Fourier Series

Samara Laliberte
Dept. of Mathematics
UMass Dartmouth
Dartmouth MA 02747
Email: slaliberte@umassd.edu

Muhammad Shams
Dept. of Mathematics
UMass Dartmouth
Dartmouth MA 02747
Email: mshams@umassd.edu

Abstract

The use of a sum of complex exponential or trigonometric periodic functions to approximate a function to almost exact precision. This tactic will result in minimal error when comparing it to the original function. Using a Fourier series allows a continuous, bounded, function to be evaluated, with uniform convergence through-out. This makes using them a useful tool in analyzing otherwise complicated functions. Starting with a simple Fourier sum of sines, a function can be almost exactly replicated as the number of coefficients are maximized. The same holds with the Fourier sum of exponential functions. This process is highly effective for continuous functions, but involves a larger error when handling discontinuous functions. The focus of this project is to understand these approximations and why there is error.

1 Uses for Fourier approximation

Fourier series are used to approximate complex functions in many different parts of science and math. They are helpful in their ability to imitate many different types of waves: x-ray, heat, light, and sound. Fourier series are used in many cases to analyze and interpret a function which would otherwise be hard to decode.

2 Approximating the Square Wave Function using Fourier Sine Series

2.1 Square Wave Function

The first function we examined which can be approximated by a Fourier series is the square wave function. This is a function which alternates between two function values periodically and instantaneously, as if the function was switched from on to off. The Square Wave function is also commonly called a step function. The function graphed from $x = -1$ to $x = 1$ is shown in Figure 1. By summing sine waves it is possible to replicate the square wave function almost exactly, however, there is a discontinuity in this periodic function, meaning the Fourier series will also have a discontinuity. It is clear in Figure 1 that the discontinuity will appear at x = 0, where the functions jumps from -1 to 1. The equation of this function is represented in Equation 1.

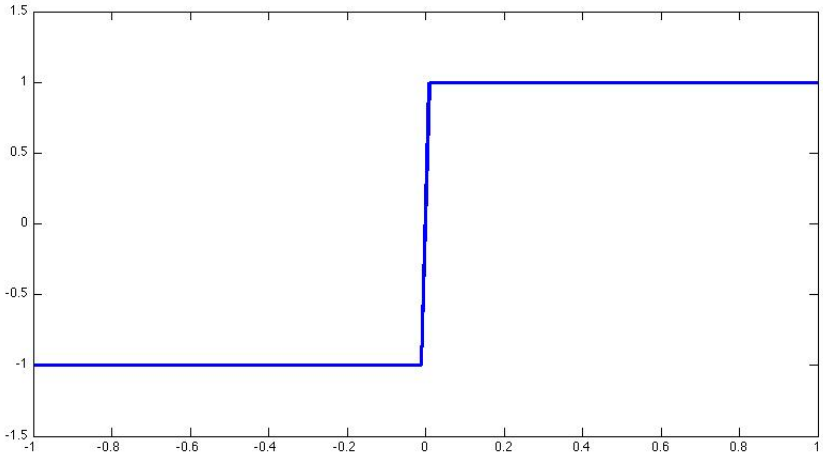

FIGURE 1: SQUARE WAVE FUNCTION

$$F(x) = \begin{cases} -1 & \text{for } -1 \leq x < 0\,, \\ 1 & \text{for } 0 \leq x \leq 1\,, \end{cases} \tag{1}$$

2.2 Fourier Sine Series

The Fourier sum of sines can be used to accurately approximate the square wave function. The more points plotted and coefficients used the closer the Fourier sum will be to looking like the square wave function. The equation of Fourier sine series used in this case is represented in Equation 2. j represents the number of coefficients used. When using the sum of sines, only odd numbered values are used, otherwise you would be adding zero every other term. The starting point, where j=1 is shown in Figure 2.

$$F(x) = 4/\pi \sum_{j=odd}^{\infty} 1/jsin(j\pi x) \tag{2}$$

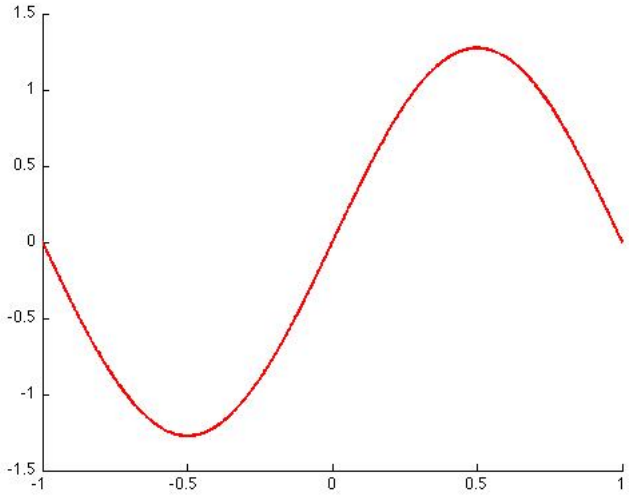

FIGURE 2: SINE WAVE

2.3 Approximating with Fourier Sine Series

A MATLAB code is used to plot the square wave function along with the Fourier sine series in order to compare the accuracy and error between the approximation and the actual function.

MATLAB CODE FOR FOURIER SUM, SQUARE WAVE AND ERROR

```
N = Number of points plotted
x = linspace(-1,1,N);
f = sign(x);
sum = 0.*x;
M = number of coefficients
for j = 1:2:M
    sum = sum + 4/pi*sin(j*pi*x)/j;
end
plot(x, sum, 'r')
hold on
plot(x,f,'LineWidth',2)
hold on
error = abs(sum-F)
Plot(x, error);
```

By changing N, the number of points plotted, and M, the number of coefficients, the accuracy of the approximation changes. For our purposes we kept the number of points plotted at 1000 to ensure the most precise graph for the number of coefficients used. We started using one coefficient, setting M equal to 1. The Fourier Series compared to the actual function is shown in Figure 3. Evaluating the function for $M = 10$ (Figure 4), $M = 50$ (Figure 5), $M = 100$ (Figure 6) and $M = 1000$ (Figure 7) it is easy to see how the series is almost perfectly approximated, but with visible discontinuity.

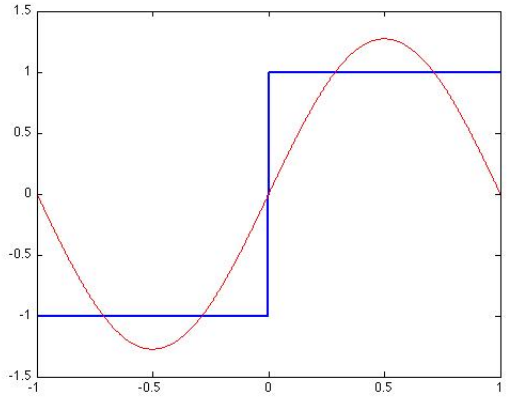

FIGURE 3: M = 1 COEFFICIENT

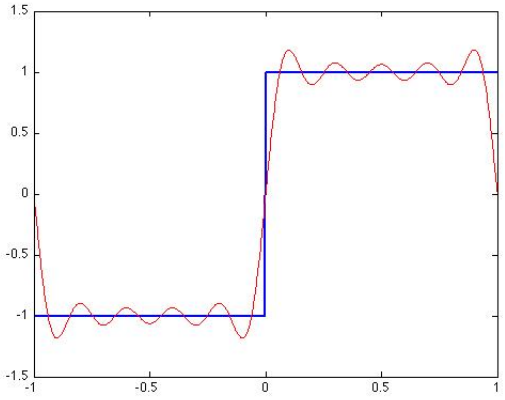

FIGURE 4: M = 10 COEFFICIENTS

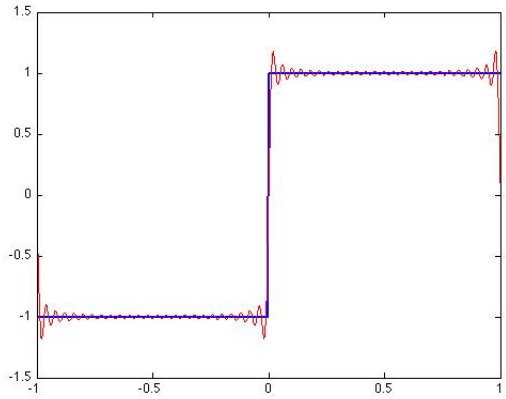

FIGURE 5: M = 50 COEFFICIENTS

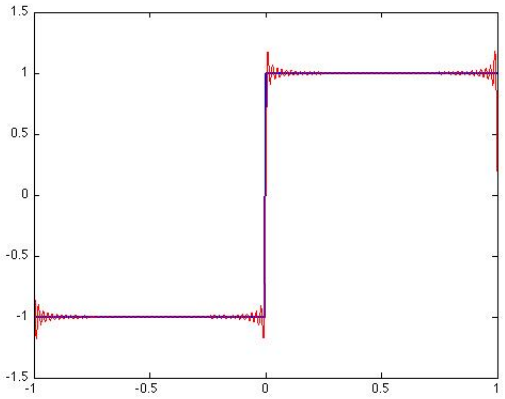

FIGURE 6: M = 100 COEFFICIENTS

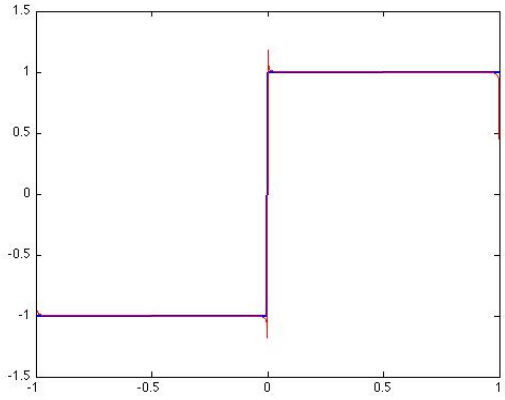

FIGURE 7: M = 1000 COEFFICIENTS

2.4 Error

There is visible error at the points: $x = 1$, $x = -1$, and $x = 0$, i.e where the function is discontinuous. Although this error appears to be minimal as more coefficients are used, it never disappears. This occurrence is referred to as the Gibb's Phenomenon. J. Willard Gibbs discovered that there will always be an overshoot at the points of discontinuity when using Fourier Series approximation. The error in $M = 50$ (Figure 8) and $M = 1000$ (Figure 9) noticeably decreases to almost zero where the function is a straight line. With that, there remains the same amount of error at the three discontinuous points. This is a product of using Fourier Series to approximate discontinuous functions.

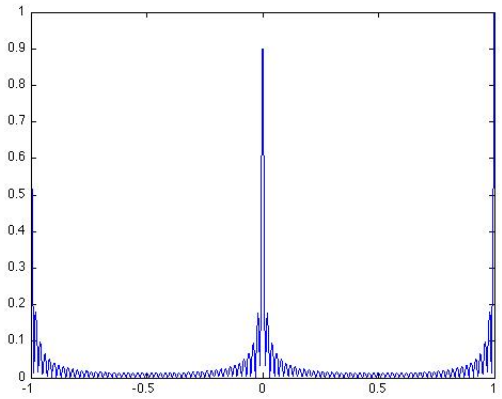

FIGURE 8: ERROR AT M = 50 COEFFICIENTS

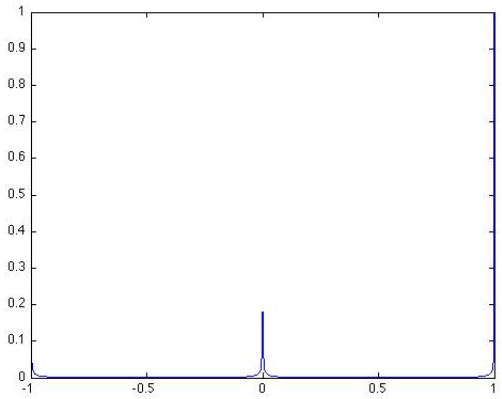

FIGURE 9: ERROR AT M = 1000 COEFFICIENTS

3 Fourier Approximation of a Line

3.1 The Line Approximated

The next function we used a Fourier sum of sines to approximate was a line. We chose the interval: 0 to 2π. The equation of the line is represented in Equation 3. The graph of the basic line is seen in Figure 10.

$$F(x) = 1/2(\pi - x) \tag{3}$$

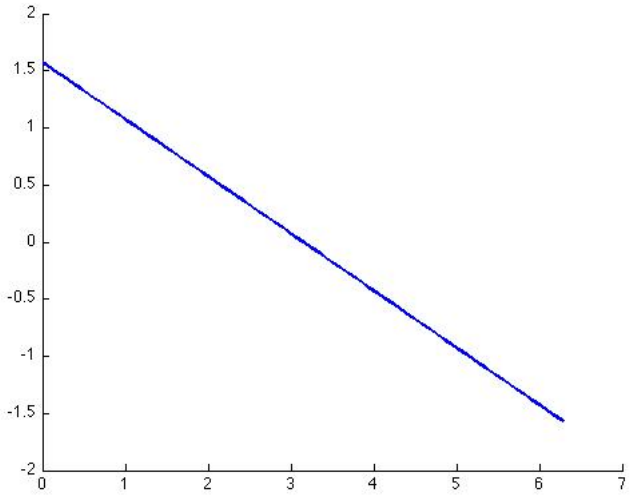

Figure 10: Graph of F(x)

3.2 Sine Series Used to Approximate line

The Sine Series used to approximate this function is slightly different than the first one used. The reasoning for this is that the original sine curve needs to be multiplied by and evaluated at different values in order to shape it into this function. Also, the curve is flipped over the x-axis in comparison to the curve in Figure 2. This allows it to start at π and move to 0, while the first function had to start at -1 and move to 1. The Sine series used is shown in Equation 4.

Just as before, the number of values used for j effect the accuracy of the approximation. The graph of the basic sine wave when $j = 1$ is represented in Figure 11.

$$\text{F(x)} = \sum_{j=odd}^{\inf}(1/j)sin(jx) \tag{4}$$

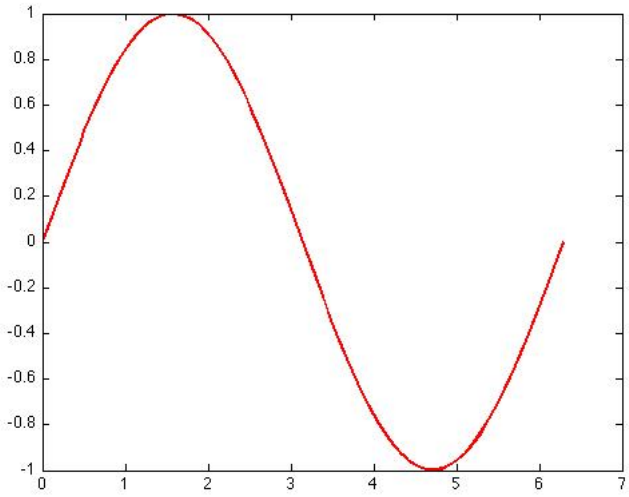

Figure 11: Initial Sine Wave

3.3 Approximating F(x)

To compare the graphs, we used a MATLAB code similar to the one used for our first function. There is a change in the boundary and the function we are using to approximate the line. The value of M ,once again, changes the number of coefficients plotted and the number of points plotted was set at 100. Increasing the value of M allows the line to be almost exactly replicated, but the Gibb's Phenomenon remains; there is a discontinuity at the endpoints. The starting point of the sine series is compared with the function it will eventual replicate in Figure 12. The function and it's approximation are shown at $M = 10$ (Figure 13), $M = 50$ (Figure 14), $M = 100$ (Figure 15) and $M = 100$ (Figure 16).

MATLAB Code for Fourier Sine Series, Line, and Error

```
x = linspace(0,2*pi,100);
sum = 0.*x;
M = number of coefficients used
for j = 1:M
    sum = sum + ((1/j)*sin(j*x));
end
F = ((1/2)*(pi-x));
plot(x, sum,'r');
hold on
plot(x, F, 'LineWidth', 2);
hold on
error = abs(sum - F);
plot(x, error,'m')
hold on
```

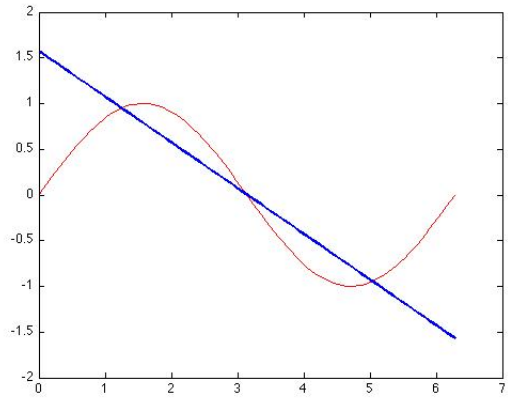

Figure 12: M = 1 Coefficient

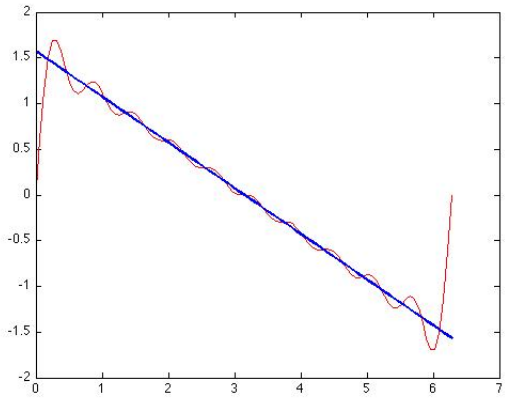

Figure 13: M = 10 Coefficients

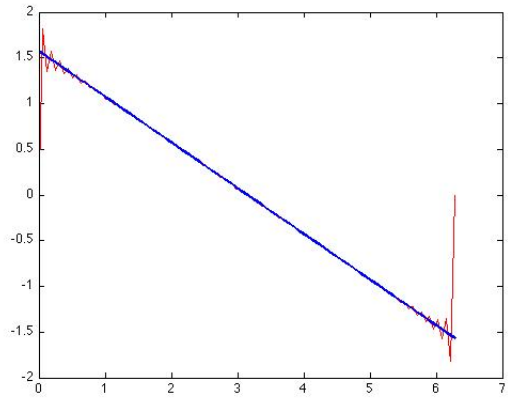

Figure 14: M = 50 Coefficients

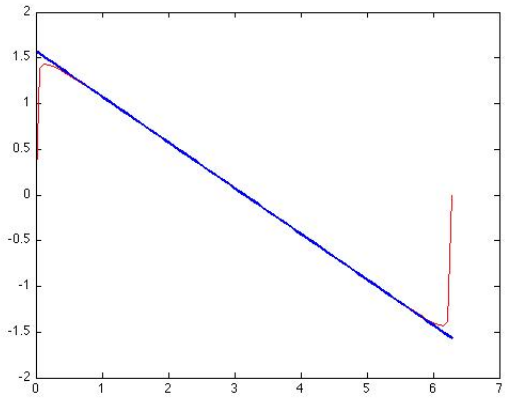

Figure 15: M = 100 Coefficients

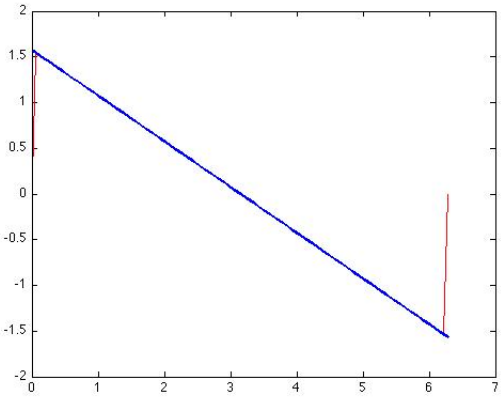

Figure 16: M = 1000 Coefficients

3.4 Error

The graph at $M = 1000$ shows the line is almost perfectly approximated, but there is a discontinuity at the ends. The reason for this discontinuity is not the same as in the first case. The function here is continuous and bounded, but

not periodic. Due to the sine function being periodic, it cannot approximate a non-periodic function with complete accuracy. The error between the function and it's approximation is shown at $M = 50$ (Figure 17) and $M = 1000$ (Figure 18). The error becomes fades through out the line but does not change at the ends, again, showing the Gibb's Phenomenon.

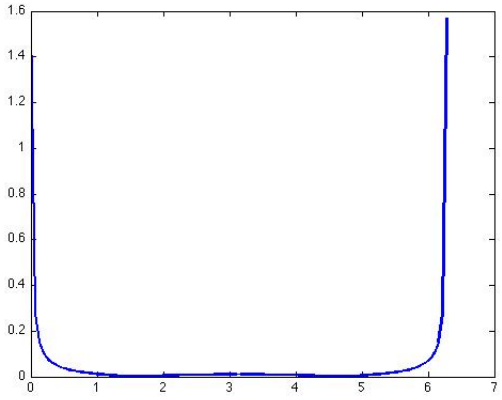

Figure 17: Error at M = 50

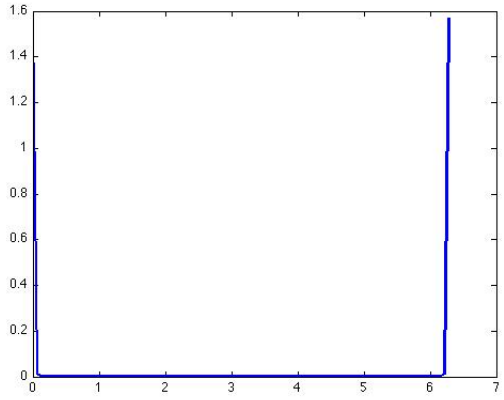

Figure 18: Error at M = 1000

4 Fourier Exponential Approximation

4.1 The Line Approximated

The next function we approximated was another line. We chose the interval -1 to 1. The equation of the line is represented in Equation 5.

$$\text{F(x)} = \text{x} \tag{5}$$

We used this simple line to show a different way of approximating that works just as well. The Fourier series we used to approximate this line was not a sum of sine waves, but instead a sum of complex exponential functions. The new function used to approximate the line is seen in equation 6.

$$\text{F(x)} = \sum_{-N}^{N} f(x)e^{i(k\pi x)} \tag{6}$$

In equation 6, k is used for the number of coefficients.

4.2 Approximating f(x) = x

In order to actually approximate $f(x) = x$ using fourier exponential coefficients we created a set of MATLAB codes. They were to be run one after the other as each code called some parameter from the previous one.

MATLAB Codes for Exponential Coefficients

```
clc
np = 100
nc = 30
k = -nc:nc;
x = linspace(-1,1,np);
f = @(x) x;
for j = 1:length(k)
C = @(x) f(x).*exp(-1i*x*k(j)*pi);
```

```
fk(j) = quad(C,-1,1,);
end
plot(k, fk);
```

This first code is used to calculate the coefficients to be used later.

```
function F = reconstruction(y,fk,k)
for x = 1:length(y)
F(x) = 0;
for j= 1:length(k)
F(x) = F(x) + fk(j)*exp(1i*k(j)*y(x)*pi);
end
end
end
```

This second code is to reconstruct the function we are approximating using the coefficients from the earlier code.

```
clc np = 100;
nc = 5;
f = @(x) x;
npy = 101;
x = linspace(-1,1,np);
y = linspace(-1,1,npy);
k = -nc:nc;
for j = 1:length(k)
C = @(x) f(x).*exp(-1i*pi*x*k(j));
fk(j) = quad(C,-1,1,1e-12);
end
F = reconstructionexpo(y,fk,k);
F = F./(2);
plot(x,f(x),'--b')
hold on
plot(y,F,'-xr')
```

This last code takes the output from the last two codes and actually approximates a function using the exponential coefficients. Our code uses nc to represent the number of coefficients used. By increasing it we can better approximate a function. The graph of the function we are approximating and the exponential function used to approximate with $nc = 1$ can be seen in figure 19, $nc = 10$ in figure 20, $nc = 25$ in figure 21, and $nc = 100$ in figure 22.

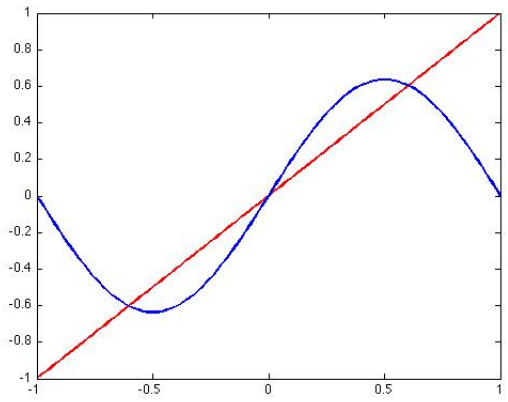

Figure 19: nc = 1 Coefficient

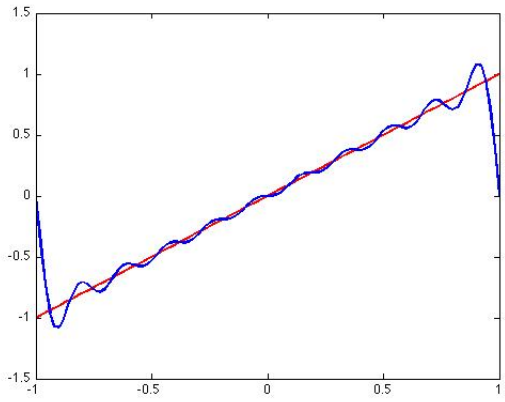

Figure 20: nc = 10 Coefficients

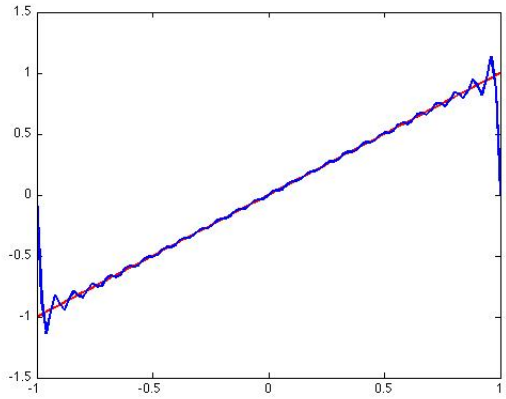

Figure 21: nc = 25 Coefficients

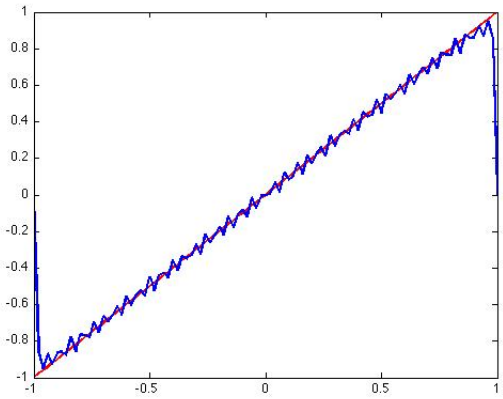

Figure 22: nc = 100 Coefficients

You can see that even with an exponential function used for approximation, the Gibbs phenomenon persists along the boundaries. You can also see that as the number of coefficients was increased the approximation got better except that in figure 22, the approximation appears to be worse than in figure 21. This would be because, with the quad function we used in MATLAB to integrate the function we see a problem as the number of coefficients exceeds 25. This can be dealt with by changing the tolerance.

5 Conclusions

To continue researching Fourier Series there are a few areas and specific problems that we would address. With our code for exponential Fourier series we would want to look into using different quad functions in order to decrease the error that is present after the coefficients reach higher than 25. Also we would look into how post processing is done.

6 Acknowledgements

Thank you Sigal Gottlieb and Saeja Kim for continuing to advise us on this project and help us when we were totally and utterly confused.

7 References

[1] Sigal Gottlieb, Jae-Hun Jung, and Saeja Kim. *A Review Article of David Gottlieb's Work on the Resolution of the Gibbs Phenomenon*, volume x of *Communications in Computational Physics*. Global-Science Press, 2010.