Virtual Pascal's Triangles: The Ballot Problem and the Method of Images

Chris McCarthy cmccarthy@bmcc.cuny.edu
Johannes Familton jfamilton@bmcc.cuny.edu

Department of Mathematics
Borough of Manhattan Community College, CUNY
New York, New York 10007
USA

Abstract

Bertrand's ballot problem asks what is the probability that the winner in a two candidate election will always be ahead in the vote count. In this paper we present a method for solving this problem using virtual Pascal's triangles, analogs of the virtual objects used in the method of images in the theory of differential equations. We supply a short script, written in the language R, which implements these methods and displays the results.

1 Introduction

Joseph Bertrand [**Bertrand1887**] in 1887 proposed and solved what is called the Ballot Problem.

Ballot Problem: An election is held to choose between Candidate A and Candidate B. Candidate A receives "a" votes. Candidate B receives "b" votes, with $a > b$. As the votes are counted, what is the probability that Candidate A will always have more votes than Candidate B?

Solution:

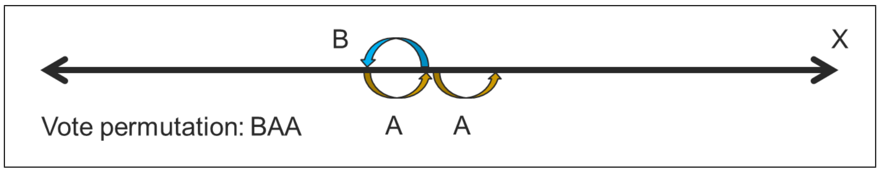
$$P(\text{A always ahead of B}) = \frac{a-b}{a+b}$$

There are many ways to prove this result. For four of them, see Mark Renault's excellent 2007 paper, "Four Proofs of the Ballot Problem"[**Renault2007**]. The standard method of proof is by lattice path reflection.

2 Lattice Paths

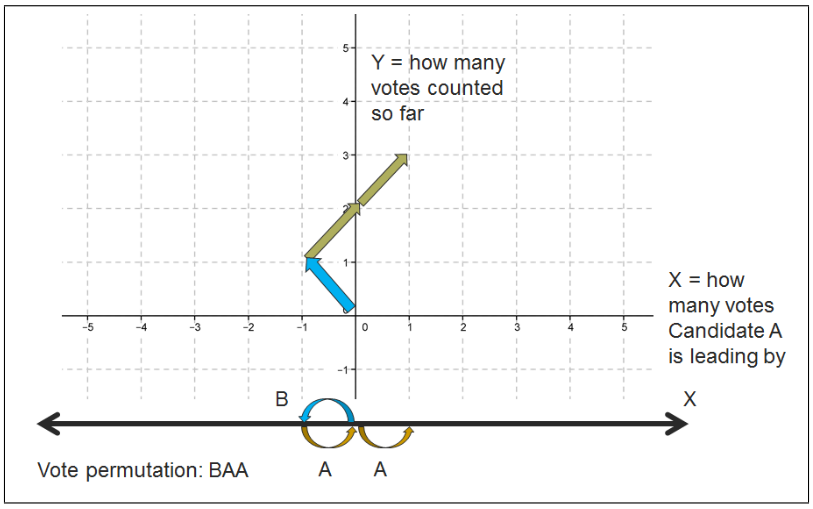
2.1 Lattice Paths and the Ballot Problem

- Each permutation of ballots is represented by a 1-dimensional lattice path starting at $x = 0$.
  $x =$ how many votes Candidate A is leading by
  $=$ (number of votes for A) $-$ (number of votes for B)



- A vote for candidate A is one step to right $(+1)$.
  A vote for candidate B is one step to left (-1).
  The lattice path will terminate at $a - b$.

The lattice paths are more easily visualized in two dimensions, with the y direction being the number of steps (i.e. votes counted).


Y = how many votes counted so far
X = how many votes Candidate A is leading by
B
X
Vote permutation: BAA
A
A

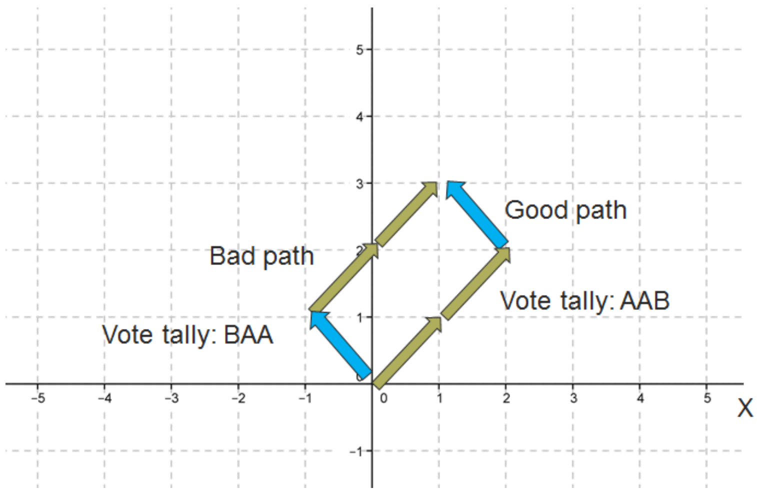
The two dimensional lattice path starts at $(0, 0)$ and terminates at $(a - b, n)$.

- Clearly, there are $\dfrac{n!}{(n-a)!a!} = C(n, a) = C(n, b)$ possible permutations of the ballots since $n = a + b$.
- Equivalently, there are $C(n, a) = C(n, b)$ lattice paths of length n from 0 to $a - b$.
- If $n = a + b$ and $x = a - b$ then (after a little algebra) $a = \dfrac{x+n}{2}$ and $b = \dfrac{n-x}{2}$.

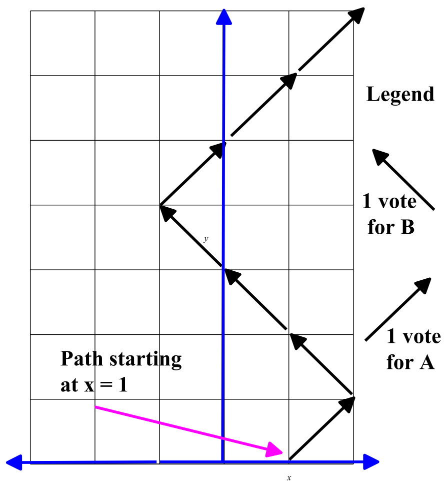
- So there are
$$C\left(n, \frac{n+x}{2}\right) = C\left(n, \frac{n-x}{2}\right) \tag{1}$$
length n lattice paths from 0 to x provided $\frac{n+x}{2}$ is an integer between 0 and n. If not, there are 0 such paths.

- By translation invariance, there are
$$C\left(n, \frac{n+(x_2-x_1)}{2}\right) = C\left(n, \frac{n-(x_2-x_1)}{2}\right) \tag{2}$$
length n lattice paths from x_1 to x_2 provided $\frac{n+(x_2-x_1)}{2}$ is an integer between 0 and n. If not, there are 0 such paths.

To solve the ballot problem we will count how many of the lattice paths from 0 to $a-b$ are "good", meaning that on that path candidate A is always in the lead, i.e. $x > 0$. A path is "bad", if at some point in the count, the candidates are tied, i.e. if $x = 0$.



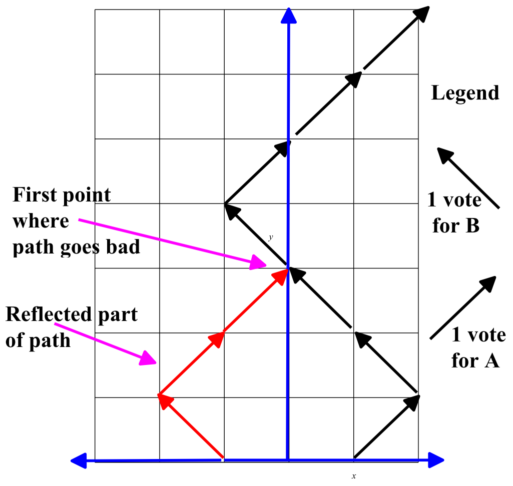
We start by assuming that the first vote counted was for Candidate A. This happens with probability $\dfrac{a}{a+b}$. Then we find the probability that A stays ahead, assuming the first vote counted was for A. This will be the ratio
$$\frac{|\text{permutations of the } n-1 \text{ remaining ballots with A staying ahead}|}{|\text{permutations of the } n-1 \text{ remaining ballots}|} \tag{3}$$
To find the denominator of (3) we count the number of paths of length $n-1$ that start at $x = 1$ and terminate at $x = a - b$. By Formula (2) there are a total of:
$$C\left(n-1, \frac{(n-1)+(x_2-x_1)}{2}\right) = C\left(n-1, \frac{(n-1)+(a-b-1)}{2}\right) = C\left(n-1, a-1\right)$$
paths of length $n-1$ from $x = 1$ to $x = a - b$.



2.2 The Reflection Method

To find the numerator of (3) we will count the number of paths of length $n-1$ that start at $x=1$ and terminate at $x=a-b$ but never intersect $x=0$. We will call such paths "good" paths. If the path intersects $x=0$, we will call it a "bad" path. To count the number of good paths, we will subtract the number of bad paths from the total number of paths.

The reflection method counts the bad paths from $x=1$ to $x=a-b$ of length $n-1$ by noting that these paths are in a one–one correspondence with the paths of length $n-1$ that go from $x=-1$ to $x=a-b$. As shown below, the bijection comes about by reflecting the part of the bad path up until it first hits the y axis.



So there are a total of:

$$C\left(n-1,\frac{(n-1)+(x_2-x_1)}{2}\right)=C\left(n-1,\frac{(n-1)+(a-b-(-1))}{2}\right)=C\left(n-1,a\right)$$

bad paths of length $n-1$ from $x=1$ to $x=a-b$.

Combining all this we get:

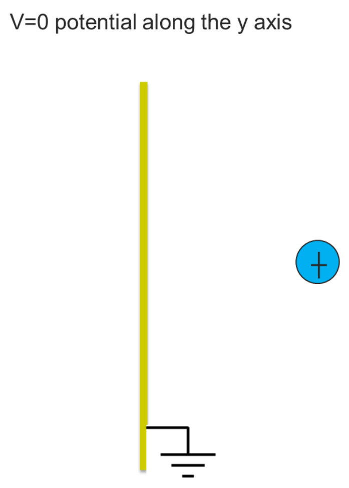
$$\begin{aligned} P(\text{A is always ahead}) &= P(\text{first vote is for A})\ P(\text{A is always ahead} \mid \text{first vote is for A}) \\ &= \frac{a}{a+b}\ \frac{C(n-1,a-1)-C(n-1,a)}{C(n-1,a-1)} \\ &= \frac{a-b}{a+b}. \end{aligned}$$

3 Method of Images

The method of images is a technique to solve boundary value problems by the introduction of fictional, virtual particles. The most well known problem solved by means of the method of images involves Poisson's equation in electrostatics.

3.1 Method of Images: Poisson's Equation

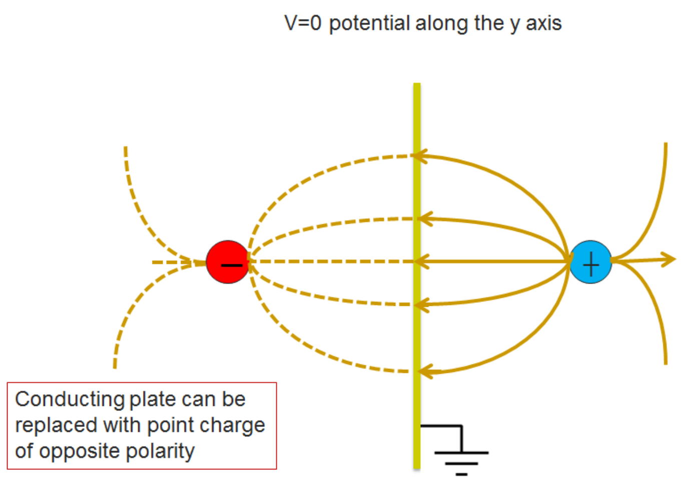
In electrostatics the method of images is used, in certain situations, to quickly solve Poisson's Equation[1] $\nabla^2\phi = \frac{\rho}{\epsilon}$ for the scalar electric potential ϕ. For us the relevant situation is a point charge located at a fixed distance to the right of an infinite grounded conducting plate. See figure below. Grounded means that the plate has a constant voltage[2], i.e. $x \in$ plate, then $\phi(x) = 0$.



Solving Poisson's Equation directly (i.e., finding ϕ) for the above setup would be difficult without the use of the following trick known as the method of images. The trick is to replace the conducting plate by a virtual charge of opposite sign located at the mirror image point of the original charge with respect to the conducting plate. See figure below.

[1]In Poisson's Equation $\rho = \rho(x)$ = charge density as a function of x; ϵ = permittivity; the electric field $E = -\nabla\phi$; for a point charge $\phi = \frac{1}{4\pi\epsilon}\frac{Q}{r}$ where Q = charge and r = the distance from the point charge.

[2]Voltage is another name for the scalar electric potential.


V=0 potential along the y axis
Conducting plate can be replaced with point charge of opposite polarity
−
+

The mathematical justification for the method of images comes from the Poisson Uniqueness Theorem. Let U be a region in R^3. If ϕ satisfies Poisson's Equation $\nabla^2\phi = \frac{\rho}{\epsilon}$ on U then ϕ is uniquely determined by its values on the boundary of U. See Griffiths [**griffiths1998electrodynamics**] for more details.

3.2 Method of Images: the Diffusion Equation

The method of images has also been applied to solving the diffusion equation [**crank1979mathematics**], [**strauss1992partial**]

$$\begin{array}{rll} u_{xx} = ku_t, & & x \geq 0, \;\; t \geq 0 \\ \text{BC} \;\; u(0,t) = 0, & & t \geq 0 \\ \text{IC} \;\; u(x,0) = \delta(x-y), & & \text{for some } y > 0 \end{array}$$

where here δ is the Dirac delta function. Physically, this is the diffusion problem for a semi-infinite tube with the end at $x = 0$ open and the initial concentration of the diffusing substance all concentrated at the single point y. It is worth noting that the diffusion equation can be derived as the limit of an unbiased random walk[3] [**varadhan1980lectures**].

4 Method of Virtual Pascal's Triangles

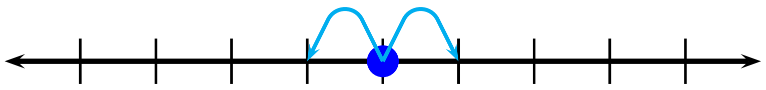
Background: We call our method of solving the ballot problem the "the method of virtual Pascal's triangles" in homage to the virtual charges from the method of images. We have not seen our method, as presented here, presented elsewhere. However, the underlying ideas are present in the literature.

In a brief footnote on page 72 of Feller's *Introduction to Probability, Vol. I.* (1968) [**Feller1968**], Feller mentions that the reflection method is equivalent to the method of images. He notes that the reflection principle is called the method of images when used in the theory of differential equations

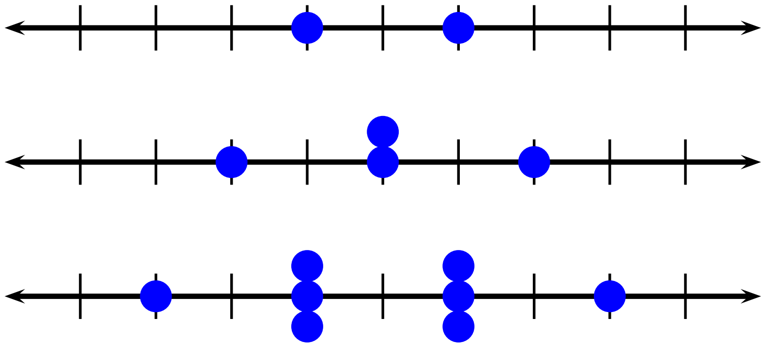
[3]A random walk is a lattice path where the direction of each step is chosen stochastically; unbiased means a step to the right is equally likely as a step to the left.

and states that this method is usually attributed to Lord Kelvin or James Clark Maxwell. Feller does not elaborate further. Zeilberger (1980) [**Zeilberger1980MR578062**], in his elegant treatment of partial difference equations, introduces a recurrence relationship, similar to the one we develop below (our operator F), which he uses, along with generating functions, to solve a higher dimension generalization of the ballot problem.

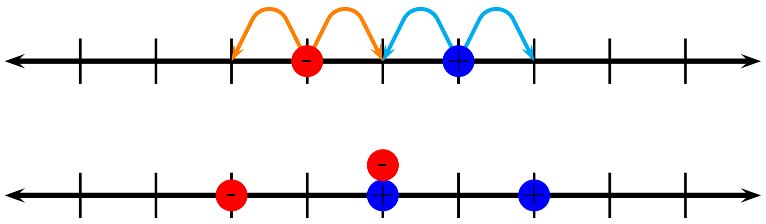
Path counting and virtual particles: We can think of the one dimensional lattice paths as being traced out by particles that must move one unit to the right or one unit to the left at each step.



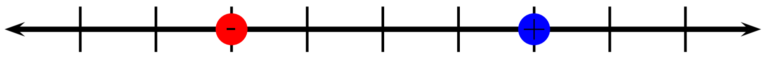
To count the number of paths that can reach a point we imagine the following. At each step we let the particle duplicate itself, with one copy going one unit to the right and one copy going one unit to the left. The number of paths reaching a point equals the number of particles found at the point.



We can imagine that the particles have signs or charges (+) or (-),



and that these can cancel each other when combined, yielding:



The net number of particles at each $x \in \mathbb{Z}$ defines a function $\mathbb{Z} \to \mathbb{Z}$ and so we let

$$\begin{aligned} S &= \{f : \mathbb{Z} \to \mathbb{Z}\} \\ &= \text{ set of all possible particle counts on the lattice } \mathbb{Z}. \end{aligned}$$

The particles propagate and trace out their lattice paths according to Pascal's rule. With that in mind we define $F : S \to S$ by

$$F[f](x) = f(x-1) + f(x+1).$$

F is just Pascal's rule[4] written as a linear operator on S. If $f(x)$ counts how many paths of length $n-1$ will reach x, then $F[f](x)$ will count how many paths of length n will reach x.

In the literature, F and its iterates are typically folded into a single recursive relation

$$\phi(x,n) = \phi(x-1,n-1) + \phi(x+1,n-1).$$

The relationship between F and ϕ is as follows. If we let $\phi(x,0) = f(x)$ be the initial condition, then

$$\phi(x,n) = F^n[f](x), \text{ where } F^n = \underbrace{F \circ F \circ \cdots \circ F}_{F \text{ iterated } n \text{ times}}.$$

We will make use of ϕ when we explain our R script or discuss boundary conditions.

The Kronecker delta function,

$$\delta(x) = \begin{cases} 1, & \text{if } x = 0; \\ 0, & \text{otherwise} \end{cases}$$

represents the integer lattice with a single particle at $x = 0$. Applying F repeatedly to $\delta(x)$ yields Pascal's triangle, which by Formula (1) is:

$$F^n[\delta](x) = \begin{cases} C\left(n, \frac{n+x}{2}\right), & \text{if } \frac{n+x}{2} \text{ is an integer between 0 and } n; \\ 0, & \text{otherwise.} \end{cases}$$

Applying F repeatedly to the translated Kronecker delta function,

$$\delta_{x_0}(x) = \delta(x - x_0)$$

yields Pascal's triangle translated, which by Formula (2) is:

$$F^n[\delta_{x_0}](x) = \begin{cases} C\left(n, \frac{n+(x-x_0)}{2}\right), & \text{if } \frac{n+(x-x_0)}{2} \in \{0,1,\ldots,n\}; \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

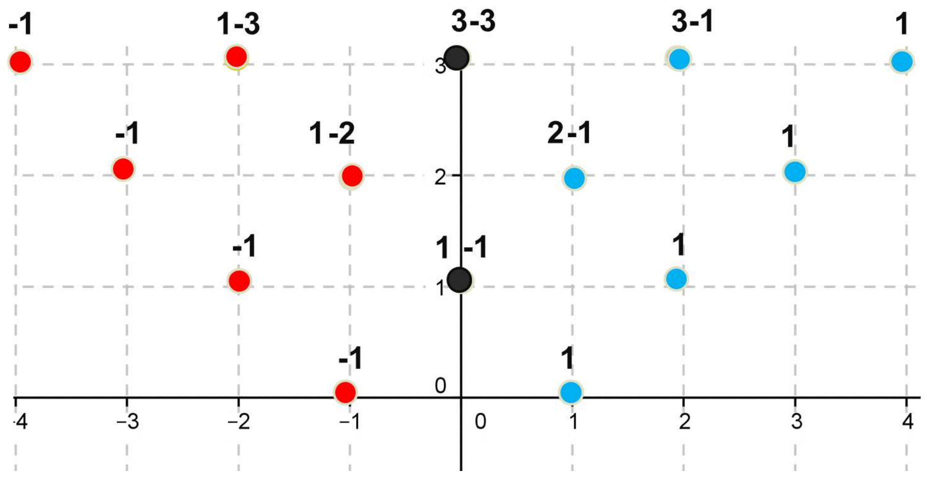
We proceed to solve the ballot problem as before, by assuming that the first vote counted is for A, giving us the initial condition $x = 1$, as we are identifying x with A's lead in the vote count. This

[4]F of course violates conservation of mass, i.e, each time F is applied the total number of particles in the system doubles. If we wish to conserve mass, for example if $f(x)$ represented the probability of reaching x via a lattice path, or the concentration of a substance at x, we would use the relationship $F[f](x) = pf(x-1) + qf(x+1)$, with $p = q = 1/2$ or more generally $p + q = 1$.

leaves $n-1$ votes to count. Hence to solve the ballot problem it suffices to count the number of length $n-1$ good paths from $x=1$ to $x=a-b$, as $a-b$ is the final vote tally.

To do this, we place a particle at $x=1$, which in our formulation, is represented by $\delta(x-1)=\delta_1(x)$, and apply Pascal's rule, meaning F, a total of $n-1$ times to δ_1. However, when a particle reaches the boundary $x=0$, i.e., when a particle's journey starts to represent a bad path, the particle (and its descendants) are no longer keeping track of a good path, and so it and they must eliminated from our counting procedure.

We accomplish this by the method of images[5]. We place a virtual negative particle at $x=-1$, which in our formulation is represented by $-\delta_{-1}(x)$, and then we repeatedly apply F to $(-\delta_{-1}+\delta_1)(x)$. The following figure shows the first few iterations, with the virtual Pascal's triangle starting at $x=-1$.



So, to get the number of good paths of length $n-1$ from $x=1$ to $x\geq 0$ we apply F a total of $n-1$ times to $-\delta_{-1}+\delta_1$. Using the linearity of F and Formula (4) we get:

$$\begin{aligned}F^{n-1}[-\delta_{-1}+\delta_1](x) &= F^{n-1}[\delta_1-\delta_{-1}](x)\\ &= F^{n-1}[\delta_1](x)-F^{n-1}[\delta_{-1}](x)\\ &= C\left(n-1,\frac{(n-1)+(x-1)}{2}\right)-C\left(n-1,\frac{(n-1)+(x--1)}{2}\right)\\ &= C\left(n-1,\frac{n+x-2}{2}\right)-C\left(n-1,\frac{n+x}{2}\right). \qquad (5)\end{aligned}$$

Note that Equation (5) implies $F^{n-1}[-\delta_{-1}+\delta_1](0)=0$ since $\dfrac{n-2}{2}+\dfrac{n}{2}=n-1$, indicating that the choice of $(-\delta_{-1}+\delta_1)(x)$ as the initial conditions, implies that

$$\phi(x,n)=F^n[-\delta_{-1}+\delta_1](x)$$

[5]The method of images, whether being applied to differential equations or lattice path counting requires a unique solution theorem. In the case of lattice path counting, it is clear, by construction, that if the boundary conditions are satisfied that the resulting lattice path count, which is given by repeatedly applying F, will be unique. It is worth noting that F is not 1 to 1 as $\ker F$ is generated by infinite sequences of the form $\ldots-1,0,1,0,\ldots$

will satisfy the boundary conditions, $\phi(0,n) = 0$ and the initial conditions $\phi(x,0) = \delta(x)$ on the domain of the original problem, $\mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0}$, as well as the master equation (Pascal's rule) $\phi(x,n) = \phi(x-1,n-1) + \phi(x+1,n-1)$. Hence, $\phi(x,n)$, which is defined on $\mathbb{Z} \times \mathbb{Z}_{\geq 0}$, when restricted to the domain of the original problem, $\mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0}$, is the solution of the original problem.

By Equation (5), the number of good paths of length $n-1$ from $x = 1$ to $x = a-b$ will be:

$$\begin{aligned} F^{n-1}[-\delta_{-1} + \delta_1](a-b) &= C\left(n-1, \frac{n+(a-b)-2}{2}\right) - C\left(n-1, \frac{n+(a-b)}{2}\right) \\ &= C\left(n-1, \frac{(a+b)+(a-b)-2}{2}\right) - C\left(n-1, \frac{(a+b)+(a-b)}{2}\right) \\ &= C\left(n-1, \frac{2a-2}{2}\right) - C\left(n-1, \frac{2a}{2}\right) \\ &= C\left(n-1, a-1\right) - C\left(n-1, a\right). \end{aligned}$$

This is the same result as gotten by the reflection method. So, as before:

$$\begin{aligned} P(\text{A is always ahead}) &= P(\text{first vote is for A})\ P(\text{A is always ahead} \mid \text{first vote is for A}) \\ &= \frac{a}{a+b}\, \frac{C(n-1,a-1) - C(n-1,a)}{C(n-1,a-1)} \\ &= \frac{a-b}{a+b}. \end{aligned}$$

5 R script

The included R[6] script [S1], explained below and reproduced in the Appendix, allows one to see the result of applying F to an initial condition n times. The script "plots" $\phi(x,i) = F^i[f](x)$ for specified x and $i = 0, 1, 2, \ldots, n$.

5.1 The Function Definitions

The first function, C, counts the number of lattice paths from x_0 to x of length n. In other words, C implements Formula (4), which is:

$$F^n[\delta_{x_0}](x) = \begin{cases} C\left(n, \frac{n+(x-x_0)}{2}\right), & \text{if } \frac{n+(x-x_0)}{2} \in \{0, 1, \ldots, n\}\,; \\ 0, & \text{otherwise.} \end{cases}$$

The function C first checks if $k = \dfrac{n+(x-x_0)}{2}$ is an integer between 0 and n. If it is, C applies R's built in binomial coefficient function, choose(n,k) $= \dfrac{n!}{(n-k)!k!}$. Otherwise C returns 0.

[6] R is an open source programming environment widely used for statistical computing and its graphics capabilities [R], [**Rcite**].

```
C <- function(n,x0,x){
     k = (n +(x-x0))/2;
     out = 0;
     if( any(k == 0:n) ){
        out = choose(n,k);
     }
     return(out)
 }
```

The next function, F, is equivalent to applying Pascal's rule n times to the initial conditions. The initial conditions being two, same sized vectors $x0$, $v0$ with $x0$ giving the particles' positions and $v0$ giving their values (or counts). If we express Pascal's rule recursively as

$$\begin{aligned} \phi(x,n) &= \phi(x-1,n-1) + \phi(x+1,n-1) \qquad x \in \mathbb{Z},\ n \in \mathbb{Z}_{>0} \\ \phi(x0,0) &= v0 \end{aligned}$$

then F returns $\phi(x,n)$ vectorized with respect to x.

```
F  <- function(n,x0,v0,x){
      out = array(dim = c(1,length(x)));
      for(j in 1:length(x)){
          out[j] = 0;
          for(k in 1:length(x0)){
              out[j] = out[j] + v0[k]*C(n,x0[k],x[j])
          }
        }
         return(out);
}
```

The next function, ArrayF, returns the $(n+1) \times$ length(x) matrix whose i,j entry is $\phi(x[j],i)$ with $i = 1,2,\ldots,n$ and $j = 0,1,2,\ldots,$ length(x).

```
ArrayF<-function(n,x0,v0,x){
        out = array(dim = c(n+1,length(x)));
        for(i in 0:n){out[i+1,] = F(i,x0,v0,x);}
        return(out)
}
```

The final function, PlotPascal, displays the non-zero members of ArrayF, as a plot, at the values given in x.

```
PlotPascal <- function(n,x0,v0,x){
               pathArray = ArrayF(n,x0,v0,x);
               plot(NULL, xlim=c(min(x),max(x)),
                          ylim=c(0,n),
                          ylab="n", xlab="x");
```

```
                grid(lwd = 2);
                for(j in 1: length(x)){
                    for(i in 0: n){
                        if(pathArray[i+1,j] !=0){
                           text(x[j],i,paste(pathArray[i+1,j]));
                        };
                    };
                 };
}
```

5.2 Examples using the R script

The following three examples illustrate the use of the R script, with details given for Example 3.

Example 1 *Find the first 12 rows of Pascal's triangle. See Figure 1.*

```
x0 = c(0);  # Example 1. Pascal's Triangle.
v0 = c(1);
n = 12;     # First 12 rows
x = -n:n;
PlotPascal(n,x0,v0,x);
```

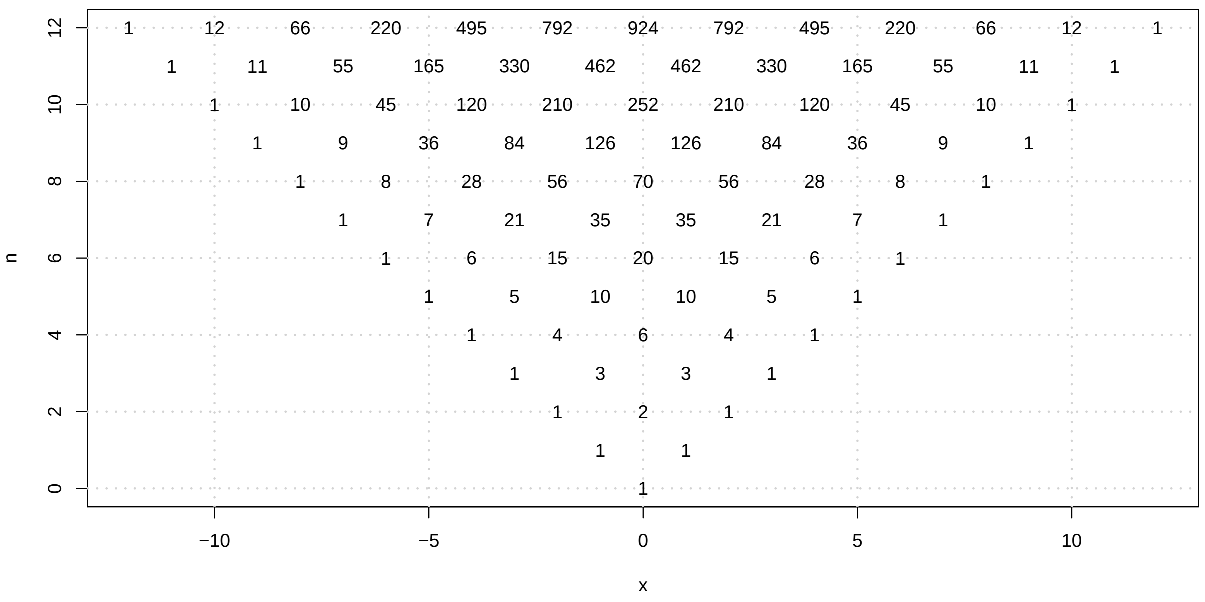


Figure 1: The first 12 rows of Pascal's triangle plotted in R.

Example 2 *Using the method of images and virtual Pascal's triangles find the number of lattice paths of lengths $n = 0, 1, \ldots, 12$ from $x_0 = 1$ to $x > 0$, which do not intersect $x = 0$. See Figure 2.*

```
x0 = c(-1, 1);  # Example 2. Using Virtual Pascal's Triangle
v0 = c(-1, 1);  #            to count lattice paths.
n = 12;
x = (-n-1):(n+1);
PlotPascal(n,x0,v0,x);
```

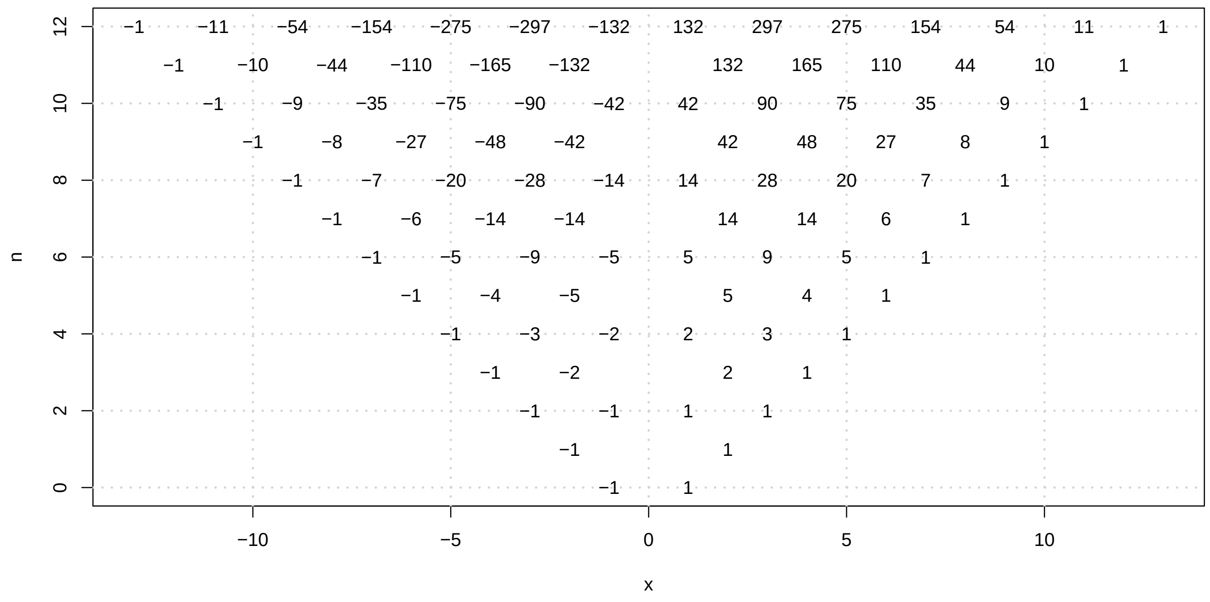


Figure 2: The number of good paths from $x = 1$ to x plotted in R. The virtual Pascal's triangle starts on the left, at $x = -1$.

Example 3 *Ballot Problem. Suppose candidate A received $a = 8$ votes and candidate B received $b = 5$ votes. Find the probability that candidate A is always ahead in the vote count.*

We let x = how many votes A is ahead of B in the vote tally. At the end of the vote tally, $x = a - b = 8 - 5 = 3$. The probability that the first vote counted is for A is

$$\frac{a}{a+b} = \frac{8}{8+5} = \frac{8}{13}.$$

If we assume that the first ballot counted is for A, then $x = 1$ and of the remaining 12 ballots, $8 - 1 = 7$ will be for A. The number of ways these 12 ballots can be permuted is the same as the number of paths of length 12 that start at $x = 1$ and end at $x = a - b = 3$, which can be found by applying F

to the initial condition $f(x) = \delta(x-1) = \delta_1(x)$ a total of $n = 12$ times. In the R script we represent this initial condition $f(x)$ by the pair of 1-vectors $c(1), c(1)$, which tell R that at $x = 1$ (from the first $c(1)$) that the value of f is 1 (from the second $c(1)$) and that otherwise $f(x) = 0$. There are

$$F^{12}[\delta_1](3) = \overbrace{F(12, \underbrace{c(1), c(1)}_{\delta_1}, c(3))}^{\text{The } F \text{ function from R script.}} = C(12, 8-1) = 792$$

distinct ways to tally those ballots, see Figure 1, row 12. Of those 792 ways to tally the votes

$$F^{12}[-\delta_{-1} + \delta_1](3) = \overbrace{F(12, \underbrace{c(-1,1), c(-1,1)}_{-\delta_{-1}+\delta_1}, c(3))}^{\text{The } F \text{ function from R script.}} = 297 \qquad (6)$$

of the ways have A always ahead, by the method of virtual Pascal's triangles, see Figure 2, row 12. For the second calculation (6), the initial condition $f(x) = \delta_{-1}(x) + \delta_1(x)$. In the R script we represent this initial condition $f(x)$ by the pair of 2-vectors $c(-1,1), c(-1,1)$, which tell R that $f(-1) = -1$ and $f(1) = 1$ and that otherwise $f(x) = 0$. Carrying out the calculations we arrive at:

$$\text{Probability that candidate A is always ahead} = \frac{8}{13} \cdot \frac{297}{792} = \frac{b-a}{b+a} = \frac{3}{13} = 0.2307692.$$

```
a = 8; b = 5;   # Example 3. Ballot Problem.
paths = F(a+b-1,c(1),c(1),c(a-b))[1,1];
paths;
goodpaths = F(a+b-1,c(-1,1),c(-1,1),c(a-b))[1,1];
goodpaths;
paste('Probability A always ahead of B = ',
 (a/(a+b))*(goodpaths/paths));
paste('Check answer. (a-b)/(a+b) = ', (a-b)/(a+b));
```

6 Conclusion

The method of images and virtual Pascal's triangles leads to appealing visual representations of standard combinatorial methods. It also provides a connection between combinatorics and the method of images from the boundary value problems of mathematical physics, especially certain problems arising from the study of the diffusion or heat equation. Our R script, which displays the number of lattice paths (both actual and virtual), is easily modifiable, and may be of interest to those studying lattice paths, integer sequences, or discrete simulations of the diffusion process.

7 Acknowledgements

We thank the Editor-in-Chief Wei-Chi Yang and the two anonymous reviewers for their support; for helping us to identify and correct our grammatical, typographic, and stylistic errors; and for suggesting that we make our R script available as a download.

Software Packages

[R] The R software package can be downloaded (GNU open source) from the R Project for Statistical Computing website https://www.r-project.org/.

Supplemental Electronic Materials

[S1] The R script discussed in this paper can be downloaded from the July 9, 2016 entry of the Math & Simulations website https://mccarthymath.commons.gc.cuny.edu/papers/virtualpascal2017ejmt/.

Appendix: The R Script

```
#---------------------------------------------
# Function Definitions
#---------------------------------------------
# C counts the number of lattice paths from x0 to x of length n
C <- function(n,x0,x){
     k = (n +(x-x0))/2;
     out = 0;
     if( any(k == 0:n) ){
        out = choose(n,k);
     }
     return(out)
 }
#---------------------------------------------
# F applies Pascal's rule n times to the vectors x0, v0
F  <- function(n,x0,v0,x){
      out = array(dim = c(1,length(x)));
      for(j in 1:length(x)){
          out[j] = 0;
          for(k in 1:length(x0)){
              out[j] = out[j] + v0[k]*C(n,x0[k],x[j])
          }
        }
         return(out);
}
#---------------------------------------------
# Returns an array of F applied i = 0, 1, 2,..., n times
ArrayF<-function(n,x0,v0,x){
        out = array(dim = c(n+1,length(x)));
        for(i in 0:n){out[i+1,] = F(i,x0,v0,x);}
        return(out)
}
#---------------------------------------------
PlotPascal <- function(n,x0,v0,x){
              pathArray = ArrayF(n,x0,v0,x);
              plot(NULL, xlim=c(min(x),max(x)),
                         ylim=c(0,n),
                         ylab="n", xlab="x");
              grid(lwd = 2);
              for(j in 1: length(x)){
                  for(i in 0: n){
                      if(pathArray[i+1,j] !=0){
                         text(x[j],i,paste(pathArray[i+1,j]));
```

```
                        };
                    };
                };
}
#-----------------------------------------------
# End Function Definitions
#-----------------------------------------------
x0 = c(0);  # Example 1. Pascal's Triangle.
v0 = c(1);
n = 12;     # First 12 rows
x = -n:n;
PlotPascal(n,x0,v0,x);
#-----------------------------------------------
x0 = c(-1, 1);  # Example 2. Using Virtual Pascal's Triangle
v0 = c(-1, 1);  #            to count lattice paths.
n = 12;
x = (-n-1):(n+1);
PlotPascal(n,x0,v0,x);
#-----------------------------------------------
a = 8; b = 5;   # Example 3. Ballot Problem.
paths = F(a+b-1,c(1),c(1),c(a-b))[1,1];
paths;
goodpaths = F(a+b-1,c(-1,1),c(-1,1),c(a-b))[1,1];
goodpaths;
paste('Probability A always ahead of B = ',
 (a/(a+b))*(goodpaths/paths));
paste('Check answer. (a-b)/(a+b) = ', (a-b)/(a+b));
#-----------------------------------------------
```