0.1636	0.1313	0.1013	0.0712	0.0369
$q=5$	0.0909	0.0758	0.0622	0.0495	0.0369	0.0231

Table 2: Computed probabilities for the three-person ballot problem for $p = 6$.

4.1 Simulating three candidates

Appendix C simulates the three-person voting process. All possible combinations of p, q, and r votes less than or equal to 12 were tested, each with $N = 10^9$ trials. For an underlying process with probability l of success, and N trials, we know that the observed number of successes will be a binomial distribution with mean Nl and variance $Nl(1-l) < N/4$. Thus the standard deviation of our probability estimate is less than $(\sqrt{N/4})/N = 1/\sqrt{4N} \approx 1.58 \times 10^{-5}$. Thus we expect our probabilities to be correct to four decimal places. Tables 2, 3, 4, 5, and 6 show the computed probabilities for $p = 6$, $p = 7$, $p = 8$, $p = 9$, and $p = 10$ respectively, and table 1 shows the probabilities for $p \leq 5$.

4.2 Analytical results for three walkers

Consider the case when $r = 1$. The total number of possible ways the votes can be counted is $(p+q+1)N(p-q, p+q)$, since any counting process can be viewed as a counting process between candidates A and B only, with C's vote inserted at one of $p+q+1$ located between the other votes.

We know that in order for A to always be ahead, he must receive the first two votes. Consider any voting process between A and B where A is always ahead. If C's vote is inserted before any votes are counted, then C will take the lead and A will not always be ahead. If C's vote is inserted after one vote has been counted, then C will tie with A, and again the condition will be violated. However, if C's vote is inserted at any later point, then it will not violate the condition, since A must have at least two votes by this stage. The total number of possible voting processes satisfying the condition in therefore $(p+q-1)F(p-q, p+q)$, and hence the probability of the condition being satisfied is

$$\frac{(p+q-1)F(p-q,p+q)}{(p+q+1)N(p-q,p+q)} = \frac{(p+q-1)(p-q)}{(p+q+1)(p+q)}.$$

For the case when $r > 1$ the reader should refer to references [2] and [3].

	$r=0$	$r=1$	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$
$q=0$	1.0000	0.7500	0.5556	0.4000	0.2727	0.1667	0.0769
$q=1$	0.7500	0.5835	0.4444	0.3272	0.2273	0.1411	0.0659
$q=2$	0.5556	0.4444	0.3484	0.2631	0.1868	0.1179	0.0559
$q=3$	0.4000	0.3272	0.2631	0.2047	0.1490	0.0964	0.0466
$q=4$	0.2727	0.2273	0.1868	0.1490	0.1128	0.0755	0.0376
$q=5$	0.1667	0.1411	0.1179	0.0964	0.0755	0.0539	0.0284
$q=6$	0.0769	0.0659	0.0559	0.0466	0.0376	0.0284	0.0179

Table 3: Computed probabilities for the three-person ballot problem for $p = 7$.

	$r=0$	$r=1$	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$	$r=7$
$q=0$	1.0000	0.7778	0.6000	0.4544	0.3334	0.2309	0.1429	0.0667
$q=1$	0.7778	0.6222	0.4906	0.3788	0.2824	0.1979	0.1238	0.0583
$q=2$	0.6000	0.4906	0.3960	0.3120	0.2360	0.1680	0.1064	0.0507
$q=3$	0.4544	0.3788	0.3120	0.2505	0.1936	0.1401	0.0902	0.0435
$q=4$	0.3334	0.2824	0.2360	0.1936	0.1531	0.1135	0.0746	0.0367
$q=5$	0.2309	0.1979	0.1680	0.1401	0.1135	0.0870	0.0591	0.0298
$q=6$	0.1429	0.1238	0.1064	0.0902	0.0746	0.0591	0.0426	0.0227
$q=7$	0.0667	0.0583	0.0507	0.0435	0.0367	0.0298	0.0227	0.0145

Table 4: Computed probabilities for the three-person ballot problem for $p = 8$.

A C++ codes for simulating the Cauchy first passage problem

This listing provides a simple C++ code for generating the distribution of first passage times for a Cauchy walk. The code accepts two command line argumerts: the drift parameter d, and a random seed. Once all walks have been simulated, the first passage probabilities for each number of walk step are printed to the standard output. Walks which did not achieve first passage in `n` steps are listed in the final line of the output.

```
#include <string>
#include <iostream>
#include <cstdio>
#include <cmath>
using namespace std;

const double p=3.1415926535897932384626433832795;
const long n=10000;     //Cutoff number of steps
const long w=10000000;  //Number of walkers
double d=0;             //Drift

inline int cauchy() {
        static double x;x=1;
        for(int i=0;i<n;i++) {
                x+=d+tan(((double(rand())+0.5)/RAND_MAX-0.5)*p);
                if (x<0) return i;
        }
        return n;
}

int main(int argc,char* argv[]) {
        srand(atoi(argv[1]));
```

	$r=0$	$r=1$	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$	$r=7$	$r=8$
$q=0$	1.0000	0.8000	0.6362	0.5000	0.3849	0.2860	0.2000	0.1250	0.0588
$q=1$	0.8000	0.6544	0.5303	0.4236	0.3301	0.2477	0.1750	0.1103	0.0523
$q=2$	0.6362	0.5303	0.4379	0.3550	0.2801	0.2129	0.1521	0.0968	0.0462
$q=3$	0.5000	0.4236	0.3550	0.2921	0.2343	0.1807	0.1307	0.0841	0.0406
$q=4$	0.3849	0.3301	0.2801	0.2343	0.1914	0.1501	0.1103	0.0719	0.0351
$q=5$	0.2860	0.2477	0.2129	0.1807	0.1501	0.1203	0.0902	0.0600	0.0298
$q=6$	0.2000	0.1750	0.1521	0.1307	0.1103	0.0902	0.0698	0.0479	0.0244
$q=7$	0.1250	0.1103	0.0968	0.0841	0.0719	0.0600	0.0479	0.0348	0.0187
$q=8$	0.0588	0.0523	0.0462	0.0406	0.0351	0.0298	0.0244	0.0187	0.0120

Table 5: Computed probabilities for the three-person ballot problem for $p = 9$.

	$r=0$	$r=1$	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$	$r=7$	$r=8$	$r=9$
$q=0$	1.0000	0.8182	0.6667	0.5387	0.4294	0.3334	0.2500	0.1764	0.1111	0.0526
$q=1$	0.8182	0.6818	0.5643	0.4625	0.3715	0.2916	0.2206	0.1569	0.0994	0.0474
$q=2$	0.6667	0.5643	0.4746	0.3926	0.3196	0.2535	0.1935	0.1388	0.0886	0.0425
$q=3$	0.5387	0.4625	0.3926	0.3297	0.2718	0.2182	0.1683	0.1217	0.0783	0.0378
$q=4$	0.4294	0.3715	0.3196	0.2718	0.2271	0.1848	0.1442	0.1055	0.0685	0.0334
$q=5$	0.3334	0.2916	0.2535	0.2182	0.1848	0.1525	0.1208	0.0897	0.0590	0.0291
$q=6$	0.2500	0.2206	0.1935	0.1683	0.1442	0.1208	0.0976	0.0739	0.0495	0.0248
$q=7$	0.1764	0.1569	0.1388	0.1217	0.1055	0.0897	0.0739	0.0576	0.0398	0.0205
$q=8$	0.1111	0.0994	0.0886	0.0783	0.0685	0.0590	0.0495	0.0398	0.0291	0.0158
$q=9$	0.0526	0.0474	0.0425	0.0378	0.0334	0.0291	0.0248	0.0205	0.0158	0.0101

Table 6: Computed probabilities for the three-person ballot problem for $p = 10$.

```
        d=atof(argv[2]);
        int i,j,c,a[n+1];
        for(i=0;i<=n;i++) a[i]=0;
        for(j=0;j<w;j++) a[cauchy()]++;
        for(i=0;i<=n;i++) {
                cout << i << "␣" << a[i]
                     << "␣" << double(a[i])/w << endl;
        }
}
```

B C++/GSL code for simulating the Cauchy first passage problem

This code performs the same task as that in appendix A but makes use of random number generating routines in the GNU Scientific Library[1] (GSL), which provide much better sources of randomness. The code accepts a single command line argument for the drift. The environment variable `GSL_RNG_TYPE` chooses the random number routine to use (in this case, set to `mrg`), and the environment variable `GSL_RNG_SEED` chooses a random seed.

```
#include <string>
#include <iostream>
#include <cstdio>
#include <cmath>
#include <gsl/gsl_rng.h>
using namespace std;
```

```
const double p=3.14159265358979323846264338327950;
const long n=100000;     //Cutoff number of steps
const long w=1000000;    //Number of walkers
double d;                //Drift
const gsl_rng_type * T;
gsl_rng * r;

inline int cauchy() {
        static double x;x=1;
        for(int i=0;i<n;i++) {
                x+=d+tan((gsl_rng_uniform(r)-0.5)*p);
                if (x<0) return i;
        }
        return n;
}

int main(int argc,char* argv[]) {
        d=atof(argv[1]);
        gsl_rng_env_setup();
        T=gsl_rng_default;
        r=gsl_rng_alloc(T);
        int i,j,c,a[n+1];
        for(i=0;i<=n;i++) a[i]=0;
        for(j=0;j<w;j++) a[cauchy()]++;
        for(i=0;i<=n;i++) {
                cout << i << " " << a[i]
                     << " " << double(a[i])/w << endl;
        }
        gsl_rng_free(r);
}
```

C C++ code for simulating three-candidate vote counting

The code below simulates a three-person voting process. It accepts one integer command line argument, which seeds the random number generator. Up to symmetry, all possible values of the three vote totals below `max` are tested, and the computed probabilities are printed to the standard output.

```
#include <string>
#include <iostream>
#include <cstdio>
#include <cmath>
using namespace std;

const int trials=200000000;     //Total number of trials
const int m=12;                 //Max value of p,q,r
```

```
//Returns a random integer between 0 and x-1
inline int randi(int x) {
        int y=(RAND_MAX/x)*x,z;
        do {z=rand();} while (z>=y);
        return z%x;
}

//Simulates a voting process with p, q, and r votes
//and returns true if the first candidate is always
//ahead
bool walk(int p,int q,int r) {
        int a,b,c,h;
        a=b=c=0;
        while(p>0||q>0||r>0) {
                h=randi(p+q+r);
                if(h<p) {a++;p--;} else
                {
                        if (h<p+q) {b++;q--;if (b>=a) return false;}
                        else {c++;r--;if (c>=a) return false;}
                }
        }
        return true;
}

int main(int argc,char* argv[]) {
        srand(atoi(argv[1]));   //Seed random number generator
        int i,j,k,l,s;
        for(i=0;i<=m;i++) {     //Loop over all possible triples
                for(j=0;j<i;j++) {
                        for(k=j;k<i;k++) {
                                s=0;
                                for(l=0;l<trials;l++) {
                                        if (walk(i,j,k)) s++;
                                }
                                cout << i << "␣" << j << "␣"
                                     << k << "␣" << s << "␣"
                                     << double(s)/trials << endl;
                        }
                }
        }
}
```

References

[1] http://www.gnu.org/software/gsl.

[2] Arumugam Muhundan, *Ballot problem with two and three candidates*, Master's thesis, Florida Atlantic University, 1990.

[3] Heinrich Niederhausen, *The ballot problem with three candidates*, European Journal of Combinatorics **4** (1983), 161–167.