Random Fourier Series

Table of Contents

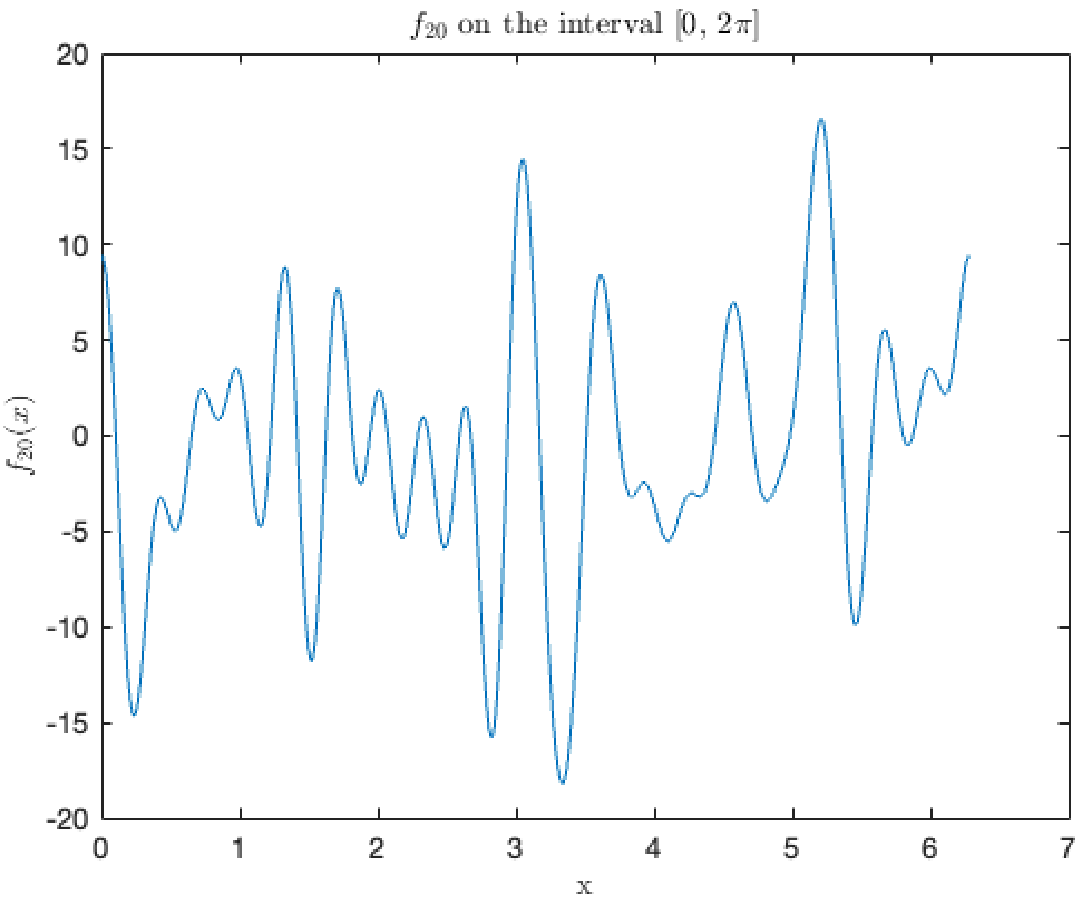
Exercise A1

In this part we create the function smooth(m) which generates what's called a smooth random function f_m using a random fourier series truncated to m terms. The random fourier series is created by sampling coefficients a_i ($0 \leq i \leq m$), b_j ($1 \leq j \leq m$) from the standard normal distribution. We test this function by plotting f_{20} in the range $[0, 2\pi]$. The function produced seems fairly 'random'. It oscillates about $f_m = 0$, with no discernible trend up or down.

```
figure(1) % set the figure

% Test the smooth function by plotting smooth(20) for 5000 points in [0, 2pi].
% We use a fixed rng seed so the same function will be generated each time.
npts = 5000;
xx = linspace(0,2*pi,npts);
seed = 1; rng(seed), fm = smooth(20);
plot(xx,fm(xx))

% Format the plot
title("$f_{20}$ on the interval [0, 2$\pi$]", 'Interpreter','latex')
xlabel('x', 'Interpreter','latex')
ylabel('$f_{20}(x)$', 'Interpreter','latex')
set(gca, 'fontsize', 13)

% Output the smooth function
type smooth.m
```

```
% This function takes m >= 1 as an input,and returns an anonymous function,
% which is a finite random Fourier series.
function fm = smooth(m)

    % First generate the random constants a_i, b_j in the order
    % a_0, a_1, b_1, ..., a_m, b_m.
    % These are drawn from the standard normal distribution.
    ao = randn;
    a_b = randn([2 m]); % generate a 2xm matrix of random numbers

    a = a_b(1,:); % take the first row for the a_is
    b = a_b(2,:); % second row for the b_js
```

```
    % Create the anonymous function for the fourier series, fm
    fm = @(x)ao;

    % We generate the fourier series iteratively by adding on each
    % sin(jx), cos(jx) term, one at a time.
    for j = 1:m
        fm = @(x)(fm(x) + sqrt(2) * (a(j)*cos(j*x) + b(j)*sin(j*x)));
    end
end
```

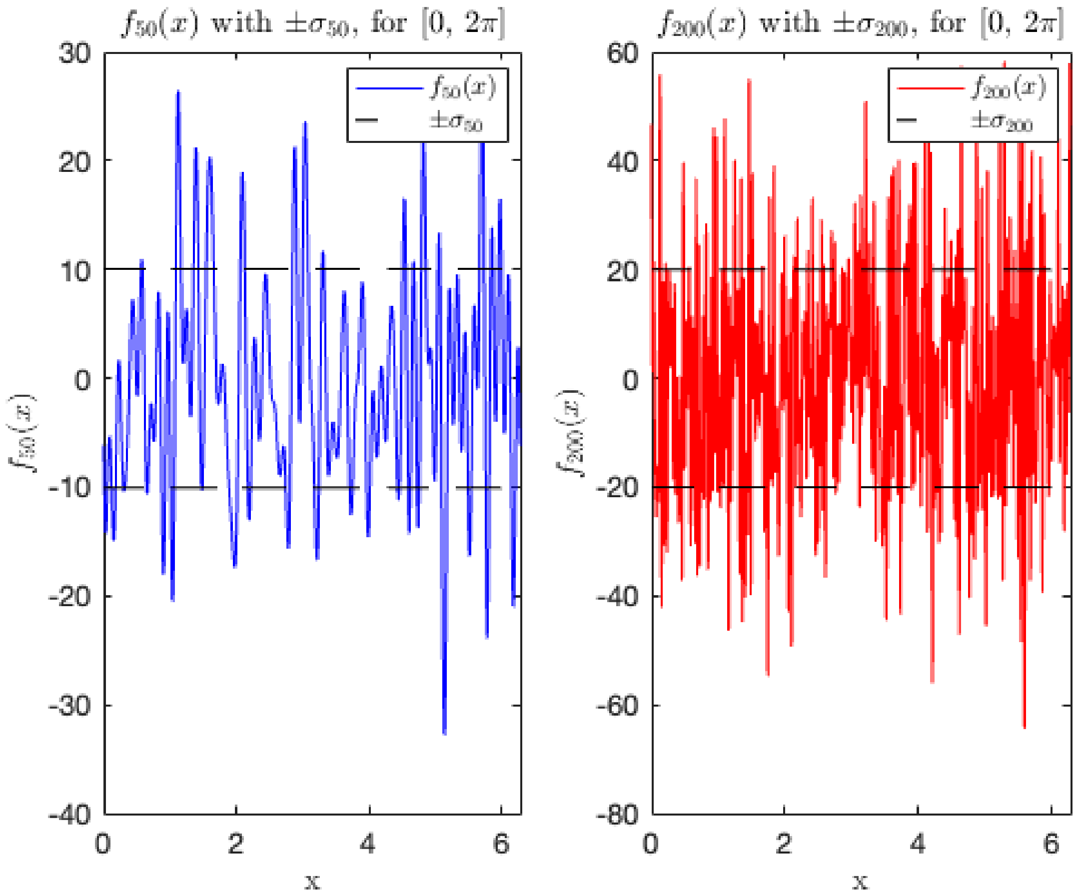


Exercise A2

In this part we treat f_m as a random variable, and investigate it's distribution. We're told the standard deviation of this distribution is $\sigma_m = \sqrt{2m+1}$. First we plot f_m for $m = 50, m = 200$ in the range $[0, 2\pi]$. We also add standard deviation to each plot. The two graphs suggest that this is an appropriate formula for the standard deviation, as the functions are mostly between $\pm\sigma_m$, with only a few outliers.

```
% Create an anonymous function which gives the standard deviation of f_m.
sigma = @(m) sqrt(2*m + 1)
```

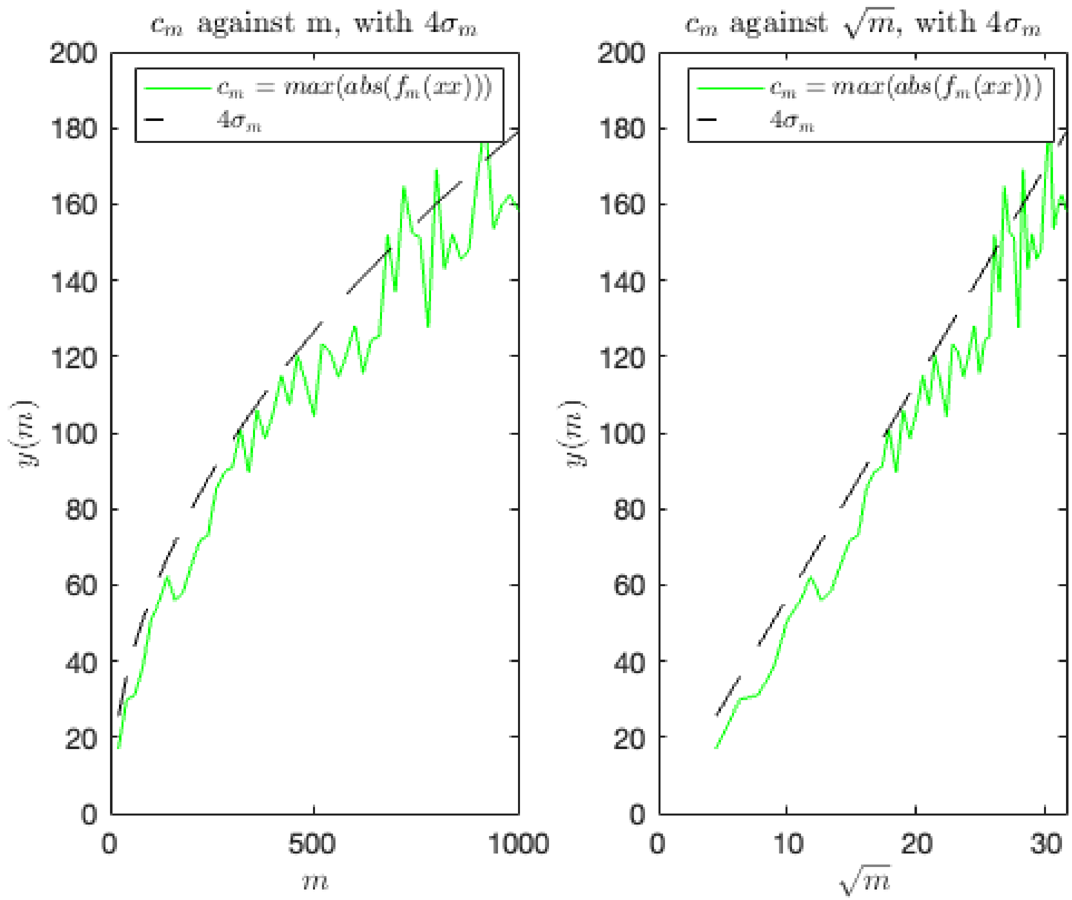
```
% Plot f_50 and f_200.
f_50 = smooth(50);
f_200 = smooth(200);

figure(2) % switch to a new figure

% --- Plot f_50
subplot(1,2,1) % set the subplot
plot(xx, f_50(xx), 'b') % plot f_50
hold on % plot +- sigma_50 as black dashed lines
fplot(sigma(50), [0, 2*pi], 'Color', 'k', 'LineStyle', '--')
fplot(-sigma(50), [0, 2*pi], 'Color', 'k', 'LineStyle', '--')

% Format the plot
title('$f_{50}(x)$ with $\pm\sigma_{50}$, for [0, 2$\pi
$]', 'Interpreter','latex')
xlabel('x', 'Interpreter','latex'), ylabel('$f_{50}
(x)$', 'Interpreter','latex')
set(gca, 'fontsize', 13)
legend('$f_{50}(x)$', '$\pm\sigma_{50}$', 'Interpreter','latex')

% --- Plot f_200
subplot(1,2,2) % set the subplot
plot(xx, f_200(xx), 'r') % plot f_200
hold on % plot +- sigma_200 as black dashed lines
fplot(sigma(200), [0, 2*pi], 'Color', 'k', 'LineStyle', '--')
fplot(-sigma(200), [0, 2*pi], 'Color', 'k', 'LineStyle', '--')

% Format the plot
title('$f_{200}(x)$ with $\pm\sigma_{200}$, for [0, 2$\pi
$]', 'Interpreter','latex')
xlabel('x', 'Interpreter','latex'), ylabel('$f_{200}
(x)$', 'Interpreter','latex')
set(gca, 'fontsize', 13)
legend('$f_{200}(x)$', '$\pm\sigma_{200}$', 'Interpreter','latex')


sigma =

  function_handle with value:

    @(m)sqrt(2*m+1)
```



Now we investigate the size of f_m: c_m (the random variable who's value is the max displacement of f_m from 0, in $[0, 2\pi]$). Specifically we look at how c_m varies with m. We plot c_m for $m = 20, 40, ..., 1000$, along with the curve $4\sigma_m$. We use both m, and $\sqrt{m}$ on the horizontal axis. In both cases we see that c_m fairly closely tracks the curve $4\sigma_m$, with only small osscilations around it. This suggests that the expectation of c_m is approximately $4\sigma_m$, and that the standard deviation of c_m is fairly small. This relationship between max value and standard deviation for f_m is quite suprising. (This code can take ~ 15 seconds to run).

```
figure(3)

% Respectively these arrays store:
ms = []; % {m : 20, 40, ..., 1000}
root_ms = []; % {sqrt(m): 20, 40, ..., 1000}
cms = []; % cm for each m

% Loop through m = 20, 40, ..., 1000
index = 1; % index variable for adding values to the above arrays
for m = linspace(20, 1000, 50)
    % generate fm
    fm = smooth(m);

    % find cm as the max deviation of fm from 0
    cm = max(abs(fm(xx)));
```

```
    % save m, sqrt(m), cm for plotting
    ms(index) = m;
    root_ms(index) = sqrt(m);
    cms(index) = cm;

    % incriment the indexing variable
    index = index + 1;
end

% --- Plot the graphs
% Start with cm against m
subplot(1,2,1) % set the subplot
plot(ms, cms, 'g') % plot cm against m
hold on % then plot 4*sigma_m
plot(ms, 4*sigma(ms), 'Color', 'k', 'LineStyle', '--')

% Format the plot
xlabel('$m$', 'Interpreter','latex')
ylabel('$y(m)$', 'Interpreter','latex')
title('$c_m$ against m, with $4\sigma_m$', 'Interpreter','latex')
legend('$c_m$ = $max(abs(f_m(xx)))$', '$4\sigma_{m}$','Interpreter','latex')
set(gca, 'fontsize', 13)

% Now plot cm against sqrt(m)
subplot(1,2,2) % set the subplot
plot(root_ms, cms, 'g') % plot cm against sqrt(m)
hold on % then plot 4*sigma_m
plot(root_ms, 4*sigma(ms), 'Color', 'k', 'LineStyle', '--')

% Format the plot
xlabel('$\sqrt{m}$','Interpreter','latex')
ylabel('$y(m)$', 'Interpreter','latex')
title('$c_m$ against $\sqrt{m}$, with $4\sigma_m$', 'Interpreter','latex')
legend('$c_m = max(abs(f_m(xx)))$', '$4\sigma_m$','Interpreter','latex')
set(gca, 'fontsize', 13)
```



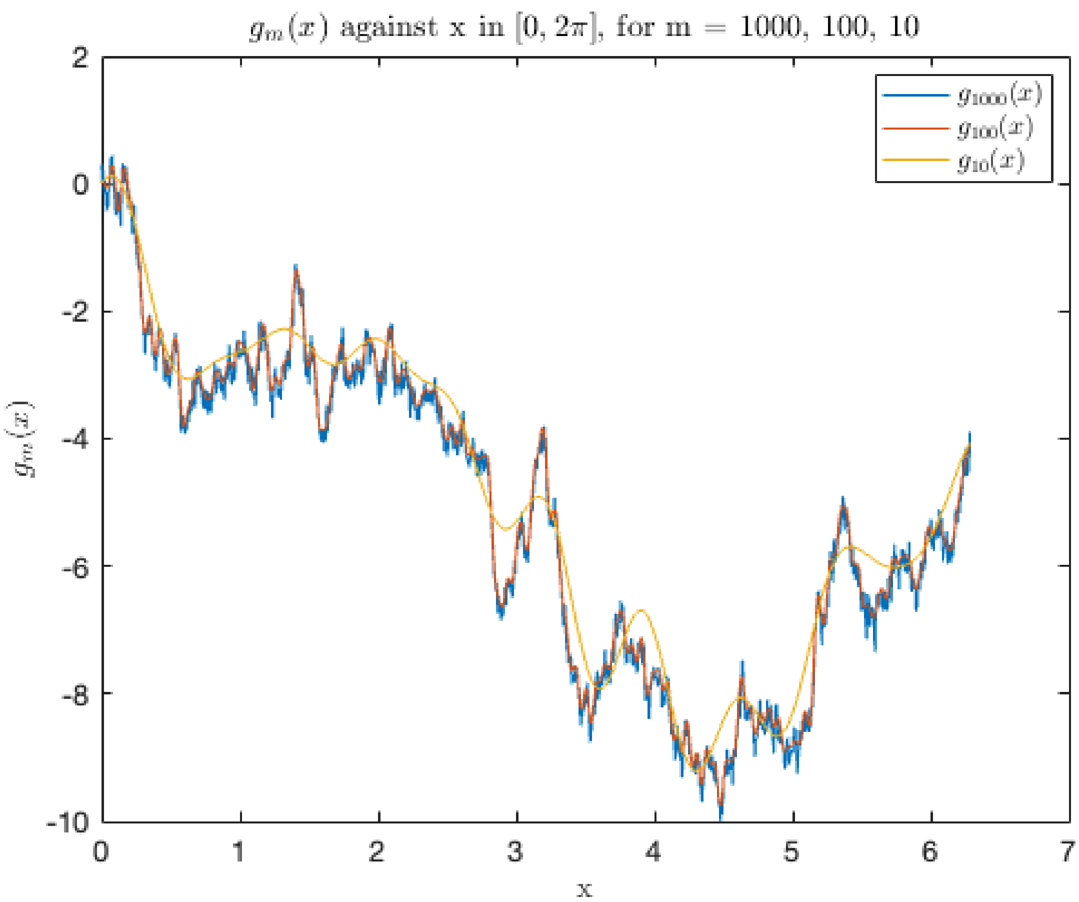
The plot against m shows that the rate of increase of c_m, decreases with m, and assuming $4\sigma_m$ is the expected value of c_m, as m tends towards infinity, so does c_m. The plot against $\sqrt{m}$ approximately gives a linear relationship for $4\sigma_m$ as $\sqrt{2m+1} \approx \sqrt{2m}$ for large, positive m. This shows that we can expect the max value reached by f_m in $[0, 2\pi]$ to scale linearly with $\sqrt{m}$. Thus, adding more terms to the random fourier series that defines f_m, causes larger oscillations. This is not immediately obvious as you could expect that adding more terms would smooth out f_m.

Exercise A3

In this part we investigate g_m (ie. the integral of f_m over $[0, x]$). This is called a smooth random walk. As m tends to infinity, g_m approaches a Brownian path. We plot g_m for $m = 10, 100, 1000$.

```
% We create an anonymous function gmxx which approximates g_m
% for x in [0, 2pi], for a given f_m. The trapezium rule is used
% to approximate the integral g_m. The statement cumsum(fm(xx))
% returns a vector of the cumulative sums of heights of the graph.
% So we multiply by the density of the grid xx, which is
% 2pi/npts, to get the cumulative area.
gmxx = @(fm) (2*pi/npts)*cumsum(fm(xx));

% Now we plot gmxx for m as decreasing powers of 10 (1000, 100, 10).
```

```
figure(4)
for k = 1:3
    rng(1) % reset the rng seed;
    f = smooth(10 ^ (4-k));

    plot(xx, gmxx(f))
    hold on
end

% Format the plot
xlabel('x', 'Interpreter', 'latex')
ylabel('$g_m(x)$', 'Interpreter', 'latex')
title('$g_m(x)$ against x in $[0, 2\pi]$, for m = 1000, 100,
 10', 'Interpreter', 'latex')
legend('$g_{1000}(x)$', '$g_{100}(x)$', '$g_{10}(x)$', 'Interpreter', 'latex')
set(gca, 'fontsize', 13)
```



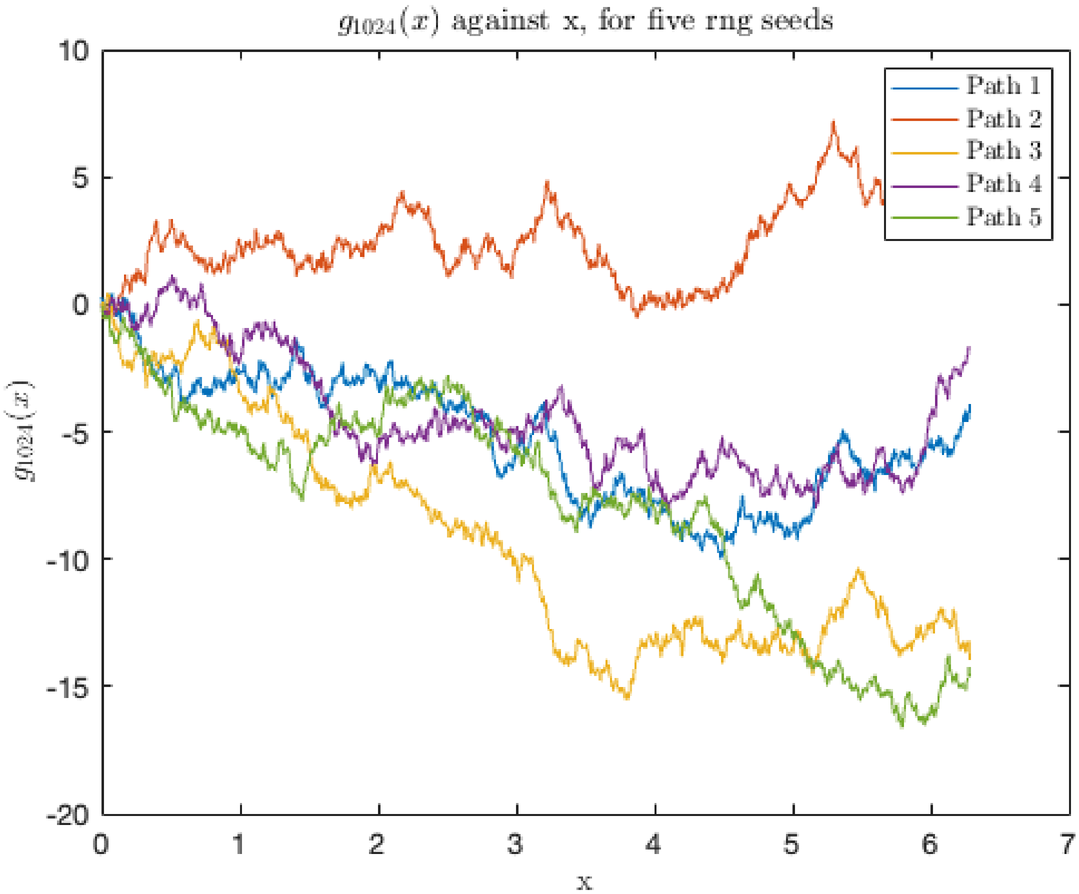
The plot contains 3 curves for $m = 10, 100, 1000$, using the same rng seed. In A2, we saw how taking a larger m caused the expected size of f_m to increase proportionally to $\sqrt{m}$, yet here the difference between g_m for $m = 10, 100$ is very small. The reason for this is that g_m is defined as an integral of f_m, so adding more sine and cosine terms has little effect. This is because the integral of $sin(jy)dy$ from $[0, x]$ oscilates between $0, 2$ with period $2\pi/j$ (and similarly for $cos(jy)$). So as j increases, the period of these osscilations decrease, and so these terms only cause

g_m to osscilate slightly over very small length scales. So as m increases, the graphs have the same approximate shape, but become less smooth. Hence it makes sense that as m goes towards infinity, these graphs become nowhere differentiable, but everywhere continuous.

Exercise A4

In this part we illustrate various Brownian paths. Firstly, a 1D Brownian path by plotting g_m for a large value of $m = 1024$, five times. The five graphs produced once again appear to be independent random walks. Although most of the graphs tend to be below 0, I expect that this is just a conincidence.

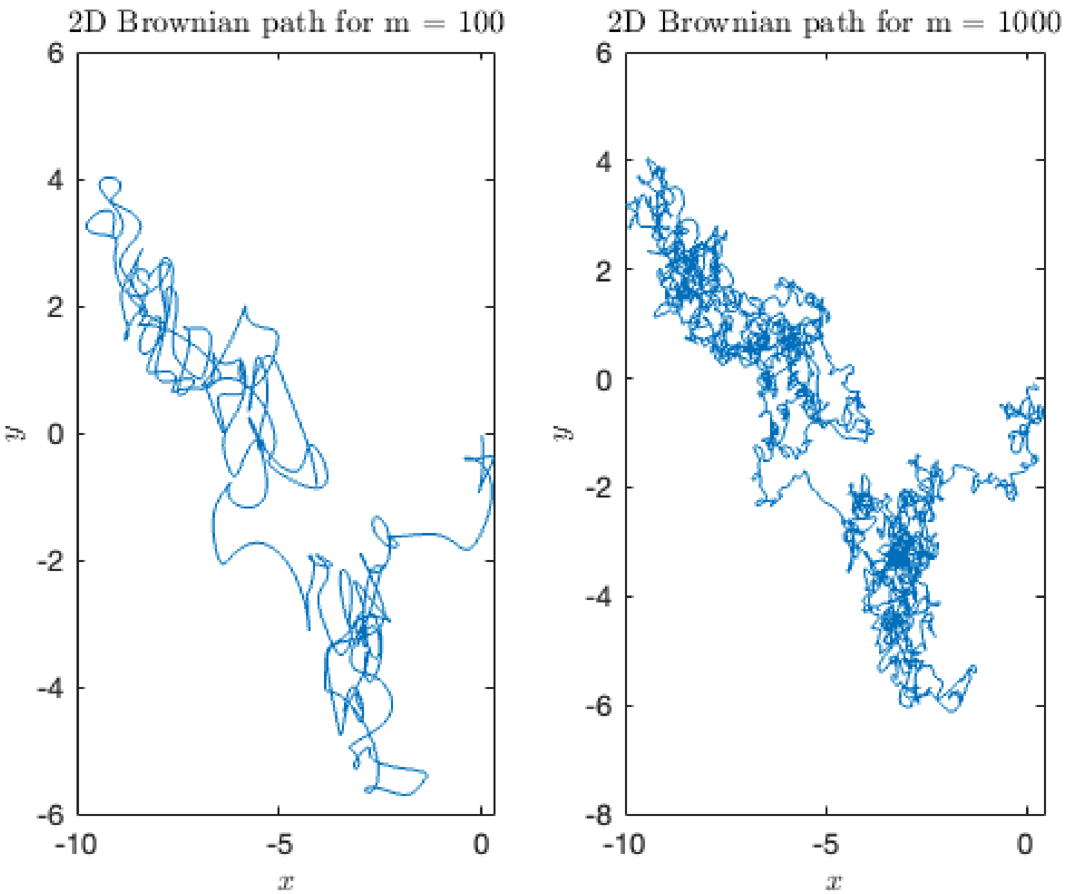
```
% We now fix m = 1024, and plot 5 different trajectories.
figure(5)
rng(1);
m = 1024;

% Loop through k = 1, ..., 5
for k = 1:5
    f = smooth(m);

    plot(xx, gmxx(f))
    hold on
end

% Format the plot
xlabel('x', 'Interpreter', 'latex')
ylabel('$g_{1024}(x)$', 'Interpreter', 'latex')
title('$g_{1024}(x)$ against x, for five rng seeds', 'Interpreter', 'latex')
legend('Path 1', 'Path 2', 'Path 3', 'Path 4', 'Path
 5', 'Interpreter', 'latex')
set(gca, 'fontsize', 13)
```



Next we illustrate 2D Brownian plots by plotting g_m against itself for two different, specified random seeds. We do this for $m = 100, 1000$.

```
% Set the figure
figure(6)

% Generate the x coordinates
% (here we use x_m, y_m for the x and y coordinates for a particular value
% of m).
rng(1)
f_100 = smooth(100);
rng(1)
f_1000 = smooth(1000);

x_100 = gmxx(f_100);
x_1000 = gmxx(f_1000);

% Generate the y coordinates
rng(2)
f_100 = smooth(100);
rng(2)
f_1000 = smooth(1000);
```

```
y_100 = gmxx(f_100);
y_1000 = gmxx(f_1000);

axis equal % set the axis to have equal scales

% Plot the graph for m = 100
subplot(1,2,1)
plot(x_100, y_100)

% Format the graph
xlabel('$x$', 'Interpreter','latex')
ylabel('$y$', 'Interpreter','latex')
title('2D Brownian path for m = 100', 'Interpreter','latex')
set(gca, 'fontsize', 13)

% Plot the graph for m = 1000
subplot(1,2,2)
plot(x_1000, y_1000)

% Format the graph
xlabel('$x$', 'Interpreter','latex')
ylabel('$y$', 'Interpreter','latex')
title('2D Brownian path for m = 1000', 'Interpreter','latex')
set(gca, 'fontsize', 13)
```



The two graphs produced appear to show two, 2D random walks. I think this is because we've seen from the previous part of A4 that g_m appears to produce independent random walks, given different rng seeds. So plotting two of these against each other will give a random walk whose x and y coordinates vary continuously and independently - ie. a 2D random walk. Because the same rng seeds are used for $m = 100, 1000$, the graphs produced have a similar overall shape. Like in the 1D case in A3, it seems that as m increases, the random walk produced becomes less smooth, and oscilates over smaller length scales.

Published with MATLAB® R2021b