Stochastic Processes and
Advanced Mathematical Finance

Approximation of Brownian Motion by Coin-Flipping Sums

Rating
Mathematically Mature: may contain mathematics beyond calculus with proofs.
Section Starter Question

Suppose you know the graph $y = f(x)$ of the function $f(x)$. What is the effect on the graph of the transformation $f(ax)$ where $a > 1$? What is the effect on the graph of the transformation $(1/b)f(x)$ where $b > 1$? What about the transformation $f(ax)/b$ where $a > 1$ and $b > 1$?

Key Concepts

1. Brownian motion can be approximated by a properly scaled “random fortune” process (i.e. random walk).

2. Brownian motion is the limit of “random fortune” discrete time processes (i.e. random walks), properly scaled. The study of Brownian motion is therefore an extension of the study of random fortunes.

Vocabulary

1. We define approximate Brownian Motion $\hat{W}_N(t)$ to be the rescaled random walk with steps of size $1/\sqrt{N}$ taken every $1/N$ time units where $N$ is a large integer.
Mathematical Ideas

Approximation of Brownian Motion by Fortunes

As we have now assumed many times, for \( i \geq 1 \) let

\[
Y_i = \begin{cases} 
+1 & \text{with probability } 1/2 \\
-1 & \text{with probability } 1/2
\end{cases}
\]

be a sequence of independent, identically distributed Bernoulli random variables. Note that \( \text{Var}[Y_i] = 1 \), which we will need to use in a moment. Let \( Y_0 = 0 \) for convenience and let

\[
T_n = \sum_{i=0}^{n} Y_i
\]

be the sequence of sums which represent the successive net fortunes of our notorious gambler. Sketch the random fortune \( T_n \) versus time using linear interpolation between the points \((n-1, T_{n-1})\) and \((n, T_n)\) to obtain a continuous, piecewise linear function. The interpolation defines a function \( \hat{W}(t) \) defined on \([0, \infty)\) with \( \hat{W}(n) = T_n \). This function is piecewise linear with segments of length \( \sqrt{2} \). The notation \( \hat{W}(t) \) reminds us of the piecewise linear nature of the function.

We will compress time, and rescale the space in a special way. Let \( N \) be a large integer, and consider the rescaled function

\[
\hat{W}_N(t) = \left( \frac{1}{\sqrt{N}} \right) \hat{W}(Nt).
\]

This has the effect of taking a step of size \( \pm 1/\sqrt{N} \) in \( 1/N \) time unit. For example,

\[
\hat{W}_N(1/N) = \left( \frac{1}{\sqrt{N}} \right) \hat{W}(N \cdot 1/N) = \frac{T_1}{\sqrt{N}} = \frac{Y_1}{\sqrt{N}}.
\]

Now consider

\[
\hat{W}_N(1) = \frac{\hat{W}(N \cdot 1)}{\sqrt{N}} = \frac{\hat{W}(N)}{\sqrt{N}} = \frac{T_N}{\sqrt{N}}.
\]
According to the Central Limit Theorem, this quantity is approximately normally distributed, with mean zero, and variance 1. More generally,

\[ \hat{W}_N(t) = \frac{\hat{W}(Nt)}{\sqrt{N}} = \sqrt{t} \frac{\hat{W}(Nt)}{\sqrt{Nt}} \]

If \( Nt \) is an integer, \( \hat{W}_N(t) \) is normally distributed with mean 0 and variance \( t \). Furthermore, \( \hat{W}_N(0) = 0 \) and \( \hat{W}_N(t) \) is a continuous function, and so is continuous at 0. At times \( t_j = j/N, t_k = k/N, t_\ell = \ell/N, \) and \( t_m = m/N \) with \( t_j < t_k \leq t_\ell < t_m \) the function differences \( \hat{W}_N(t_k) - \hat{W}_N(t_j) \) and \( \hat{W}_N(t_m) - \hat{W}_N(t_\ell) \) are the differences \( (T_k - T_j)/\sqrt{N} \) and \( (T_m - T_\ell)/\sqrt{N} \), hence independent.

Altogether, this should be a strong suggestion that \( \hat{W}_N(t) \) is an approximation to Standard Brownian Motion. We will define the very jagged piecewise linear function \( \hat{W}_N(t) \) as \textbf{approximate Brownian Motion}.

\textbf{Theorem 1.} The limit of the rescaled random walk defining approximate Brownian Motion is Brownian Motion in the following sense:

1. \[ P\left[ \hat{W}_N(t) < x \right] \to P[\hat{W}(t) < x] \text{ as } N \to \infty. \]

2. More generally, the limit of the rescaled random walk defining approximate Brownian Motion is Brownian Motion in the following sense:

\[ P\left[ \hat{W}_N(t_1) < x_1, \hat{W}_N(t_2) < x_2, \ldots, \hat{W}_N(t_n) < x_n \right] \to P[\hat{W}(t_1) < x_1, \hat{W}(t_2) < x_2, \ldots, \hat{W}(t_n) < x_n] \]

as \( N \to \infty \) where \( t_1 < t_2 < \cdots < t_n \). That is, the joint distributions of \( \hat{W}_N(t) \) converges to the joint normal distribution

\[ f(x_1, x_2, x_2 ; \ldots ; x_n, x_n) = p(x_1, t)p(x_2-x_1, t_2-t_1) \cdots p(x_n-x_{n-1}, t_n-t_{n-1}) \]

of the Standard Brownian Motion.

With some additional foundational work, a mathematical theorem establishes that the rescaled fortune processes actually converge to the mathematical object called the Standard Brownian Motion as defined in the previous section. The proof of this mathematical theorem is beyond the scope of a text of this level, but the theorem above should strongly suggest how this
can happen, and give some intuitive feel for the approximation of Brownian motion through the rescaled coin-flip process.

Using a scaled random walk is not the only way to approximate Brownian Motion. Other approximations of the Wiener process use “global” approximations such as Fourier series (or more generally $L^2[0,T]$ expansions in an orthogonal basis) or Bernstein polynomials. The Fourier series representation is also known at the Karhunen-Loève expansion of the Wiener process, for elementary details, see [3]. For the representation as a limit of Bernstein polynomials, see Introduction to Bernstein Polynomials and Brownian Motion or [4]. Both of these approximations use ideas from probability theory and analysis which are beyond the scope of this book. When one only needs to simulate the position of a sample path of Brownian Motion at one or even several time points, then the scaled random walk approximation is simple and the accuracy can be estimated with the Central Limit Theorem or the Berry-Esseen Theorem. If one needs information on a whole sample path of Brownian Motion, then the “global” methods are more appropriate approximations. The “global” methods are not only more mathematically sophisticated, they also are more expensive in terms of processing and rely on the underlying implementations of the mathematical functions used.

**Sources**

Algorithms, Scripts, Simulations

Algorithm

Simulate a sample path of the Wiener process as follows, see [3]. Divide the interval $[0, T]$ into a grid of $N + 1$ nodes $0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = T$ with $t_{i+1} - t_i = \Delta$. The nodes may be indexed from either 0 to $N$ or from 1 to $N + 1$ depending on the language. Create a Bernoulli random walk $T_0, T_1, T_2, \ldots, T_N$ of $N + 1$ steps with $T_0 = 0$. Given a value $x \in [0, T]$ the prior node with $t_k \leq x$ is $\lfloor x/\Delta \rfloor$ for 0-based arrays (or $\lfloor x/\Delta \rfloor + 1$ for 1-based arrays). The subsequent node with $x \leq t_{k+1}$ is $\lceil x/\Delta \rceil$ for 0-based arrays (or $\lceil x/\Delta \rceil + 1$ for 1-based arrays). Then define the value of the approximation function $\hat{W}_N(x)$ by linear interpolation between the values of the random walk at $T_k$ and $T_{k+1}$.

A feature of this $N + 1$-step random walk scaling approximation algorithm is that it creates the approximation as a function on $[0, T]$. This function can then be plotted with a function plotting routine on any time grid on the interval $[0, T]$. If the time grid used for plotting on $[0, T]$ has less than $N$ points, then some of the information in the $N$-step scaling approximation is ignored, and the plotted function will be less representative of the approximation than it could be. If the time grid on $[0, T]$ is greater than $N$ points, then the plotted function will just represent the linear interpolation between the random-walk points at $T_j = jT/N$ and no new information is represented.

Depending on the internal plotting routines used by the language, plotting the approximation function $\hat{W}_N(t)$ can result in plot artifacts. One simple artifact may be horizontal segments in the plot. If the plotting algorithms attempt to use adaptive point selection to densely position a greater portion of a fixed number of plotting points in a region of rapid variation, then other regions will have fewer plotting points. Those regions with fewer plotting points will miss some of the information in that region. Depending on the language, the plotting routine may use smoothing or some other nonlinear interpolation between plotting points which will result in curved segments instead of a piecewise linear function. If the intention is to plot an approximate Brownian Motion, then there are more direct and efficient ways to create and plot the $N + 1$ coordinate pairs $(jT/N, \sqrt{T/\Delta} S_j)$ defining the vertices of the piecewise linear scaled random walk approximation with an appropriate amount of information. (This is approach taken in the GeoGebra script, because the language does not support building a separate function
as in the other languages.) Here the intention is to first to demonstrate the creation of the approximation function as a piecewise linear function, then second to use the function to plot a graph.

**Scripts**

**R**

R script for approximation

```r
p <- 0.5
N <- 400
T <- 1

S <- array(0, c(N+1))
rw <- cumsum( 2 * ( runif(N) <= p)-1 )
S[2:(N+1)] <- rw

WcaretN <- function(x) {
  Delta <- T/N

  # add 1 since arrays are 1-based
  prior = floor(x/Delta) + 1
  subsequent = ceiling(x/Delta) + 1

  retval <- sqrt(Delta)*(S[prior] + ((x/Delta+1) - prior)*(S[subsequent] - S[prior]))
}

plot(WcaretN, 0,1, n=400)
```

# # alternative plotting method
# # time point grid
# t <- seq(0,T, length=N+1)
# # approximate Wiener process at time point grid
# W <- sapply(t, WcaretN)
# plot(t, W, type = "l")
Octave

Octave script for approximation

```octave
p = 0.5;

global N = 400;
global T = 1;

global S
S = zeros(N+1, 1);
S(2:N+1) = cumsum( 2 * (rand(N,1)<=p) - 1 );

function retval = WcaretN(x)
    global N;
global T;
global S;
    Delta = T/N;

    # add 1 since arrays are 1-based
    prior = floor(x/Delta) + 1;
    subsequent = ceil(x/Delta) + 1;

    retval = sqrt(Delta)*(S(prior) + ((x/Delta+1) - prior).* (S(subsequent) - S(prior)));
endfunction

fplot (@WcaretN, [0 ,T])
```

Perl

Perl PDL script for approximation

```perl
use PDL::NiceSlice;

$p = 0.5;

$N = 400;
```
$T = 1;

# the random walk
$S = zeros( $N + 1 );
$S ( 1 : $N ) .= cumusumover( 2 * ( random($N) <= $p ) - 1 );

# function WcaretN interpolating random walk
sub WcaretN {
  my $x = shift @_;
  $Delta = $T / $N;

  $prior = floor( $x / $Delta );
  $subsequent = ceil( $x / $Delta );

  $retval =
    sqrt($Delta)
    * ($S($prior)
        + ( ( $x / $Delta ) - $prior )
        * ($S($subsequent) - $S($prior)));
}

# file output to use with external plotting programming
# such as gnuplot, R, octave, etc.
# Start gnuplot, then from gnuplot prompt
# plot "wienerprocess.dat" with lines
$M = 300;
$tgri = zeros( $M + 1 )->xlinvals( 0, $T );
$W = W caretN($tgri);

open( F, ">wienerprocess.dat" ) || die "cannot_write: $!";
foreach $j ( 0 .. $M ) {
  print F $tgri->range( [ $j ] ), "", $W->range( [ $j ] ), "n";
}
close(F);

SciPy Scientific Python script for approximation

import scipy
p = 0.5
N = 400
T = 1.

# the random walk
S = scipy.zeros(N+1)
S[1:N+1] = scipy.cumsum( 2*( scipy.random.random(N) <= p ) - 1 )

def WcaretN(x):
    Delta = T/N
    prior = scipy.floor(x/Delta).astype(int)
    subsequent = scipy.ceil(x/Delta).astype(int)
    return scipy.sqrt(Delta)*(S[prior] + (x/Delta - prior)*(S[subsequent] - S[prior]))

M = 300
tgrid = scipy.linspace(0, T, M+1)
W = WcaretN(tgrid)

# optional file output to use with external plotting programming
# such as gnuplot, R, octave, etc.
# Start gnuplot, then from gnuplot prompt
# plot "wienerprocess.dat" with lines
f = open('wienerprocess.dat', 'w')
for j in range(0,M+1):
    f.write(str(tgrid[j]) + ' ' + str(W[j]) + '
')

f.close()
Problems to Work for Understanding

1. Flip a coin 25 times, recording whether it comes up Heads or Tails each time. Scoring \( Y_i = +1 \) for each Heads and \( Y_i = -1 \) for each flip, also keep track of the accumulated sum \( T_n = \sum_{i=1}^{n} T_i \) for \( i = 1 \ldots 25 \) representing the net fortune at any time. Plot the resulting \( T_n \) versus \( n \) on the interval \([0, 25]\). Finally, using \( N = 5 \), plot the rescaled approximation \( \hat{W}_5(t) = \left(1/\sqrt{5}\right)T(5t) \) on the interval \([0, 5]\) on the same graph.

2. (a) What are the slopes of the linear segments of \( \hat{W}_N(t) \)?
   (b) Is the slope necessarily undefined at the nodes defining \( \hat{W}_N(t) \)?
   (c) Explain how a plotting routine could create a horizontal segment in the plot of \( \hat{W}_N(t) \).

3. Modify one of the scripts in the following ways:
   (a) Plot more than one Wiener process sample path on the same set of axes.
   (b) Change \( p \) to a value greater than 0.5 and describe the effect on the sample paths. Change \( p \) to a value less than 0.5 and describe the effect on the sample paths.
   (c) Plot the approximation functions for increasing values of \( N \) and describe the effects on the sample paths.
   (d) For a fixed, large value of \( N \), plot the approximation at increasing values of the number of plotting points which are not divisors of \( N \) and describe the effects on the sample paths.
   (e) Change the value of \( T \) and plot the resulting sample paths, both without increasing the value of \( N \) and increasing the value of \( N \) proportionally to \( T \) and describe the resulting sample paths.

4. Iacus uses a different random walk step function approximation to Brownian Motion:

\[
\bar{W}_N(t) = \frac{S_{\lfloor Nt \rfloor}}{\sqrt{N}}.
\]

Create scripts that plot this random walk approximation for \( N = 10 \), \( N = 100 \) and \( N = 1000 \).
5. Under the assumption of the theorem that the joint distributions of $\hat{W}_N(t)$ converges to the joint normal distribution is true, show then that

$$\mathbb{P}\left[ \frac{S[N]}{\sqrt{N}} < x \right] \rightarrow \mathbb{P}[W(t) < x].$$

Reading Suggestion:

References


Outside Readings and Links:

1.

I check all the information on each page for correctness and typographical errors. Nevertheless, some errors may occur and I would be grateful if you would alert me to such errors. I make every reasonable effort to present current and accurate information for public use, however I do not guarantee the accuracy or timeliness of information on this website. Your use of the information from this website is strictly voluntary and at your risk.

I have checked the links to external sites for usefulness. Links to external websites are provided as a convenience. I do not endorse, control, monitor, or guarantee the information contained in any external website. I don’t guarantee that the links are active at all times. Use the links here with the same caution as you would all information on the Internet. This website reflects the thoughts, interests and opinions of its author. They do not explicitly represent official positions or policies of my employer.

Information on this website is subject to change without notice.
Steve Dunbar’s Home Page, http://www.math.unl.edu/~sdunbar1
Email to Steve Dunbar, sdunbar1 at unl dot edu
Last modified: Processed from \LaTeX{} source on August 26, 2013