Math 248 - Project 2:
Diffusion on a Grid (Brownian Motion)

Josh Derrow & Michael Berry

5/11/25

Problem Statement:

In this project we are studying and analyzing Brownian motion, which is the random movement of very small particles suspended in a fluid (liquid or gas). The motion continues as long as the particle is suspended in the fluid and has no predictable direction or pattern. However, it has been observed that after an uncertain amount of time, the particle almost always returns to its starting position. For one-dimensional Brownian motion the particle can only move left and right, but for two-dimensional Brownian motion the particle can move up, down, left, or right. The focus of this project will be to analyze the iteration and max distance distributions for both Brownian motions, as well as other statistical data such as the average number of iterations (steps) and the average max distance reached.

Solution Procedure:

For C grade functionality, we made a simple Matlab function (brownmotion1d.m) that has an initial random movement to move the particle from the origin, then continuous random movements until the particle returns to the origin. At each iteration, we call rand() to get a random number between 0 and 1 to determine the one-dimensional movement direction. The particle moves left (position - 1) if rand() is less than 0.5, and the particle moves right (position + 1) if rand() is greater than 0.5. Once the particle returns to the origin, the total number of iterations and the maximum distance away from the origin reached is returned.

For B grade functionality, we made another Matlab function (brownmotion2d.m) very similar to brownmotion1d.m except this one has a 25% chance of moving up, down, left, or right instead of a 50% chance of moving just left or right. Like before, this function has an initial random movement to move the particle from the origin, then continuous random movements until the particle returns to the origin. However this time at each iteration, we call rand() to get a random number between 0 and 1 to determine the two-dimensional movement direction. The particle moves up (position + [0, 1]) if rand() is less than 0.25, down (position - [0, 1]) if rand() is less than 0.5, left (position - [1, 0]) if rand() is less than 0.75, and the particle moves right (position + [1, 0]) if rand() is greater than 0.75. After each random movement the Euclidean distance from the origin is calculated ($dist = \sqrt{pos(1)^2 + pos(2)^2}$) and this value is compared against the current max distance to see if the current distance is greater. If the current value is greater, then the overall max distance is set to the current distance. Once the particle returns to the origin, the total number of iterations and the maximum distance away from the origin reached is returned.

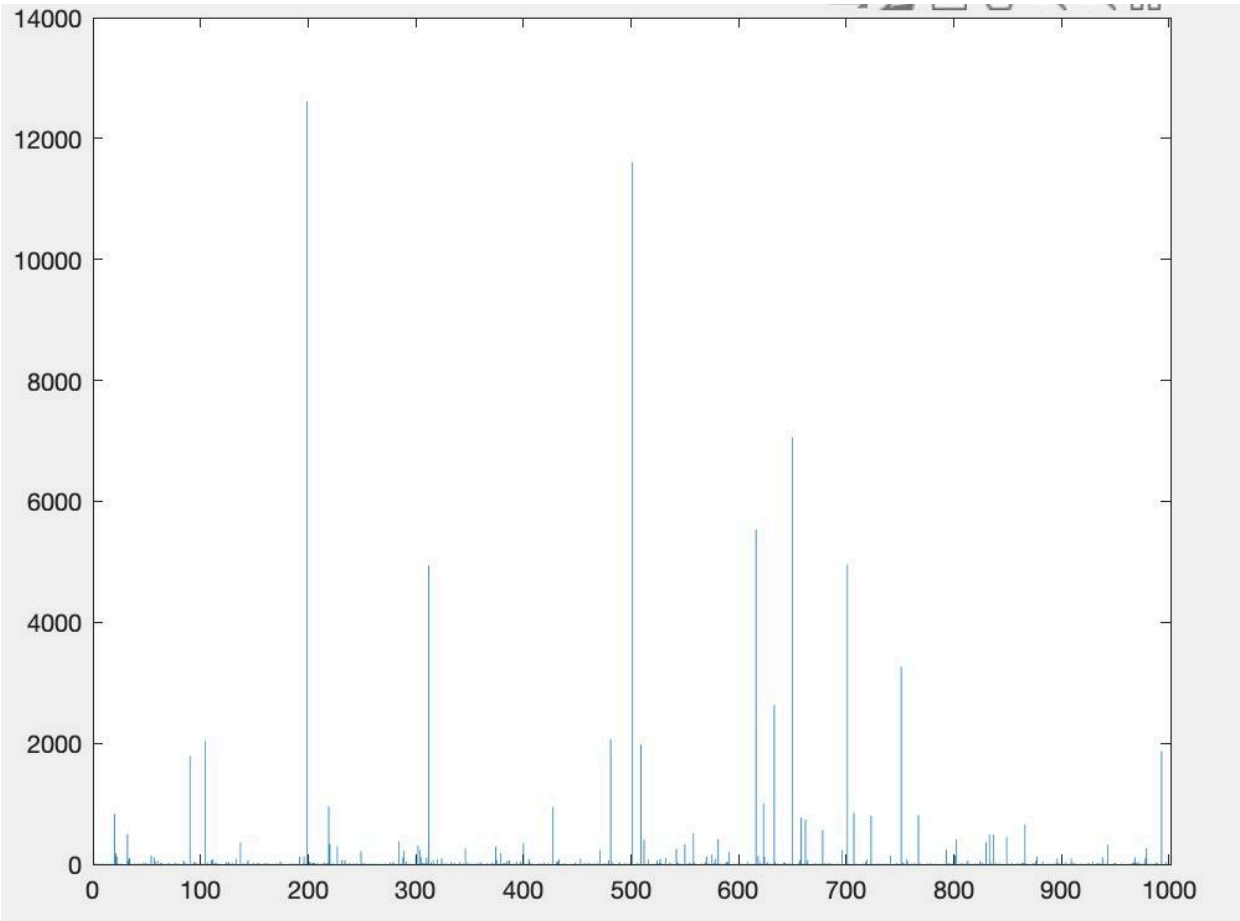
For A grade functionality, like B and C grades, we made a Matlab function randpercent.m that would take as input a percentile p (between 1 and 99) that would be used as the new distance moved each iteration. For this function we looped until the position went back to 0 or if the number of iterations reached 16,000,000. We chose this number because I observed from part C that the highest amount of iterations was ~15,000,000. The function returned, in an array, the number of iterations and the max distance achieved. To test the peculiarities of this problem, I wanted to graph the ratio of successful tests vs total tests for each natural number up to 100 when I loop through this code 1000 times. Also, I tested certain p values (20,25,50, and 60) to find their average number of iterations required to return to 0 when I have the upper limit set to 1,000,000 and when it's set to 16,000,000. This allowed me to see the "true" average (when the

outliers aren't so big) as compared to a skewed higher average where outliers can be 16x greater.

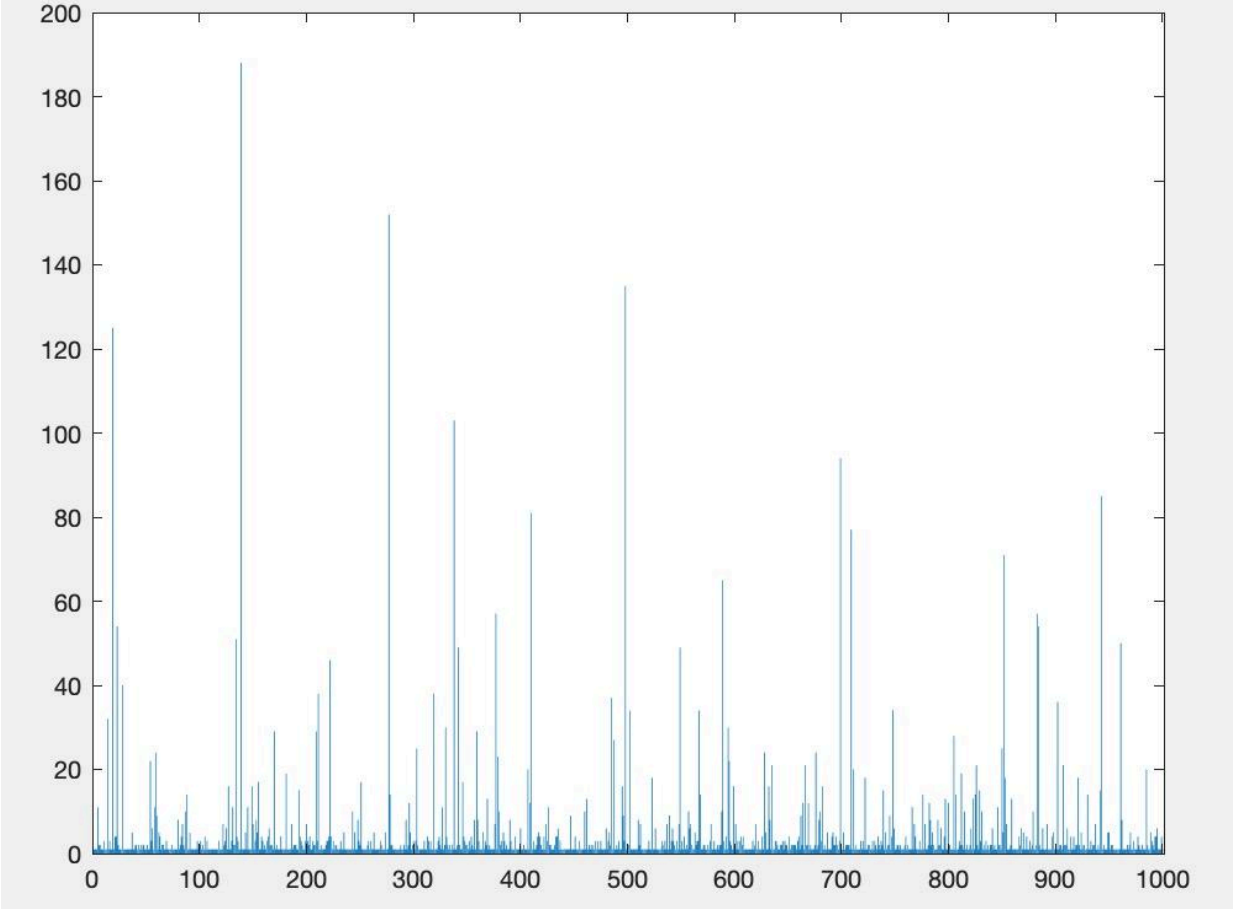
Results and Conclusions:

For C grade functionality, we wrote a test script (testbrownmotion1d.m) that calls brownmotion1d 1000 times and aggregates the results. While testing brownmotion1d.m, I set the upper limit of the number of iterations to 20,000 (otherwise there would always be a case in the millions messing with the averages and bar graph).

1D Iteration Distribution:

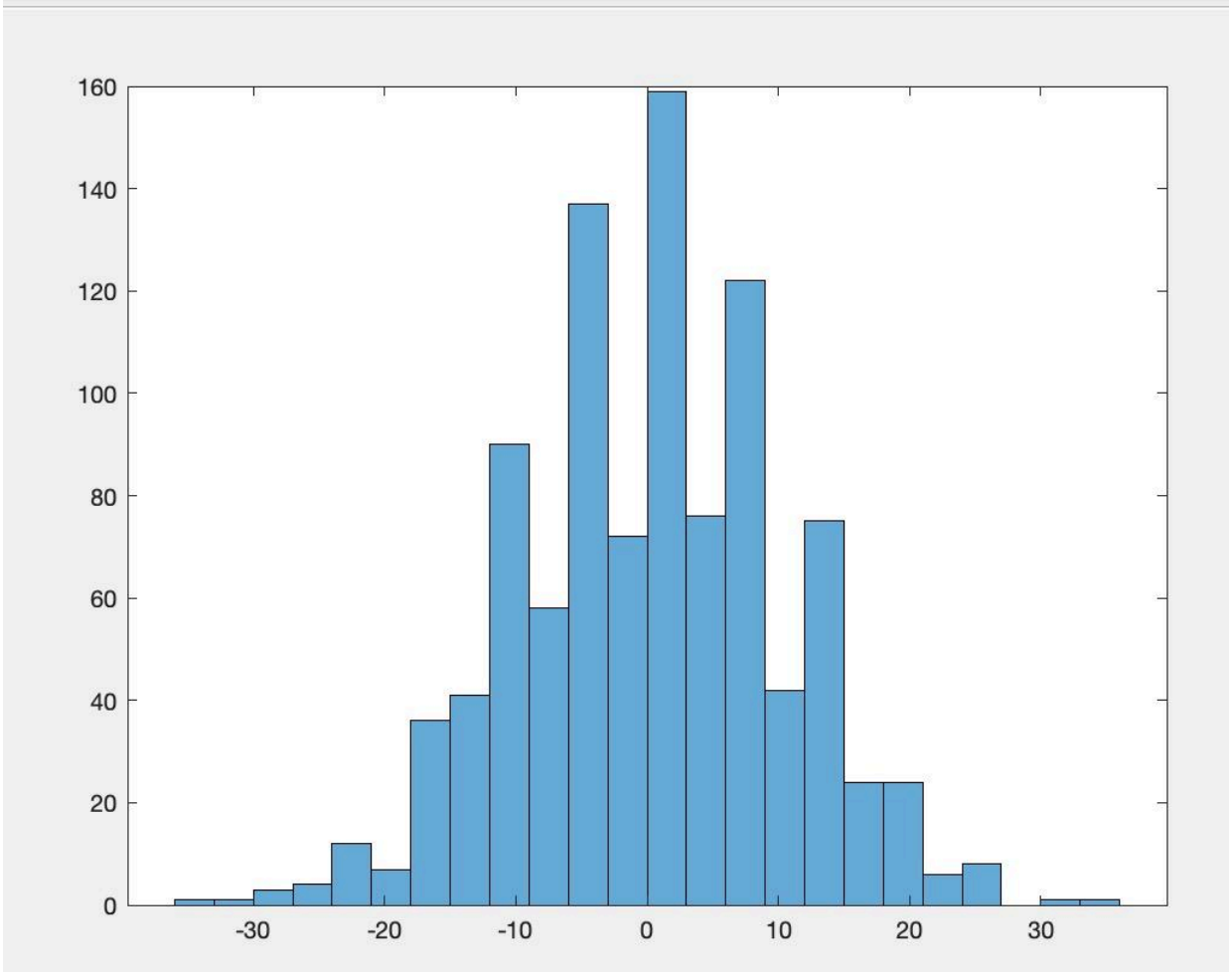


1D Distance Distribution:



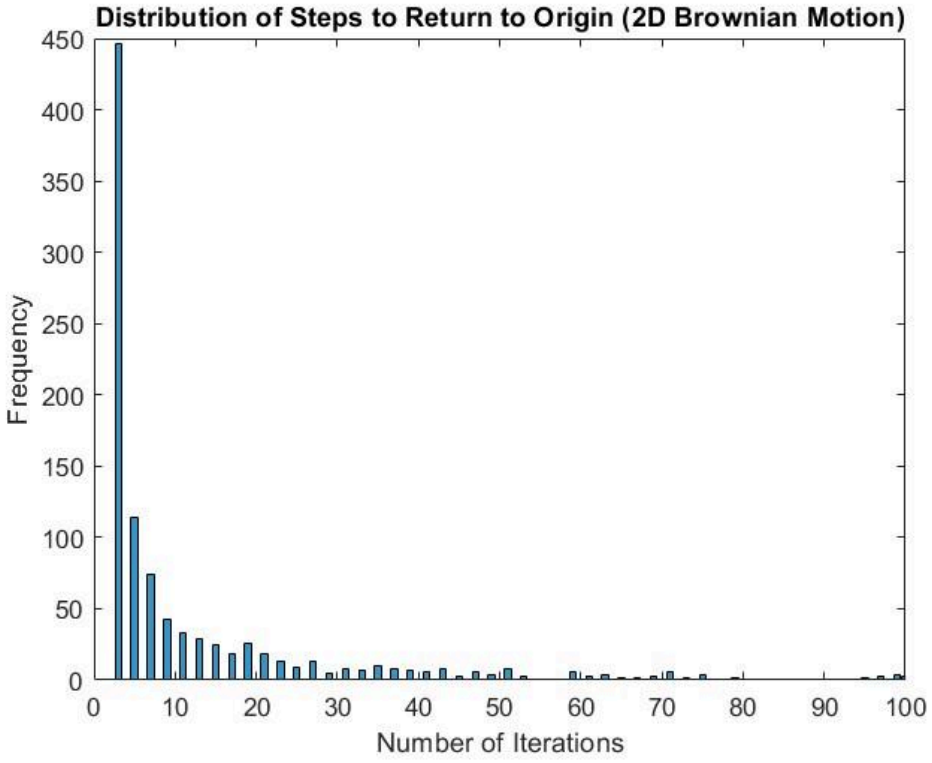
For one-dimensional Brownian motion, we observed an average of 133.76 iterations and an average max distance of 5.08 away from the origin.

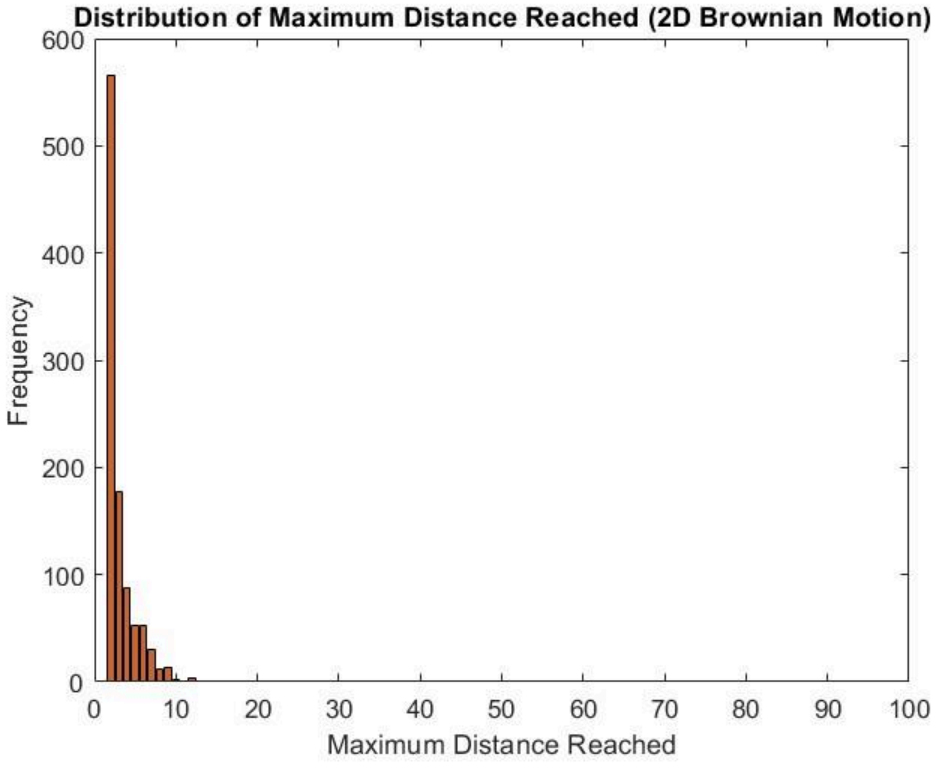
1D Particle Position Distribution:



This histogram represents the position of the particle after running stepbrownmotion() a thousand times with n = 100. This curve is similar to the bell curve. When I run the code for n = 200 or 400, the curve is still similarly shaped to the bell curve.

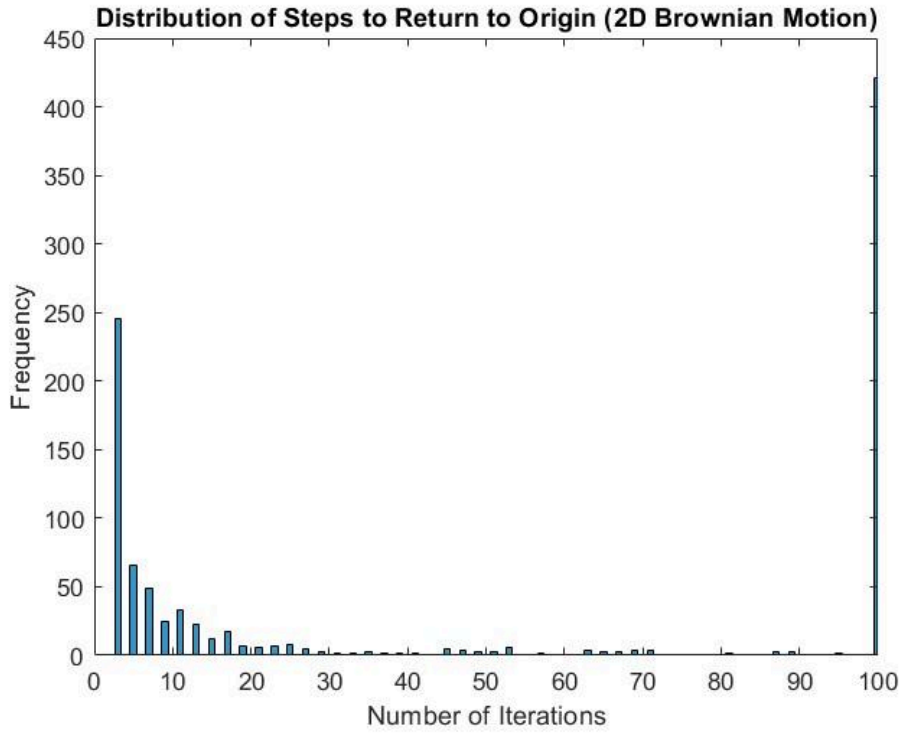
For B grade functionality, we wrote two very similar test scripts (testbrownmotion2d.m & teststepbrownmotion2d.m) to the previous testbrownmotion1d.m that calls brownmotion2d.m and stepbrownmotion2d.m 1000 times each and aggregates the results. Below are the iteration and distance distributions for brownmotion2d.m:

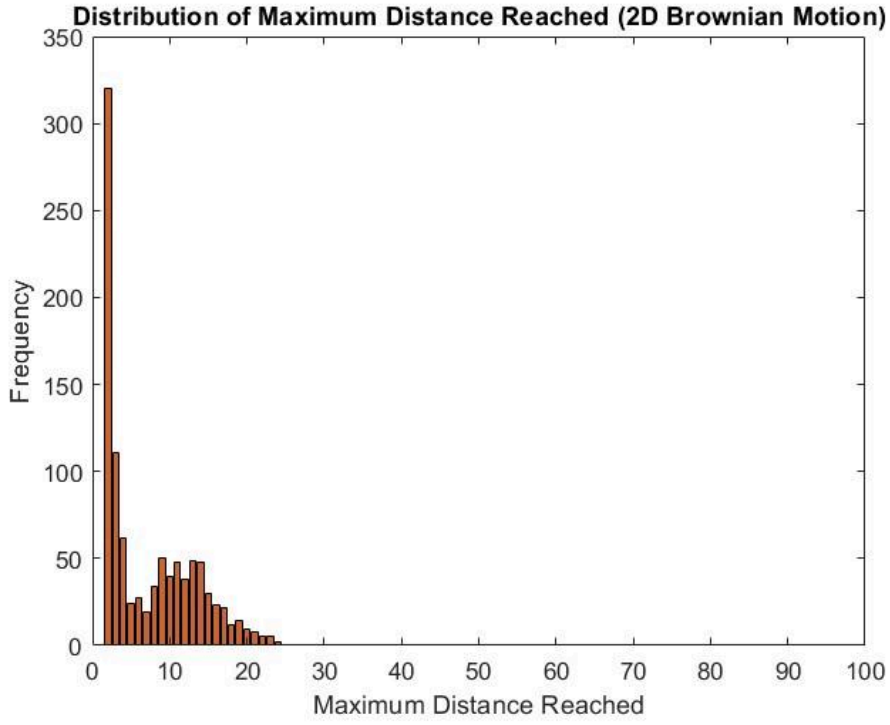

Distribution of Steps to Return to Origin (2D Brownian Motion)
Frequency
Number of Iterations
0
50
100
150
200
250
300
350
400
450
0
10
20
30
40
50
60
70
80
90
100


Distribution of Maximum Distance Reached (2D Brownian Motion)
Frequency
Maximum Distance Reached
0
100
200
300
400
500
600
0
10
20
30
40
50
60
70
80
90
100

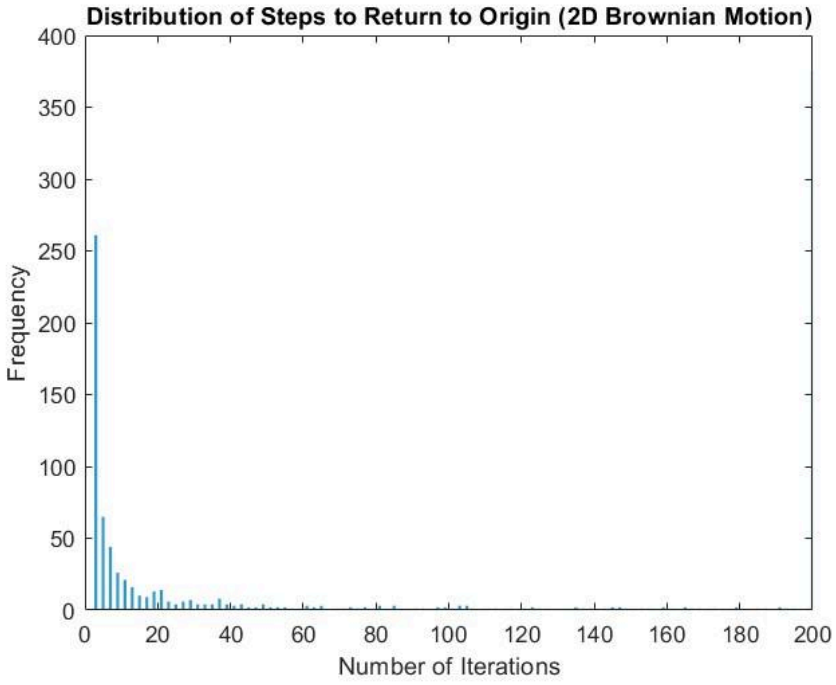
With these tests (testbrownmotion2d.m), we observed an average number of 13.792 iterations and an average maximum distance of 2.362 away from the origin. Below are the iteration and distance distributions for stepbrownmotion2d.m:

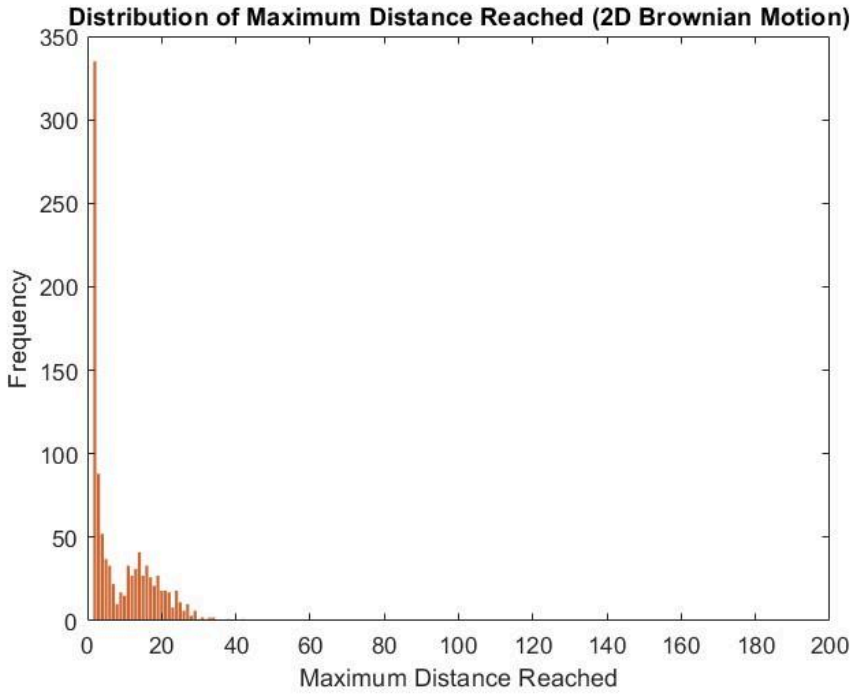
Iteration limit = 100:



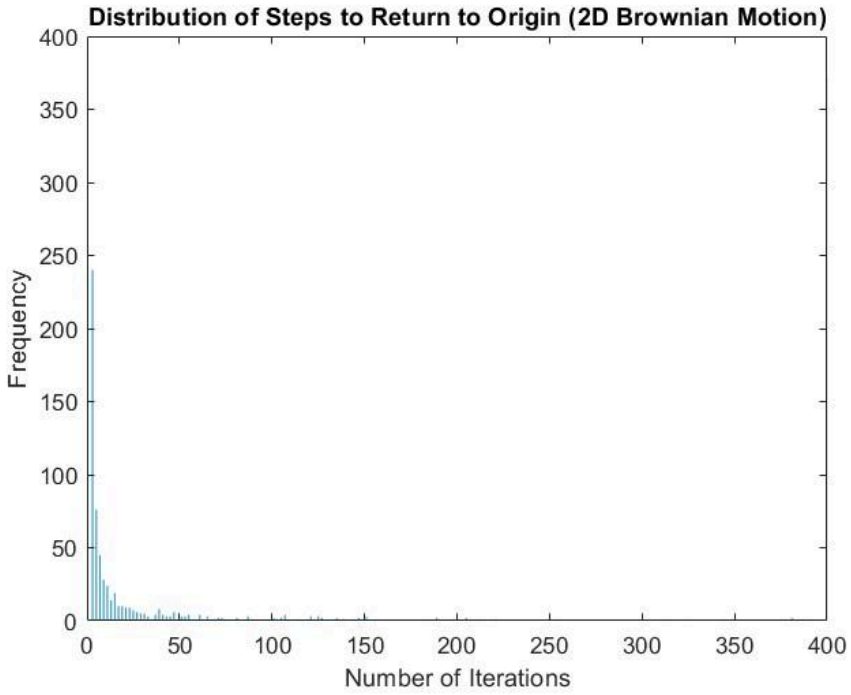


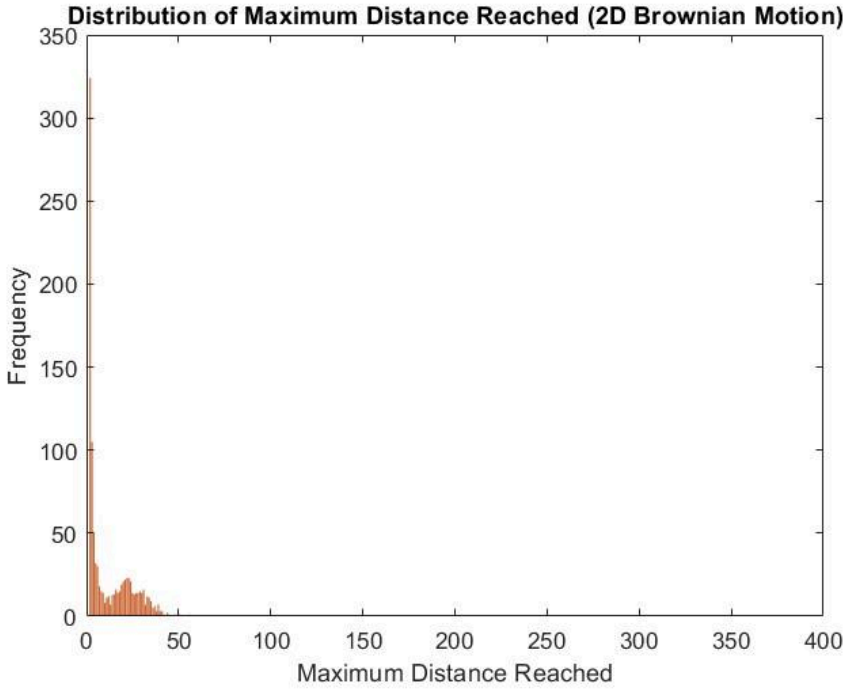
Iteration limit = 200:





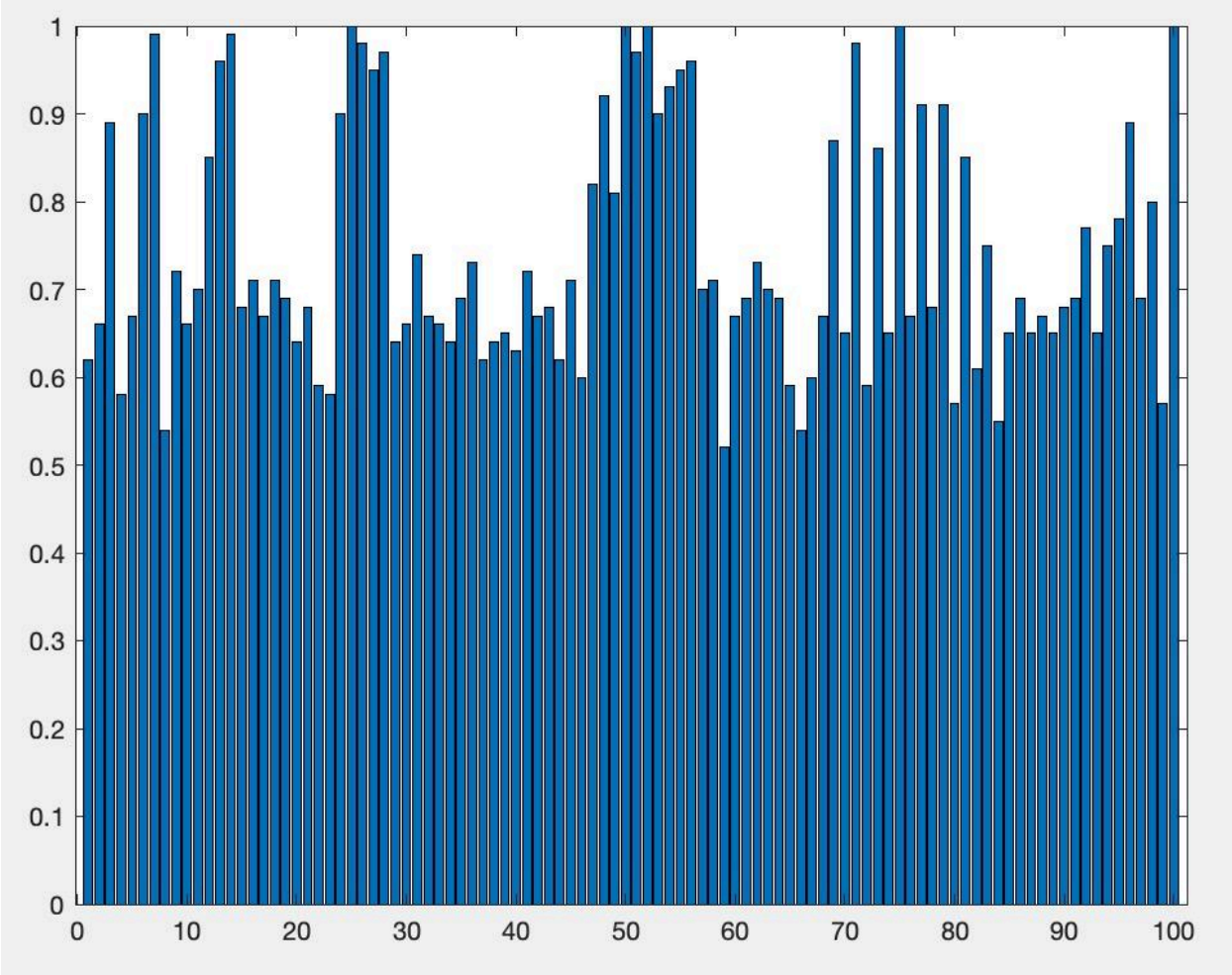
Iteration limit = 400:





Based on these results, the iteration limit does not seem to strongly influence the iteration and distance distributions, as all of the graphs look pretty similar even though the iteration limit is doubling each time.

For A grade functionality, I decided to test the case of 1-dimensional Brownian motion with it skewing to the left or right by a given p percentage. For each p, I use a for loop that goes to 1000. My definition of a successful test is if the number of iterations returned is less than 16,000,000. I chose this number because the highest number of iterations I found in the C-grade section was ~15,000,000. For p = 50, the ratio of successful tests vs total tests was (usually) 1. A couple times while testing, the ratio was 0.999. I think this just means that my upper limit is too small for extreme outliers, not that it's not guaranteed to always return to 0. Other p numbers that (usually) had a ratio of 1 include p = 12.5, 25, 37.5, 62.5, 75, 87.5. These are all number representing ⅛, ¼ , ⅜, etc. For p = 33, the ratio was 0.6500, 0.6750, and 0.6480 showing that ratios are not set in stone, at least not statistically after repeating 1000 times. Below is a bar graph with the x - axis representing the p and the y - axis representing the ratio of successful tests vs total tests. The limit to this graph is that only natural numbers are shown. The p number with the lowest ratio is 59 with the ratio 0.52. It seems that there is always at least a 50% chance of running a successful test on a 1-d scale with a consistent p to change the distance covered.



p	Average iterations required to return to 0 after running 1000 times (upper limit = 1,000,000)	Average iterations required to return to 0 after running 1000 times (upper limit = 16,000,000)
20	2.7080	5.4240e+06
25	302.7640	1.5992e+03
50	949.6560	1.7152e+04
60	2.7220	5.1680e+06

This shows that numbers with a lower success ratio(20, 60) will average very few steps when outliers are not factored in. But that number greatly jumps when bigger outliers are allowed in, of course, but their factor is around 10e3 higher than the numbers with a higher success ratio(25,50) with their upper limit increased as well.

Matlab Code:

C Grade Functionality:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NAME: Josh Derrow & Michael Berry
% JMU-EID: derrowjb & berrymw
% DATE: May 11, 2025
%
% PROGRAM: brownmotion1d.m
% PURPOSE: Simulates one-dimensional brownian motion.
%
% VARIABLES:
%    out = return array containing the total number of iterations simulated
%          and the maximum distance away from the origin reached
%    pos = current position of the particle
%    iter = current iteration of the simulation
%    maxdist = maximum distance away from the origin reached
%
% JMU PLEDGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function out = brownmotion1d()
    out = [0, 0];
    pos = 0;
    iter = 1;
```

```
    maxdist = 1;

    % Initial movement
    if rand() < 0.5
        pos = pos - 1;
    else
        pos = pos + 1;
    end

    % Further movement until the particle returns to the origin
    while pos ~= 0
        if rand() < 0.5
            pos = pos - 1;
        else
            pos = pos + 1;
        end

        iter = iter + 1;

        if abs(pos) > maxdist
            maxdist = abs(pos);
        end
    end

out(1) = iter;
out(2) = maxdist;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NAME: Josh Derrow & Michael Berry
% JMU-EID: derrowjb & berrymw
% DATE: May 11, 2025
%
% PROGRAM: testbrownmotion1d.m
% PURPOSE: Tests the one-dimensional brownian motion simulation.
%
% VARIABLES:
%    iters = an array containing the number of iterations for each run of
%            brownmotion1d
%    dists = an array containing the maximum distance reached for each run
%            of brownmotion1d
%    totaliters = total number of iterations across all calls to
%                 brownmotion1d; used to compute the average
```

```
%    totalmaxdist = total maximum distance reached across all calls to
%                   brownmotion1d; also used to compute the average
%    upperlimit = threshold value to limit the number of iterations of
%                 brownmotion1d
%    k = temporary array that stores the current number of iterations and
%        the current maximum distance reached
%    averageiters = average number of iterations per run
%    averagemax = average maximum distance per run
%    freqiters = an array storing the frequencies of each iteration count
%                (freqiters(i) = the number of times iteration is equal to
%                i (min i is 2))
%    freqdists = an array storing the frequencies of each maximum distance
%                reached (freqdists(i) = the number of times the max
%                distance is i (min i is 1))
%    iteroverthous = a counter for the runs that had over 1000 iterations
%    distoverthous = a counter for the runs that had maximum distances over
%                    1000
%    curriter = temporary current value of iters(i)
%
% JMU PLEDGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

iters = zeros(1, 1000);
dists = zeros(1, 1000);
totaliters = 0;
totalmaxdist = 0;
upperlimit = 20000;

for i = 1:1000
   k = brownmotion1d();

   while k(1) > upperlimit
      k = brownmotion1d();
   end

   iters(i) = k(1);
   dists(i) = k(2);
   totaliters = totaliters + k(1);
   totalmaxdist = totalmaxdist + k(2);
end

averageiters = totaliters / 1000;
averagemax = totalmaxdist / 1000;
```

```
disp(averageiters);
disp(averagemax);

freqiters = zeros(1, 1000);
freqdists = zeros(1, 1000);
iteroverthous = 0;
distoverthous = 0;

for i = 1:1000
   curriter = iters(i);
   if curriter > 1000
      iteroverthous = iteroverthous + 1;
      continue
   end

   currdist = dists(i);
   if currdist > 1000
      distoverthous = distoverthous + 1;
      continue
   end

   freqiters(curriter) = freqiters(curriter) + 1;
   freqdists(currdist) = freqdists(currdist) + 1;
End
bar(iters)
bar(dists)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NAME: Josh Derrow & Michael Berry
% JMU-EID: derrowjb & berrymw
% DATE: May 11, 2025
%
% PROGRAM: stepbrownmotion1d.m
% PURPOSE: Simulates one-dimensional brownian motion with a user-inputted
%          iteration limit.
%
% VARIABLES:
%    n = iteration limit
%    out = return array containing the total number of iterations simulated
%          and the maximum distance away from the origin reached
%    pos = current position of the particle
%
% JMU PLEDGE
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%

function out = stepbrownmotion1d(n)
   pos = 0;

   for i = 1:n
      if rand() < 0.5
         pos = pos - 1;
      else
         pos = pos + 1;
      end
   end

   out = pos;
end
```

B Grade Functionality:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
% NAME: Josh Derrow & Michael Berry
% JMU-EID: derrowjb & berrymw
% DATE: May 11, 2025
%
% PROGRAM: brownmotion2d.m
% PURPOSE: Simulates two-dimensional brownian motion.
%
% VARIABLES:
%    out = return array containing the total number of iterations simulated
%          and the maximum distance away from the origin reached
%    pos = current position of the particle
%    iter = current iteration of the simulation
%    maxdist = maximum Euclidean distance away from the origin reached
%    upperlimit = threshold value to limit the number of iterations of
%                 brownmotion2d
%    r = random value between 0 and 1
%    dist = current Euclidean distance away from the origin
%
% JMU PLEDGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%

function out = brownmotion2d()
   out = [0, 0];
```

```
pos = [0, 0];
iter = 1;
maxdist = 1;
upperlimit = 100;

% Initial movement
r = rand();
if r < 0.25
   pos = pos + [0, 1];  % Move up
elseif r < 0.5
   pos = pos + [0, -1]; % Move down
elseif r < 0.75
   pos = pos + [-1, 0]; % Move left
else
   pos = pos + [1, 0];  % Move right
end

% Further movement until the particle returns to the origin
while (any(pos ~= 0)) && (iter < upperlimit)
   r = rand();
   if r < 0.25
      pos = pos + [0, 1];  % Move up
   elseif r < 0.5
      pos = pos + [0, -1]; % Move down
   elseif r < 0.75
      pos = pos + [-1, 0]; % Move left
   else
      pos = pos + [1, 0];  % Move right
   end

   iter = iter + 1;

   % Euclidean distance from origin
   dist = sqrt(pos(1)^2 + pos(2)^2);
   if dist > maxdist
      maxdist = dist;
   end
end

if iter == upperlimit
   fprintf("Particle did not return to the origin after %d iterations\n", upperlimit);
end

fprintf("Number of iterations simulated: %d\n", iter);
```

```
    fprintf("Maximum distance away from the origin reached: %.3f\n", maxdist);
    out(1) = iter;
    out(2) = maxdist;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NAME: Josh Derrow & Michael Berry
% JMU-EID: derrowjb & berrymw
% DATE: May 11, 2025
%
% PROGRAM: testbrownmotion2d.m
% PURPOSE: Tests the two-dimensional brownian motion simulation.
%
% VARIABLES:
%    iters = an array containing the number of iterations for each run of
%            brownmotion2d
%    dists = an array containing the maximum distance reached for each run
%            of brownmotion2d
%    totaliters = total number of iterations across all calls to
%                 brownmotion2d; used to compute the average
%    totalmaxdist = total maximum distance reached across all calls to
%                   brownmotion2d; also used to compute the average
%    upperlimit = threshold value to limit the number of iterations of
%                 brownmotion2d
%    k = temporary array that stores the current number of iterations and
%        the current maximum distance reached
%    averageiters = average number of iterations per run
%    averagemax = average maximum distance per run
%    freqiters = an array storing the frequencies of each iteration count
%                (freqiters(i) = the number of times iteration is equal to
%                i (min i is 2))
%    freqdists = an array storing the frequencies of each maximum distance
%                reached (freqdists(i) = the number of times the max
%                distance is i (min i is 1))
%    iteroverthous = a counter for the runs that had over 1000 iterations
%    distoverthous = a counter for the runs that had maximum distances over
%                    1000
%    curriter = temporary current value of iters(i)
%
% JMU PLEDGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
iters = zeros(1, 1000);
dists = zeros(1, 1000);
totaliters = 0;
totalmaxdist = 0;
upperlimit = 100000;

for i = 1:1000
    k = brownmotion2d();

    while k(1) > upperlimit
        k = brownmotion2d();
    end

    iters(i) = k(1);
    dists(i) = k(2);
    totaliters = totaliters + k(1);
    totalmaxdist = totalmaxdist + k(2);
end

averageiters = totaliters / 1000;
averagemax = totalmaxdist / 1000;
fprintf("Average number of iterations: %.3f\n", averageiters);
fprintf("Average max distance reached: %.3f\n", averagemax);

iteroverthous = 0;
distoverthous = 0;

for i = 1:1000
    curriter = iters(i);
    if curriter > 1000
        iteroverthous = iteroverthous + 1;
        continue
    end

    currdist = dists(i);
    if currdist > 1000
        distoverthous = distoverthous + 1;
        continue
    end
end

edges = 0:100;
freqdists = histcounts(dists, edges);
freqiters = histcounts(iters, edges);
```

```
% Distribution plotting:
% Iterations
figure;
bar(1:100, freqiters, 'FaceColor', [0.2 0.6 0.8]);
xlim([0 100]);
xlabel('Number of Iterations');
ylabel('Frequency');
title('Distribution of Steps to Return to Origin (2D Brownian Motion)');

% Distances
figure;
bar(1:100, freqdists, 'FaceColor', [0.8 0.4 0.2]);
xlim([0 100]);
xlabel('Maximum Distance Reached');
ylabel('Frequency');
title('Distribution of Maximum Distance Reached (2D Brownian Motion)');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NAME: Josh Derrow & Michael Berry
% JMU-EID: derrowjb & berrymw
% DATE: May 11, 2025
%
% PROGRAM: stepbrownmotion2d.m
% PURPOSE: Simulates two-dimensional brownian motion with a user-inputted
%          iteration limit.
%
% VARIABLES:
%    n = iteration limit
%    out = return array containing the total number of iterations simulated
%          and the maximum distance away from the origin reached
%    pos = current position of the particle
%    iter = current iteration of the simulation
%    maxdist = maximum Euclidean distance away from the origin reached
%    r = random value between 0 and 1
%    dist = current Euclidean distance away from the origin
%
% JMU PLEDGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function out = stepbrownmotion2d(n)
    out = [0, 0];
```

```
pos = [0, 0];
iter = 1;
maxdist = 1;

% Initial movement
r = rand();
if r < 0.25
   pos = pos + [0, 1];  % Move up
elseif r < 0.5
   pos = pos + [0, -1]; % Move down
elseif r < 0.75
   pos = pos + [-1, 0]; % Move left
else
   pos = pos + [1, 0];  % Move right
end

% Further movement until the particle returns to the origin
while (any(pos ~= 0)) && (iter < n)
   r = rand();
   if r < 0.25
      pos = pos + [0, 1];  % Move up
   elseif r < 0.5
      pos = pos + [0, -1]; % Move down
   elseif r < 0.75
      pos = pos + [-1, 0]; % Move left
   else
      pos = pos + [1, 0];  % Move right
   end

   iter = iter + 1;

   % Euclidean distance from origin
   dist = sqrt(pos(1)^2 + pos(2)^2);
   if dist > maxdist
      maxdist = dist;
   end
end

if iter == n
   fprintf("Particle did not return to the origin after %d iterations\n", n);
end

fprintf("Number of iterations simulated: %d (Limit: %d)\n", iter, n);
fprintf("Maximum distance away from the origin reached: %.3f\n", maxdist);
```

```
    out(1) = iter;
    out(2) = maxdist;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NAME: Josh Derrow & Michael Berry
% JMU-EID: derrowjb & berrymw
% DATE: May 11, 2025
%
% PROGRAM: teststepbrownmotion2d.m
% PURPOSE: Tests the two-dimensional brownian motion simulation with an
%          iteration limit.
%
% VARIABLES:
%    iters = an array containing the number of iterations for each run of
%            brownmotion2d
%    dists = an array containing the maximum distance reached for each run
%            of brownmotion2d
%    totaliters = total number of iterations across all calls to
%                 brownmotion2d; used to compute the average
%    totalmaxdist = total maximum distance reached across all calls to
%                   brownmotion2d; also used to compute the average
%    upperlimit = threshold value to limit the number of iterations of
%                 brownmotion2d
%    k = temporary array that stores the current number of iterations and
%        the current maximum distance reached
%    averageiters = average number of iterations per run
%    averagemax = average maximum distance per run
%    freqiters = an array storing the frequencies of each iteration count
%                (freqiters(i) = the number of times iteration is equal to
%                i (min i is 2))
%    freqdists = an array storing the frequencies of each maximum distance
%                reached (freqdists(i) = the number of times the max
%                distance is i (min i is 1))
%    iteroverthous = a counter for the runs that had over 1000 iterations
%    distoverthous = a counter for the runs that had maximum distances over
%                    1000
%    curriter = temporary current value of iters(i)
%
% JMU PLEDGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
iters = zeros(1, 1000);
dists = zeros(1, 1000);
totaliters = 0;
totalmaxdist = 0;
upperlimit = input("Please enter an upper iteration limit: ");

for i = 1:1000
    k = stepbrownmotion2d(upperlimit);

    while k(1) > upperlimit
        k = stepbrownmotion2d(upperlimit);
    end

    iters(i) = k(1);
    dists(i) = k(2);
    totaliters = totaliters + k(1);
    totalmaxdist = totalmaxdist + k(2);
end

averageiters = totaliters / 1000;
averagemax = totalmaxdist / 1000;
fprintf("Average number of iterations: %.3f\n", averageiters);
fprintf("Average max distance reached: %.3f\n", averagemax);

iteroverthous = 0;
distoverthous = 0;

for i = 1:1000
    curriter = iters(i);
    if curriter > 1000
        iteroverthous = iteroverthous + 1;
        continue
    end

    currdist = dists(i);
    if currdist > 1000
        distoverthous = distoverthous + 1;
        continue
    end
end

edges = 0:upperlimit;
freqdists = histcounts(dists, edges);
```

```
freqiters = histcounts(iters, edges);

% Distribution plotting:
% Iterations
figure;
bar(1:upperlimit, freqiters, 'FaceColor', [0.2 0.6 0.8]);
xlim([0 upperlimit]);
xlabel('Number of Iterations');
ylabel('Frequency');
title('Distribution of Steps to Return to Origin (2D Brownian Motion)');

% Distances
figure;
bar(1:upperlimit, freqdists, 'FaceColor', [0.8 0.4 0.2]);
xlim([0 upperlimit]);
xlabel('Maximum Distance Reached');
ylabel('Frequency');
title('Distribution of Maximum Distance Reached (2D Brownian Motion)');
```

A Grade Functionality:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NAME: Josh Derrow & Michael Berry
% JMU-EID: derrowjb & berrymw
% DATE: May 11, 2025
%
% PROGRAM: randpercent.m
% PURPOSE: Simulates one-dimensional brownian motion biased p percent to
%          the left and 100 - p percent to the right.
%
% VARIABLES:
%    p = user-inputted motion bias
%    out = return array containing the total number of iterations simulated
%          and the maximum distance away from the origin reached
%    pos = current position of the particle
%    maxiter = maximum number of iterations limit
%    percent = motion bias as a percent
%    iter = current iteration of the simulation
%    maxdist = maximum distance away from the origin reached
%
% JMU PLEDGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

function out = randpercent(p)
```

```
    pos = 0;
    maxiter = 16000000;
    percent = p / 100;
    iter = 1;
    maxdist = percent;

    % Initial movement
    if rand() < 0.5
        pos = pos - percent;
    else
        pos = pos + percent;
    end

    % Further movement until the particle returns to the origin
    while (pos ~= 0) && (iter < maxiter)
        if rand() < 0.5
            pos = pos - percent;
        else
            pos = pos + percent;
        end

        iter = iter + 1;

        if abs(pos) > maxdist
            maxdist = abs(pos);
        end
    end

    out = [iter, maxdist];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NAME: Josh Derrow & Michael Berry
% JMU-EID: derrowjb & berrymw
% DATE: May 11, 2025
%
% PROGRAM: testrandpercent.m
% PURPOSE: test randpercent.m with specific p's to find their average number of iterations
%required to return to position 0
%
%
% JMU PLEDGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=1000;
```

```
iteration = 0;
for i=1:n
   k=randpercent(20); % here I am testing p=20
   iteration = iteration + k(1);
   disp(i);
end
disp(iteration/n) %average number of iterations
```

Honor Pledge:

All work done on this project is our own and we each did our fair share of the work required for this analysis. We both worked on the analysis document, Michael completed A and C grade functionality (1D Brownian motion & Biased Brownian motion), and Josh completed B grade functionality (2D Brownian motion) and code style formatting.