TRACING CLOSED CURVES WITH EPICYCLES: A FUN APPLICATION OF THE DISCRETE FOURIER TRANSFORM

Juan Carlos Ponce Campuzano

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

Abstract

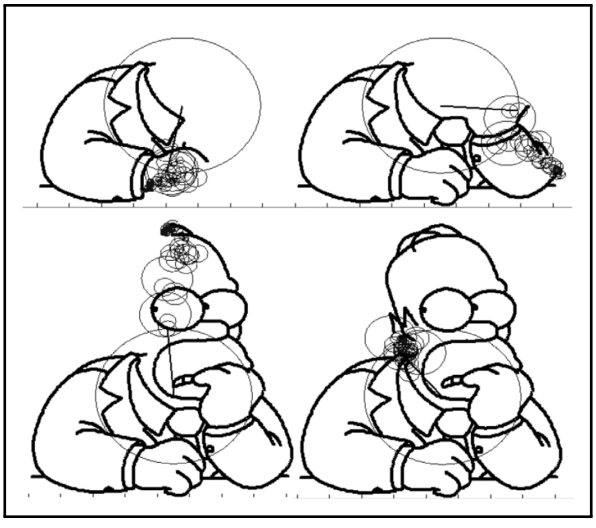
The Discrete Fourier Transform has many applications in our modern digital world. In particular, it allows us to approximate periodic functions by means of trigonometric polynomials which provides the required information to define a system of epicycles that can be animated to trace out closed curves. In this paper, I present a method in GeoGebra to create artistic animations consisting of systems of epicycles tracing out closed curves. The geometric construction presented here can also be used as an introductory learning activity to study the Discrete Fourier Transform from a geometric point of view.

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

1 INTRODUCTION

In 2008 Santiago Ginnobili y Christián C. Carman, inspired by the work of Hanson (1965) decades earlier, published an example of a system of epicycles tracing out an orbit resembling the famous cartoon character, Homer Simpson (Ginnobili and Carman (2008)).

Figure 1. Homer Simpson's orbit with 1000 epicycles by Ginnobili and Carman (2008).



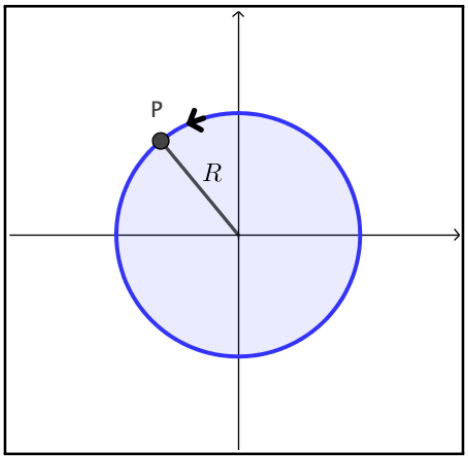
In that same year, Ginnobili published an online video with an animated version of this orbit in which we can observe an assemblage of 1000 circles with different radii, each one of them rotating at different speeds and slowly tracing out Homer Simpson's orbit, see Figure 1. To fully appreciate how striking this animation is, I highly recommend the reader to watch the video by Ginnobili (2008). This intricate animation relies on the deep insights arising from the Fourier Transform. More specifically on the Discrete Fourier Transform.

How did Ginnobili and Carman manage to make this animation? How can we calculate the radii of the circles? How can we calculate the speed at which each circle rotates? But most importantly, how is this animation related to the Discrete Fourier Transform? In this paper I will discuss the mathematics required to answer all these questions and I will provide a method to create in GeoGebra artistic animations consisting of systems of epicycles tracing out closed curves.

2 WHAT ARE EPICYCLES?

Consider a single point P on the edge of a rotating disk. If we traced it out for a full cycle, that point describes a circle:

Figure 2. The point P with an anticlockwise rotation describes a circle.



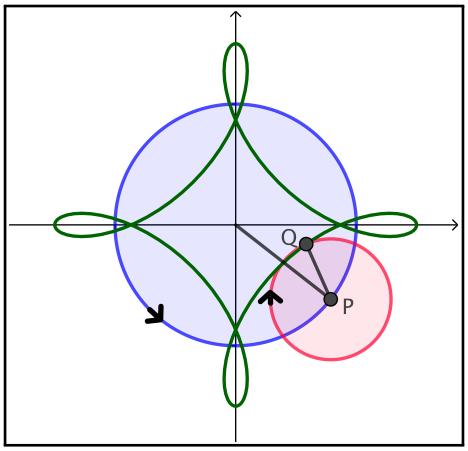
We can represent the motion of this point P considering the center of the circle as the origin of the Cartesian coordinate system and using parametric equations:

$$\begin{cases} x(t) &= R\cos(\omega t) \\ y(t) &= R\sin(\omega t), \end{cases}$$

where R is the radius of the disk and ω is the speed at which the disk is rotating. If ω is positive, the rotation is anticlockwise, and if it is negative, the rotation is clockwise.

Now let's look at something a bit more complex: Let's add another disk with radius half of the original disk and centred at P (point rotating on the original disk) but this time rotating clockwise (three times faster) with its own point Q rotating on its circumference. If we trace the locus of this second point, we can observe a shape resembling a flower, shown in Figure 3.

Figure 3. The point Q describes a flower curve.

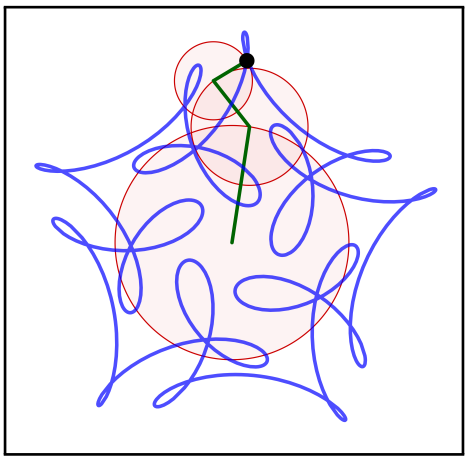


Luckily, it is not difficult to describe this new path: the overall position is just the sum of the contributions from each disk. More formally, the point Q at time t is the position at time t due to the first disk plus the position at time t due to the second disk. In other words

$$\begin{cases} x(t) = R_1 \cos(\omega_1 t) + R_2 \cos(\omega_2 t) \\ y(t) = R_1 \sin(\omega_1 t) + R_2 \sin(\omega_2 t) \end{cases}$$

Here R_1 is the radius of the first, inner disk and R_2 is the radius of the second, outer disk. We'll be using subscript notation throughout to keep track of both the radii of the disks (R_k) and the speed that they're rotating at ω_k. Thus, in the example shown in Figure 3 we used the values $R_1 = 1$, $R_2 = \frac{1}{2}$, $\omega_1 = 1$, and $\omega_2 = -3$.

Figure 4. A more complex curve traced out by epicycles.



By adding extra disks and varying their speeds and sizes, you can get increasing complex curves, called *epicycles*. For example, the curve shown in Figure 4 is defined as

$$\begin{cases} x(t) = R_1 \cos(\omega_1 t) + R_2 \cos(\omega_2 t) + R_3 \cos(\omega_3 t) \\ y(t) = R_1 \sin(\omega_1 t) + R_2 \sin(\omega_2 t) + R_3 \sin(\omega_3 t) \end{cases}$$

where

$$R_1 = 1, R_2 = \frac{1}{2}, R_3 = \frac{1}{3}, \omega_1 = 1, \omega_2 = 6, \text{ and } \omega_3 = -14.$$

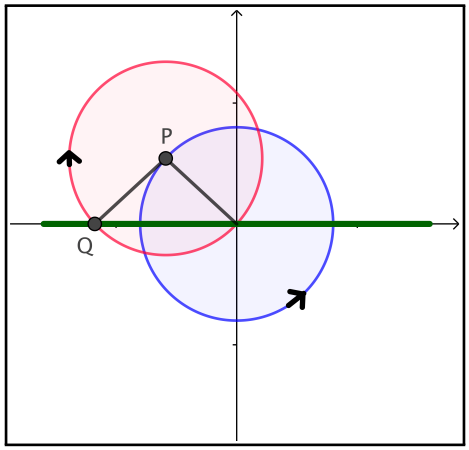
In general, epicycles are curves defined by the equations:

$$\begin{cases} x(t) = \sum_{k=1}^{N} R_k \cos(\omega_k t + \phi_k) \\ y(t) = \sum_{k=1}^{N} R_k \sin(\omega_k t + \phi_k) \end{cases}$$

This new symbol ϕ_k indicates how much the disk k is initially rotated at time $t = 0$, and we called a *phase offset*. If we do not specify a phase offset, we can only describe epicycles where the circles start aligned.

At first glance, it seems that the geometric shapes described by epicycles are smooth closed curves. But, do all epicycles have to be *curvy* or closed? To answer this question, consider two disks with the same radius rotating in opposite directions. With this setup we can create a straight line:

Figure 5. The point Q a straight line.



In this case, because the sizes of the circles are the same, the y components cancel each other out, and we get:

$$\begin{cases} x(t) = R\cos(\omega t) + R\cos(-\omega t) = R\cos(\omega t) + R\cos(\omega t) = 2R\cos(\omega t) \\ y(t) = R\sin(\omega t) + R\sin(-\omega t) = R\sin(\omega t) - R\sin(\omega t) = 0 \end{cases}$$

Here $y(t)$ is always zero. This means that we draw a straight line along the x axis. Thus, we can create epicycles that are simple curves like a circle, a simple straight line, a flower curve, and crazy complex curves like the one shown in Figure 4). Furthermore, it seems like the more disks we have, the more complex the curves can get.

In this context, if we have enough disks, and we choose the right sizes for each of the disks, and have them spin at the appropriate speeds, can we create any closed shape/curve? Like a cat or the greek letter π, shown in Figures 6 – 7.

Figure 6. Epicycles tracing a cat curve.

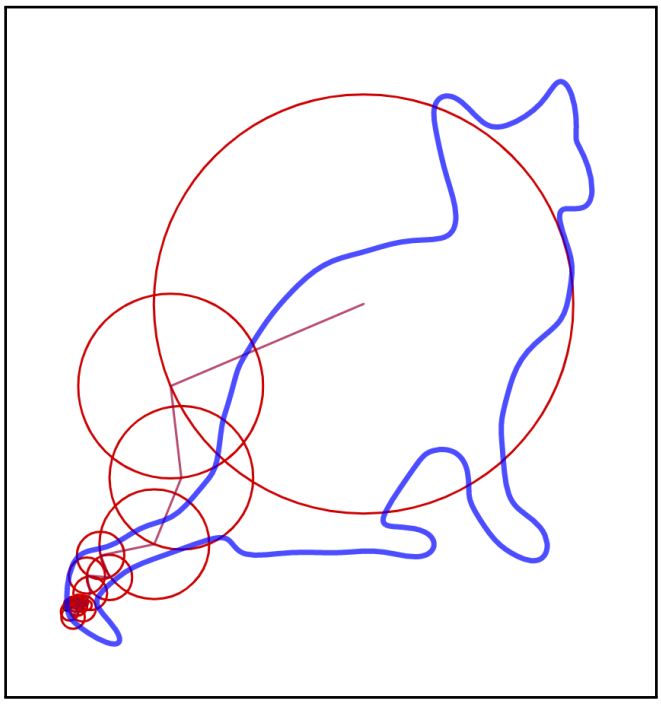
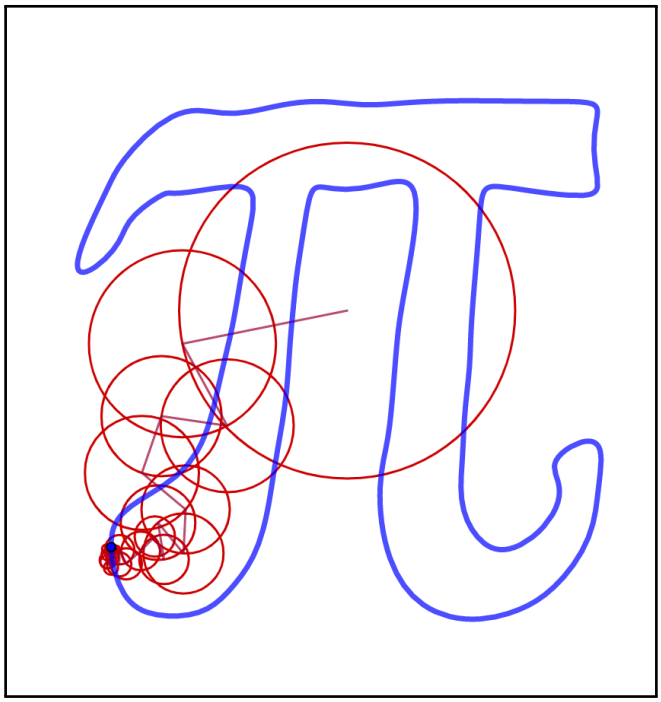


Figure 7. Epicycles tracing the Greek letter π.



As the reader may have guessed by this point, the answer is yes! Using epicycles, we can draw any closed shape, including Homer Simpson, as Ginnobili and Carman did back in 2008.

There are still a couple of unanswered questions though - how exactly do we determine what disks to use, and how fast to spin them? For the epicycle drawing of the Greek letter π above, how do we determine the values for the various R_k, ω_k, and ϕ_k to make the drawing come out right? Let's answer these questions.

3 HOW CAN WE TRACE OUT CLOSED CURVES WITH EPICYCLES?

In the previous section we saw how to trace out continuous closed curves using rotating disks and varying their speed and sizes. The problem we want to solve now is: *Given any continuous close curve/shape, is it possible to trace it out using epicycles?*

To start, instead of describing the continuous curve/shape that we want using parametric equations, let's say that we just specify some points, and then find a formula for *connecting those points* into the shape that we want. This means that our input is a finite set of points (x_k, y_k), and we want to find epicycles that best connect those points. That is, we want to determine the values of R_k, ω_k, and ϕ_k so the expression

$$\begin{cases} x(t) = \sum_{k=1}^{N} R_k \cos(\omega_k t + \phi_k) \\ y(t) = \sum_{k=1}^{N} R_k \sin(\omega_k t + \phi_k) \end{cases} \tag{1}$$

comes as close to the points as possible. Then we can use the points to draw out the shape we want, and consequently we will have our epicycles. But how can we determine the values of R_k, ω_k, and ϕ_k?

Surprisingly, we can use the *Discrete Fourier Transform* (DFT) to solve this problem in a way that is not only elegant, but also very easy for computers to perform. In this case, instead of thinking about our points defined on the real plane, we can think about those points as complex numbers. Thus we can write

$$p_k = x_k + iy_k$$

and the closed curve

$$x(t) + iy(t) = \sum_{k=1}^{N} R_k(\cos(\omega_k t + \phi_k) + i \sin(\omega_k t + \phi_k)) \tag{2}$$

will come as close to the complex points p_k as possible. Notice also that equation (2) can be rewritten, by using Euler's formula $e^{it} = \cos t + i \sin t$, as follows:

$$x(t) + iy(t) = \sum_{k=1}^{N} R_k e^{i\omega_k t + \phi_k}.$$

Even better, if we allow X_k to be a complex number, then the formula above becomes:

$$\sum_{k=1}^{N} X_k e^{i\omega_k t}.$$

Finally, rather than choosing N arbitrary speeds ω_k, we will make the speeds to be

$$0 \text{ (stationary)}, 1x, -1x, 2x, -2x, 3x, -3x, \ldots N/2x, -N/2x.$$

With this simplification, we can state our problem as follows: Find the complex numbers X_k to minimize the *mean squared error* between the complex points p_k and the curve

$$p(t) = \sum_{k=0}^{N} X_k e^{\frac{2\pi i k}{N} t}.$$

As it turns out, the form above almost identically matches the form of the DFT and the process described above is known as *trigonometric interpolation*.

4 Drawing epicycles to trace out closed curves using the Discrete Fourier Transform

The Discrete Fourier Transform and its inverse are defined as

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

$$\text{IDFT:} \quad x_n = \sum_{k=0}^{N-1} X_k \cdot e^{ik\frac{2\pi}{N}n}, \quad n = 0, 1, \ldots, N-1 \tag{4}$$

which can be rewritten, by using Euler's formula, as follows

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

$$x_n = \sum_{k=0}^{N-1} X_k \left[\cos\left(k\,\frac{2\pi}{N}\,n \right) + i\,\sin\left(k\,\frac{2\pi}{N}\,n \right) \right] \tag{6}$$

It is well-known that the DFT can be used for *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. In particular, we can use the DFT to approximate periodic functions by means of trigonometric polynomials using only a finite number of sampled function values, and the same procedure works for approximating parametric curves of the form

$$\begin{cases} u(t) \\ v(t) \end{cases} \qquad t \in [0, 2\pi]. \tag{7}$$

which can rewritten in its complex form as $z(t) = u(t) + i\,v(t),\ t \in [0, 2\pi]$ (see Ponce Campuzano (2022) or Stein and Shakarchi (2003) for more details).

First, consider a set of N points belonging to the parametric curve

$$\begin{aligned}\mathbf{x} &= \{u_0 + i\,v_0, u_1 + i\,v_1, \cdots, u_{N-1} + i\,v_{N-1}\} \\ &= \{x_0, x_1, x_2, \cdots, x_{N-1}\}.\end{aligned}$$

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. This is the necessary information to create a system of epicycles that will trace out the parametric curve (7).
The radii of each circle is the amplitude of X_k, that is, its *modulus*. This can be calculated with the formula

$$R_k = \left|X_k\right| = \left|\mathbf{Re}(X_k) + i\,\mathbf{Im}(X_k)\right| = \sqrt{(\mathbf{Re}(X_k))^2 + (\mathbf{Im}(X_k))^2} \tag{8}$$

where $\mathbf{Re}(X_k)$ and $\mathbf{Im}(X_k)$ are the *real* and *imaginary* components of X_k. On the other hand, the phase of X_k is its *argument*, that is

$$\phi_k = \mathbf{Arg}(X_k) = \arctan\left(\frac{\mathbf{Im}(X_k)}{\mathbf{Re}(X_k)}\right) \tag{9}$$

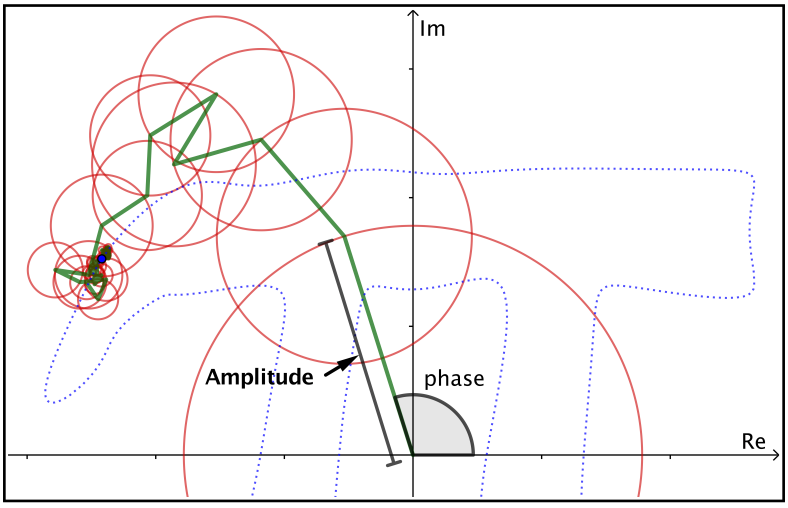
Finally, using the IDFT, we can calculate the trigonometric polynomial to approximate the parametric curve $z(t)$.

The image shown in Figure 8 demonstrates how a simple sum of complex numbers in terms of phases/amplitudes can be nicely visualized as a set of concatenated circles in the complex plane. Each radius is a vector representing a term in the sum:

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

Adding the summands corresponds to simply concatenating each of these red vectors in complex space.

Figure 8. Geometric interpretation of the addition of complex numbers as vectors.



4.1 The square orbit

To see how this works we will determine a system of epicycles tracing out a square curve and find its interpolatory trigonometric polynomial.

Step 1: Define dataset

First we defined the sample values:

$$\begin{aligned} x \;=\; \{&(2,2),\ (2,3/2),(2,1),(2,1/2),(2,0),(2,-1/2),(2,-1),(2,-3/2),(2,-2),(3/2,-2),\\ &(1,-2),(1/2,-2),(0,-2),(-1/2,-2),(-1,-2),(-3/2,-2),(-2,-2),(-2,-3/2),\\ &(-2,-1),(-2,-1/2),(-2,0),(-2,1/2),(-2,1),(-2,3/2),(-2,2),(-3/2,2),\\ &(-1,2),(-1/2,2),(0,2),(1/2,2),(1,2),\ (3/2,2)\} \end{aligned}$$

The points in our dataset form a loop distributed approximately at equal space around the boundary of a square of side 4.

Step 2: Calculate the coefficients X_k

The number of elements in our dataset is $N = 32$, so we need use the low frequency version of the DFT for $N = 2m$ even. The reader can easily adjust this equation for $N = 2m + 1$ odd (for more details about this topic see Ponce Campuzano (2022)). Thus we have that

$$X_k \;=\; \frac{1}{N}\sum_{n=-m}^{m-1} x_{n+m}\left[\cos\left(k\,\frac{2\pi}{N}\,n\right) - i\,\sin\left(k\,\frac{2\pi}{N}\,n\right)\right] \tag{10}$$

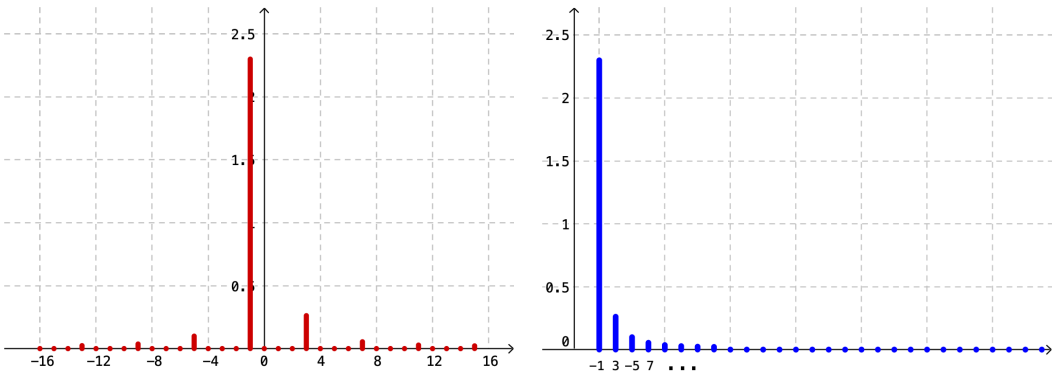
for $k = -\left(\frac{N}{2}\right), -\left(\frac{N}{2}-1\right), \ldots, \frac{N}{2}-1$.

Step 3: Calculate modulus, phase and frequency of each X_k

The *modulus* and *phase* are calculated using equations (8) and (9). The frequency in this case is $k = -\left(\frac{N}{2}\right), -\left(\frac{N}{2}-1\right), \ldots, \frac{N}{2}-1$. The frequency domain is shown in Figure 9 (a).

Figure 9. Frequency domain.

(a) Unsorted. **(b)** Sorted by magnitude.



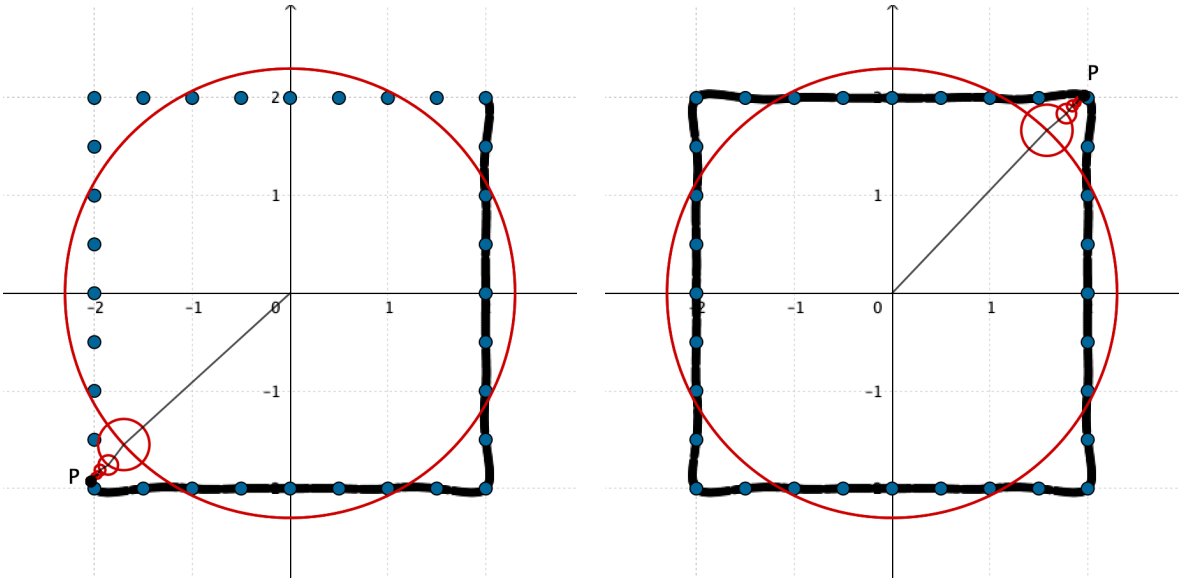
Step 4: Define systems of epicycles

With the information from Step 3 we can now define the required complex numbers (considered as vectors) to add them together, tip to tail, to construct the system of epicycles.

Remark 1. *Here we can sort the coefficients X_k by its amplitude from the largest to the lowest so have a better depiction of the system of epicycles. In Figure 9b is plotted the sorted frequency domain.*

Thus we obtain a system of epicycles sorted by size which is shown in Figure 10. The image shows the trace of the resulting point P at different positions on its cycle. Notice how this point traces out the closed curve passing through each point defined in our dataset.

Figure 10. Almost completing a full cycle.



(a) Half of a cycle. **(b)** System of epicycles tracing out a square curve.

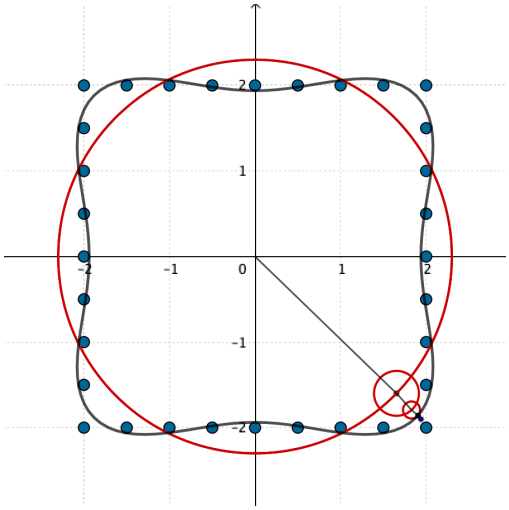
Step 5: Calculate the interpolatory trigonometric polynomial

Finally, we use the low frequencies version of the IDFT for $N = 2m$ even to calculate the interpolatory trigonometric polynomial. The reader can easily adjust this equation for $N = 2m + 1$ odd (for more details see Ponce Campuzano (2022)). Then

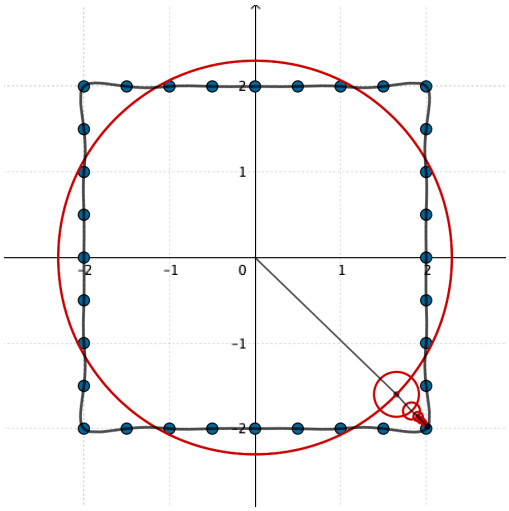
$$z(t) = x(t) + i\,y(t) \approx \sum_{k=-m}^{m-1} X_{k+m}\Big[\cos(k\,t) + i\sin(k\,t)\Big], \quad t \in [0, 2\pi]. \tag{11}$$

In Figure 11 we can appreciate the parametric curve for different values of N. Notice also that since we have sorted the terms by size (amplitude of X_k), we can appreciate that only using the largest coefficients X_k in the sum Fourier sum determines a quite close approximation to the square curve. The other coefficients become negligible from $N = 9$. 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.

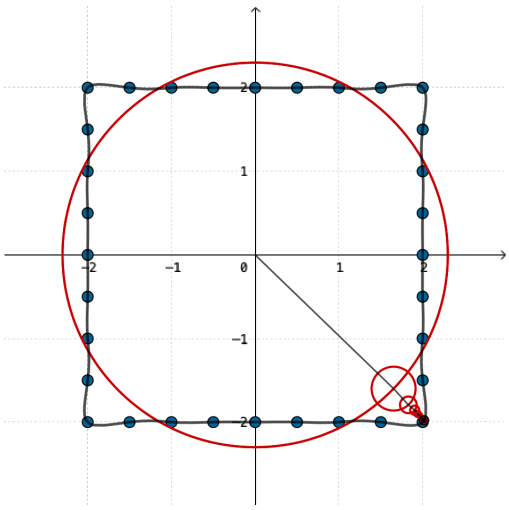
Figure 11. System of epicycles tracing out a square curve.



(a) $N = 3$.



(b) $N = 8$.



(c) $N = 32$.

5 FINAL COMMENTS

The intricate animations of orbits being traced out by a system of epicycles relies on the basic geometric interpretation of sums of complex numbers due to the application of the Discrete Fourier Transform. In this paper, we have shown how we can use the DFT to obtain the necessary information to construct a system of epicycles for tracing out closed curves.

Since Ginnobili and Carman published online their animation of Homer Simpson's orbit, this fun application of the DFT has become quite popular for people interested in mathematics and its applications. Nowadays it is common to find on the internet people sharing their implementation of these animations using different programming languages; some examples are Azad (nd); Sanderson (2019); Shiffman (2019); Swanson (2019).

If the number of points in the dataset is less than 300, GeoGebra can handle easily all the calculations. However, if the number of sample points is greater than 300, I recommend to use a different mathematical software or programming language. To fully appreciate the construction and animation, I also recommend the reader to interact with the live demos and GeoGebra source code available online in Ponce Campuzano (2018). I also included the GeoGebra script for the square curve in the Appendix – Code.

Figure 12. Epicycles tracing out the GeoGebra logo.

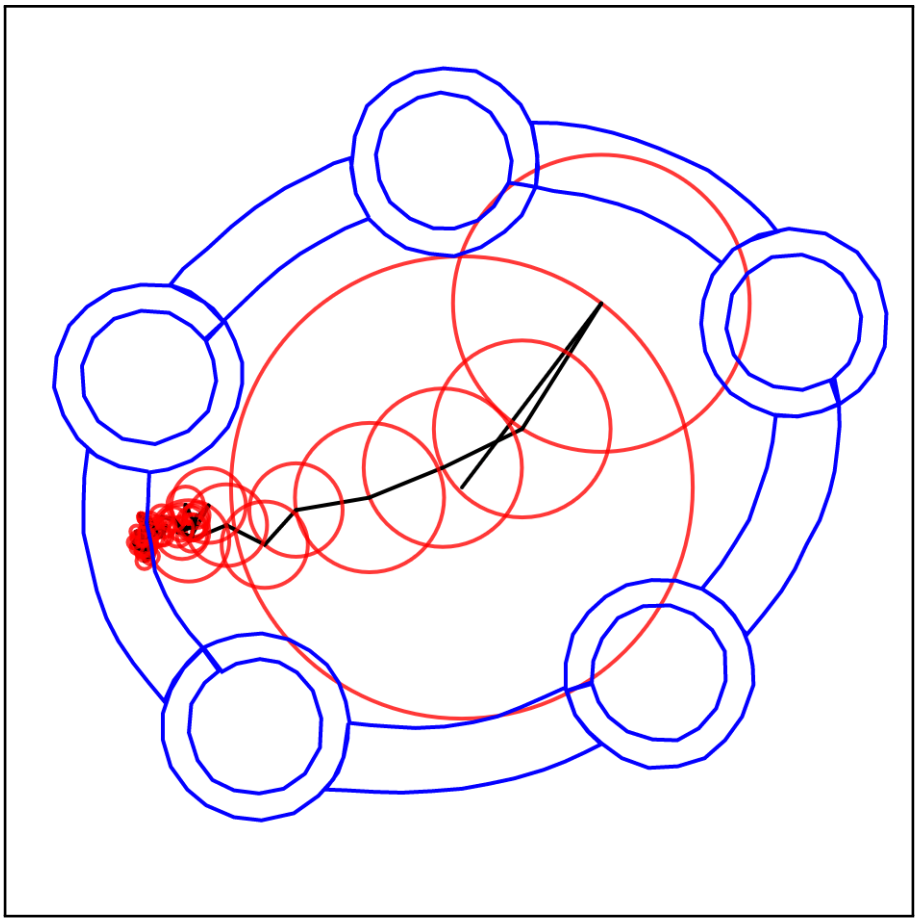


Figure 13. Rainbow epicycles tracing out a T-rex using the method described in Ponce Campuzano (2021).

ACKNOWLEDGMENT

The author wishes to thank Thijs for his assistance to improve the GeoGebra script and performance of the online applets. In addition, he would like to thank the reviewers for their suggestions to improve this paper, and the NAGJ editors for their support and assistance.

REFERENCES

Azad, K. (n.d.). An interactive guide to the fourier transform. *Better Explained*. `https://tinyurl.com/36d722j2` [Online; accessed 25-Oct-2022].

Ginnobili, S. (2008). Ptolemy and Homer (Simpson). `https://youtu.be/QVuU2YCwHjw`. [Online; accessed 25-Oct-2022].

Ginnobili, S. and Carman, C. C. (2008). Deferentes, epiciclos y adaptaciones. *Associação de Filosofia e História da Ciência do Cone Sul (AFHIC)*, 5:399–408.

Hanson, N. R. (1965). The mathematical power of epicyclical astronomy. *Isis*, 51:150–158.

Ponce Campuzano, J. C. (2018). Fourier series and discrete fourier transform. *GeoGebra*. `https://www.geogebra.org/m/t9uspumz#chapter/501202` [Online; accessed 25-Oct-2022].

Ponce Campuzano, J. C. (2021). On coloring different objects of the same class. *North American GeoGebra Journal.*, 9:31–35.

Ponce Campuzano, J. C. (2022). Trigonometric interpolation using the discrete fourier transform. *North American GeoGebra Journal.*, 10:1–13.

Sanderson, G. (2019). But what is a fourier series? from heat flow to circle drawings. https://youtu.be/r6sGWTCMz2k. [Online; accessed 25-Oct-2022].

Shiffman, D. (2019). Coding challenges #130.1, #130.2 and #130.3: Drawing with fourier transform and epicycles. https://youtu.be/MY4luNgGfms. [Online; accessed 25-Oct-2022].

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

Swanson, J. (2019). An interactive introduction to fourier transforms. http://www.jezzamon.com/fourier/index.html. [Online; accessed 25-Oct-2022].

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.

APPENDIX - CODE

```
#=======================================
#  Define discrete data signal and Setup
#=======================================
Sx = {(2, 2), (2, 3/2), (2, 1), (2, 1/2), (2, 0), (2, -1 / 2), (2, -1), (2,-3/2)
    , (2, -2), (3/2, -2), (1, -2), (1 / 2, -2), (0, -2), (-1/2, -2), (-1,-2),
    (-3/2, -2), (-2, -2), (-2, -3/2), (-2, -1), (-2, -1 / 2), (-2, 0), (-2,1 / 2)
    , (-2, 1), (-2, 3 / 2), (-2, 2), (-3 / 2, 2), (-1, 2), (-1 / 2, 2), (0,2), (1
     / 2, 2), (1, 2), (3 / 2, 2)}

N = Length(Sx)
LN = Sequence(N)

#=========================================================
#  Shift zero-frequency component to centre of spectrum
#=========================================================

Lk = LN - 1 - Div(N, 2)

#=========================================
#  Calculate DFT: LX is a list of frequencies
#=========================================

LX = 1 / N * Zip(Sum(Zip((abs(P); arg(P) - 2 * pi * k * (n-1) / N), P, Sx, n, LN
    )), k, Lk)

#=========================================
#  Lj : frequency index sorted by size (abs)
#=========================================

LAs = Reverse(Sort(Zip((abs(X), n), X, LX, n, LN)))
Lj = Zip(y(As), As, LAs)

#=======================================
#  Use the first M greatest frequencies
#=======================================

M  = Slider(1, N, 1, 1, 160, false, true, false, false)
SetValue(M, N)
Lm = Sequence(1, M)

Lks = First(Zip(Element(Lk, j), j, Lj), M)
LXs = First(Zip(Element(LX, j), j, Lj), M)
LRs = First(Zip(x(As), As, LAs), M)

#=========================
#  Plot M epicycles
#=========================

speed = 0.5
t  = Slider(0, 2 * pi, 0.0099, speed, 150, false, true, false, false)

LC1= Zip(Sum(Zip((abs(X); arg(X) + k * t), X, First(LXs, m), k, Lks)), m, Lm )
LC = Join({(0, 0)}, LC1)
```

```
Plast = Last(LC)
PolyL = Polyline(LC)
Epicycles = Zip(Circle(Element(LC, m), R), m, Lm, R, LRs)

#=============================================================
#  Parametric curve: Inverse of Discrete Fourier Transform
#=============================================================

fx(x) = Sum(Zip( x(X) * cos(k * x) - y(X) * sin(k * x), X, LXs, k, Lks))
fy(x) = Sum(Zip( x(X) * sin(k * x) + y(X) * cos(k * x), X, LXs, k, Lks))
orbit = Curve(fx(t), fy(t), t, 0, 2 * pi)

#==========================
#  Extras: Settings
#==========================

SetValue(t, 0)
StartAnimation(t, true)

SetCaption(M, "n")
SetLabelMode(M, 9)
SetVisibleInView(M, 1, true)
SetVisibleInView(M, 2, false)

SetVisibleInView(t, 1, true)
SetVisibleInView(t, 2, false)

SetVisibleInView(LX , 1, false)
SetVisibleInView(LAs, 1, false)
SetVisibleInView(LXs, 1, false)
SetVisibleInView(LC1, 1, false)
SetVisibleInView(LC , 1, false)
SetVisibleInView(fx, 1, false)
SetVisibleInView(fy, 1, false)

ShowLabel(PolyL, false)
ShowLabel(orbit, false)
```