COSC2500 Assignment 6

Ryan White
s4499039

20th of November 2020

R6.1

The Matlab function given in Appendix 0.1 yields the following values for Monte Carlo Approximation of the area of an n-dimensional ball:

Dimensions	Mean Volume (u^D)	Standard Deviation (u^D)	Approx. Quoted Value (u^D)	Error (u^D)
1	2	0	2	0
2	3.142	0.01273	3.142	0
3	4.186	0.01590	4.189	-0.003
4	4.948	0.01471	4.935	0.013
5	5.274	0.01191	5.264	0.01
6	5.186	0.00839	5.168	0.018
7	4.725	0.006085	4.725	0
8	4.088	0.004138	4.059	0.029
9	3.297	0.002501	3.299	-0.002
10	2.592	0.001588	2.550	0.042

Table 1: Comparison of n-Ball Volume Values using Monte Carlo Integration
Source of quoted values: Wikipedia

where the quoted values were sourced via wikipedia. One can say with 95% confidence that the true value lies within 1.96 standard deviations of the mean. In comparison with the quoted values, all but the 6th, 8th, and 10th dimensional volumes are within 1.96 standard deviations of the mean.

R6.2

a. For the probability $p = 0.7$ of the random walk going up one unit per step, the probability of it reaching the top boundary (as opposed to the bottom boundary) for a range of intervals is

Interval	Probability	Calc. Probability	Error
$[-2, 5]$	0.8162	0.8185	-0.0023
$[-5, 3]$	0.9870	0.9867	0.0003
$[-8, 3]$	0.9988	0.9990	0.0002

Table 2: Probabilities of Biased Random Walks reaching Top Boundary

The above was taken from $n = 10000$ trials, with the code used shown in Appendix 0.2.

b. The mean value for $y(1)$ for step sizes $h = [0.1, 0.01, 0.001]$ and the SDE

$$dy = B_t dt + (9y^2)^{1/3} dB_t$$

are

Step Size	Approx. Value	Error
0.1	0.0068	0.0148
0.01	0.0012	0.0190
0.001	0.0112	0.0180

Table 3: Mean Values for solution $y(1)$ for Stochastic DE with $n = 5000$

As the table implies, step size seems to have little impact on the error (and the solution) of the value for $y(1)$. The code used to model the Euler-Maruyama method is shown in Appendix 0.3.

Appendices

0.1 Matlab Code for R6.1

```
for t = 1:10
    montecarlo(t, 1000)
end

function x = montecarlo(D, N)
    points = rand(D, N);
    for k = 1:1000
        count = 0;
        for n = 1:N
            if sqrt(sum(points(:, n).^2)) <= 1
                count = count + 1;
            end
        end
        ratio = count / N;
        volume(k) = ratio;
    end
    m = mean(volume) * 2^D; s = std(volume);
    formatSpec = ['The function returned a mean volume of %d with a standard '...
        'deviation of %d for %d Dimensions, \n and %d points across %d iterations.'];
    disp(sprintf(formatSpec, m, s, D, N, k));
end
```

0.2 Matlab Code for R6.2a

```
n = 10000;
count = 0;
p = 0.7; q = 1 - p;
int = [-8, 3];
for i = 1:n
    w = 0;
    while w ~= int(2)
```

```
        if rand > p
            w = w - 1;
        else
            w = w + 1;
        end
        if w == int(1)
            break
        end
    end
    if w == int(2)
        count = count + 1;
    end
end
ratio = count / n;
actual = ((q / p)^(abs(int(1))) - 1) / ((q / p)^(abs(int(2)) + abs(int(1))) - 1);
err = ratio - actual;
```

0.3 Matlab Code for R6.2b

```
clear all
h = [0.1, 0.01, 0.001]; n = 5000;
h = h(3);
for k = 1:n
    clear y Bt
    y(1) = 0; Bt = zeros(1, 1/h); dt = 1/h;
    z = randn(dt, 1);
    for i = 2:(dt)
        dBt(1, i) = z(i) * sqrt(h);
        Bt(1, i) = sum(dBt(1:i));
        y(i) = y(i-1) + Bt(1, i-1) * h + (9 * y(i-1)^2)^(1/3) * dBt(1, i-1);
    end
    Y(k) = y(end);
end
sol = mean(Y);
err = std(Y) / sqrt(n);
```