**The Beauty of Fourier**

[[Juan Vera](https://medium.com/@vxnuaj?source=post_page-----65389914ab2f--------------------------------)](https://medium.com/@vxnuaj?source=post_page-----65389914ab2f--------------------------------)

[Juan Vera](https://medium.com/@vxnuaj?source=post_page-----65389914ab2f--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F8e7494ed753a&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40vxnuaj%2Fthe-beauty-of-fourier-65389914ab2f&user=Juan+Vera&userId=8e7494ed753a&source=post_page-8e7494ed753a----65389914ab2f---------------------post_header-----------)

15 min read

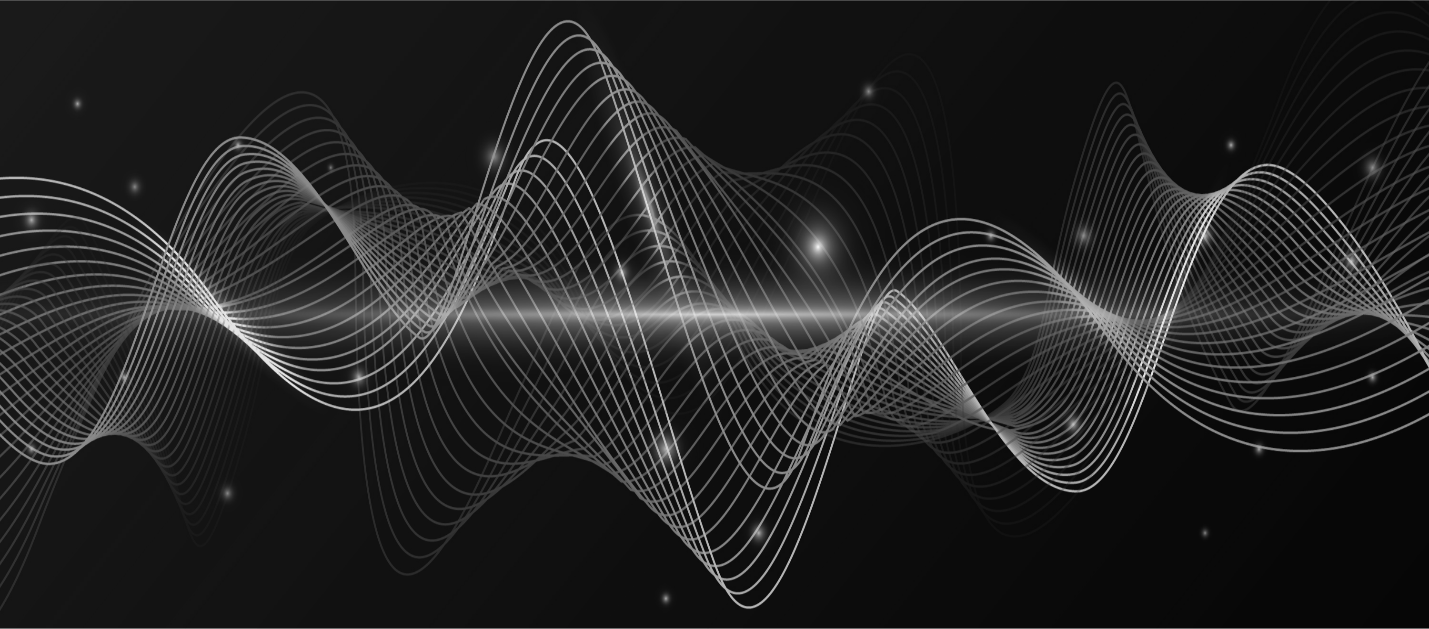
·

Jan 20

441

3

*An in depth look into principles & mathematics behind****Fourier Analysis***



The [**Fourier** **series**](https://en.wikipedia.org/wiki/Fourier_series) & it’s accompanying derivations are one of the **most** **fascinating** **real** **world** **applications** of mathematics.

I’ve always been a proponent of understanding mathematics to understand the world around us.

From using [**linear algebra**to design a **neural network**](https://www.youtube.com/watch?v=w8yWXqWQYmU&t=1231s), [**chaos theory**](https://www.pnas.org/doi/10.1073/pnas.231384098) to understand the **solar system,**and [**string theory**](https://www.youtube.com/watch?v=kF4ju6j6aLE)to understand the fundamental **building blocks of the universe**, mathematics is everywhere.

Now of course, those are the more complex & technical applications of mathematics.

*But what about applied math to our lives?*

Well, perhaps you listen to binaural beats while you study. Or you’ve gotten an x-ray done before. Or perhaps you’ve seen an Electroencephalogram (EEG) test in action.

Whatever the case may be, there’s always been mathematical algorithms that facilitate their use.

*But which algorithm(s)?*

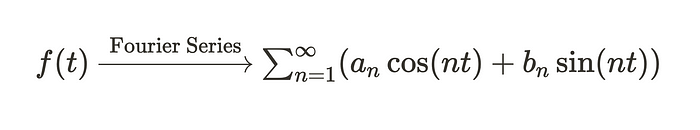
Welp, one of them is the [**Fourier series**](https://en.wikipedia.org/wiki/Fourier_series)(& it’s derivations)**.**

**A Touch of Intuition**

At it’s core, the Fourier series & its derivations are mathematical algorithms designed to transform a signal from the **time** **domain** to the **frequency** **domain**.

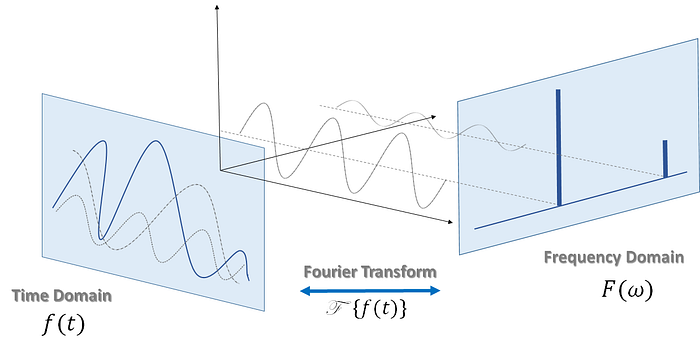
*A signal in the****time******domain****is analyzed over a period of time.  
A signal in the****frequency******domain****is analyzed over a power spectrum of frequencies*

It does so by taking in a signal function and decomposing it into a summation of sinusoidal functions that have varying frequencies are varying levels of power.



A Very Basic Representation of the Fourier Series.

From the Fourier Series, we can derive the **Continuous** [**Fourier Transform**](https://en.wikipedia.org/wiki/Fourier_transform), which can allow us to isolate individual frequency components & their corresponding power that make up a signal.



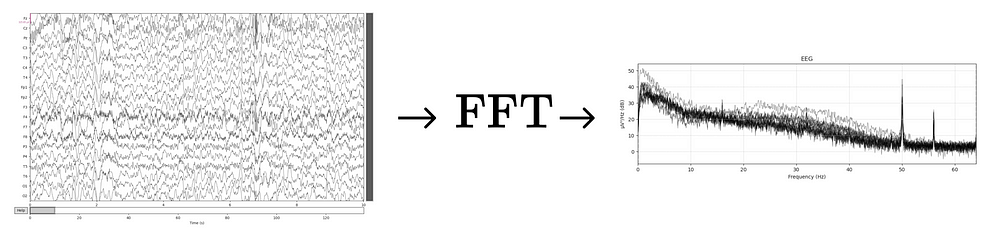
A Visualization of the Fourier Transform.

This transform is extremely useful as once we convert a signal from the time domain to the frequency domain, we can remove unwanted frequency components from our signal.

As an example, say we use EEG to record brain signals for analysis of ADHD.

Typically, signals that are recorded through EEG are very noisy and aren’t very reliable — at least the **raw EEG signal** isn’t.

Well we can use another derivation of the Fourier series, [**the Fast Fourier Transform**](https://en.wikipedia.org/wiki/Fast_Fourier_transform)*,*in order to break down the raw EEG signal from the time domain into the frequency domain.

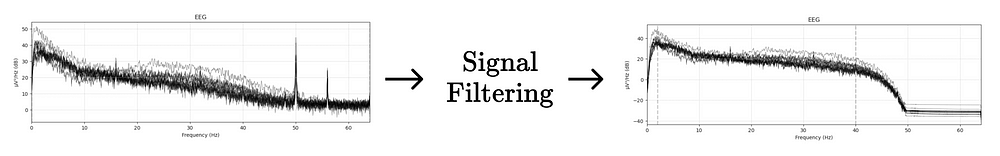


Time Domain (left) & Frequency Domain (right)

From there, we can remove a specific frequency or frequency ranges from our dataset by implementing bandpass, bandstop, or notch filters.

You can clearly see HUGE spikes in the frequency domain of the EEG dataset (which are likely from power line interference).

Those are the frequencies we remove by applying filters.



Filtering an EEG Dataset | Thanks to FFT.

*This is exactly what I did on a previous project to characterize****ADHD****.  
If you’re curious, feel free to check it out in this*[***article***](https://medium.com/@vxnuaj/using-mne-to-characterize-adhd-d5540438dcf3)*or*[***video***](https://youtu.be/uzCuzB5ixn8?si=Z8ADqo_YL1C5obMS)***.***

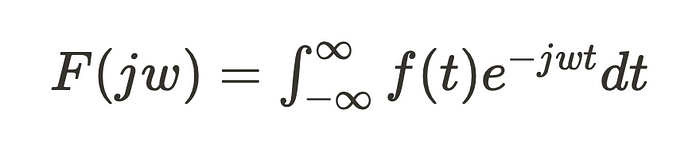
The same can be done in a multitude of other applications such as audio engineering, image processing, medical diagnostics, and speech recognition.

**It isn’t limited.**

Now, you may have noticed that I mention different derivations of the Fourier Series, some of them being the **Fourier Transform** and the **Fast Fourier Transform.**

Different derivations of the Fourier Series are applicable in different contexts. Some are more practical in use than others and some are more for mathematical theory.

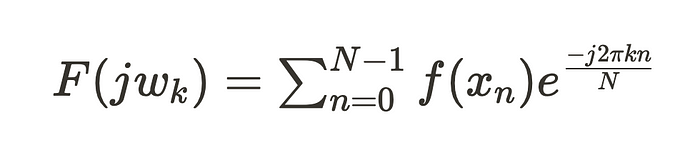
For example, the **Continuous Fourier Transform** takes continuous aperiodic signals over an infinite series.



The Continuous Fourier Transform Formula

But computers can’t work with continuous signals & it’s very likely you’ll be needing to work with more finite samples of a signal.

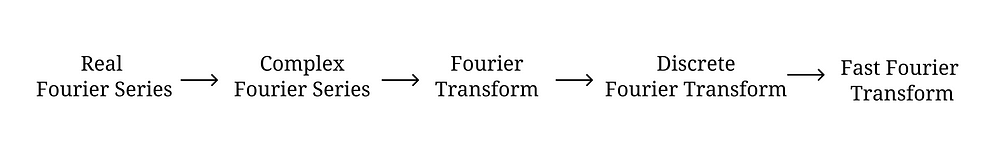
So instead, we can use the Discrete Fourier Transform which takes in a finite signal over *N*samples.



The Discrete Fourier Transform Formula

**Derivations like these make our lives way easier.**

Now, you can easily mathematically derive each transform subsequently from the previous.



It’s all interconnected.

*As I worked on*[***projects***](https://www.youtube.com/watch?v=uzCuzB5ixn8&t=125s)*in the EEG & signal processing space, I was fascinated by agility that the Fourier Series had to be molded for differing use-cases.*

*So I decided to look into & learn the mathematics underlying each transform.*

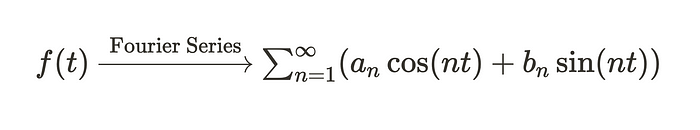
Let’s take a deeper look & understand how this works.

**The Real Fourier Series**

The Real Fourier Series serves as a foundational tool for the other derived transforms.

As mentioned earlier, it allows us to express continuous

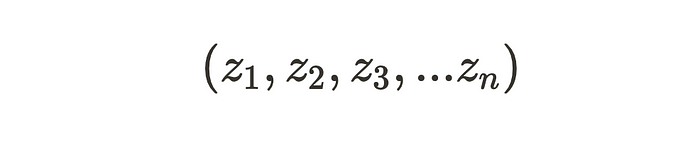
periodic signals, meaning signals that repeat over a period of time, into a sum of sines and cosines.



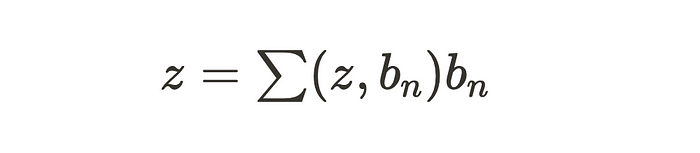
Basic Representation of the Real Fourier Series

Breaking down a function into sinusoids allows us to analyze a function more precisely given that we’re now able to understand the underlying components that make up the function.

Now, let’s say you have vector **z**represented by components 𝐳**ₙ.**



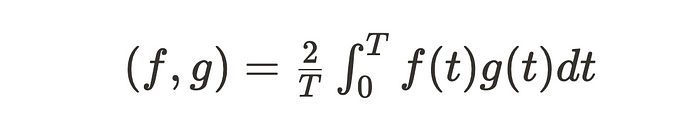
This vector can be represented in an [**orthonormal basis**](https://www.youtube.com/watch?v=ZJu26chXEiw),**b**ₙ, using the formula,



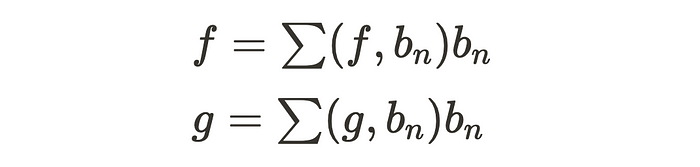
*The inner,****(z, bₙ)****, is the*[***dot product***](https://youtu.be/LyGKycYT2v0?t=51)*between****z****and the basis vector,****bₙ.***

So, this formula can be applied to functions.

For example, lets say we have functions ***f*** and ***g*** and their product is an integral. Both functions are periodic with a period of ***T.***

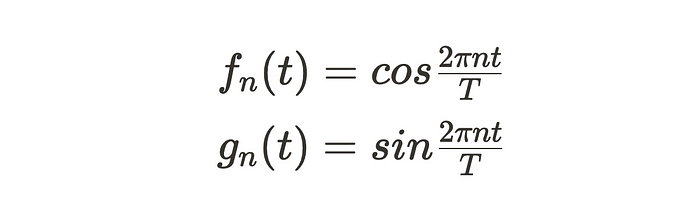


By applying the earlier formula, functions ***f***and ***g***can be rewritten as,



So the Real Fourier series allows us to rewrite a set of functions using the basis **bₖ = {fₙ}⋃{gₙ}.**

The basis function **bₙ** is a combination of two different sets of functions, **fₙ**and **gₙ**which can be written as,

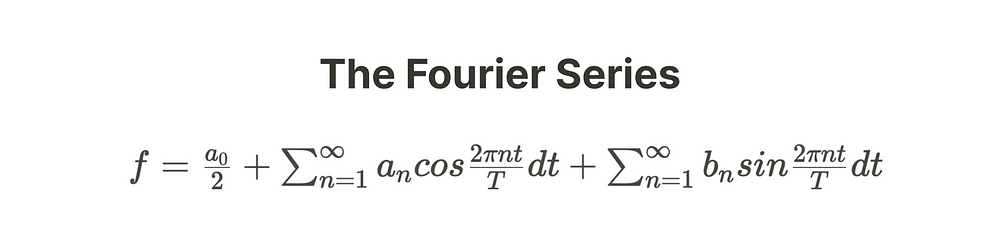


where n ≥ 0.

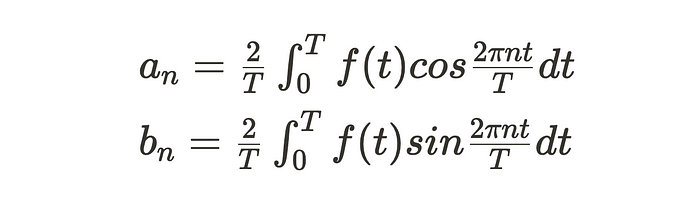
So let’s apply this to a **new function.**

Now let’s say we have a signal defined as ***f.***

We can rewrite the function ***f,*** with the basis change (demonstrated above), into the Real Fourier series.



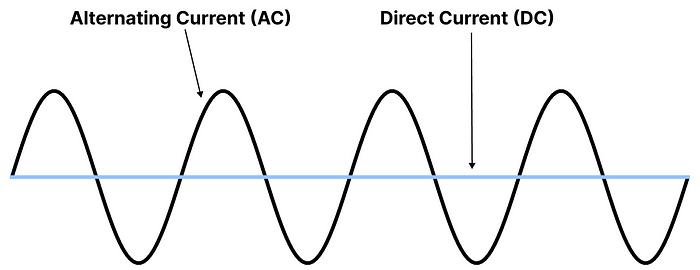
Where the inner coefficients ***aₙ*** and ***bₙ***are defined by the following.



When applied to a signal & it’s corresponding frequencies, ***n***is an index denoting a frequency component, ***t***is time, ***T***is a full period, and coefficients ***aₙ*** & ***bₙ***are the **amplitudes** of different frequency components.

To clarify, **a₀**is outside of the summations of the Real Fourier series as it represents the direct current (DC) component of a signal. It’s a constant term that represents the average value of the periodic signal over one period.

Using this term in the Real Fourier Series can prove to be useful for setting the “center” or “baseline” of a signal. From there, we can analyze the alternating current (AC) from the baseline more effectively.



So ultimately, the Real Fourier Series is a summation of the product of coefficient **aₙ**or **bₙ**anda sinusoid.

Interesting right?

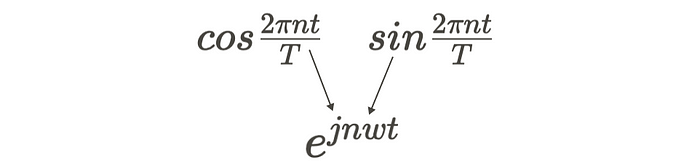
Well, **this is just the beginning.**

**The Complex Fourier Series**

The Complex Fourier Series is actually identical to the Real Fourier Series, at least in their purpose. Both of them aim to represent a periodic signal as a sum of sinusoids.

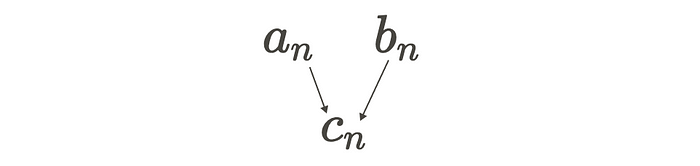
**But the key difference lies in *how* they represent a signal.**

Rather than taking the individual summations of sines and cosines and then adding them up together, the Complex Fourier Series represents them under a ***singular*** ***complex exponential.***



Just like this :)

Coefficients **aₙ**and **bₙ**can also be captured under a singular variable, **cₙ.**



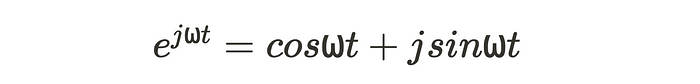
So, the **Complex** **Fourier** **Series** is a more compact and simple representation of the Real Fourier Series.

**Let’s begin with the derivation.**

*Keep in mind, when applied to a real signal:*

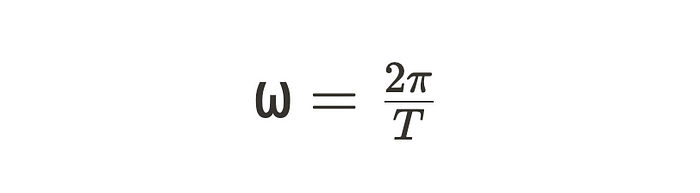
***n****is the index denoting frequency components****t****is time****T****is the full period of the signal*

So luckily for us, Leonhard Euler (a really interesting mathematician), granted us with the ***Euler formula.***



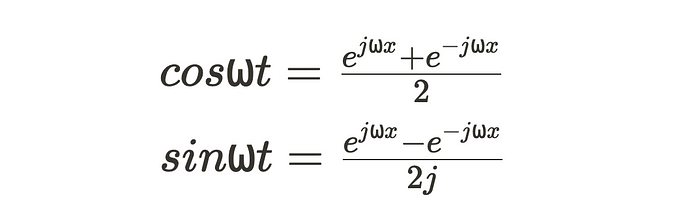
***j***is the [imaginary unit.](https://en.wikipedia.org/wiki/Imaginary_unit)

**⍵**is set to the angular frequency:



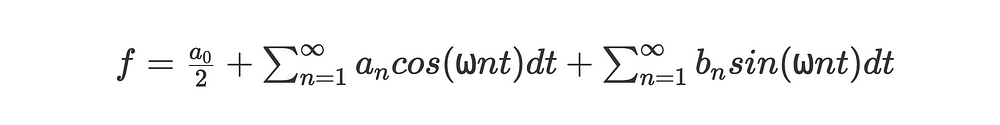
It basically denotes the equivalence between the summation of 2 sinusoids and a **complex** **exponential**.

Through **Euler’s formula**, we can derive the trig identities for sine and cosine using exponentials.

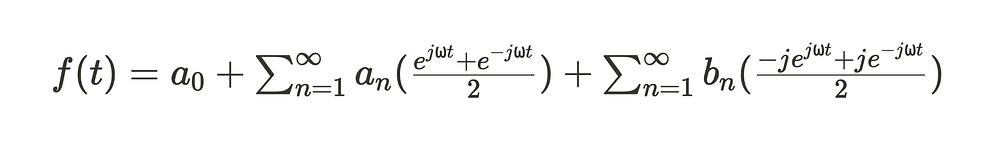


*If you’re curious, you can check out the full derivation*[***here***](https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/core-mathematics/pure-maths/algebra/euler-s-formula-and-euler-s-identity.html)

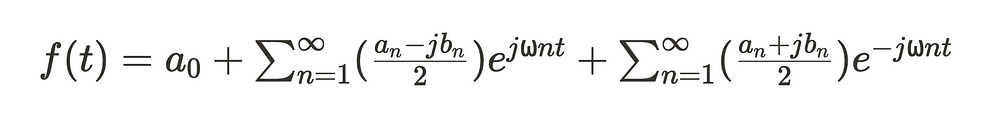
So given the **Real Fourier series**(rewritten in terms of **⍵**)**,**



and **Euler’s Formula** we can rewrite it as

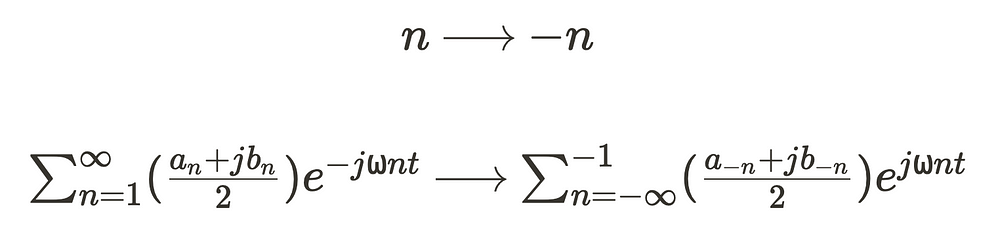


which can be simplified to

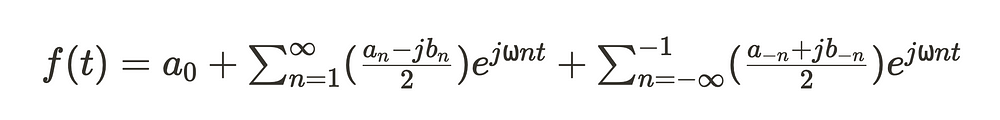


You might notice that the **second** **summation** has a negative exponential. But we want it to be a positive exponential to get both summations in terms of eʲ⍵ⁿᵗ.

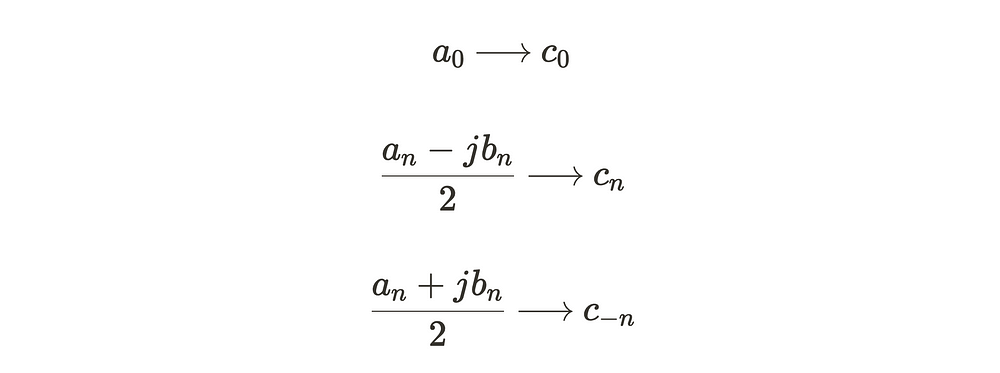
So, we can turn***n*** to ***negative n*** and rewrite the summation



Now we have:



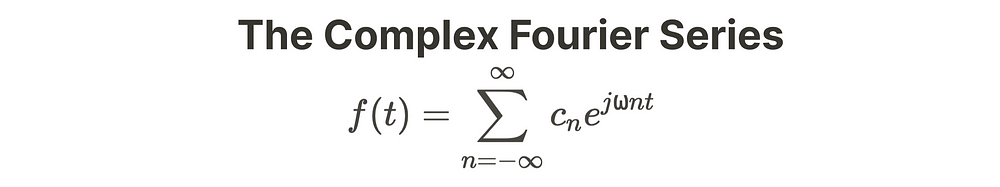
To even further simplify this, we can rewrite the inner expressions which hold coefficients***aₙ***and***bₙ*** in terms of a new coefficient, ***cₙ.***



Now, the coefficient ***cₙ***essentially represents all possible **complex** **coefficients** ranging from -∞ to ∞.

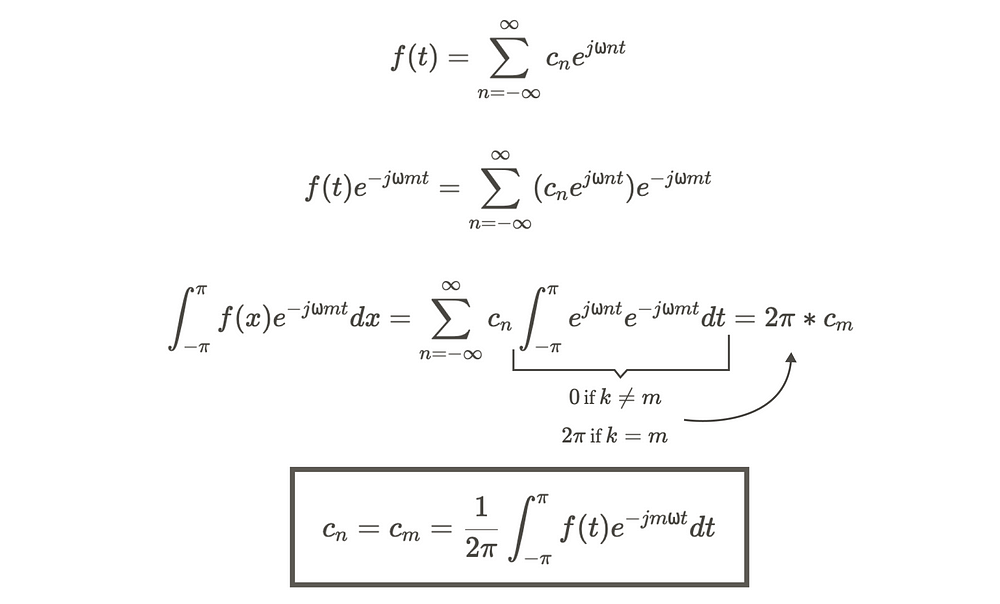
So given this, we don’t need to represent our equation in separate summations. We can join them together in a under a singular summation using coefficient ***cₙ.***

And we get the **Complex** **Fourier** **Series**.



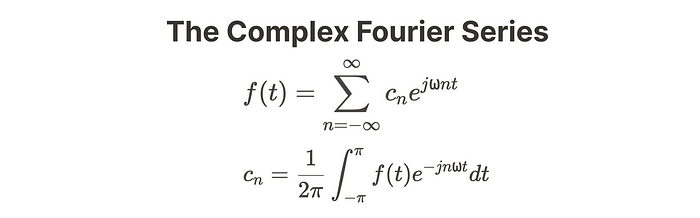
Now, the notation for the coefficient **cₙ**can be derived through the **Complex Fourier Series.**

This derivation will allow us to find the coefficient **cₙ**without solving for **aₙ**and **bₙ.**



Btw, we use **m** as a dummy variable. It’s equivalent to***n,*** treat it as a placeholder. We can substitute m with n.

Ultimately, as the Complex Fourier Series, we have:



Taken over **period T** or **2π**, from **-π to π.**

Just as a quick reminder, all the Complex Fourier Series is, is another way, in addition to the Real Fourier Series, to represent a continuous periodic signal as a sum of sinusoids.

The difference lies in the use of complex exponentials and it’s simplicity.

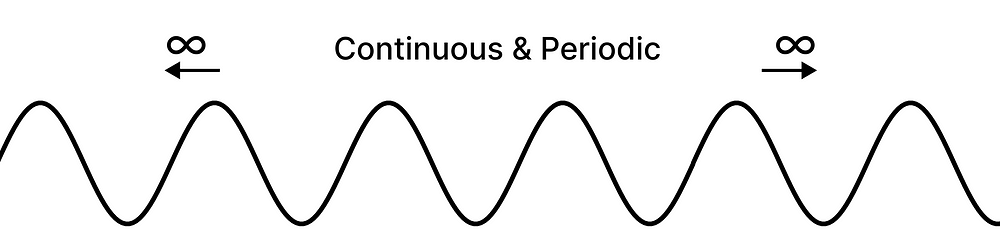
Now, onto the **Continuous** **Fourier Transform**

**The Continuous Fourier Transform**

By now, you might be asking, ***“what the hell is the difference between all these derivations?”***

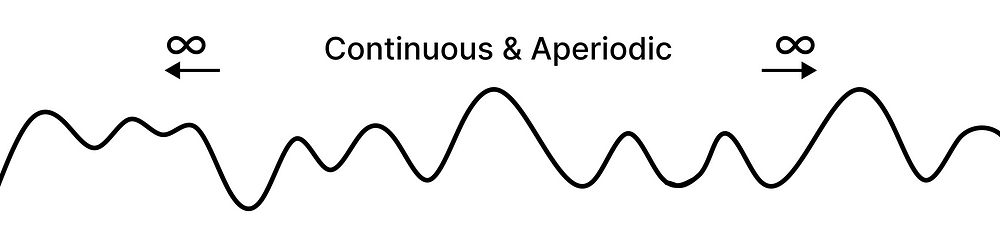
Well, the difference lies in **how they’re applied**and**what they can be applied to.**

The Real & Complex Fourier Series were both meant for continuous and periodic signals. Meaning **uniform** signals that are infinite in length.



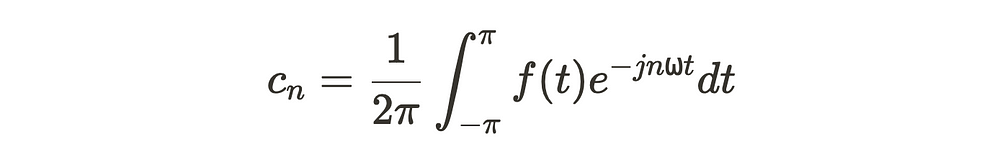
Other derivations are meant for different types of signals such as **discrete**signals or **aperiodic** signals.

In the case of the **Continuous** **Fourier Transform,**it’s designed for **continuous & infinite**yet **aperiodic**signals.



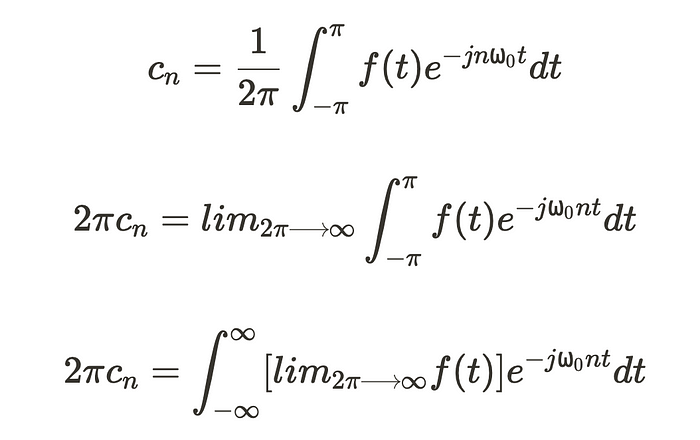
**So, let’s dive into the derivation**

Let’s take another look at our complex coefficient **cₙ.**

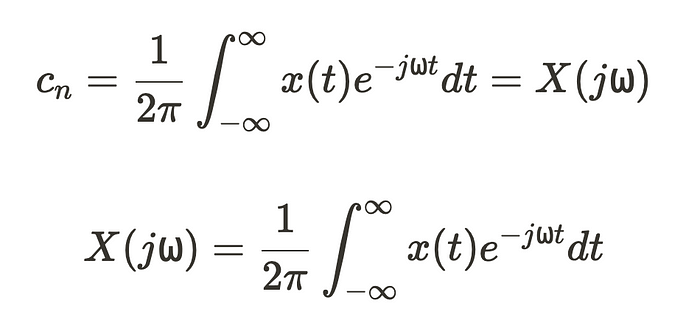


Essentially, to get the Continuous Fourier Transform, we take **cₙ**and it’s limits to infinity, turning it from a discrete variable to a continuous transform.

So we can directly begin by rearranging this equation and taking it’s limits from -∞ to ∞.

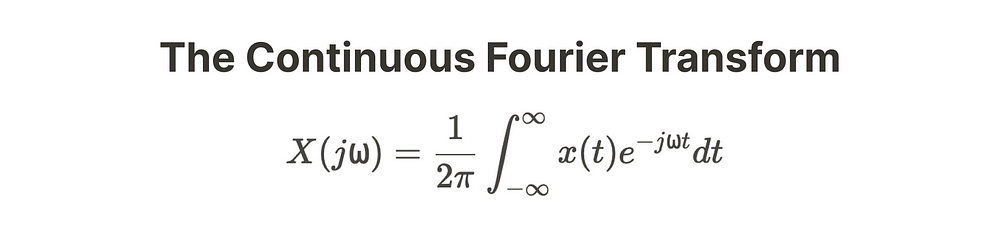


At this point, in the exponential ***e⁻⁻ʲʷ⁰ⁿᵗ***, ⍵₀ and ***n*** are multiplied into ⍵ to ultimately convert coefficient **cₙ**from discrete to continuous.



Here, ***x(t)*** represents ***f(t)*** as it’s limit goes to infinity. ***X(j⍵)*** is the signal defined in the frequency domain, characterizing it’s power at specific frequencies.

So there you have it.



In essence, this transform converts a **continuous** & **aperiodic** **signal**, ***x(t),***from the time domain to the frequency domain, ***X(j⍵).***

But note, while this transform is **aperiodic**and may appear to be increasingly useful… caveat!

It’s continuous!

***How will a computer, operating on binary 1s and 0s, take in continuous data?***

Luckily, that’s why we have the **Discrete Fourier Transform**

**The Discrete Fourier Transform**

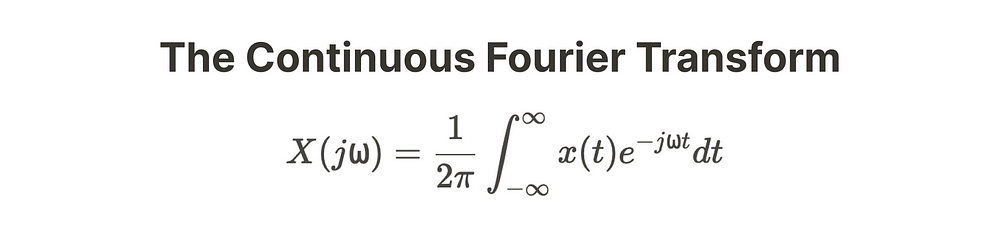
The **Discrete Fourier Transform** lays out the foundation for sampling and transforming real data with **computers**.

Let’s use Electroencephalography (EEG) as an example.

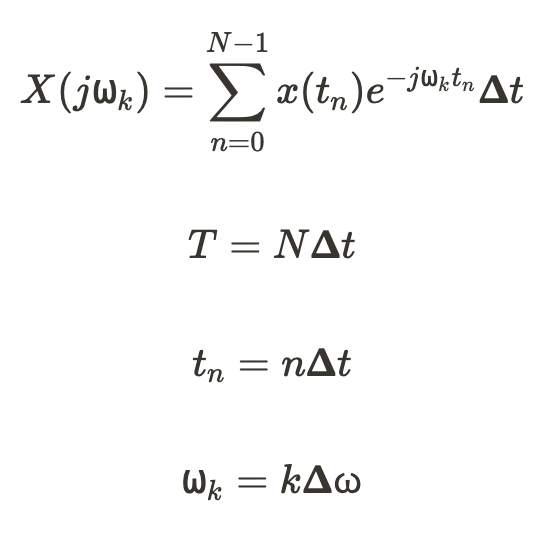
In EEG the data is taken in as discrete values at specific intervals, dependent on the **sampling** **rate**. Given this, a computer is easily able to work with this data since it functions on discrete values

The **Discrete Fourier Transform** would allow us to use the discrete data collected in that EEG and transform that data from the time domain to the frequency domain for **better** **analysis** on a **computer**.

To derive the **Discrete Fourier Transform**we takein the **Continuous Fourier Transform,**



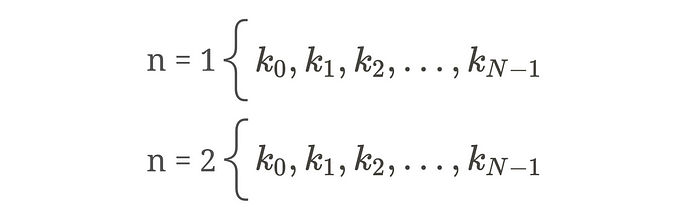
and sample it over a finite interval of***N*** samples.



* Periodicity, ***T,*** is the product of total samples (***N***) and the sampling interval, *𝚫****t***.
* Discrete time point (***tₙ***) is the product of index, ***n,***and the sampling interval *𝚫****t***.
* ⍵, as a whole represents the frequency bin at **kth**frequency component.
* ⍵ₖ represents the specific frequency at **nth**index within the ⍵ frequency bin.

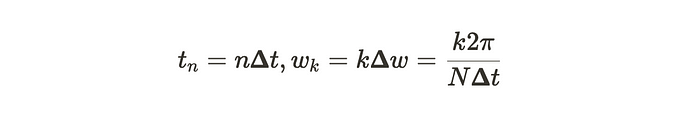
***Cool, but where the heck did k come from???***

*Well variable***k***represents the***kth** *frequency component. At each discrete time sample,***n,***we’ll have***k***amount of aperiodic frequency components that make up the signal.*

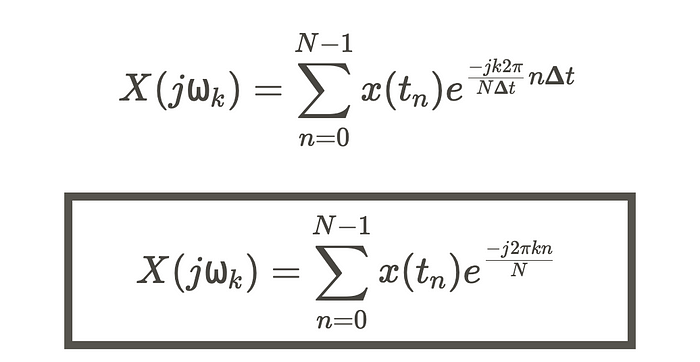


*Unfortunately, if we have a high value N, a computer has to go an immense amount of calculations which can pose problems… we’ll get into this later.*

Therefore, using



we can rewrite our equation to:



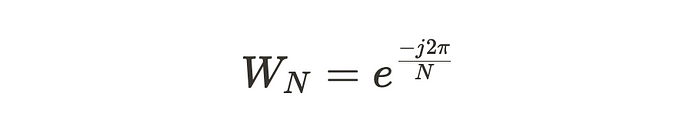
**And we get the DFT.**

But hold up, we’re not done yet!

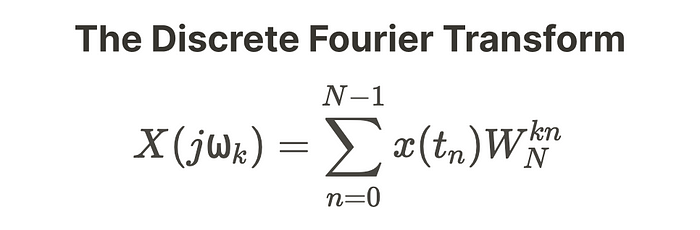
The equation for the DFT can be simplified and expressed through what’s called the [**Roots of Unity**](https://brilliant.org/wiki/roots-of-unity/#:~:text=A%20root%20of%20unity%20is,group%20theory%2C%20and%20number%20theory.)**.**

Essentially what a [**Root of Unity**](https://brilliant.org/wiki/roots-of-unity/#:~:text=A%20root%20of%20unity%20is,group%20theory%2C%20and%20number%20theory.) is, is a complex number than when raised to say the **nth** integer, when **n** is positive integer, the result of the exponential is 1.

So, we can express our equation using the Nth Root of Unity,



further simplifying it into,



Now, if you recall what I mentioned earlier,

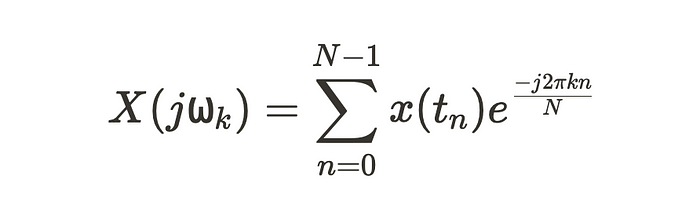
*“Unfortunately, if we have a high value N, a computer has to go an immense amount of calculations which can pose problems… we’ll get into this later.”*

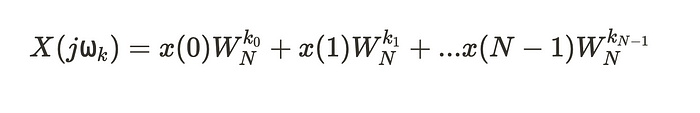
The DFT can prove to be ineffective when computing the spectrum / frequency domain of a signal.

The computation time, load, and complexity is too much!

Here’s why.

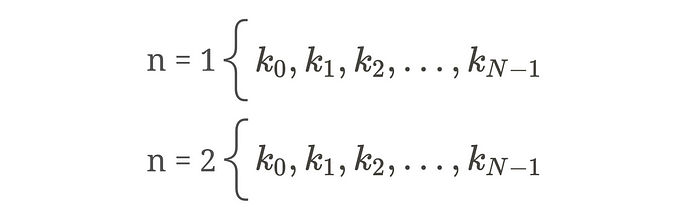
Let’s take a look at our DFT equation once more.





For each ***k***value we have to deal with **N-1** computations.

If we take another look at an earlier visual, to get more perspective,



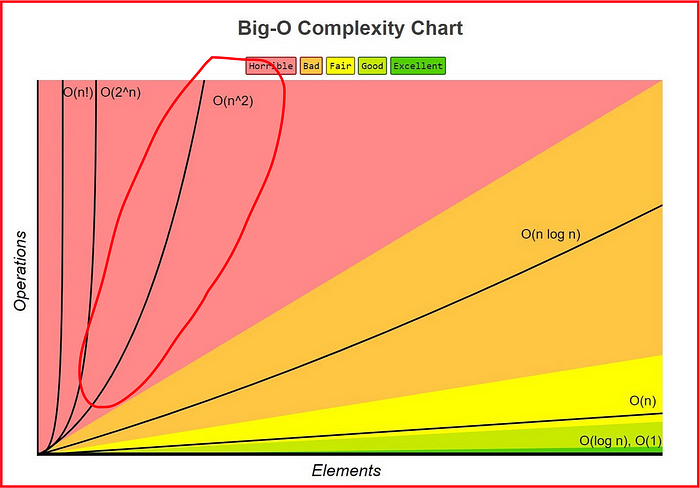
it’s clear that if we have a higher index **N**, we’d have to deal with an increasing amount of **k**computations per **n.**

The variable **N**also decides how many indices **n**our computer will decide to utilize. It’s not only **k**computations we have to worry about but **n**as well.

So ultimately, you’d need to perform **N²**computations when you consider both **k**and **n.**

If you’ve ever heard of Big O Notation,**you know this is bad.**

Let me just show you.



Over time, as our value ***N***increases, our overall computation complexity will exponential increase.

If we had **3000** pieces of sampled data in our dataset, it’d take a total of **9,000,000**total computer operations to iterate through the entire dataset. And remember, computational complexity **exponentially increases.**

When compared to the lower Big-O complexities, this is **enormous**.

But that’s why we have the **Fast Fourier Transform :).**

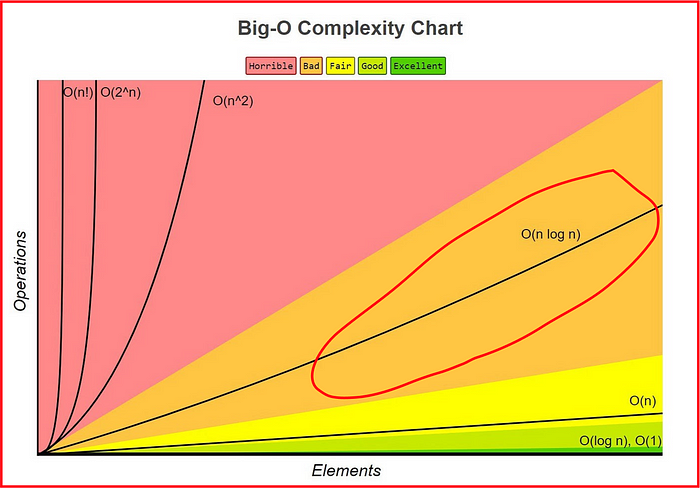
**Fast Fourier Transform**

So to mitigate the issue of the DFT’s exponentially increasing computational complexity as ***N***increases, some slight modifications were needed.

John Tukey, a mathematician, was involved in a discussion with President John F. Kennedy surrounding the possibility of detecting nuclear tests by the Soviet Union with sensors surrounding the country.

He realized, in order to utilize those sensors, they needed an algorithm of less computational complexity.

Alongside James Cooley, they published a [***paper***](https://www.ams.org/journals/mcom/1965-19-090/S0025-5718-1965-0178586-1/S0025-5718-1965-0178586-1.pdf) outlining the usability of the FFT in order to allow for decreased computational complexity from ***N²***to ***n log***₂ ***n,***which is an extreme improvement.

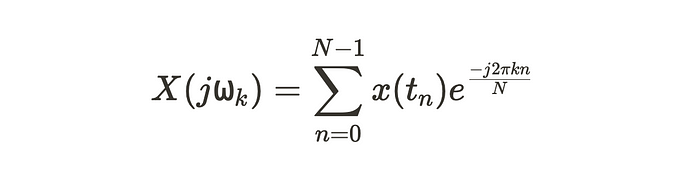


As you can see, computations increase in a more **linear**fashion rather than **exponential.**

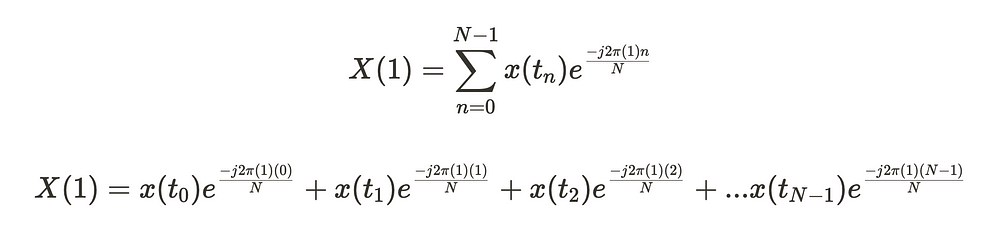
This makes things way more efficient and feasible, especially when we have a high amount of datapoints to compute.

Let me show you why.

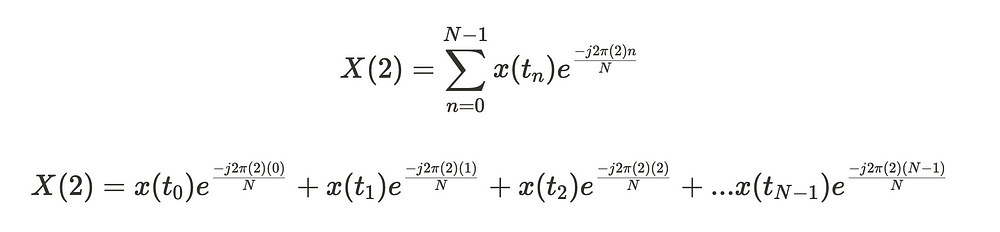
Let’s take a look at the **DFT**once more.



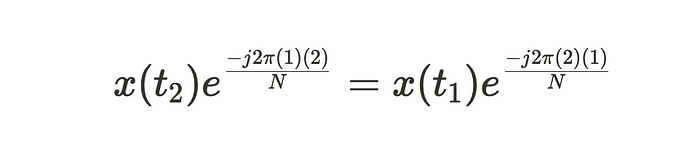
Let’s say we want to compute the second Fourier coefficient, **X(1).**



Now, let’s say we also want to compute the third Fourier coefficient, **X(2).**

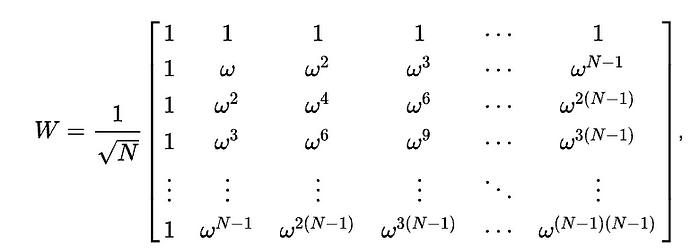


If you haven’t noticed, the third coefficient, where **X(1),**is the same as the second coefficient, where **X(2).**



So ultimately, rather than a computer calculating the same exponential that yields the same result **twice**across multiple **k values,**it computes an exponential using the **FFT,**to then store that exponential in it’s memory for future access.

This allows for increased computing speeds and better use of computational resources.



The DFT Matrix

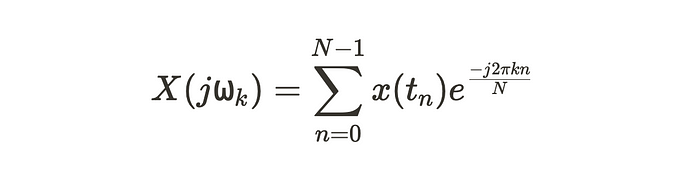
*You can see, repetitions of ⍵ in the DFT Matrix.  
Each ⍵, would be computed only once using the FFT algorithm*

Doing so over an increasing amount of **k**values, increases the amount of computational resources saved per the **FFT** algorithm

*Ok… How do we do this?*

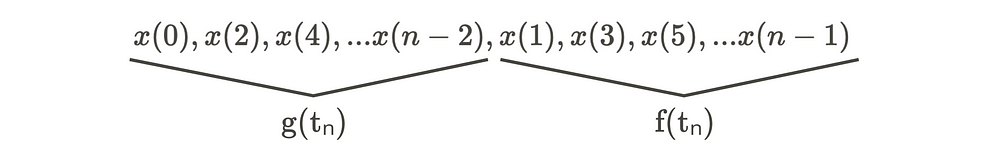
Well the **FFT** isn’t too complicated. Think of the **FFT** as an algorithm that more efficiently computes the **DFT.**Not as a completely new formula.

Let’s take a look at the **DFT** yet again.

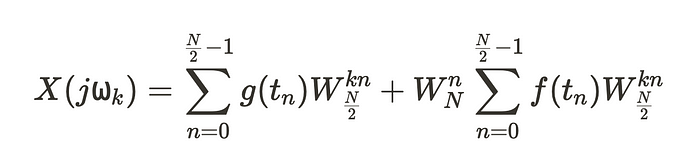


So, this equation can be divided into 2 separate sequences by the **division N/2.**One sequence would contain all even **n** values and the other would contain all the odd.

Let’s call the sequences **g(tₙ)**and **f(tₙ).**

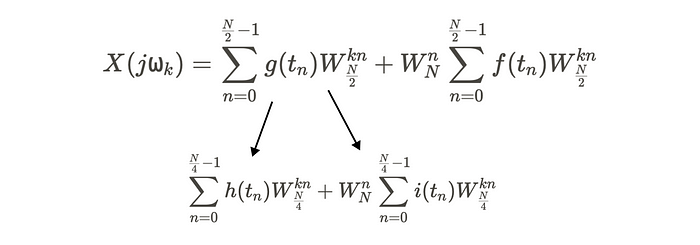


Given this, DFT can be broken down into 2 separate summations.



From here, we can break our summation down even further.

Let’s define the next functions as **h(tₙ)**and **i(tₙ).**



Each time we divide a summation into two new separate summations, we half our total number of operations to calculate the DFT.

**In essence, this is the process that makes up the FFT algorithm.**

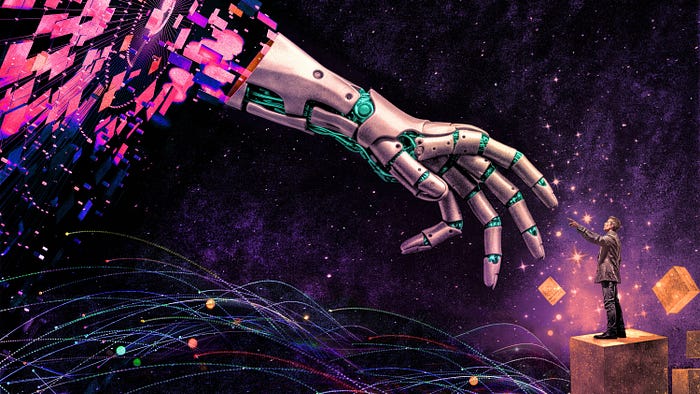
**So what?**

*What’s the point of all these complicated formulas and algorithms?*

All of these transforms serve as a fundamental basis for *why*we’re able to interact with complex forms of data.

It serves as an **essential** **tool** for the various processes and systems we interact with such as X-Rays, electroencephalography, phone calls, weather prediction, and much more.

More importantly, fundamental mathematics like these transforms make way for the creation of new and improved technologies such as brain-computer interfaces, quantum computing, smart grids, and even AI.



Understanding the mathematical processes behind emerging technologies gives you a true understanding of how you can begin to work with them and even improve them.

**Hope you enjoyed this read :)**

Feel free to contact me on [**twitter**](https://twitter.com/vxnuaj) or **[linkedin](https://www.linkedin.com/in/vxnuaj/" \t "_blank)** if you have any questions!

Also, feel free to subscribe to my [**newsletter**](https://vxnuaj.substack.com/)where I send out bi-weekly updates on what I’m working on!

[Mathematics](https://medium.com/tag/mathematics?source=post_page-----65389914ab2f---------------mathematics-----------------)

[Applied Mathematics](https://medium.com/tag/applied-mathematics?source=post_page-----65389914ab2f---------------applied_mathematics-----------------)

[Signal Processing](https://medium.com/tag/signal-processing?source=post_page-----65389914ab2f---------------signal_processing-----------------)

[Math](https://medium.com/tag/math?source=post_page-----65389914ab2f---------------math-----------------)