TRIGONOMETRIC INTERPOLATION USING THE DISCRETE FOURIER TRANSFORM

Juan Carlos Ponce Campuzano

School of Mathematics and Physics, The University of Queensland, Australia

Abstract

The Fourier Transform (and all its versions discrete/continuous/finite/infinite), covers deep and abstract mathematical concepts, and can easily overwhelm with detail. In this paper I provide some intuitive ideas of how the Discrete Fourier Transform (and its version with low frequencies) works and how we can use it to approximate real periodic functions and parametric closed curves by means of trigonometric interpolation.

Keywords: trigonometric, interpolation, Discrete Fourier Transform, GeoGebra scripting

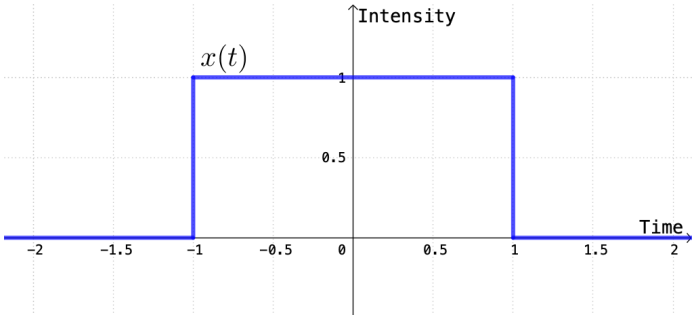
1 INTRODUCTION

Fourier Analysis has become an indispensable tool to represent signals and systems in science and engineering. In particular, the so called *Fourier Transform*:

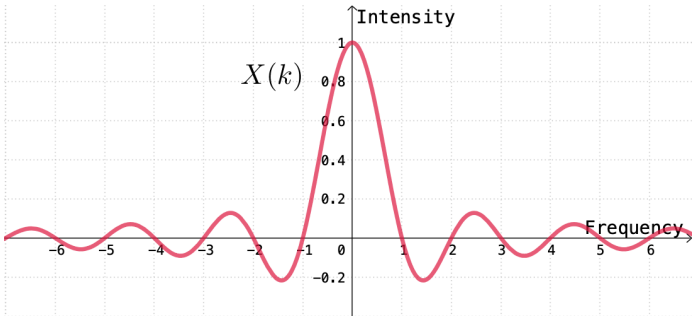
$$X(k) = \int_{-\infty}^{\infty} x(t)e^{-2\pi itk}dt \tag{1}$$

is used to decompose a signal $x(t)$ in terms of its sinusoidal components, which enable us to determine the output of a system faster that other methods, in addition to other advantages. At first glance, with all the mathematics involved, the Fourier transform (1) can be quite intimidating as it looks complex and difficult. But, it is not! It is similar to find the total amount of a set of coins. To make an analogy, let us say that we have a box with coins of 5, 10, 25 and 50 cent denominations. We can take one by one and add its value to a partial sum. We find the amount after the values of all the coins are added. An alternative way to do this is to decompose the coins into the four denominations and count the number of coins in each. Multiplying the number of different coins by their value and adding results in the amount.

In the representation of a function in the form $x(t)$, the variable t is the independent variable (but not always) in a certain domain, designated as the *time domain*. In the representation of a function in the form $X(k)$, variable k, which represents the frequency index of a frequency component, is the independent variable in the *frequency domain*, see Figure 1. Each representation specifies the given function. In particular, the frequency-domain representation turns out to be a convenient and efficient way for analysing signals.



(a) Time domain.



(b) Frequency domain.

Figure 1. Signal representations.

There are four different versions of the Fourier Transform to suit the different types of signals. In particular, the most frequently used in practice is the *Discrete Fourier Transform* (DFT), together with its *inverse* (IDFT). The DFT is the only version in which signals are represented in finite and discrete form in both the time and frequency domains, and, hence, implementable using a digital system. Modern digital media, (sound, video or images), are based on discrete data. Physical devices can generate signals only over a finite period and can generate frequency components of a finite order only. Then, for all practical purposes, all waveforms generated by physical devices can be represented by the DFT adequately. Furthermore, fast algorithms are available for its implementation. Since the DFT uses only a finite number of discrete sinusoids to represent the signals, it is easier to understand and the visualisation of the reconstruction of the waveforms is much simpler.

A well-known application of the DFT in signal processing is *trigonometric interpolation*, that is, the process of finding a function defined as a sum of sines and cosines of given periods, which goes through a given data set. The interpolation of large amounts of equally-spaced data by trigonometric polynomials can produce very accurate results. In fact, it is a very appropriate approximation technique in areas involving digital filters, antenna field patterns, quantum mechanics, optics, and in numerous simulation problems.

Until the middle of the 1960s, however, the method had not been extensively applied due to the number of arithmetic calculations required for the determination of the Fourier coefficients in the approximation. The interpolation of N data points by the direct-calculation technique, considering the DFT and its inverse, requires approximately N^2 multiplications and N^2 additions. The approximation of many thousands of data points is not unusual in areas requiring trigonometric interpolation, so the direct methods for evaluating the Fourier coefficients require multiplication and addition operations numbering in the millions.

In 1965, a paper by James W. Cooley and John W. Tukey in the journal *Mathematics of Computation* described a different method of calculating the Fourier coefficients in the interpolating trigonometric polynomial, see Cooley and Tukey (1965). This method reduces the computing time of the DFT to a time proportional to $N \log_2 N$ multiplications and $N \log_2 N$ additions, provided N is chosen in an appropriate manner. For a problem with thousands of data points, this reduces the number of calculations from millions to thousands.

The method described by Cooley and Tukey is known either as the *Cooley-Tukey algorithm* or the *Fast Fourier Transform* (FFT) algorithm and has led to a revolution in the use of interpolatory trigonometric polynomials and, according to Gilbert Strang, is "the most important numerical algorithm of our lifetime" (Strang, 1994, p.253). The FFT algorithm is wired into computers and widely explained in the literature, see for example Brigham (1988); Stein and Shakarchi (2003); Sundararajan (2018).

In this paper I provide some intuitive ideas of how the Discrete Fourier Transform works and how we can use it to approximate real periodic functions and parametric closed curves by means of trigonometric interpolation. In this case, GeoGebra turns out to be a powerful tool since we can easily implement the DFT, compute the trigonometric interpolation and plot the results.

2 THE DISCRETE FOURIER TRANSFORM

The DFT and its inverse are perhaps among the deepest insights ever made with many applications in our modern digital world. Unfortunately, its meaning is buried within dense equations:

$$\text{DFT:} \quad X_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n \cdot e^{-ik\frac{2\pi}{N}n}, \quad k = 0, 1, \ldots, N-1 \tag{2}$$

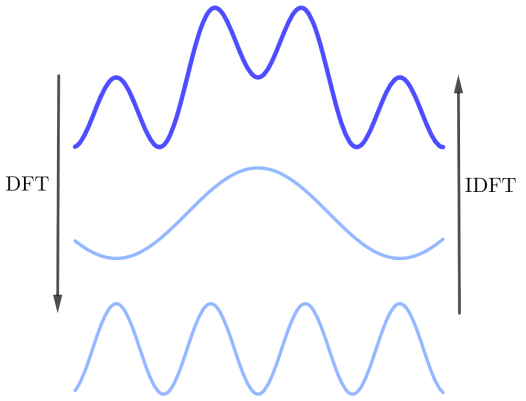
$$\text{IDFT:} \quad x_n = \sum_{k=0}^{N-1} X_k \cdot e^{ik\frac{2\pi}{N}n}, \quad n = 0, 1, \ldots, N-1 \tag{3}$$

which can be rewritten, by using Euler's formula $e^{it} = \cos t + i \sin t$, as follows

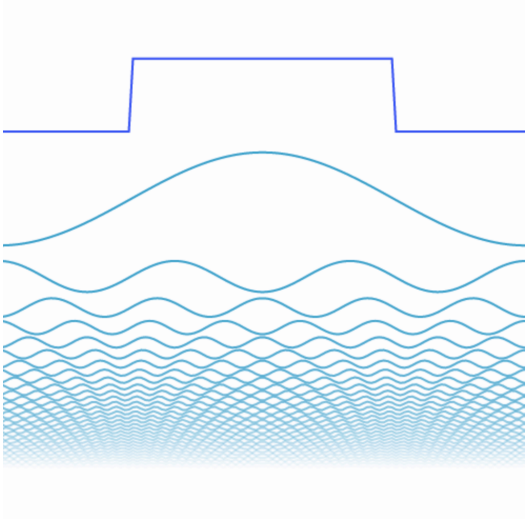
$$X_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n \left[\cos\left(k \, \frac{2\pi}{N} \, n \right) - i \, \sin\left(k \frac{2\pi}{N} \, n \right) \right] \tag{4}$$

$$x_n = \sum_{k=0}^{N-1} X_k \left[\cos\left(k \, \frac{2\pi}{N} \, n \right) + i \, \sin\left(k \, \frac{2\pi}{N} \, n \right) \right] \tag{5}$$

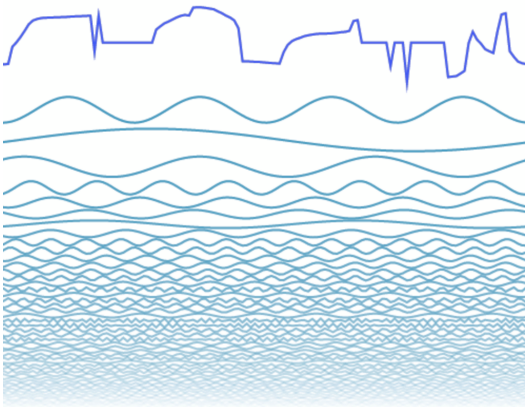
In essence, the DFT allows us to split any periodic function into a number of sinusoid functions or waves. Consider for example a periodic function $x(t)$ shown in the graph at the top in Figure 2a. This wavy pattern here can be split up into sinusoid functions, shown at the bottom in the same Figure 2a. On the other hand, the IDFT allows us to recover the original function by adding up the two sinusoid functions below. But this example seems quite simple. Can we use the same process for more complicated functions? What about functions that do not even look like they are made of sinusoids?



(a) A simple wave.



(b) The square wave.



(c) A general wave function.

Figure 2. Waves at the top are the result of the sum of the sinusoid functions shown below.

Let us consider, for instance, the periodic square function which graph is shown at the top in Figure 2b. It might be hard to see it but this wave also can be split up into sinusoid functions. However, we need a lot of them this time (technically an infinite amount to perfectly represent it), see sinusoid functions in Figure 2b. As we add up more and more sinusoid functions the pattern gets closer and closer to the square wave we started with. More surprisingly is that the DFT can be applied to any periodic function no matter how complicated this is, see for example the wave shown in Figure 2c, and the IDFT allows us to recover that periodic function from the sinusoid waves. This decomposition of a waveform relies on the remarkable fact that the complex exponentials form an *orthogonal basis*.

Remark 1. Orthogonality of two complex sequences *means that the sum of point-wise products of a sequence and the conjugate of the other sequence is zero or a constant over a specified interval. For details see (Sundararajan, 2018, p. 37).*

3 TRIGONOMETRIC POLYNOMIAL APPROXIMATION

As we mentioned before, a well-known application of the DFT in signal processing is *trigonometric interpolation*. To show how the DFT works in this context, in this section we will approximate the representation of a signal f from a discrete set of values.

Consider for example the function $f : [0, 2\pi] \to \mathbb{R}$ defined by

$$f(t) = 2t - \frac{t^2}{\pi}. \tag{6}$$

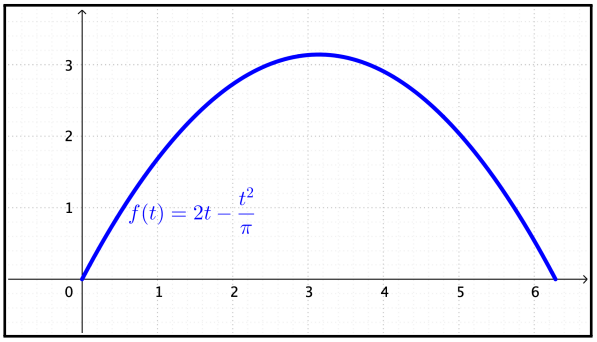


Figure 3. Graph of $f(t) = 2t - t^2/\pi$.

Given N sampled values of f, our goal is to find the *Fourier sum* $p(t)$ defined on $[0, 2\pi]$ such that

$$f(t) \approx p(t) = X_0 + X_1 e^{it} + X_2 e^{2it} + \cdots + X_{N-1} e^{(N-1)it} = \sum_{k=0}^{N-1} X_k e^{ikt}. \tag{7}$$

The symbol $\approx$ means that the function $f(t)$ and the sum $p(t)$ agree on the sample points:

$$f(t_n) = p(t_n), \quad n = 0, 1, 2, \ldots N-1.$$

Therefore, $p(t)$ can be viewed as a (complex-valued) *interpolating trigonometric polynomial* of degree $\leq n-1$ for the sample data $x_n = f(t_n)$.

Remark 2. *Since our function (6) is real, we only need to use the real part of* $p(x)$*, that is*

$$f(t) \approx p(t) = X_0 + X_1 \cos t + X_2 \cos 2t + \cdots + X_{N-1} \cos{(N-1)\, t}. \tag{8}$$

where X_k *are real constants.*

To further simplify the problem we divide the interval $0 \leq x \leq 2\pi$ in equal parts, that is, we will use the partition

$$t_0 = 0, t_1 = \frac{2\,\pi}{N}, t_2 = \frac{4\,\pi}{N}, \cdots, t_k = \frac{2\,k\,\pi}{N}, \cdots, t_{N-1} = \frac{2(N-1)\pi}{N}.$$

Now we can use the real part of the DFT defined in the expression (4) to calculate the coefficients X_k for every k from the set of the sampled function values

$$\mathbf{x} = \{x_0, x_1, \ldots, x_{N-1}\}.$$

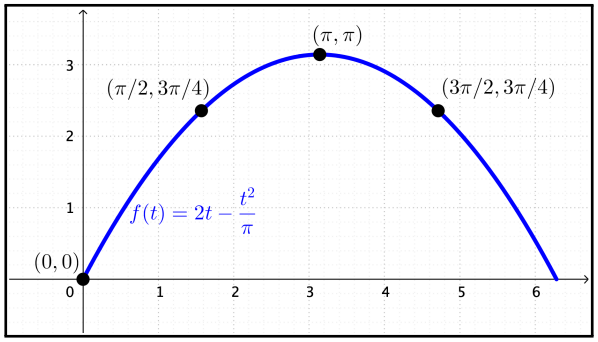


Figure 4. Graph of $f(t)$ with the values of $\mathbf{x}$.

For instance, for $N = 4$, we have the following values

$$\mathbf{x} = \left\{ f_0(0), f_1\left(\frac{\pi}{2}\right), f_2(\pi), f_3\left(\frac{3\pi}{2}\right) \right\} = \left\{ 0, \frac{3\pi}{4}, \pi, \frac{3\pi}{4} \right\}.$$

Using the real part of the DFT defined in (4) we have that

$$\begin{aligned}
X_0 &= \frac{1}{4}\left[0 \cdot \cos{(2\pi/4 \cdot 0 \cdot 0)} + (3\pi/4) \cdot \cos{(2\pi/4 \cdot 0 \cdot 1)}\right. \\
&\quad \left. +(\pi) \cdot \cos{(2\pi/4 \cdot 0 \cdot 2)} + (3\pi/4) \cdot \cos{(2\pi/4 \cdot 0 \cdot 3)}\right] = \frac{5\pi}{8} \\
X_1 &= \frac{1}{4}\left[0 \cdot \cos(2\pi/4 \cdot 1 \cdot 0) + (3\pi/4) \cdot \cos(2\pi/4 \cdot 1 \cdot 1)\right. \\
&\quad \left. +(\pi) \cdot \cos(2\pi/4 \cdot 1 \cdot 2) + (3\pi/4) \cdot \cos(2\pi/4 \cdot 1 \cdot 3)\right] = -\frac{\pi}{4} \\
X_2 &= \frac{1}{4}\left[0 \cdot \cos(2\pi/4 \cdot 2 \cdot 0) + (3\pi/4) \cdot \cos(2\pi/4 \cdot 2 \cdot 1)\right. \\
&\quad \left. +(\pi) \cdot \cos(2\pi/4 \cdot 2 \cdot 2) + (3\pi/4) \cdot \cos(2\pi/4 \cdot 2 \cdot 3)\right] = -\frac{\pi}{8} \\
X_3 &= \frac{1}{4}\left[0 \cdot \cos(2\pi/4 \cdot 3 \cdot 0) + (3\pi/4) \cdot \cos(2\pi/4 \cdot 3 \cdot 1)\right. \\
&\quad \left. +(\pi) \cdot \cos(2\pi/4 \cdot 3 \cdot 2) + (3\pi/4) \cdot \cos(2\pi/4 \cdot 3 \cdot 3)\right] = -\frac{\pi}{4}.
\end{aligned} \tag{9}$$

Therefore, the trigonometric interpolation is given by

$$p(t) = \frac{5\pi}{8} - \frac{\pi}{4}\cos t - \frac{\pi}{8}\cos 2t - \frac{\pi}{4}\cos 3t. \tag{10}$$

3.1 Implementation in GeoGebra

In the previous example we have used only four sampled points. If we want to consider more points, we need to use the computer to help us with all the computations. In this case, GeoGebra turns out to be a powerful tool since we can easily implement the DFT and calculate the trigonometric interpolation using mainly two commands `Sequence` and `Sum`. See (North American GeoGebra Journal Staff, 2021; GeoGebra, 2021c) for detailed tutorials on *list and sequence* commands.

Remark 3. *The code in each step can be typed directly in the input box of GeoGebra (in both versions online and desktop). The reader can also access the online interactive version in (Ponce Campuzano, 2021).*

Step 1: Define variable to set the number of points.

```
N = 4
```

Step 2: Define function. Here we use the command `If` to define the function in interval $[0, 2\pi]$. See (GeoGebra, 2021b) for details.

```
f(t) = If(0<=t<=2 pi, 2 * t - t^2 / pi)
```

Step 3: Define partition points.

```
Lx = Sequence(f(2 * k * pi / N), k, 0, N-1)
```

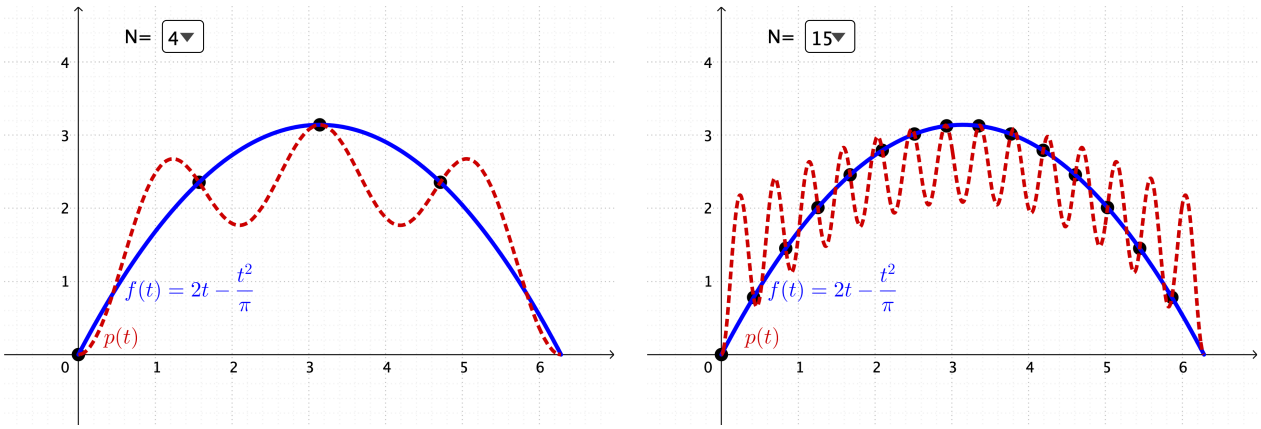
Step 4: Calculate the real component of the Fourier coefficients. Here we use the command `Element` to select an element from the list `Lx`. See (GeoGebra, 2021a) for details.

```
FReal = Sequence(1/N * Sum(Sequence(Element(Lx, n+1) * cos( 2 * pi / N * k * n),
    n, 0, N-1 )), k, 0, N-1)
```

Step 5: Define trigonometric interpolation.

```
p = If(0<=t<=2 pi, Sum(Sequence(Element(FReal, k+1) * cos(k * t), k, 0, N-1)))
```

In Figure 5, we can compare the function with the interpolation points indicated, and the discrete Fourier representations for both $N = 4$ and $N = 15$ points. In Figure 6 we can appreciate the frequency components. That is, each summand of the trigonometric interpolation $p(t)$, defined in (10), is plotted separately.



(a) $N = 4$. **(b)** $N = 15$.

Figure 5. Graph of $f(t)$ and the trigonometric interpolation $p(t)$.

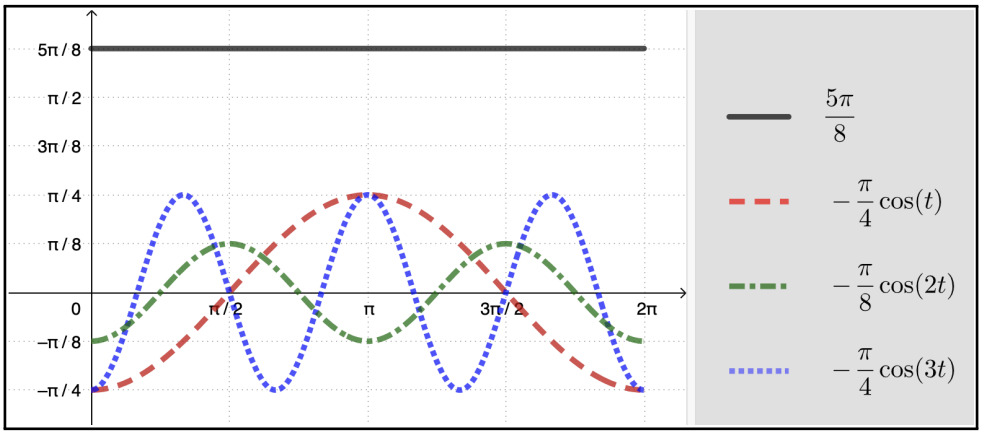


Figure 6. Frequency components of $p(t)$.

Notice that the resulting graphs point out a significant difficulty with the Discrete Fourier Transform. While the trigonometric polynomials do indeed correctly match the sampled function values, their pronounced oscillatory behaviour makes them completely unsuitable for interpolation away from the sample points.

However, this difficulty can be rectified. The problem is that we have not been paying sufficient attention to the frequencies that are represented in the Fourier sum. Indeed, the graphs in Figure 5 show a behaviour commonly known as *aliasing*, that is, low and high frequency exponentials can have the same sample data, but differ wildly in between the sample points. While the first half of the summands in (7) represent relatively low frequencies, the second half do not, and can be replaced by equivalent lower frequency, and hence less oscillatory exponentials. Namely, if $0 < k \leq \frac{1}{2}N$, then

$$e^{-ikt} \quad \text{and} \quad e^{i(N-k)t}$$

have the same sample values, but the former is of lower frequency than the latter. Thus, for interpolatory purposes, we can replace the second half of the summands in the Fourier sum (7) by their low frequency alternatives. If $N = 2m + 1$ is odd, then we take

$$\begin{aligned} \hat{p}(t) &= X_{-m}e^{-imt} + \cdots + X_{-1}e^{-t} + X_0 + X_1e^{it} + \cdots + X_me^{imt} \\ &= \sum_{k=-m}^{m} X_k e^{ikt} \end{aligned} \tag{11}$$

as the equivalent low frequency interpolation. On the other hand, if $N = 2m$ is even then

$$\begin{aligned} \hat{p}(t) &= X_{-m}e^{-imt} + \cdots + X_{-1}e^{-t} + X_0 + X_1e^{it} + \cdots + X_me^{i(m-1)t} \\ &= \sum_{k=-m}^{m-1} X_k e^{ikt}. \end{aligned} \tag{12}$$

In both cases, the Fourier coefficients with negative indices are the same as their high frequency alternatives

$$X_{-k} = X_{N-k}.$$

Returning to the previous example, for interpolatory purposes, we should replace (10) by the equivalent low frequency interpolation using the Discrete Fourier Transform with low frequencies for

$N = 2m$ even defined as

$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n \left[\cos\left(\frac{2\pi}{N}\,k\,n\right) - i\,\sin\left(\frac{2\pi}{N}\,k\,n\right)\right] \tag{13}$$

but for $k = -N/2, \ldots, N/2 - 1$. Thus we obtain the values

$$\mathbf{X} = \left\{-\frac{\pi}{8}, -\frac{\pi}{4}, \frac{5\pi}{8}, -\frac{\pi}{4}\right\}.$$

Now we can replace (10) by the equivalent low frequency interpolation

$$\hat{p}(t) = -\frac{\pi}{8}\cos(-2\cdot t) - \frac{\pi}{4}\cos(-1\cdot t) + \frac{5\pi}{8}\cos(0\cdot t) - \frac{\pi}{4}\cos(1\cdot t) \tag{14}$$

We can easily calculate in GeoGebra the low frequency coefficients and trigonometric interpolation for $N = 2m$ even. We just need to replace the steps 4 and 5 described above by the following:

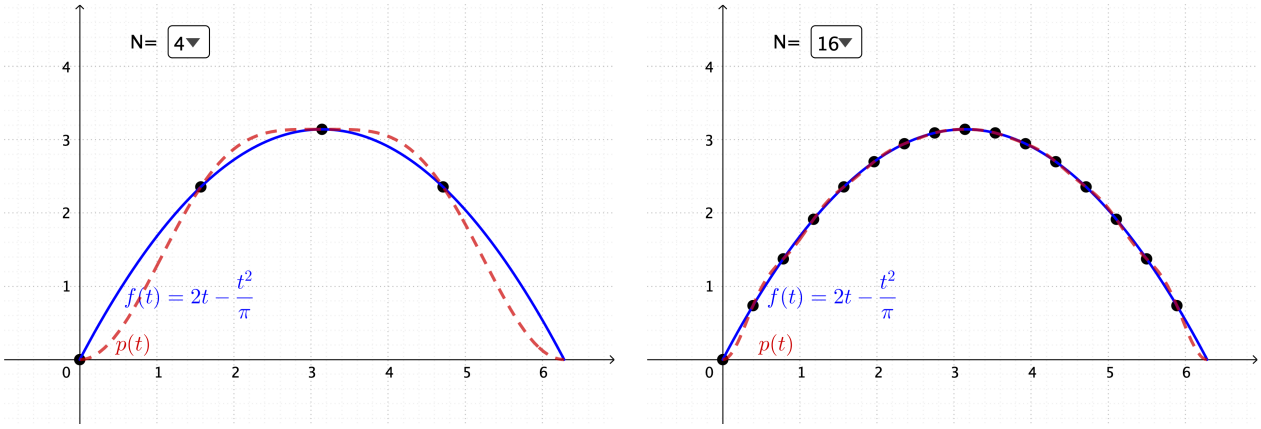
Step 4 (even): Real component of the Fourier coefficients with low frequency for N even.

```
FReEven = Sequence( 1/N * Sum(Sequence( Element(Lx, n+1) * cos(2 * pi / N * k *
    n), n, 0, N-1)), k, -N/2, N/2-1)
```

Step 5 (even): Define trigonometric interpolation for low frequencies for N even.

```
pEven = If(0<=t<=2 pi, Sum(Sequence(Element(FReEven, k + N/2 + 1) * cos(k * t),
    k, -N/2, N/2-1)))
```

Graphs of $\hat{p}(t)$ for $N = 4$ and $N = 16$ with low frequency trigonometric interpolations are shown in Figures 7a and 7b.



(a) $N = 4$. **(b)** $N = 16$.

Figure 7. Graph of $f(t)$ and the even trigonometric interpolation with low frequencies.

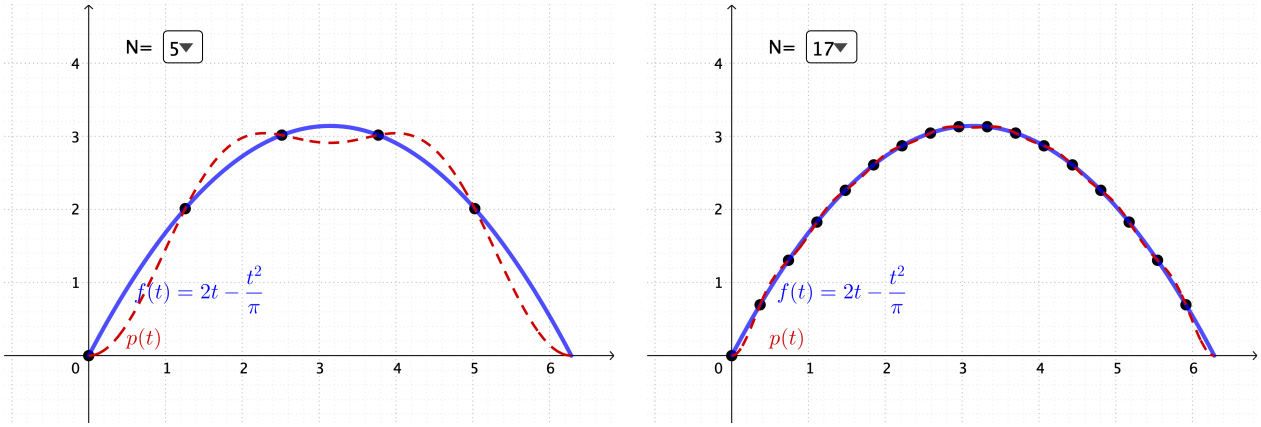
The graphs of $\hat{p}(t)$ for $N = 5$ and $N = 17$ are shown in Figures 8a and 8b. These were computed with the following:

Step 4 (odd): Real component of the Fourier coefficients with low frequency for N odd.

```
FReOdd = Sequence( 1/N * Sum( Sequence( Element(Lx, n+1) cos(2 * pi/N * k * n),
    n, 0, N-1)), k, -(N-1)/2, (N-1)/2)
```

Step 5 (odd): Define trigonometric interpolation for low frequencies for N odd.

```
pOdd = If(0<=t<=2 pi, Sum(Sequence(Element(FReOdd, k + (N-1)/2 + 1)*cos(k * t),
    k, -(N-1)/2, (N-1)/2)))
```



(a) $N = 5$. **(b)** $N = 17$.

Figure 8. Graph of $f(t)$ and the odd trigonometric interpolation with low frequencies.

Thus, by utilising only the lowest frequency exponentials, we have successfully suppressed the aliasing issues, resulting in a quite reasonable trigonometric interpolation to the given function f. Also, as mentioned in section 2, we have split f into a number of sinusoid functions, these are the frequency components of $\hat{p}(t)$, see Figure 9.

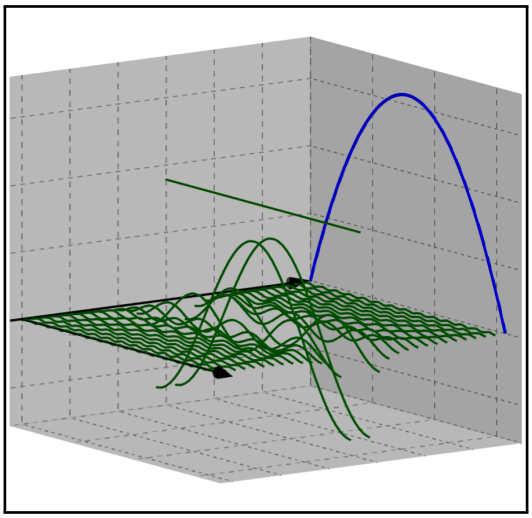


Figure 9. Frequency components of $\hat{p}(t)$ for $N = 30$.

4 THE CAT CURVE

We have approximated a real function defined on an interval using the DFT (with low frequencies). The most surprising thing here is that we do not really need to know the function f from the beginning. In fact, we only need to know a finite number of sampled function values

$$x_n = f(t_n), \quad n = 0, 1, 2, \dots N-1,$$

to reconstruct it by means of a sum of sinusoidal functions. Of course, the greater the number of sampled function values is, the more accurate the approximation will be.

Furthermore, the same procedure can be used to approximate parametric curves of the form

$$\begin{cases} u(t) \\ v(t) \end{cases} \quad t \in [0, 2\pi]. \tag{15}$$

which can rewritten in its complex form as $z(t) = u(t) + i\, v(t),\ t \in [0, 2\pi]$.
To do this we just need to consider a set of N points belonging to the parametric curve:

$$z(t_0), z(t_1), \cdots, z(t_{N-1}).$$

For each n, let $x_n = z(t_n)$. Thus we have a new sampled function of complex values

$$\mathbf{x} = \{x_0, x_1, x_2, \cdots, x_{N-1}\}.$$

Then we can apply the DFT to obtain the Fourier coefficient X_k which encodes the *amplitude* and *phase* of a sinusoidal wave with frequency k. Finally, we use the IDFT to calculate the interpolatory trigonometric polynomial. That is

$$z(t) = x(t) + i\, y(t) \approx \sum_{k=0}^{N-1} X_k \Big[\cos(k\, t) + i \sin(k\, t) \Big], \quad t \in [0, 2\pi]. \tag{16}$$

To obtain better results, it is recommended to use the low frequency version of both the DFT and IDFT.

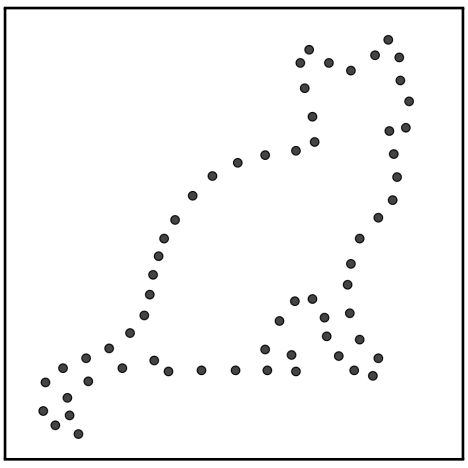
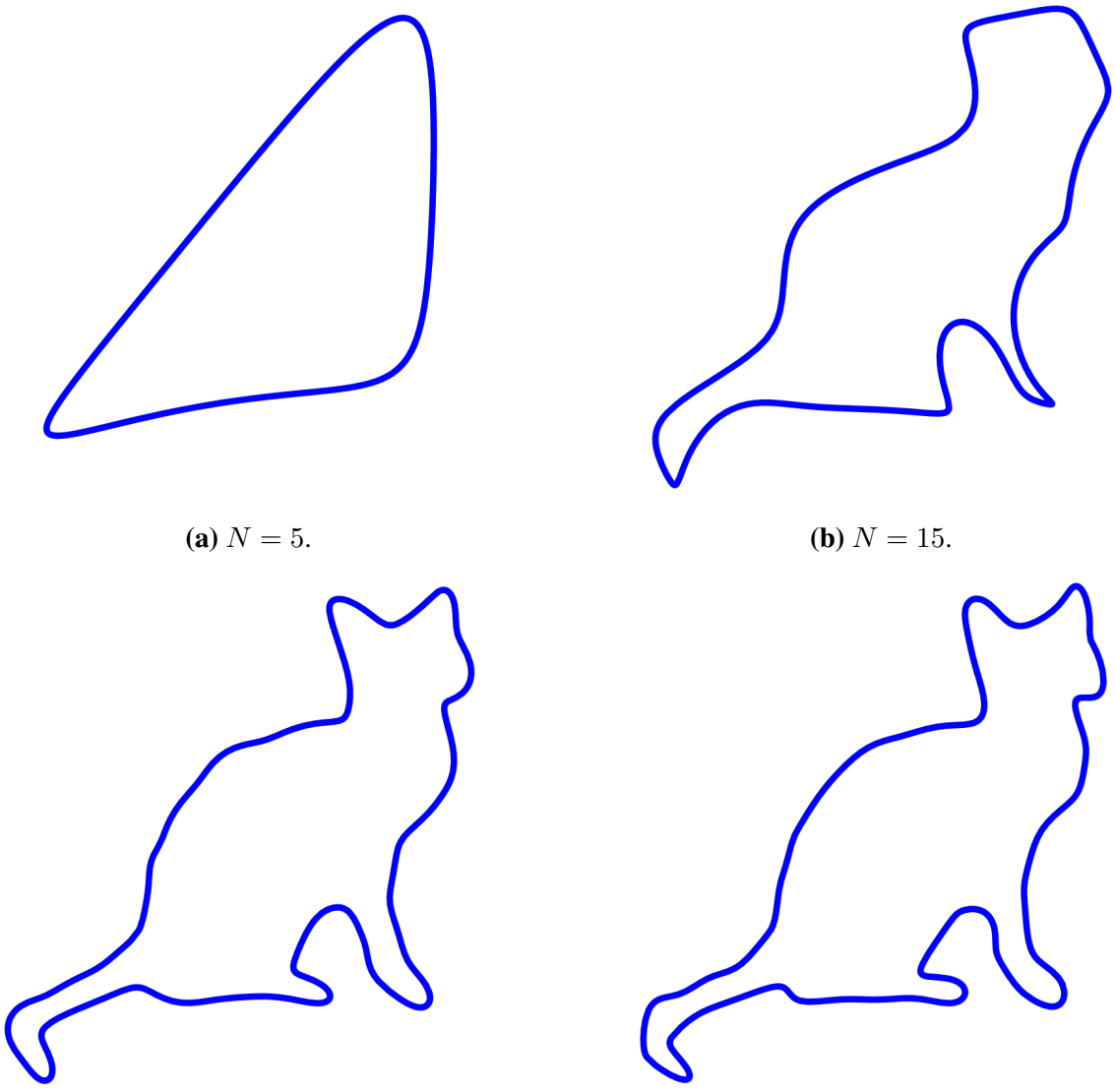


Figure 10. 63 points distributed approximately at equal space resembling the shape of a Cat.

For example, Figure 10 shows a set of 63 points on the plane resembling the shape of a Cat. If we calculate the trigonometric interpolation, we obtain the parametric curve with the shape of a Cat, as shown in Figure 11. The complete details about the construction and GeoGebra script can be consulted in Ponce Campuzano (2021). In this example, I have sorted the terms X_k by size (amplitude) to show that only using the largest coefficients X_k in the sum Fourier determines a quite close approximation to the Cat curve. The other coefficients become negligible from $N = 37$. This is in fact another application of the DFT in which shapes (in this case closed curves) can be adequately described by much fewer than N coefficients.



(a) $N = 5$.

(b) $N = 15$.

(c) $N = 37$.

(d) $N = 63$.

Figure 11. A trigonometric interpolation of a closed curve with the shape of a Cat.

5 FINAL COMMENTS

The Fourier Transform has several flavours (discrete/continuous/finite/infinite), covers deep and abstract mathematical concepts, and can easily overwhelm with detail. In this paper we have talked about how the Discrete Fourier Transform (and its version with low frequencies) allows us to split a periodic function into a number of sinusoid functions. In particular, we have shown, with the aid of GeoGebra, how to use the DFT to approximate periodic functions by means of trigonometric polynomials using only a finite number of sampled function values. This procedure has many applications in signal processing in our modern digital world. The examples presented here can be used potentially as an introductory learning activity to study, understand, and appreciate the profound insights of the DFT.

REFERENCES

Brigham, E. O. (1988). *The fast Fourier transform and it applications*. Prentice-Hall, Inc. USA.

Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation.*, 19:297–301.

GeoGebra (2021a). Element Command. `https://wiki.geogebra.org/en/Element_Command`. [Online; accessed 27-Dec-2021].

GeoGebra (2021b). If Command. `https://wiki.geogebra.org/en/If_Command`. [Online; accessed 27-Dec-2021].

GeoGebra (2021c). Sequence Command. `https://wiki.geogebra.org/en/Sequence_Command`. [Online; accessed 27-Dec-2021].

North American GeoGebra Journal Staff (2021). Lists and sequences. *North American GeoGebra Journal*, 9(1):25–30.

Ponce Campuzano, J. C. (2021). Discrete fourier transform for trigonometric interpolation. *GeoGebra*. `https://www.geogebra.org/m/hcsvacfj` [Online; accessed 27-Dec-2021].

Stein, E. M. and Shakarchi, R. (2003). *Fourier Analysis: An Introduction*. Princeton University Press Oxford.

Strang, G. (1994). Wavelets. *American Scientist*, 82(3):250–255.

Sundararajan, D. (2018). *Fourier Analysis - A Signal Processing Approach*. Springer Singapur.

Juan Carlos Ponce Campuzano, (jcponcemath@gmail.com), teaches mathematics and works on the design and integration of online learning modules and interactive mathematical applets for the School of Mathematics and Physics at the University of Queensland. Juan's professional interests include the design and construction of open source mathematics applets and online interactive books. Learn more about his projects at the following link: `https://www.jcponce.com/p/projects.html`.