Received April, 2020; | Revised April, 2020; | Accepted April, 2020

DOI: xxx/xxxx

A GENERAL INFORMATIVE ESSAY FOR GENERAL ENTHUSIASTS AND PRACTITIONERS ON FFT IN STATISTICS & FINANCE

Fast Fourier Transform as an Engine for the Convolution of PDFs and Black-Scholes Option Pricing Using Python

Gene Boo*[1]

Correspondence
*Gene Boo Email: gene.boo@aol.com

Abstract

This paper is an attempt by an FFT enthusiast to document *one of the most magical mathematical engines from the end of the previous millennium*, which created the possibility of sound/video/photo editing in the frequency domain, with a focus on statistical convolutions and an example on constant volatility Black-Scholes pricing with minimal pre-transformations.

KEYWORDS:
Fast Fourier Transform; FFT; FT; DFT; Signal; Convolution; Cyclical; Probability Density Function; Option Pricing; Quantitative Finance; Black Scholes; Garman Kohlhagan; History; Frequency; Time; Music; Digital Signal Processing; DSP; Python; NumPy; SciPy; MatPlotLib; VBA; Excel

1 | INTRODUCTION

The author of this paper is an employee in the Malaysian banking sector, who started his quantitative training and career in Brussels, Belgium, where he had been exposed to the wondrous FFT algorithm in the course of pursuing his hobbies - electronic music production and video game algorithms. As of this moment of essay-writing, the frightful Covid-19 pandemic is still ongoing, and us in Malaysia, like the rest of the world, are in a locked-down state - what better moment to use our time saved from commuting to and from work, to pen a little bit about the FFT in hopes of seeing it get picked up for more common use in the Asian financial sector.

The banking and actuarial sectors often rely on probabilistic models for the purposes of pricing and risk management, and the most commonly encountered distributions in financial mathematics in Asia are probably the Normal and Lognormal distributions.

With Normal distributions, convolution is rather easy as the sum of N Normals is really just a single Normal with the means and variances of the variates summed up. The same cannot be mentioned for the Lognormal. The Lognormal distribution is frequently used in Geometric Brownian Motion models for option pricing - such as Black-Scholes. If $ln(X)$ follows a Normal distribution, X follows a Lognormal distribution.

It is well known that where there are many summed Lognormals, the final distribution tends toward the Normal, although not quite. With lower sums, the distribution looks "Lognormal-ish", but it is not quite there neither. What if the model requires an arbitrary sum of Lognormals - e.g. Basket Options? Much research has been conducted on this, where some of the more recent ones attempt approximations like Gauss-Hermite expansion, and application of Laplace Inversions using contour integration[1] - complicated algebra!

[1]https://arxiv.org/pdf/1606.07300.pdf

The takeaway from the previous paragraph is that as simple as Lognormals seem to be, summing them is not trivial. While it is great as a research topic, it only serves as a bottleneck in industries, which then turns to brute force approaches such as Monte Carlo, yielding oscillating results - definitely a problem in post-utility such as computing capital allocation weights, and besides, the engine is most definitely not one for the impatient.

If we had a reliable engine that can sum up not just Lognormals but any arbitrary shape describing random variates stably and very quickly - it could be an industry game-changer. The FFT algorithm when used as a convolution engine, does precisely that, and is quick and accurate as well. The FFT engine is capable of convolving any arbitrary signal, hence empirical distributions of any sort and other types of shape-functions can also be combined with precision. While frequently seen in science and engineering, its use is not yet popularised in the Asian financial sectors to date. However, with the description above, one can already imagine its potential use in various financial topics encountered today where probability distributions are frequented - option pricing, XVA estimation, credit loss probability estimation, operational risk compound loss distribution estimation, liquidity risk profiling etc. to name a few.

The author hopes that this introductory essay may serve to stir up some interest in the topic whilst remaining a fun-read, such that the banking industry may consider to start adopting FFT approaches in some of their modelling efforts - be it for combining pdfs, or obtaining more precise option prices other than those generated by Monte Carlo methods, to calculating huge covariance matrices, or obtaining stable tail-risk estimates such as VaR and Expected Shortfall for use in capital allocation etc.

It draws examples and concepts from the author's other hobbies (audio synthesis and Digital Signal Processing (DSP)) with some simple examples and Python 3 code. On the history section, the reader is cautioned not to put 100% confidence in its accuracy as there are varying versions that the author has encountered.

Real-world FFT topics and applications can involve very advanced mathematics and coding structures, so please do not let the simplicity of this essay cause an underestimation of its complexity and far-reached use.

On that note - let's dilly-dally no longer. Join us on our FFT journey - we hope you enjoy the ride!

2 | SOME INTERESTING HISTORY

The Fast Fourier Transform (FFT) is a well known and highly popular method in the world of engineering and physics and many other schools where signals are measured over time. We celebrate its discovery in 1965 by James Cooley and John Tukey, but actually, the theory of Fourier Transform (FT) and the concept of the FFT was penned long ago by the legendary Johann Carl Friedrich Gauss - of whom the Normal distribution was named after, displaying a case of Stigler's Law, as is the Fourier Transform to Joseph Fourier. Just to provide an inkling of its importance - without the FT and FFT, our world would not be so technologically driven as it is today - with computers, screens, rocketships, supercars, club music and remixes, computer animations, video games, fridges that can sense depletion and order your milk via the Internet, facial recognition software, fraud detection, encryption and compression - you name it. Hence that is why those born during the era of the Baby-boomers and Gen-X could observe the sudden technological jump, revolutionising the world - rendering a complete face-lift. This is all thanks to the skeleton key of FFT opening the portals of possibilities. This magical key opened the door to enable us to work on arbitrary signals and waveforms, allowing solutions to many previously-thought-unsolvable problems in almost all scientific and mathematical disciplines.

2.1 | The Wizards of FFT

Johann Carl Friedrich Gauss is purportedly the true inventor of the underlying recursive FFT algorithm, although there was a lack of hardware and software infrastructure in the very early 19th Century for its use to become popularised (Understatement-of-the-Millennium!) His 'FFT' work was then not as recognised as his other works, as it was way before its time, but it has since found its throne in the 20th and 21st Century (and probably way further to come). Gauss had initially used it as an interpolation tool for asteroid trajectories. This work was published posthumously.

IBM's James Cooley and Princeton's John Tukey published a paper in 1965, reinventing the FFT algorithm for use on a computer. Like much technology invented during the mid 20th Century, it had been primarily intended for military use, although Cooley was misdirected and informed that it was for crystallography. When their paper was published, it had been around the era of the Analog-Digital Converter (ADC), which conveniently implemented and spread the use of the algorithm. ADCs have today shrunk to the size of mini metal-oxide-semiconductor microchips, and are commonly found soldered to or directly integrated within music equipment such as modern analogue synthesizers, which enable harddisk recording or waveform manipulation such as digital filtering (post synthesis), or simply to enable the sending of the signal pre-converted to digital form, to a computer or digital synthesizer to avoid cable losses and further processing on the receiver machine. Analogue signals weaken, distort or deteriorate if they have to traverse copper or gold wires - also hence fibre optic cables of Sony/Philips Digital Interface (S/PDIF) as the new platinum standard - retaining the quality of original signals much closer to perfection. All sound cards have an ADC built on. ADCs are of course integrated within all our mobile telecommunicative devices today to convert waveforms picked up by the microphone or light by the camera, into digitally sampled signals. This is complementary to Digital-to-Analogue Converters (DAC), whose purpose is to reverse the function - i.e. playback a digital waveform to speakers or generating the appropriate LED matrix for viewing signals on-screen.

However, as it is with most works of great transcendental mathematics, the idea behind the development of FFT is derivative of other ground works. We must thank a few more legendary 17th and 18th Century European 'Magicians' - Roger Cotes known for his Newton-Cotes Quadrature[2] formulae (enabling us to perform numerical integration at good precision using Lagrange basis polynomials), Leonhard Euler from whom constant e got its honoured name-sake "Euler's number" - yet another case of Stigler's Law, and Jean-Baptiste Joseph Fourier who further developed the trigonometric series works by Euler, Bernoulli (who is the actual discoverer of the constant e) and d'Alembert, into the Fourier Series (FS).

2.2 | Bernoulli's Contribution

Bernoulli discovered the constant e while experimenting with concepts on compound interest. He was also amongst the founders of complex logarithms.

2.3 | Cotes' Contribution

British mathematician Roger Cotes had a very important finding - $ix = ln(cos(x) + isin(x))$ - do compare it to the Euler formula below. We all know how $e^{ln(x)} = x$, thus it gets interesting.

2.4 | Euler's Contribution

Euler predated Gauss, and he is well known for his equations $e^{i\pi} + 1 = 0$ and $e^{i\pi} = cos(\pi) + isin(\pi)$. These are the Euler identity and Euler formula respectively - usually π is represented generally by x for the Euler formula, but by substituting π for x as a special case, the identity equation would be the outcome. Just for the reader to draw a parallel, the Fourier Transform is often formulated as $\mathcal{F}\{f(t)\} = F(x) = \int_{-\infty}^{\infty} f(t)e^{-2\pi itx}dt$ where $i = \sqrt{-1}$. One can immediately see the resemblance of the 2, both involving constants i, e and π, hence a notion about the logical development of the Fourier formula.

The journey from the Euler formula of a circle traversing through the real and complex plain, to derivation of the identity simultaneously kick-started branches of many other discoveries of mathematics as it establishes links between the constants e, π, 1, i, and functions of $\sqrt{}$ and exponentiation, primarily in the further development of calculus in higher dimensions, without which we would not be able to solve many physics problems, nor develop technology as we see today such as the emergence of computers for use with designs and analyses. The trigonometric functions relate the triangle to the circle, and knowledge of various basic identities[3] can help with algebraic manipulations.

[2] https://mathworld.wolfram.com/Newton-CotesFormulas.html
[3] http://math2.org/math/trig/identities.htm

2.5 | Fourier's Contribution

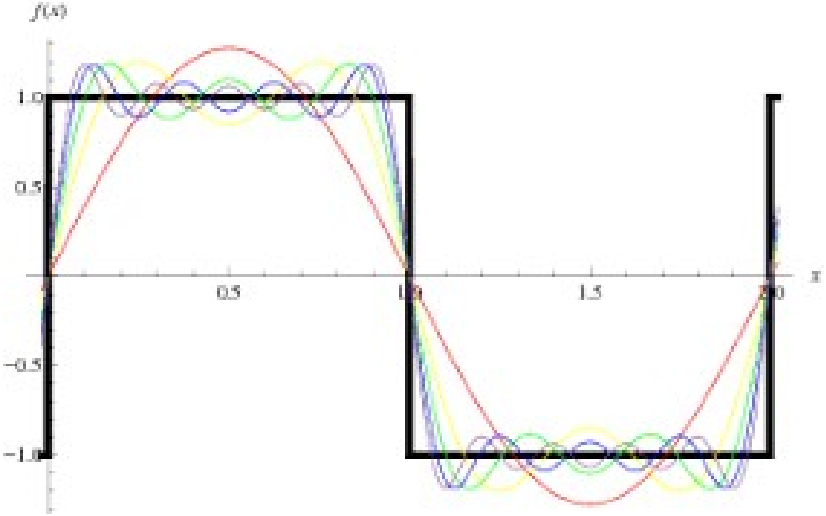
Fourier had discovered in the late 18th to early 19th Century that he could represent any continuous shape using the combination of trigonometric functions $\cos(x)$ and $\sin(x)$ in the form of a trigonometric series, having built upon the initial works of Euler, Bernoulli and d'Alembert. The purpose was to solve the heat equation $\frac{\partial u}{\partial t} = \alpha(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2})$, which apart from measurement of temperature distribution also found other general applications in today's stochastic mathematics involving Brownian Motion via the Forward Kolmogorov[4] (or Fokker-Planck[5]) Equations - which in turn provide an engine to solve various partial differential equations (pde), a recent famous one being the Black-Scholes equation $\frac{\partial V}{\partial t} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 V}{\partial S^2} = rV - rS\frac{\partial V}{\partial S}$ for option pricing.

3 | THE FOURIER SERIES

The FS, making up a study known as harmonic analysis, expands a periodic function $f(x)$ in terms of the infinite sum of $\cos(x)$ and $\sin(x)$, by exploiting the orthogonality of the 2 functions. One could form any 2D shape inclusive of waveforms such as the sawtooth, square[6] and triangular - widely used in electronic music synthesizers. There are very creative videos on Youtube [7] that trace the silhouette of Matt Groening's cartoon character Homer Simpson using a combination of Fourier analysis and Ptolemy's system of epicycles - i.e. circle mapped onto the circumference of a larger 'mother' circle recursively (looks like the cross section of an engine turning a wheel). Interestingly the idea of using epicycles was already present during B.C. times, used in early astronomic studies by Hipparchian, Ptolemaic, and Copernican systems. One can visualise for example, the trajectory of the Earth circling the Sun, with another circle made by the moon circling the Earth.

3.1 | Gibb's Phenomenon

The approximation by FS of non-continuous functions such as the square waveform tend to yield 'ringing' near the edges of the discontinuity. This is known as the Gibb's Phenomenon. Graphically, we see the sinusoids near the discontinuity take a larger amplitude overshoot in order to U-turn for the discontinuity drop or rise. We feature here a picture of it cited from Wolfram[8].



3.2 | The Journey from FS to FT

$$f(x) = 0.5a_0 + \sum_{n=1}^{\infty} a_n cos(nx) + b_n sin(nx)$$

[4]https://www.sciencedirect.com/topics/mathematics/kolmogorov-equation
[5]https://demonstrations.wolfram.com/BrownianMotionIn2DAndTheFokkerPlanckEquation/
[6]https://demonstrations.wolfram.com/FourierCoefficientsOfASquarePulse/
[7]https://www.youtube.com/watch?v=D-ogjQP_TBs
[8]https://mathworld.wolfram.com/GibbsPhenomenon.html

where

$$a_0 = \frac{\int_{-\pi}^{\pi} f(x)dx}{\pi}$$
$$a_n = \frac{\int_{-\pi}^{\pi} f(x)cos(nx)dx}{\pi}$$
$$b_n = \frac{\int_{-\pi}^{\pi} f(x)sin(nx)dx}{\pi}$$

$\{a_0, a_n, b_n\}$ form the series coefficients, $sin(x)$ and $cos(x)$ form a complete orthogonal system over $[-\pi, \pi]$ and $n = 1, 2, 3, ...$ [9]. Note that the formulation is all done for real numbers. Orthogonality (perpendicularity) is important as it means the decomposition is made up of independent functions, hence combination can be made directly - same principle as Eigenvalue decomposition. Hence the next step is to make it complex. Recalling Euler's formulae - which provide the way to formulate an exponential as weighted trigonometric functions cos(x) and sin(x) :

$$e^{ix} = cos(x) + isin(x)$$
$$e^{-ix} = cos(-x) + isin(-x)$$

Lead to:

$$cos(x) = Re(ix) = \frac{e^{ix} + e^{-ix}}{2}$$
$$sin(x) = Im(ix) = \frac{e^{ix} - e^{-ix}}{2i}$$

Hence when applied to the FS:

$$f(x) = 0.5a_0 + \sum_{n=1}^{\infty} a_n cos(nx) + b_n sin(nx)$$
$$= 0.5a_0 + \sum_{n=1}^{\infty} \frac{a_n - ib_n}{2} e^{inx} + \frac{a_n + ib_n}{2} e^{-inx}$$
$$= \sum_{-\infty}^{\infty} c_n e^{inx}$$

where complex coefficients $c_n = \frac{\int_{-\pi}^{\pi} f(x)e^{-inx}dx}{2\pi}$, $n = 0, \pm1, \pm2, ...$, making it algebraically simpler than the original equation.

Thanks to the effort of many FT-savvy netizens who are also coding wizards, we can now develop an intuition to these formulae by viewing animation of functions mapped to epicycles.[10]

The idea behind FS is to decompose signals into their component frequencies and harmonics (higher frequencies usually of lower amplitude/volume that are integer multipliers on the base frequency), that make up the full frequency, by essentially mapping the signal in time domain to epicycles and integrating it by simultaenously rotating all these circles at their given independent frequencies[11]. Applying the inverse FT (iFT) allows us to get back the signal. Hence for applications like sound editing/cryptography/blending/filtering, it is easy to just apply the FT, work on the frequency representation, then apply iFT to get the transformed signal in time domain. This also opens up a doorway to the world of convolution as will be featured later, as it is also a transformation to the frequency, by a function that happens to be the FT of the other signal.

While this version of the FS starts to resemble the FT, it is still limited to periodic functions and discrete frequency spectrum. To create a continuous version, we let $x = wt$ and change dx to dt. Now we have more control of the variable, by looking at it as a frequency w on a period t - which of course could have been done that way from the beginning. Now, to 'emulate' continuity, we let the period T tend to ∞. That way, w - the fundamental frequency will become an infinitesimally small dw, with n having to tend to ∞. [$n\,dw$] tends to w, and c_n becomes $c(w)dw$ to preserve the integral as discrete tends toward continuous.

$$c(w)dw = \lim_{T\to\infty} \int_{\frac{-T}{2}}^{\frac{T}{2}} f(t)e^{iwt}dt$$

[9] https://mathworld.wolfram.com/FourierSeries.html
[10] https://www.youtube.com/watch?v=r6sGWTCMz2k
[11] https://en.wikipedia.org/wiki/Fourier_series#/media/File:Fourier_series_square_wave_circles_animation.gif

$$= \frac{dw}{2\pi} \int_{-\infty}^{\infty} f(t) e^{iwt} dt$$

This is the basic gist of getting from the FS to FT. Formally, there are a few more steps to perform and also some scaling with 2π to meet modern convention, but the idea on the development is clear.

4 | THE FOURIER TRANSFORM

Displayed here is the Hertz form definition of FT (forward FT, aka Analysis equation):

$$\mathcal{F}\{f(t)\} = F(w) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i t w} dt$$

where $i = \sqrt{-1}$ and iFT (aka Synthesis equation):

$$\mathcal{F}^{-1}\{F(w)\} = f(t) = \int_{-\infty}^{\infty} \mathcal{F}(w) e^{2\pi i t w} dw$$

4.1 | FT as a Convolution Engine

Convolution - one of the important usages of the FT:
Let $f(t)$ and $g(t)$ functions belong to the space of real numbers $\mathbb{R}^n$, with F and G being their respective FTs, and $h(z)$ be the linear convolution of f and g:

$$F(w) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i t w} dt$$

$$G(w) = \int_{-\infty}^{\infty} g(t) e^{-2\pi i t w} dt$$

The usual linear convolution formula:

$$h(z) = \int_{-\infty}^{\infty} f(t) g(z-t) dt$$

Applying the FT on $h(z)$ and Fubini's iterated integral theorem for double integrals:

$$H(w) = \mathcal{F}\{h(z)\} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (f(t) g(z-t) dt) e^{-2\pi i z w} dz$$

let $y = z - t$ and hence $dy = dz$ and reapplying Fubini:

$$= \int_{-\infty}^{\infty} f(t) \left(\int_{-\infty}^{\infty} g(z-t) e^{-2\pi i z w} dz \right) dt$$

$$= \int_{-\infty}^{\infty} f(t) \left(\int_{-\infty}^{\infty} g(y) e^{-2\pi i (y+t) w} dy \right) dt$$

$$= \int_{-\infty}^{\infty} f(t) e^{-2\pi i t w} \left(\int_{-\infty}^{\infty} g(y) e^{-2\pi i y w} dy \right) dt$$

$$= \left(\int_{-\infty}^{\infty} f(t) e^{-2\pi i t w} dt \right) \left(\int_{-\infty}^{\infty} g(y) e^{-2\pi i y w} dy \right)$$

$$H(w) = F(w)G(w)$$

showing that FT of the convolution function is simply an element-wise multiplication between the FTs of each function. Using FT to perform a convolution is known as cyclical or circular convolution, as it also has a wrap-around effect, being mapped to a circle. If we use the numerical FFT engine to convolve, we can see this easily if we take the exact N samples of the signal and perform a convolution directly. The output will not be as expected - a work-around known as 'zero-padding' which will be discussed later has to be in place.

Imagine the implications, for we can now specify convolutions by taking the iFT of the convolved $H(w)$ to get the new function that describes the convolution of $f(t)$ and $g(t)$. This means, a function that describes how each element of f affects each element of g, commutatively! In probability theory it would be equivalent to finding the combined density kernel of summing 2 random variates of arbitrary density kernels. In sound, it would be a mix-down of 2 signals i.e. like kick drum played together with a snare. Still, analytically performing the algebra and calculus may lead to unwieldy roadblocks for many types of functions, so the true power does not really lie here. Instead, the true power lies halfway in between - where if we have a numerical algorithm that can perform very quick FT and iFT, that takes away the analytical evaluation steps, allowing us to perform this exact same task to a very good approximation, without overly wrecking our brains.

4.2 | FFT Engine

FFT is that key. The algorithm computes the Discrete Fourier Transform (DFT) and inverse (iDFT) of a sequence - very quickly. It works by decomposing the sequence or signal into the frequency components that make it up. While DFT and its inverse can be applied directly mathematically, it quickly hits the 'curse of dimensionality' and hence rendering it not practical for larger scale problems.

If we were to approach DFT computation using linear algebra, we could create a square DFT matrix by specifying a transformation $W = \frac{w^{jk}}{\sqrt{N}}$ Vandermonde matrix (matrix containing geometric progression terms $V_{i,j} = \alpha_i^{j-1}$), of (N^2) dimensions holding powers of the complex exponential terms $w = e^{\frac{-2\pi i}{N}}$, and compute the matrix multiplication $X = Wx$, with x being a vector containing equi-distanced samples up to N points of the original signal and X being the Discrete-Time Fourier Transform (DTFT) vector also up to N points:

$$W = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{N-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{N-1} & w^{2(N-1)} & \cdots & w^{(N-1)^2} \end{pmatrix}$$

This is good for smaller N but gets computationally heavy as N grows large. In computer science and mathematics we say that the DFT computation takes on an order of complexity of $O(N^2)$, as the reader can observe in the formulation above - for each additional sample to be transformed, you have roughly its square more operations to perform. Note as well that we are dealing with complex matrix multiplications. Each row of W is really the decomposed component waveform as we see in an FS, and we are combining these with the original signal using matrix multiplication to get the summed products of each component waveform applied to all samples of the signal, which will make up the DTFT vector X of the signal. Since this is complex multiplication, we will have both real cos and imaginary sin numbers each element of X.

The top row in the matrix being $w^0 = 1$ multiplied and summed across the signal vector is really what is known as the 'total DC-offset or DC-bias' and the follow-up division by the scalar $\sqrt{N}$ will be the 'mean DC-offset', which measures the average value of the waveform over a single period. This is of course well related to the electrical concept of Direct Current and Voltage - like in Alternating Current (AC), we want the sum to be 0. Mathematically it is really equivalent to normalising an integral like dividing the Normal distribution by $\sqrt{2\pi}$, as it is the value of the total integral space. In sound engineering, we also want the DC offset to be 0, because a signal with DC-bias - representable by a waveform whose mean hovers above/below the 0 dB line, would be played back with potentially a lot of clicks and distortion, not a clean high-fidelity sound, unless of course the artist is creating a 'Vinyl-warmth Lofi Hip-hop' tune. Amplitude (volume) can also be lost, as the DC-offset affects the 'energy' of the system. Hence in remastering sessions using Sound Forge or Audacity, one might choose to remove the DC-offset of the waveform using normalisation and/or compression techniques - which really are applications of linear operations on the signal