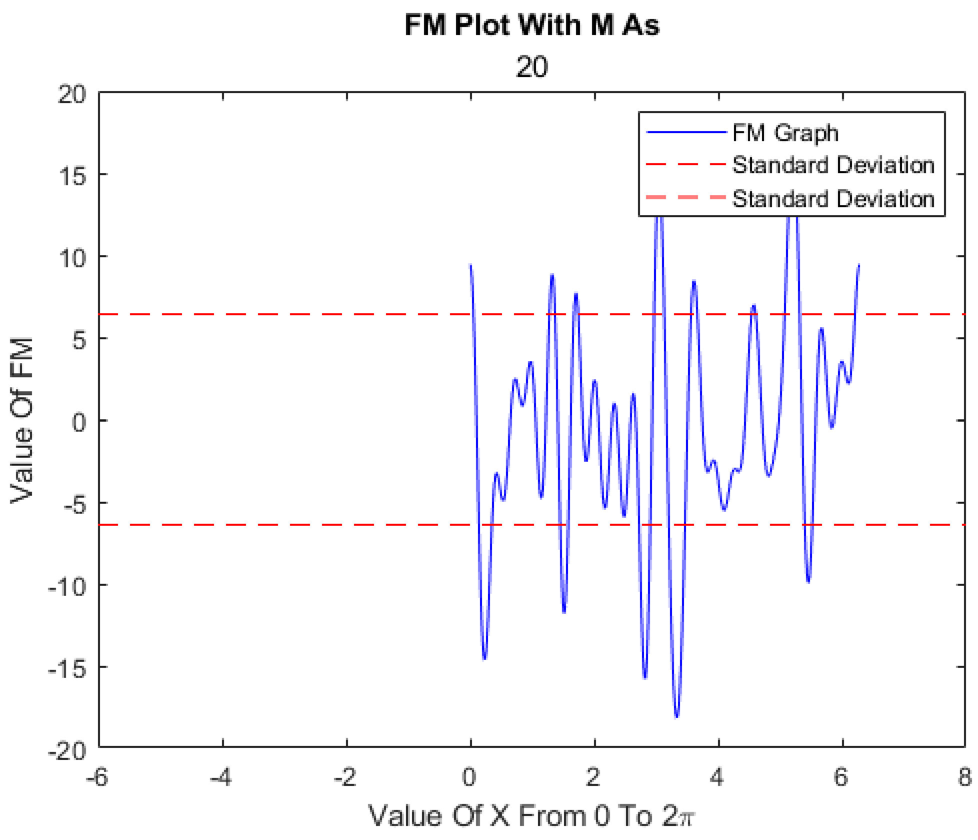
Random Fourier Series

Table of Contents

Exercise 1

In this exercise I have created a function that creates an anonymous function corresponding to the finite random Fourier series. I have then plotted this function with value of $m = 20$ from 0 to 2π. I have also included the standard deviation of this function for convenience. I have ensured that the same anonymous function is reproduced by making sure that Matlab's random number seed is always set to 1. This works as the random numbers I generate to create the anonymous function are reproduced.

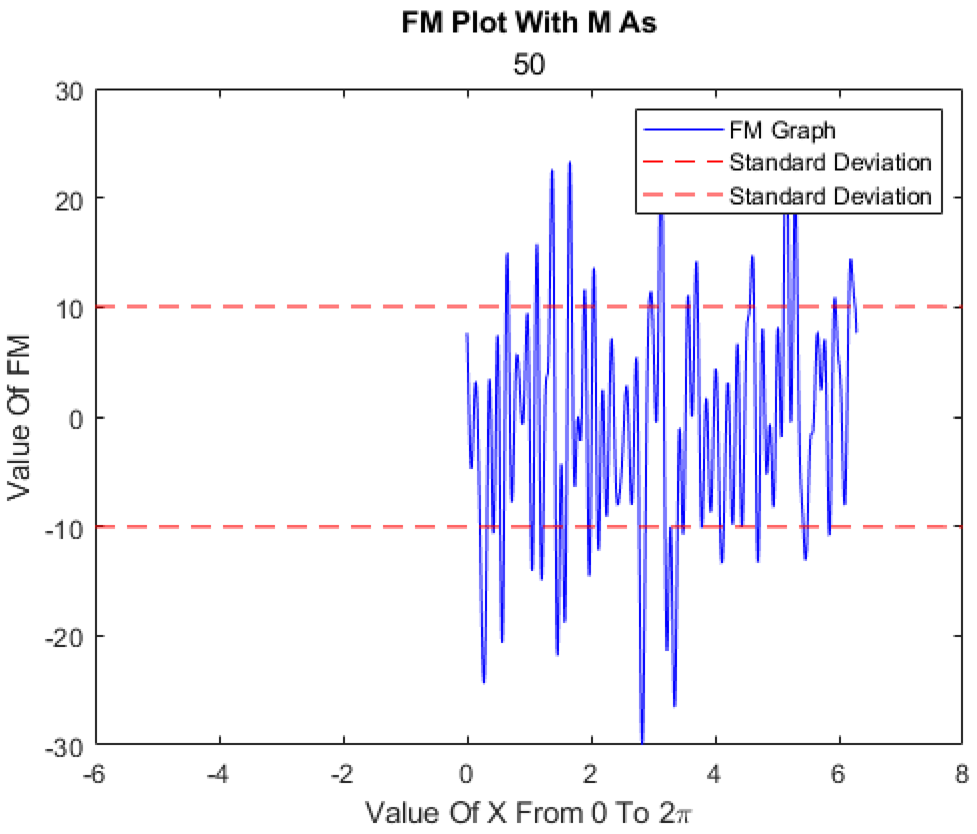
```
fmplot(20)
```

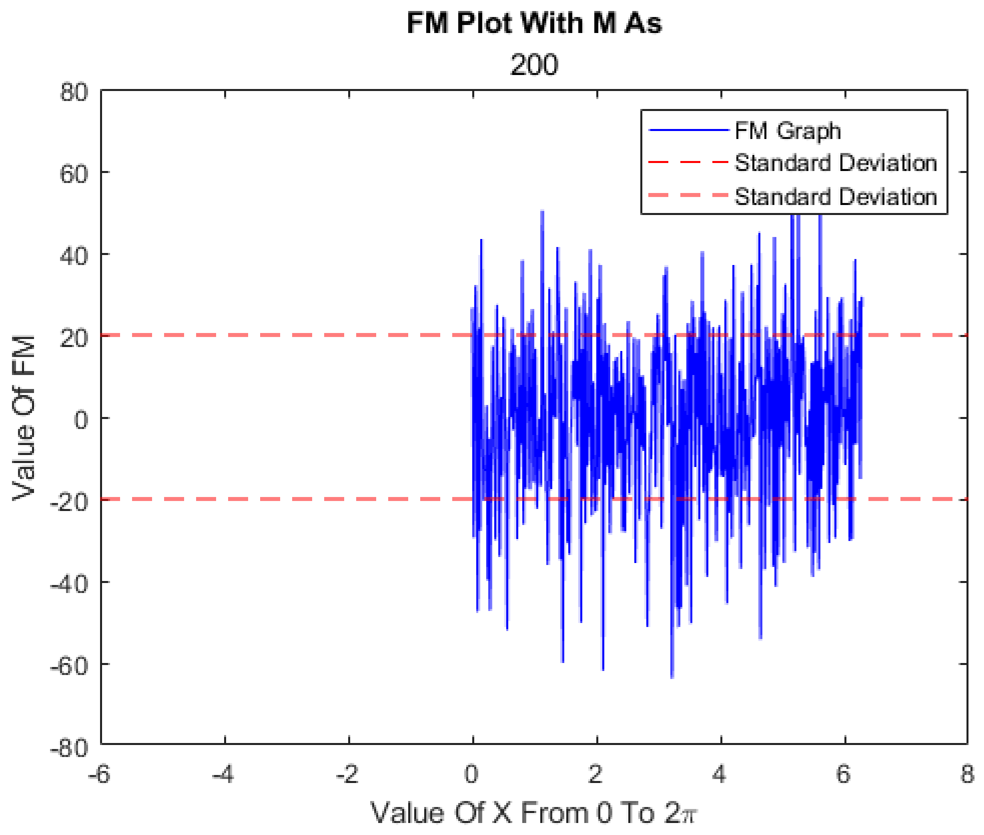


Exercise 2

In this exercise I once again plot my anonmyous function with m = 50 and m = 200. I include the standard deviation again as well. This is to illustrate how $\sigma =$ the square root of $2m + 1$.

```
fmplot(50)
fmplot(200)
```

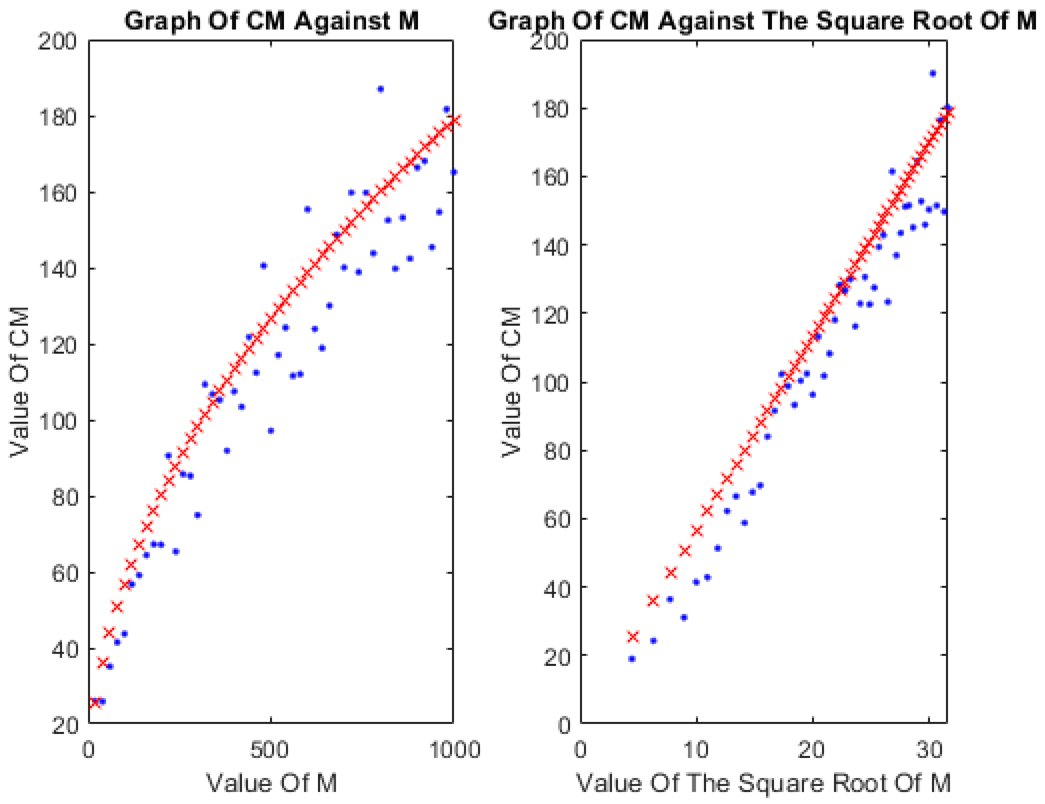




By looking at the graphs we can deduce this standard deviation is correct. This is because as the value of m increases we see that the spread of our smooth function increases. The dashed lines representing the standard deviation contain the bulk portion of our function, which is the values of the smooth function close to the mean.

In this next section I have plotted the maximum absolute value of the smooth function for values of m = 20,40,...,1000 against m on one subplot and against the square root of m on another subplot. On both subplots I have also plotted 4σ against its respective axis.

```
cmplotm()
cmplotsqrtm()
```

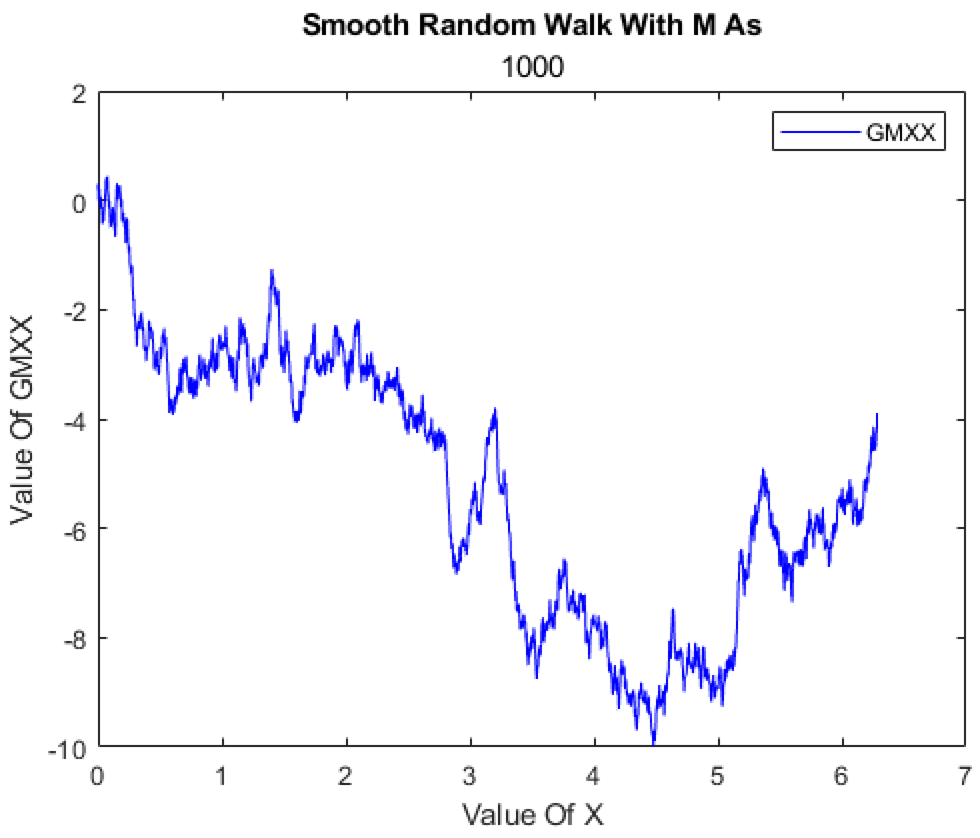


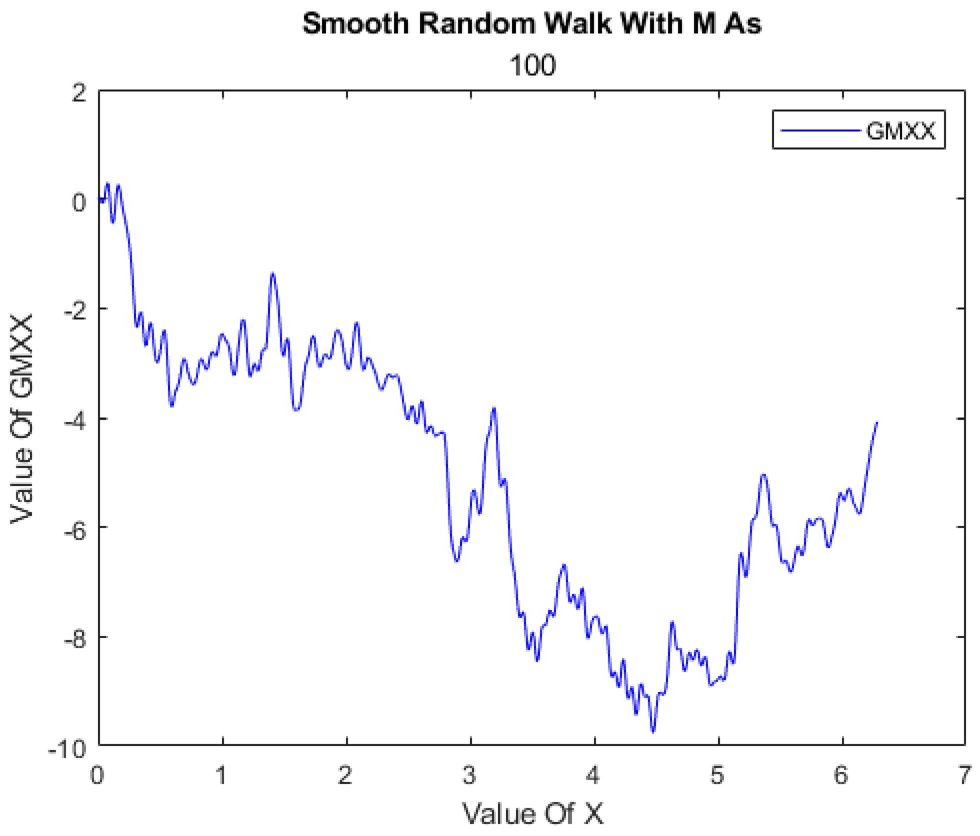
By looking at these two subplots we can see that the maximum absolute value of the smooth function (blue dots) scales with 4σ (red crosses) as m increases. In the second subplot we see there is linear growth. This is to illustrate how the size of the smooth function scales with the square root of m.

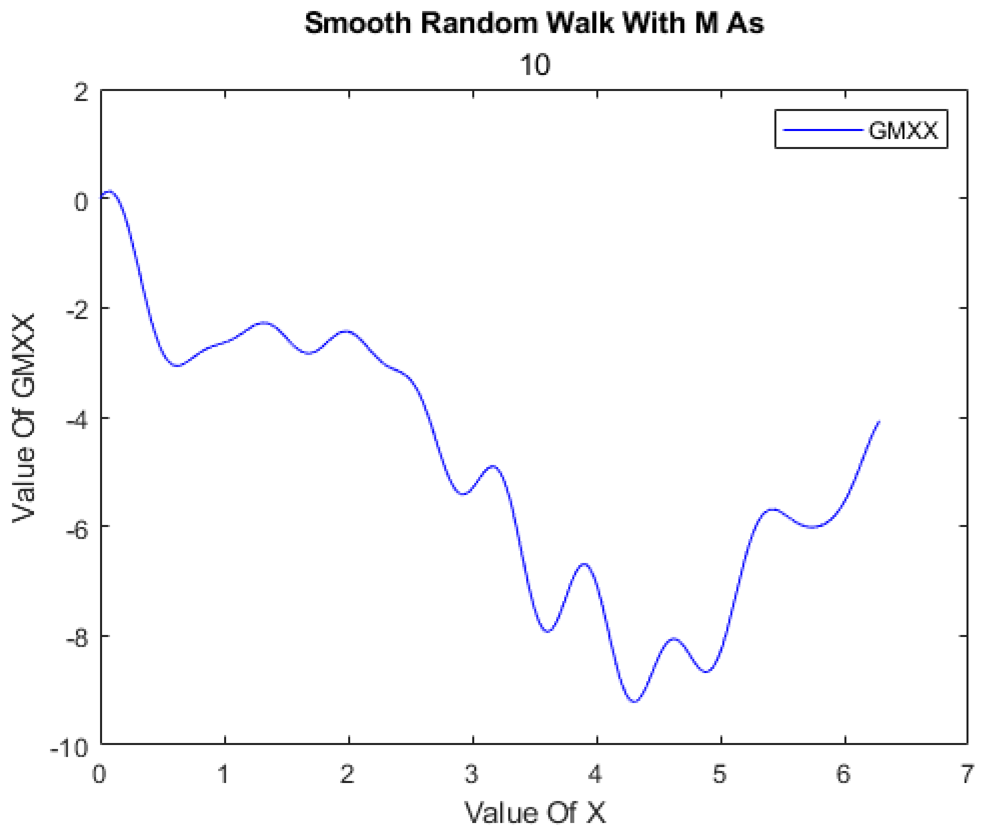
Exercise 3

In this exercise I plot three smooth random walks that approaches a Brownian path as $m \to \infty$. In this case I will plot three random walks where the random number seed is always set to 1 when creating my smooth random function.

```
cumsumplot()
```


Smooth Random Walk With M As
1000
GMXX
Value Of GMXX
Value Of X
2
0
-2
-4
-6
-8
-10
0
1
2
3
4
5
6
7


Smooth Random Walk With M As
100
GMXX
Value Of GMXX
Value Of X
2
0
-2
-4
-6
-8
-10
0
1
2
3
4
5
6
7

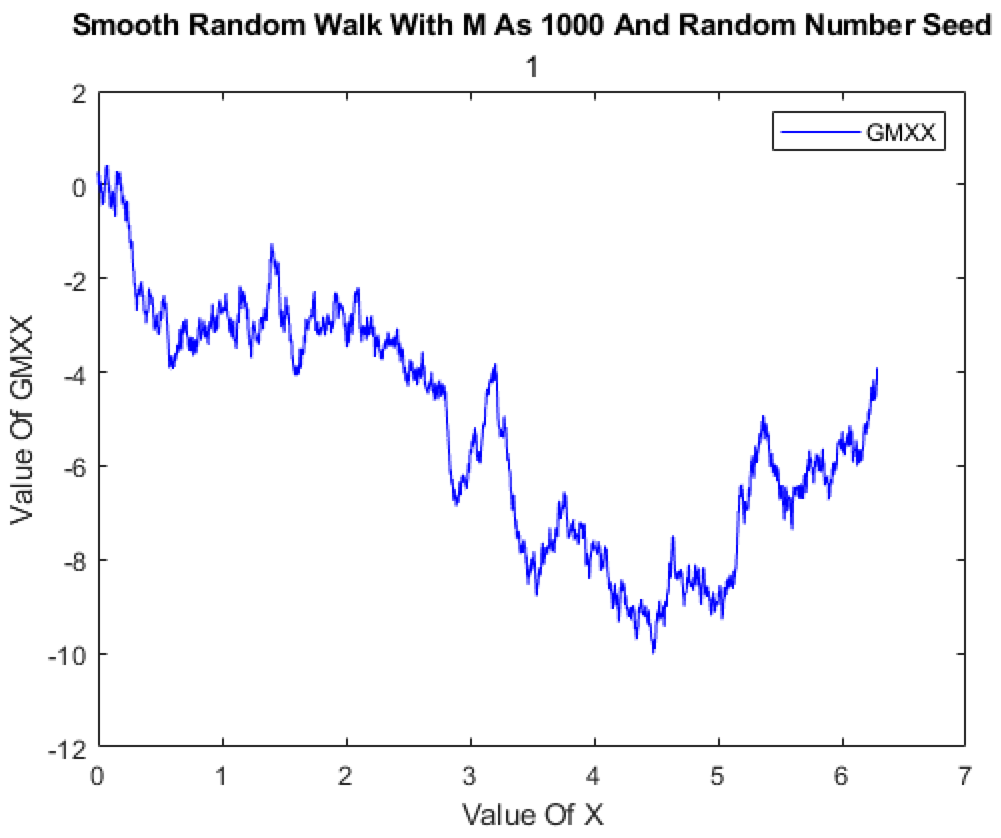


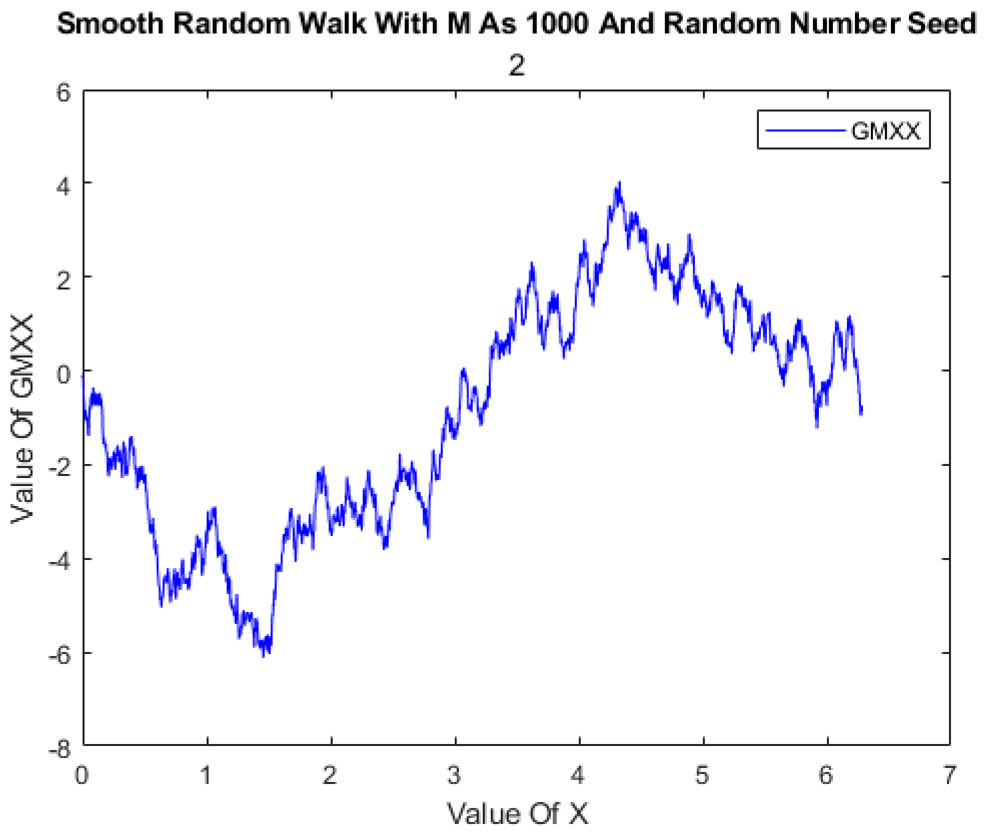
By looking at the graphs we can see that despite having different values of m they all produce a graph where the shape of the graphs are similar. This is because since the random number seed is always the same, the first portion of the smooth random functions is the same to some degree. We see that with a higher value of m the smooth random walk is more defined and that the graph becomes less smooth.

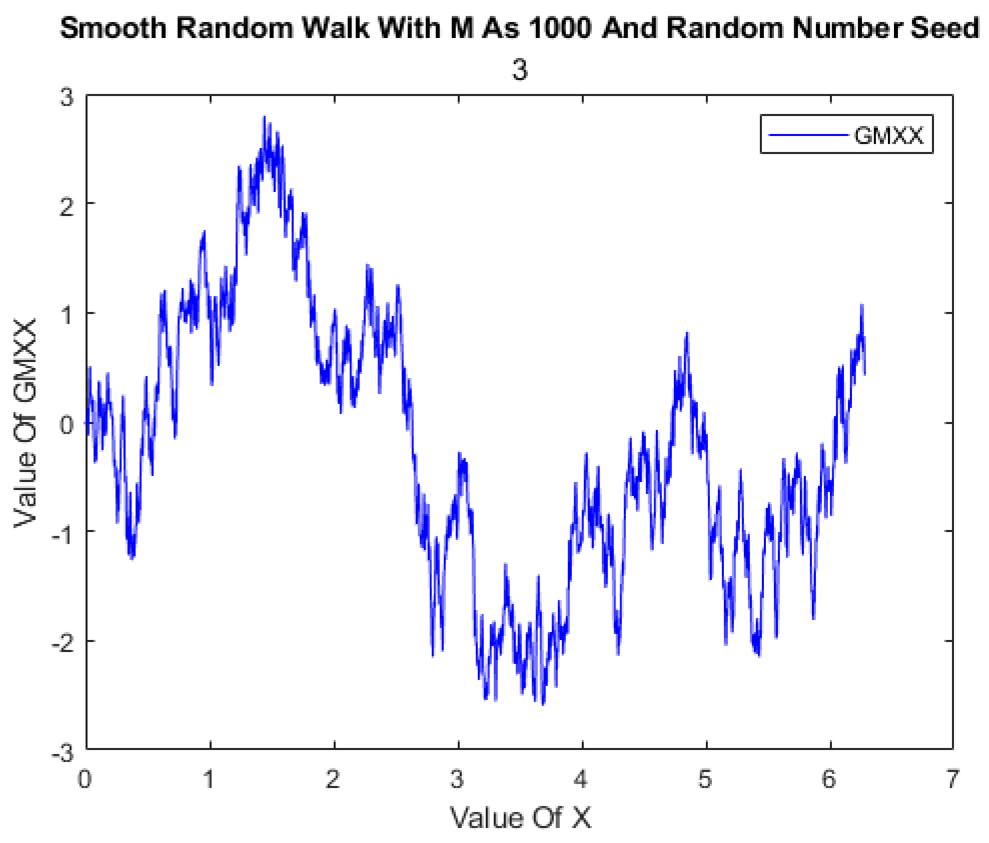
Exercise 4

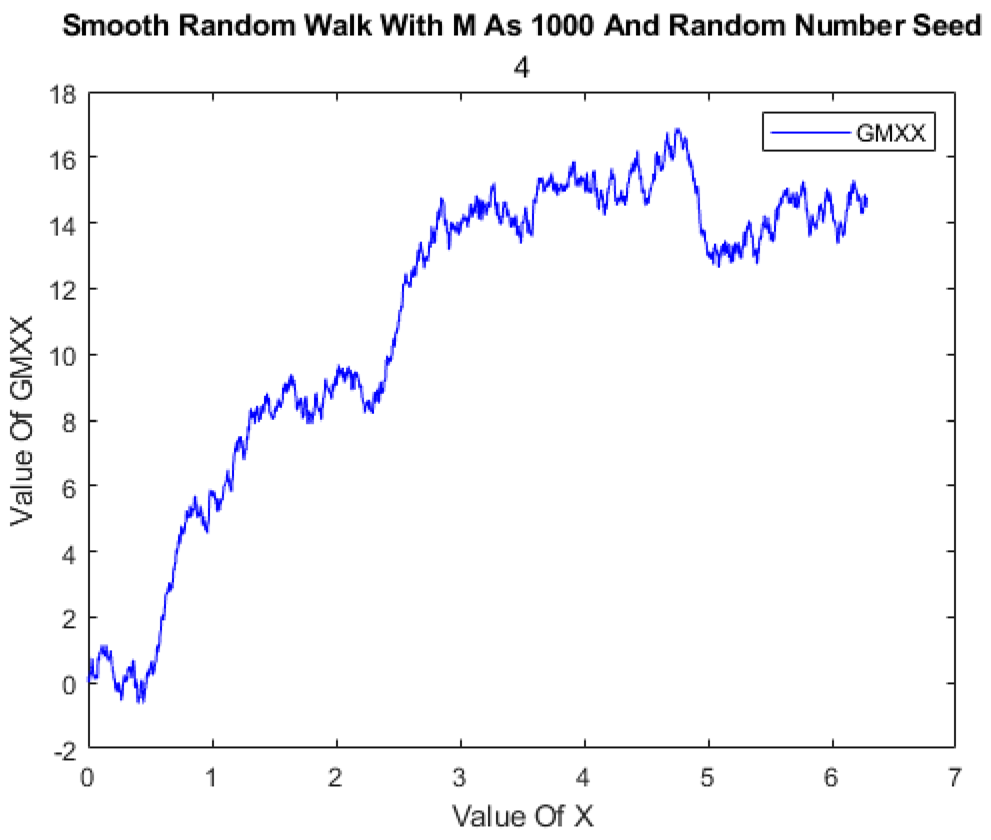
In this exercise I have created 5 different Brownian paths with $m = 1024$ and making sure this time that the random number seed is not reinitialised. The concept behind is that Brownian paths are meant to model random behaviour that evolves after time.

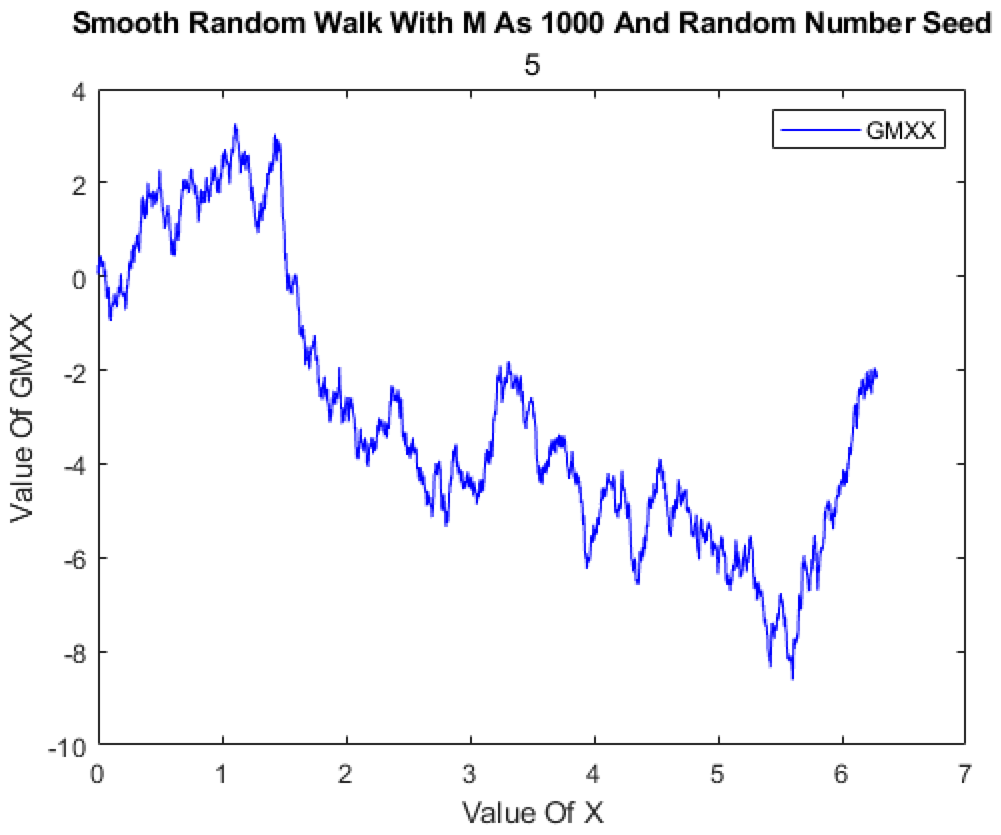
```
fivebrownpaths()
```


Smooth Random Walk With M As 1000 And Random Number Seed
1
GMXX
Value Of GMXX
Value Of X
2
0
-2
-4
-6
-8
-10
-12
0
1
2
3
4
5
6
7


Smooth Random Walk With M As 1000 And Random Number Seed
2
GMXX
Value Of GMXX
Value Of X
6
4
2
0
-2
-4
-6
-8
0
1
2
3
4
5
6
7


Smooth Random Walk With M As 1000 And Random Number Seed
3
GMXX
Value Of GMXX
Value Of X
3
2
1
0
-1
-2
-3
0
1
2
3
4
5
6
7

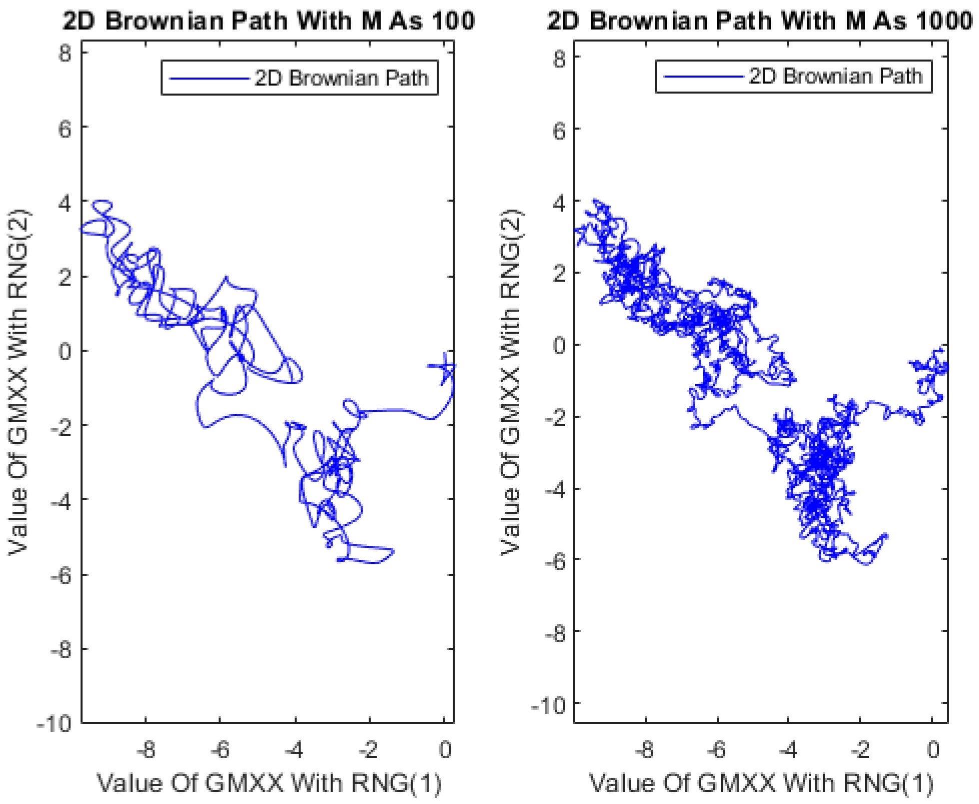

Smooth Random Walk With M As 1000 And Random Number Seed
4
GMXX
Value Of GMXX
Value Of X
18
16
14
12
10
8
6
4
2
0
-2
0
1
2
3
4
5
6
7



By looking at the graphs we can see that this time the different graphs do not resemble each other. This is because since the random number seed is different, the smooth random functions are completely different. Comparing these graphs and the ones from before, we can note that the Brownian paths start from 0, which is an axiom for Brownian motion.

In this next section I illustrate $2D$ Brownian paths. This is done by plotting a smooth random walk with RNG(1) against another with RNG(2). I have produced one subplot with $m = 100$ and another with $m = 1000$.

```
twodbrownianpaths100()
twodbrownianpaths1000()
```



Looking at these graphs we can see they both produce a similar shape. The second subplot is more detailed and filled in as it uses a higher value of m. The plots look weird and this highlights how Brownian motion is meant to be truly random.

Description Of Functions

This section shows the code written to create the functions that I have used to complete the exercises.

```
type smooth.m
type fmplot.m
type cmplotm.m
type cmplotsqrtm.m
type cumsumplot.m
type fivebrownpaths.m
type twodbrownianpaths100.m
type twodbrownianpaths1000.m


% This function produces an anomynous function fm corresponding to the
% smooth random function defined by the random Fourier series. This
% function takes an integer as an argument. It then computes independent
% samples of the standard normal distribution and then uses these values to
% create the anonymous function using the Fourier series formula.
function fm = smooth(m) % Takes a value m
r = randn(2*m+1,1); % Generates the random samples in an array
fm = @(x) 0; % Defining the function
fm = @(x) fm(x) + r(1); % Appends the function with our first value
count = 1; % A variable to keep check of the index we are on
for j = 2:2*m + 1 % A for loop to compute the series
    if mod(j,2) == 0 % Checks if we are using the a or b value
        fm = @(x) fm(x) + (2.^0.5 * r(j) * cos(count*x));
        % Appends the function
    else
        fm = @(x) fm(x) + (2.^0.5 * r(j) * sin(count*x));
        count = count + 1; % Updates what index we are on
    end
end
fm = @(x)fm(x);
end

% This function plots the anonymous function that we created using 5000
% values equally spaced in the interval 0 to 2pi. It takes an argument to
% determine what anonymous function it should produce. This function also
% reproduces the same graph for the same value of m as well as illustrating
% numerically the standard deviation for a value of m.
function fmplot(m)
npts = 5000;
xx = linspace(0,2*pi,npts); % Creates the values of x
seed = 1; rng(seed), fm = smooth(m); % Generates the anonymous function
% The seed ensures that the same anonmyous function is produced for the
% same value of m
figure(m); % Defines which figure to plot this graph
plot(xx,fm(xx),'Color','blue','DisplayName','FM Graph')
% Plots fm against xx
hold on
t = num2str(m);
y = (2*m + 1)^0.5; % Calculates standard deviation
fplot(y,'--','Color','red','DisplayName','Standard Deviation')
fplot(-y,'--','Color','red','DisplayName','Standard Deviation')
title('FM Plot With M As', t)
xlabel('Value Of X From 0 To 2\pi')
ylabel('Value Of FM')
legend()
% Appropriate labelling
end

% This function plots the highest absolute value of the smooth function fm
% for each value of m against m.
function cmplotm()
npts = 5000;
xx = linspace(0,2*pi,npts);
% Generates enough data points to have a clear idea of what is our highest
% absolute value for fm
for m = 20:20:1000 % Uses a for loop to increment our values of m by 20
    % from 20 to 1000
    fm = smooth(m); % Generates our smooth function
    cm = max(abs(fm(xx)));
    % Finds the highest absolute value of our function
    figure(1000);
    subplot(1,2,1); % Our plot will be on a subplot
    plot(m,cm,'.','Color','blue') % Plots our value against m
    hold on
    y = 4*(2*m + 1)^0.5;
    plot(m,y,'x','Color','red') % Plot 4 times the standard deviation for
    % each value of m for comparison
end
title('Graph Of CM Against M')
xlabel('Value Of M')
ylabel('Value Of CM')
% Appropriate labels
end

% This function plots the highest absolute value of the smooth function fm
% for each value of m against the square root of m.
function cmplotsqrtm()
npts = 5000;
xx = linspace(0,2*pi,npts);
for m = 20:20:1000
    fm = smooth(m);
    cm = max(abs(fm(xx)));
    figure(1000);
    subplot(1,2,2); % Plots the graph right next to our other subplot
    plot(m^0.5,cm,'.','Color','blue')
    % Plots the value against the square root of m
    hold on
    y = 4*(2*m + 1)^0.5;
    plot(m^0.5,y,'x','Color','red')
end
title('Graph Of CM Against The Square Root Of M')
xlabel('Value Of The Square Root Of M')
ylabel('Value Of CM')
% Appropriate labels
end

% This function plots the indefinite integral of our smooth function from 0
% to x against x. It makes sure to use the same random number seed to
% produce three different smooth functions with differernt values of m. It
% plots three different smooth random walks respective to our three
% different smooth functions.
function cumsumplot()
npts = 5000;
xx = linspace(0,2*pi,npts);
% Generates points for x
for i = 1:3 % A for loop to produce our three different functions
    seed = 1; rng(seed), fm = smooth(10^(4-i));
    % Produces the smooth function with the same random number generator
    figure(i); % Plots on different figures
    gmxx = @(fm) (2*pi/npts)*cumsum(fm(xx));
    % Function for the indefinite integral
    plot(xx,gmxx(fm),'Color','blue','DisplayName','GMXX') % Plots the smooth
 random walk
    t = num2str(10^(4-i));
    title('Smooth Random Walk With M As', t)
    xlabel('Value Of X')
    ylabel('Value Of GMXX')
    legend()
    % Appropriate labels
end
end

% This function plot five Brownain paths with the same value of m but
% different random number seeds
function fivebrownpaths()
npts = 5000;
xx = linspace(0,2*pi,npts);
for i = 1:5 % For loop for our five different plots
    seed = i; rng(seed),fm = smooth(1024);
    % Generates our random function, making sure to change the random
    % number seed
    figure(i);
    gmxx = @(fm) (2*pi/npts)*cumsum(fm(xx));
    plot(xx,gmxx(fm),'Color','blue','DisplayName','GMXX')
    % Plots our Brownian path
    t = num2str(i);
    title('Smooth Random Walk With M As 1000 And Random Number Seed', t)
    xlabel('Value Of X')
    ylabel('Value Of GMXX')
    legend()
    % Appropriate labels
end
end

% This function illustrates 2D Brownian paths. It plots gmxx with rng(1)
% against gmxx with rng(2) with fm taking the value of 100
function twodbrownianpaths100()
npts = 5000;
xx = linspace(0,2*pi,npts);
seed = 1; rng(seed), fm1 = smooth(100);
seed = 2; rng(seed), fm2 = smooth(100);
% Our two different smooth functions
gmxx = @(fm) (2*pi/npts)*cumsum(fm(xx));
% Function for gmxx
figure(999);
subplot(1,2,1);
plot(gmxx(fm1),gmxx(fm2),'Color','blue','DisplayName','2D Brownian Path')
% Plots gmxx with rng(1) against gmxx with rng(2)
axis equal
% Makes the x and y components equally scaled
title('2D Brownian Path With M As 100')
xlabel('Value Of GMXX With RNG(1)')
ylabel('Value Of GMXX With RNG(2)')
legend()
end

% This function illustrates 2D Brownian paths. It plots gmxx with rng(1)
% against gmxx with rng(2) with fm taking the value of 1000
function twodbrownianpaths1000()
npts = 5000;
```

```
xx = linspace(0,2*pi,npts);
seed = 1; rng(seed), fm1 = smooth(1000);
seed = 2; rng(seed), fm2 = smooth(1000);
gmxx1 = @(fm) (2*pi/npts)*cumsum(fm(xx));
figure(999);
subplot(1,2,2);
plot(gmxx1(fm1),gmxx1(fm2),'Color','blue','DisplayName','2D Brownian Path')
axis equal
title('2D Brownian Path With M As 1000')
xlabel('Value Of GMXX With RNG(1)')
ylabel('Value Of GMXX With RNG(2)')
legend()
end
```

Published with MATLAB® R2021b