A FINITE REPRESENTATION OF AN INFINITELY SMALL WORLD

CHRIS RACKAUCKAS

ABSTRACT. A common symbol of "it's a small world" is the cutout of people holding hands. These figures were transformed into functions defined on the interval $[0, 8]$ and thus a generalized interval fourier decomposition. The constants were then solved using a finite sum approximation. These values were then used to generate a function that repeats indefinitely that draws an approximation to this figure with a periodicity of 8. Since this method solved for a finite sum approximation, it is a finitely approximated solution of an infinite version of the "it's a small world" cutout, hence a finite representation of an infinitely small world.

1. Introduction to the Problem

The picture above is a very common image. It shows unity among people, "small the world is", and represents how we should all come together as one. Although we think of the cutout like this, it is lacking in one significant property: it is finite and the number of people is much less than the population. Thus I set out to construct an infinite version of this image. Being that infinite actual representations must either take an infinite amount of whatever is used to construct it, or if the infinite representation does take a finite amount of paper then the size of the persons would be some geometric series where the people are getting smaller. Since I want to show that all people are the same and actually be able to make this construction, an actual physical representation was not possible. Thus I decided to create a representation in function space.

2. The Image as a Function

To solve this problem in function space, I first had to describe what the function was. A few abstractions were used to ensure that the function could be described in the function space of $\mathscr{R}^2$. All of the people were to be described using what in the image above was the "male body". Most cutouts do not distinguish between male and female and so this does not seem to stray from the original (also, it is so passe to believe that all women must be wearing a dress!). Also, the heads were changed to be a rectangle. This is because it would have been impossible for the function to get to the side of the head while staying on the body which would have been necessary to split the circle into two function. However, this will not end up as a problem since the fourier decomposed version will round out the edges and thus give a pretty circular looking head. Lastly, although many cutouts use a version with the hands going down, I chose to have the hands going up like in the image above. All of the same steps would apply to either image.

To transform this image into a function, I had to make sure it passed the vertical line test. Since no one function would pass the vertical line test (using non-parametric functions), I split the image into 4 functions. One describes the inside of the legs, the other the outside of the legs (and gives a waist line), another describes the bottom of the arms (and gives the same waist line), and the last gives the top of the arms and the head. These functions were labeled f_1 through f_4 respectively. By drawing this image on a graph of width 8 and height 8, I solved for these functions to be:

$$f_1(x) = \begin{cases} 0 & \text{if } 0 \le x \le 2 \text{ or } 6 \le x \le 8 \\ 2x - 4 & \text{if } 2 \le x \le 4 \\ -2x + 12 & \text{if } 4 \le x \le 6 \end{cases}$$

$$f_2(x) = \begin{cases} 0 & \text{if } 0 \le x \le 1 \text{ or } 7 \le x \le 8 \\ 4x - 4 & \text{if } 1 \le x \le 2 \\ 4 & \text{if } 2 \le x \le 6 \\ -4x + 28 & \text{if } 6 \le x \le 7 \end{cases}$$

$$f_3(x) = \begin{cases} -\frac{3}{2}x + 7 & \text{if } 0 \le x \le 2 \\ 4 & \text{if } 2 \le x \le 6 \\ \frac{3}{2}x - 5 & \text{if } 6 \le x \le 8 \end{cases}$$

$$f_4(x) = \begin{cases} 7 & \text{if } 0 \le x \le 1 \text{ or } 7 \le x \le 8 \\ -x + 8 & \text{if } 1 \le x \le 2 \\ 6 & \text{if } 2 \le x \le 3 \text{ or } 5 \le x \le 6 \\ 8 & \text{if } 3 \le x \le 5 \\ x & \text{if } 6 \le x \le 7 \end{cases}$$

The image below shows functions f_1 through f_4 respectively then plots them together using Mathematica.

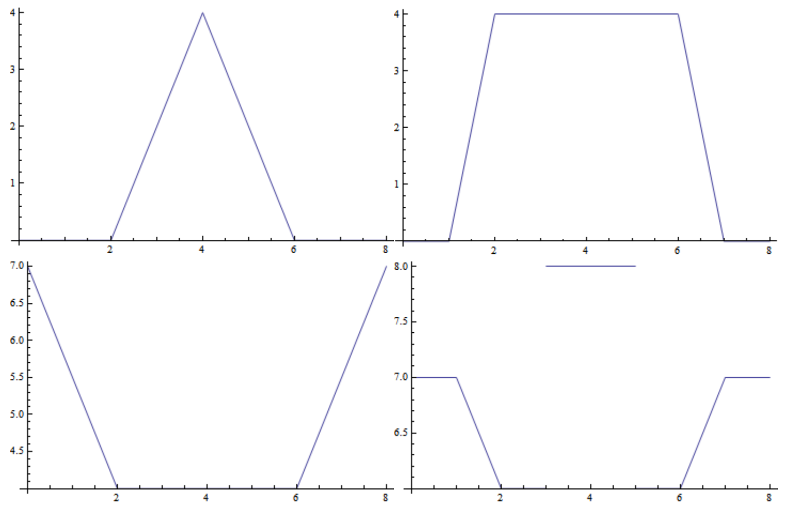

FIGURE 2.1. The plots of f_1 through f_4

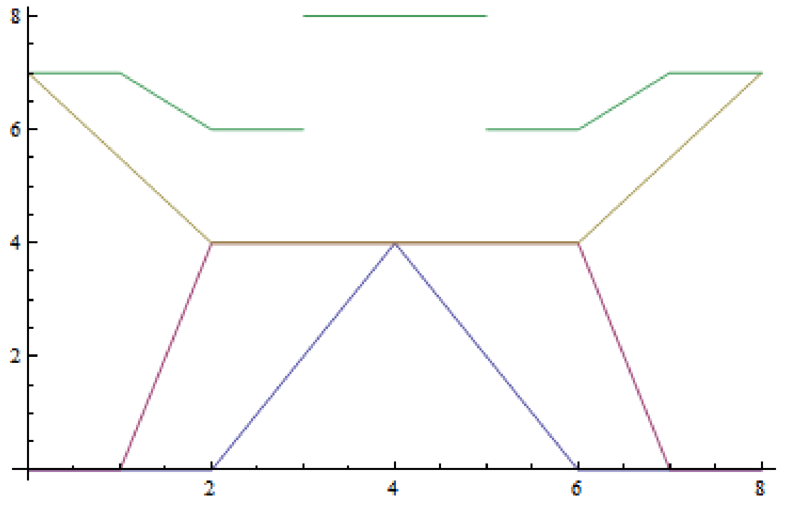

FIGURE 2.2. A plot of f_1 through f_4 together

3. Fourier Decomposition

The next step was to decompose these functions to infinite periodic functions using what is known as a Fourier decomposition. The Fourier Decoposition method is defined as writing the function $g_i(x)$ as an infinite summation sine and cosine functions. The method is described as writing

$$g_i(x) = \frac{A_0}{2} + \sum_{n=1}^{\infty}(A_{n_i}\cos(nx) + B_{n_i}\sin(nx))$$

where

$$A_{n_i} = \frac{1}{\pi}\int_{-\pi}^{\pi} f_i(x)\cos(nx)dx$$

and

$$B_{n_i} = \frac{1}{\pi}\int_{-\pi}^{\pi} f_i(x)\sin(nx)dx.$$

However, this Fourier decomposition requires the function to be defined on the interval $[-\pi,\pi]$. Thus I instead had to use a different version of the Fourier Decomposition called the Generalized Interval Fourier Decomposition[2] which defines the summed function as a summation of functions defined on the interval $[0,L]$ as

$$g_i(x) = \frac{A_0}{2} + \sum_{n=1}^{\infty}(A_{n_i}\cos(\frac{2\pi nx}{L}) + B_{n_i}\sin(\frac{2\pi nx}{L}))$$

where

$$A_{n_i} = \frac{2}{L}\int_0^L f_i(x)\cos(\frac{2\pi nx}{L})dx$$

and

$$B_{n_i} = \frac{2}{L}\int_0^L f_i(x)\sin(\frac{2\pi nx}{L})dx$$

Notice that this simply resizes the periodicity of the functions to match the length of the interval and thus is an intuitive extension of the standard Fourier Decomposition.

4. Infinite Sum Representation

In order for these to be calculated, I needed these integrals to be put into a summation form. First take note that the integral of a piecewise function is simply the sum of the integrals on the different intervals. Since each interval of every f_i function can be written as some linear function $ax+b$, it suffices to solve for what these integrals would be.

Noting that the sine function can be representated as a summation by

$$\sin(x) = \sum_{k=0}^{\infty}\frac{(-1)^k x^{2k+1}}{(2k+1)!}$$

by using a U-substituion and solving we see that

$$\frac{2}{L}\int_r^s b\cos(\frac{2\pi nx}{L})dx = \frac{2b}{L}\sum_{k=0}^{\infty}\frac{(-1)^k(s^{2k+1}-r^{2k+1})(\frac{2\pi n}{L})^{2k}}{(2k+1)!}$$

and by using integration by parts we see that

$$\frac{2}{L}\int_r^s ax\cos(\frac{2\pi nx}{L})dx = \frac{2a}{L}\sum_{k=0}^{\infty}\frac{(-1)^k(s^{2k+2}-r^{2k+2})(\frac{2\pi n}{L})^{2k}}{\frac{(2k+2)!(2k+1)!}{(2k+2)!-(2k+1)!}}$$

The derivations are included in the notes. It is interesting to see that the ax integral can be put into one summation. This is due to the fact that the integration requires the one portion be integrated once and the other integrated twice. However, while the one that is integrated twice is a cosine while the other is a sine, the u-substitution occurs on the integrated twice version twice while the u-substition occurs on the integrated once version just once. Thus while the summation of the cosine contains only the $2k$ values and the sine contains the $2k+1$ values, after the u-substitions cancel out the n terms a bit, it turns out that both parts are defined as summations with the values of n only being even (since the cosine term loses n twice, one for each u-substition, and the sine term loses n once for its single u-substitution). Thus the value $\frac{(2k+2)!(2k+1)!}{(2k+2)!-(2k+1)!}$ is simply finding the common denominator to add together these two terms to make a single function.

Notice by putting this together we get the equation necessary for computation:

$$\frac{2}{L}\int_r^s (ax+b)\cos(\frac{2\pi nx}{L})dx = \frac{2}{L}[a\sum_{k=0}^{\infty}\frac{(-1)^k(s^{2k+2}-r^{2k+2})(\frac{2\pi n}{L})^{2k}}{\frac{(2k+2)!(2k+1)!}{(2k+2)!-(2k+1)!}} + b\sum_{k=0}^{\infty}\frac{(-1)^k(s^{2k+1}-r^{2k+1})(\frac{2\pi n}{L})^{2k}}{(2k+1)!}]$$

Notice that for the functions we are interested in, functions defined on $[0,8]$, $L=8$. Thus these sums are sufficient for the computation.

5. A Note on the Accuracy

When I first solved the equations, I solved the equations using just the first 3 terms in the summation for both a and b. At $n=1$this solves the integral of $(2x-4)\cos(x)$ from 2 to 4 (using $L=\frac{1}{2\pi}$) as approximately 15.39. However, using WolframAlpha I saw that the actual value is approximately -1.178585. Why such the large discrepency?

Upon investigation, I noticed that there were some signs that this error gives that would end up being useful for computation. First of all, notice that both of the summations are alternating series. Also notice that since the factorial function grows quicker than the expoential function, this gives us the fact that each term of the series is smaller than the next. Thus what we see with the errors is that the estimation of the A_{n_i} starts as a large value and then decreases k increases. Since in general for a Fourier series the $n+1$th term is smaller than the nth term, we can easily see if we did not solve the sum of the k values enough if we see this did not decrease. A simple of data not ran on enough k's is included. What is shown is the sum for solving $(2x-4)\cos(\frac{2\pi nx}{L})$ where $n=3$ and $L=8$. Notice how the total value of the sum starts high and decreases with each k. If k was large enough, this would have approached its actual value, approximately 0.18012654869748941859, but since the sum was not run long enough, its value is higher than the summed value for the $n = 2$ term which was 0.81056946913870223475, which indicates that the max k used for the summation must be increased.

6. Computation

Since the accuracy required was far too many terms to be computed by hand, the summations were solved using a program in Python. The method was to solve out the terms using a given maximum k and check to see if the terms were decreasing and if not increase the maximum k value by hand. I did not resort to having the computation increment the k values since each run of the program already takes approximately an hour, and thus letting the computer find a good k could have made the program run on a standard PC for more than a day. Also, normal floats and integers were not able to be used for computation. This is because the integers from factorials of large numbers quickly created integers larger than what could be represented by 32 bits and the decimal precision required for much of the division was much more than a 32 bit float could offer. Thus the computation had to be done using Long and Decimal formats. Python automatically converts int types to long types which have infinite accuracy (which was why I chose Python, I knew this from Crypotography exercises). However, some conversions had to occur to have the Decimal class used instead of floats since operations between decimals and floats were undefined (though operations between longs and Decimals are defined). In particular, a decimal accuracy must be set for the computation. I noticed a decimal accuracy of 200 siginificant figures. Thus by using the decimal class for the computation, I was guarenteed a computation that was correct to 200 siginificant figures (which, gives at least 50 significant figures in the decimal place even for the largest of numbers seen in the computation).

A sample of how the computation was done for a simple function is included. That program was then extended to be able to compute the integrals of an arbitrary number of functions defined piecewise by linear functions. By trial and error using the facts mentioned in the Note on Accuracy, I found that a good maximum k value for the computation was 2000. Thus the final computation consisted of using infinite accuracy integers called longs, 200 significant figure decimals, used the first 2000 terms of the summation, and solved for the first 10 A_{n_i} terms for each function f_1 through f_4 (notice that the first 10 terms means solving for A_{0_i} through A_{9_i}). This computation took approximately an hour for a single run (making debugging be a pretty lengthly process!).

7. Final Results

The program was run and the final results were obtained. The results are included. Notice that any value $|A_{n_i}| \leq 10^{-10}$ is set equal to 0 to account for errors in the computation. The transformed functions can thus be approximated as:

$$\begin{aligned} g_1(x) = 1 + &-1.62113893827740437022\cos(\frac{\pi x}{4}) \\ &+ 0.81056946913870223475\cos(2\frac{\pi x}{4}) \\ &- 0.18012654869748941859\cos(3\frac{\pi x}{4}) \\ &- 0.06484555753109615893\cos(5\frac{\pi x}{4}) \\ &+ 0.09006327434874469275\cos(6\frac{\pi x}{4}) \\ &- 0.03308446812811030948\cos(7\frac{\pi x}{4}) \\ &- 0.02001406096638769847\cos(9\frac{\pi x}{4}) \end{aligned}$$

$$\begin{aligned} g_2(x) = 2.5 &- 2.29263667300302521816\,2\cos(\frac{\pi x}{4}) \end{aligned}$$

$$\begin{aligned}
g_4(x) = 6.875 &- 0.32715714790635031829\cos(\frac{\pi x}{4}) \\
&+ 0.83926213965225634361\cos(2\frac{\pi x}{4}) \\
&- 0.36378979074689778757\cos(3\frac{\pi x}{4}) \\
&- 0.10132118364233828611\cos(4\frac{\pi x}{4}) \\
&+ 0.15713689650139041578\cos(5\frac{\pi x}{4}) \\
&- 0.18969077220200814945\cos(6\frac{\pi x}{4}) \\
&+ 0.14031374247654027899\cos(7\frac{\pi x}{4}) \\
&- 0.09295910712547210659\cos(9\frac{\pi x}{4})
\end{aligned}$$

The results were then put into Mathematica to plot the generated functions. The functions are plotted in order and on top of the function they are approximating:

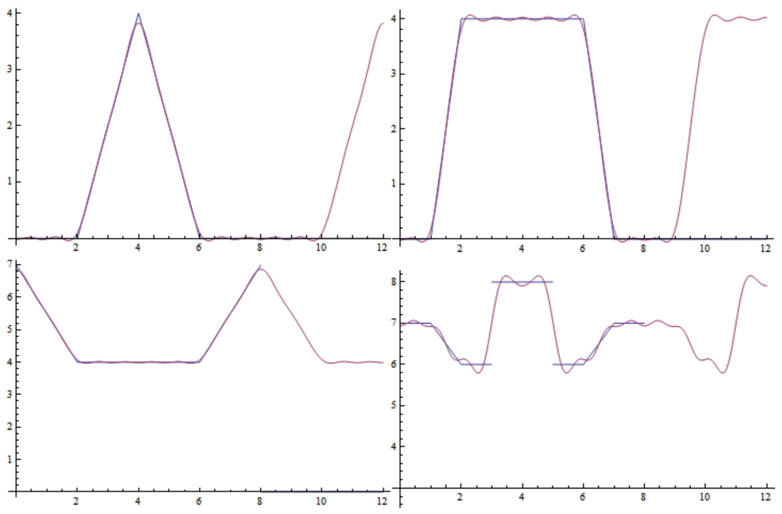

Figure 7.1. Functions g_1 through g_4 plotted on top of f_1 through f_4 respectively

Notice that f_4, being the most complex function and including a discontinuity, is the least well approximated. However, when these are all put together, these form a picture that I would say is quite pretty:

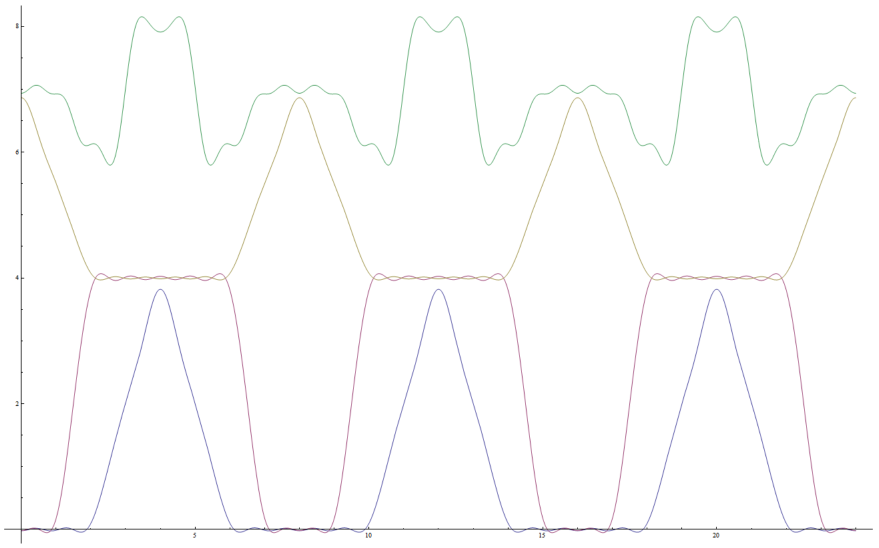

FIGURE 7.2. The finite sum representation of the infinite functions g_1 through g_4

8. Summary

The small world cutout is quite a lovely image, but as humans we can do better than just a finite representation. Thus I created a representation of the cutout in function space using a Fourier Decomposition. After dealing with many computational problems due to accuracy, final equations were produced. With the finite approximation of the infinitely defined function, I draw out the picture to make what I would call a lovely picture that expresses how infinitely many people are one. This is complete unity. This is the finite representation of the infinitely small world.

References

[1] Khamsi, M. A. "Fourier Series: Basic Results." S.O.S Math. MathMedics, LLC. Web. 16 May 2012. <http://www.sosmath.com/fourier/fourier1/fourier1.html>.

[2] Olver, Peter. "Chapter 12, Fourier Series." Peter Olver's Home Page. University of Minnesota. Web. 16 May 2012. <http://www.math.umn.edu/~olver/am_/fs.pdf>.

[3] Weisstein, Eric W. "Fourier Series." Wolfram MathWorld. Wolfram Research, Inc. Web. 16 May 2012. <http://mathworld.wolfram.com/FourierSeries.html>.