Generating random variables from a Lorentzian distribution

Colin Gordon

August 22, 2023

Abstract

This paper will cover a methodology used to generate random values for a Lorentzian distribution in my powershell scripts on this github repository. The motivation was to build a simplistic market model to generate possible accumulation functions for investing in the S&P 500. The original model consisted of finding the underlying distribution of interest rates to be a Lorentzian and build the accumulation function from randomly generated points from generated continual interest rates. As expected the results come out as quasi step functions where random jumps are due to tail end interest rates dominating accumulation functions. A more sophisticated model will take into account auto-correlations between interest rates and would assess the relationship between interest rates by performing convolution analysis with time differences. Future ideas will utilize a probability distribution constituting two interest rates and a difference in time to determine future interest rates via monte-carlo and possibly the student-t's distributions.

1 Introduction

I was interested in modeling accumulation functions for the S&P 500 index. I wanted to start with a simple probabilistically independent model for the accumulation function and from there introduce covariance in the next model. When I generated histogram Figure 1 from the S&P 500, it seemed to form a semi-martingale that looked very similar to a Lorentzian distribution. Since Lorentzian distributions are Levy-T stable, I figured it would be worth creating a simulation using a Lorentzian distribution.

2 Fitting Lorentzian Distribution

A Lorentzian distribution is a type of probability distribution function that was originally motivated from physics. The original use case was to model a spherical light source projected onto a flat plane such as a CCD camera. The spherical projection on a flat surface can be seen from the trigonometric resemblance in the equation (1).

$$\rho(x) = \frac{\gamma}{\pi(\gamma^2 + (x-\mu)^2)} \tag{1}$$

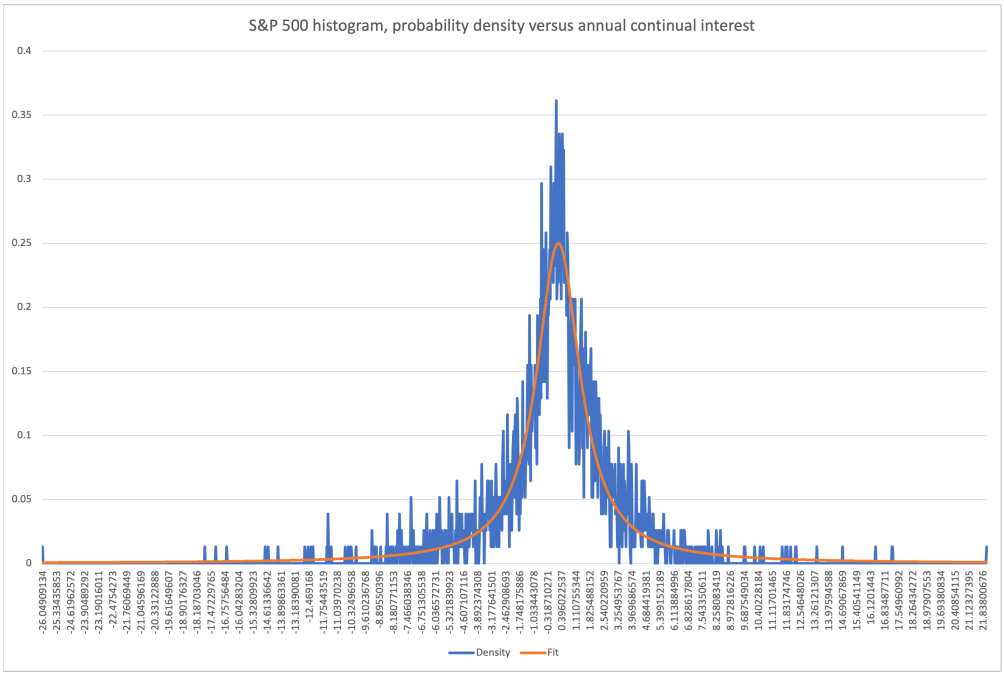

Figure 1: This is a histogram of the continual rate of interest for the S&P 500 over the last 10 years with data from mid July 2013-2023 with a Lorentzian fit.

I took data from [2] over the last ten years of the S&P 500 index. I have found that histogram of the continual short term interest rates seemed to closely resemble a Lorentzian distribution. especially since it is narrow at the center and mostly concave up from the center outwards unlike a Gaussian distribution.

2.1 Histogram Proceedure for Short term continual annual interest

The definition of continual annual interest is that if you have some accumulation function with respect to time for a fund $a(t)$ The definition of continual annual interest is

$$\delta = \frac{1}{a}\frac{da}{dt} = \frac{dLog(a)}{dt} \tag{2}$$

For a small time step of h we can estimate the first derivative of a function from data with

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + O(h^2) \tag{3}$$

where $O(h^2) = \sum_{n=1}^{\infty} \frac{h^{2n}}{(2n+1)!} f^{(2n+1)}(x)$ is calculated from a Taylor series expansion around $h = 0$. Since $h << 1$ year we will estimate it by assuming $O(h^2) \to 0$. The data analyzed was organized by date, with the latest date being in the top row. if the top row(latest date) is $i = 0$ and the last row(earliest date) is $i = N - 1$ then I calculated the continual interest by equation (4).

$$\delta[i] = \frac{Log(Data[i].OpenPrice) - Log(Data[i+1].OpenPrice)}{Data[i].OpenTime - Data[i+1].OpenTime} \tag{4}$$

where $\delta[i]$ is the continual interest rate at time $t[i] = \frac{Data[i].OpenTime + Data[i+1].OpenTime}{2}$. Note that time units were converted from days to years for continual annual interest rates. This creates an array of times and interest rates that we can organize into a histogram. To create a histogram I first found the standard deviation, the min and the maximum of the values of $\delta[i]$. The histogram function I had built will let the user define the fraction of the standard deviation that defines the size of a bin/partition. In the case for the data in figure 1, I used $\frac{1}{20}$th of the standard deviation to partition the data. From there you define the total number of bins as Ceiling$[\frac{\text{Max}-\text{Min}}{\text{Partition Size}}]$. I organized the data into a hash where the key is the continual interest rate in the middle of the bin and the value being an object with the bin's min, max, frequency and density. Density being defined as

$$\text{Density} = \frac{\text{Frequency}}{\text{Total Number of data points} * \text{Bin size}} \tag{5}$$

After calculating the histogram, I used excel to do a quick plot of the data which you see in blue. the x axis being the midpoint of the bin and the y axis being the calculated density of each bin.

2.2 Fitting the Lorentzian Distribution

To fit a Lorentzian distribution I used the standard methodology of calculating χ^2 by taking the difference of the calculated density with a normalized distribution and minimizing the χ^2 with respect to γ and μ. I was going to do this using Broyden-Fletcher-Goldfarb-Shanno minimization (BFGS) on both variables. Turns out there was a way to do it directly with excel with the solver tool which creates the orange curve seen in figure 1. The solver tools best fit for the variables in equation (1) is $\gamma = 1.268911575$, $\mu = 0.216918871$ with a calculated $\chi^2 = 0.41398057$. relaxing the assumption that the best fit was not normalized only decreased the χ^2 distance to $\chi^2 = 0.40764135$ which is less than a 2 percent difference. This would imply that our normalized Lorentzian fit is a rather good guess expected from the definition of density imposed on the data. If we were to use the squared difference between the data and the fit as a euclidean distance then we can define an analogues inner-product for this sample space by defining

$$\langle f_1 | f_2 \rangle = \sum_i f_1(x_i) f_2(x_i) \tag{6}$$

Which every innerproduct has a standard norm $||f_1||^2 = \langle f_1 | f_1 \rangle$. Every norm has a metric associated which is $||f_1 - f_2|| = \text{dist}(f_1, f_2)$. This distance under this metric is the same as χ^2. Using this innerproduct and norm as a metric for our sample vector space, we can calculate the effective euclidean angle between the fit and the data with this equation.

$$cos(\theta) = \frac{\langle f_1 | f_2 \rangle}{||f_1|| ||f_2||} \tag{7}$$

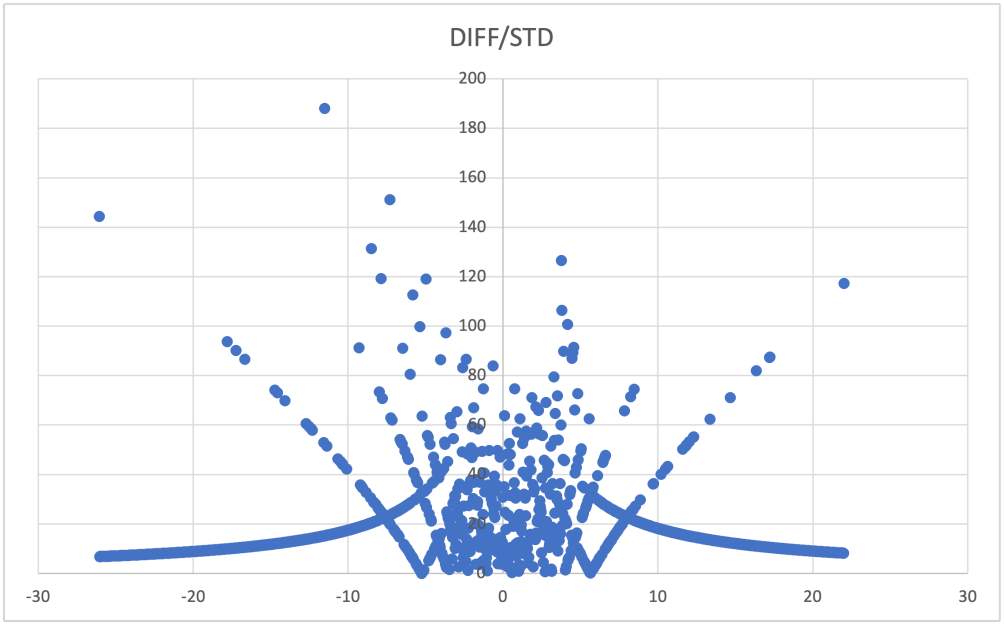

Figure 2: This is a histogram of the difference between data value and fit divided by the standard deviation using a binomial distribution

the calculated angle between the fitted distribution and the data is about $\theta \approx 0.301$ radians. This makes $cos^2(\theta) \approx 0.912$ which Born's interpretation would say that this fitted curve explains about 91.2% of continual interest rate data while the rest of the data would be explained by an orthogonal basis to the Lorentzian under this innerproduct.

We can further illustrate how well our data fits by creating figure 2. The figure 2 estimates the binomial standard deviation for each bin by using $\sigma = \sqrt{NP(1-P)}$. $P =$ Density $*$ Bin Size, $N =$ total number of data points $= 2494$. The difference calculated is $|$Fit density $-$ Empirical Density$| *$ Bin Size $* N$. The idea was that if each continual interest rate was randomly selected in a probabilistic fashion, the chance of landing in the bin would be $P =$ Density $*$ Bin Size. Therefore the spread of the data would be related to a binomial distribution where NP points land in the bin and $N(1-P)$ points land outside of the bin. The difference being on order of 10s to 100s of standard deviations means that our noise is far from being a true Lorentzian. This is due mainly on that fact that data is not randomly distributed and there exists a lot of correlations in the underlying data that would not be captured by the semi-martingale. The generation of noise is out of the scope of this particular paper but will be explored in future endeavors.

2.3 Summarizing Empirical analysis

Since that our distribution has several points of contact showing that we can fit a normalized Lorentzian distribution well within reason, it stands that we can utilize a Lorentzian as a standard probability distribution function for generating fund data representing funds like the S&P 500. In the next section I will discuss on how to compute random variables for a Lorentzian distribution.

3 Simulating a Lorentzian distribution

This section describes how you can simulate a Lorentzian distribution using a uniform pseudo random number generator to generate values that will form a histogram with a Lorentzian distribution. Note: I am currently working on a paper to generalize the methodology that I have used to generate random variables from a Lorentzian distribution. The methodology in that paper was inspired by what I have done for this distribution specifically. Hence some of the sections here are from that generalized papers, namely 3.1-3.4.2.

3.1 Pseudo Random number generators

Pseudo random number generators in computers are created by many different methodologies. One example is to solve the differential equation of a logistic curve with an Euler method, the sink equilibrium point can cause the value to vary in a near random fashion near the equilibrium. By design random number generators should produce a uniform distribution of numbers within some interval $[a, b]$. for this paper we will choose the interval to be $[0, 1]$. Note that we can always transform a number that is in an interval of $[a, b]$ to an interval $[0, 1]$ by taking the number $x \in [a, b]$ to make it the number $y = \frac{x-a}{b-a}$ which will be a random number in $y \in [0, 1]$ or invertibly go from $x = a + (b - a)y$. We will utilize the pseudo random number generators in two ways. One will be to partition the interval [0,1] into sub intervals where the length of the intervals is the probability of a sub interval of the real number line. if the pseudo random number lands in that interval then we map that to an associate sub interval on the real number line. from that sub interval we randomly choose a number in that interval to be the randomly generated number that has a background probability density function.

3.2 Partitioning $[0, 1]$

You can constructively partition the number line from $[0, 1]$ with probabilities of intervals in a way where the length of an interval equates to a probability. We can order subintervals of the real number line in a way where we constructively build intervals in $[0, 1]$.

First we define our probability distribution in a way where the maximum occurs at $x = \mu$ which is usually an axis of symmetry or mean of the distribution. There is also another factor that represents the width and flatness of such a distribution to which we can call γ for Lorentzian or σ for Gaussian distributions. This is like the standard deviation of the distribution. with these type of descriptive variables we can adjust the distribution to a unitless version where $\Delta = \frac{1}{\gamma}(x-\mu)$. This allows to computationally create a bunch of points for one type of the distribution and save it to memory. when we need a random point we just randomly pick a generated point of Δ and use γ and μ to calculate the associate $x = \gamma\Delta+\mu$. doing this type of transformation we can define our probability density function using change of variable. hence $\rho(\Delta)d\Delta = \rho(x)dx$ this means our distribution is now centered at the axis of symmetry, mean or weakest case maximum by construction and $\frac{1}{\gamma}dx = d\Delta$ making Δ unitless.

By construction, we can take a finite set of mutually exclusive intervals $\{I_k\}$ that cover $\mathbb{R}$ for the variable Δ. Since the intervals are mutually exclusive and cover the universal set then

$\sum_k P(I_k) = P(\cup_k I_k) = P(\mathbb{R}) = 1$. hence we can constructively partition $[0,1]$ by defining.

$$P_0 := 0, P_k := P(I_k) + P_{k-1} \tag{8}$$

In essence $P_r = \sum_{k=0}^{r} P(I_k)$. If r is the number of the last interval then we know $P_r = 1$. hence $[0,1]$ can be partitioned with a set of mutually exclusive intervals $\{\tilde{P}_k\}$ of the forms $[P_{k-1}, P_k]$, $[P_{k-1}, P_k)$, $(P_{k-1}, P_k]$, or (P_{k-1}, P_k) depending on how we define our I_k intervals and topological continuity we will impose. (closed intervals are closed, open are open, half and half are half and half). The nature of these intervals also have the length of $P(I_k)$. Constructively we pick $[P_{k-1}, P_k)$ usually unless we are dealing with edge cases.

The imposed relationship is that there is a function $\tilde{g} : \{\tilde{P}_k\} \to \{I_k\}$ that is one to one and onto. hence we can define a well-defined surjective relationship such that $g : [0,1] \to I_k$ if $x \in \tilde{P}_k$ then $g(x) = I_k$

3.3 combining partitioning with pseudo random number generators

The construction of the function g allows us to take any number from the interval $[0,1]$ and project that number to a unique interval in the real number line. If a number in the interval $[0,1]$ was choosen randomly in a fair matter, then the probability of that number being in the interval $\tilde{P}_k$ is $P(I_k)$. This is due to the fact that the length of the interval is $P(I_k)$ by construction and the length of the universe $[0,1]$ is 1. Since pseudorandom number generators are designed to have a near uniform distribution of generated numbers in an interval, this function g will can take a generated number y and turn that number into an interval $g(y)$. If we choose our intervals of the real number line to be small enough in a matter where $\rho(\Delta)$ has an extremely small variance across the interval, then randomily choosing another variable Δ from the interval $g(y)$ will approximate the likelyhood of that variable Δ being choosen from the background PDF $\rho(\Delta)$.

3.4 Generating subintervals of the real number line

there are two methods of how to create a subintevals $\{I_k\}$ on the real number line that have a small variance in $\rho(\Delta)$ for $\Delta \in I_k$. One method requires $\rho(\Delta)$ to be differentiable and non-zero but is the faster of the two. the other is to stretch the interval in a way where the max and min differ within a certain limit. The main idea is that we want to pick an interval where

$$sup[\rho(I_k)] - inf[(\rho(I_k))] \leq h\langle\rho(I_k)\rangle \tag{9}$$

where $\langle\rho(I_k)\rangle$ is the average over the interval, h is some scale factor, sup is the supremum/max and inf is the inferium/min of the interval. The equation (9) defines a constraint of the interval that relates the to an idea colloquially called percentage/relative difference h. This idea means that the highest mountain and the lowest trough has an elevation difference proportional to h. A smaller choice of h will result in a smaller relative elevation difference. Choosing a value of $h = 0.01$ would mean that the probability density will vary at most 1

percent within the interval making the density function have a near uniform probability distribution across the interval. This constraint allows us to use the random number generator once again where our Δ can be randomly chosen from I_k in an fair fashion with loosing a controlled tiny sliver of error from non-uniformity.

3.4.1 Differential method

This method approximates the relative difference by using the linear approximation methods from calculus. let I_k be an interval where the boundary points are Δ_{k-1} and $\Delta_k = \Delta_{k-1} + \delta$ which $\delta > 0$. if ρ is differentiable over the interval I_k then $\rho(\Delta_0 + \delta) - \rho(\Delta_{k-1}) = \rho'(\Delta_{k-1})\delta$. in general the sign of the derivative signifies if the function is locally decreasing or increasing. for sufficiently small enough intervals the function is strictly increasing or decreasing. hence

$$sup[\rho(I_k)] - inf[(\rho(I_k))] = |\rho(\Delta_{k-1} + \delta) - \rho(\Delta_{k-1})| \approx |\rho'(\Delta_{k-1})|\delta \tag{10}$$

if δ is sufficiently small then we can approximate $\langle \rho(I_k) \rangle \approx \rho(\Delta_{k-1})$ and setting the $\leq$ to an equals sign we get.

$$|\rho'(\Delta_{k-1})|\delta \approx h\rho(\Delta_{k-1}) \tag{11}$$

the equation (11) can be used to generate δ in turn generate Δ_k by the following equation.

$$\Delta_k = \Delta_{k-1} \pm h\frac{\rho(\Delta_{k-1})}{|\rho'(\Delta_{k-1})|} \tag{12}$$

where we have $\pm$ depending if we want to increase or decrease our delta values. Note that the function divided by the derivative is the same as the reciprocal of $\frac{dLog(\rho)}{d\Delta}$

3.4.2 Direct methods

A direct approach maybe needed if the derivative is near zero. You can do this by directly substituting the absolute difference in (10) to (11) to get

$$|\rho(\Delta_k) - \rho(\Delta_{k-1})| \approx h\rho(\Delta_{k-1}) \tag{13}$$

the solution could be done analytically or utilize a standard root finding method, like Secant/midpoint/Newton methods. Another methodology that can be used is

$$|Log(\frac{\rho(\Delta_k)}{\rho(\Delta_{k-1})})| \approx h \tag{14}$$

Depending on the type of distribution one methodology maybe easier than the other. to give an example of each. For the center of the Lorentzian $\rho(\Delta) = \frac{1}{\pi(1+\Delta^2)}$ such that $\gamma = 1$ and $\mu = 0$ where $\Delta_{k-1} = 0$ then

$$|\frac{1}{\pi(1+\Delta_k^2)} - \frac{1}{\pi})| \approx h\frac{1}{\pi} \tag{15}$$

which Δ_k can be solved analytically directly as $\Delta_k^2 = \frac{h}{1-h}$ which if we want to be suave we can make it a bit simpler $\Delta_k = h^{\frac{1}{2}}$ since this number is slightly smaller than the fraction.

for a Gaussian $\rho(\Delta) \sim e^{-\frac{\Delta^2}{2}}$ the Log method would be beneficial at $\mu = 0$, $\Delta_{k-1} = 0$ and $\sigma = 1$

$$|Log(\frac{\rho(\Delta_k)}{\rho(0)})| = |\frac{-\Delta_k^2}{2} - 0| = \frac{\Delta_k^2}{2} \approx h \tag{16}$$

Direct method implementation is more accurate at employing intervals that maintain the relative difference being a maximum of h. however the smaller h is the difference accuracy diminishes. to save on computation time it is in general to employ the method in (14) than in the direct methods unless the derivative is absolutely zero or the change of Delta is so great that you would like to restrain it. On a computer you can catch the zero condition by assuming it is non-zero and try calculating. the zero in a denominator will result in an error which will activate a catch that would implore one of the two methods listed above associated with a root finding method like the secant method. on the other hand, you can use an if condition to change the values if the change of Delta is too big.

3.4.3 Methods applied to Lorentzian distribution

If (1) is a generic Lorentzian distribution, we can simplify the distribution by defining the variable $\Delta = \frac{1}{\gamma}(x - \mu)$. Doing so would center the maximum at the center of the distribution and make the distribution unitless. The distribution where $\rho(\Delta)d\Delta = \rho(x)dx$ would be defined like this

$$\rho(\Delta) = \frac{1}{\pi(1 + \Delta^2)} \tag{17}$$

Using this as our distribution, we can always map Δ back to x by inverting the definition. This allows us to only need to generate points for one distribution to get a random set of points. As discussed in (15) we can use this to estimate our first partition of $[0, 1]$ defined to be where $\Delta_0 = h^{\frac{1}{2}}$ covers the range $(-\Delta_0, \Delta_0)$. This makes $P_1 = P(\Delta \in (-\Delta_0, \Delta_0)) = \int_{-\Delta_0}^{\Delta_0} \frac{1}{\pi(1+\Delta^2)} d\Delta = \frac{2}{\pi} tan^{-1}(\Delta_0)$. So our interval is $\tilde{P}_1 = [0, \frac{2}{\pi} tan^{-1}(\Delta_0))$.

As for the rest of the partitions, I will use the differential method to point describe them. The δ term from equation(11), can be calculated from this distribution as

$$\delta = h\frac{\Delta^2 + 1}{2\Delta} \geq \frac{h}{2}\Delta \tag{18}$$

using the slightly smaller region would mean the percentage difference is even smaller than what our limit provides, and it will give us a way to compute Δ_k from a direct computation. if $\delta = \frac{h}{2}\Delta$ then $\Delta_{k+1} = (1 + \frac{h}{2})\Delta_k$ which is an exponentially growing $\Delta_k = \Delta_0(1 + \frac{h}{2})^k$. The resulting equation for Δ_k for all counting numbers k is

$$\Delta_k = h^{\frac{1}{2}}(1 + \frac{h}{2})^k \tag{19}$$

From this equation we can systematically generate partitions of the interval $[0, 1]$. Due to the even symmetry of the Lorentzian distribution we know that a $P(\Delta \in [\Delta_k, \Delta_{k+1})) = P(\Delta \in (-\Delta_{k+1}, -\Delta_k])$. We can exploit this property by flipping the sign of the interval between every even and odd partition. thus $P_2 = P_1 + P(\Delta \in [\Delta_0, \Delta_1))$ making $\tilde{P}_2 = [P_1, P_2)$.

While the negative region would be $P_3 = P_2 + P(\Delta \in (-\Delta_1, -\Delta_0]) = P_2 + P(\Delta \in [\Delta_0, \Delta_1))$ which can take the interval $\tilde{P}_3 = [P_2, P_3)$. To generalize it as a formula then

$$P_{2n} = P_{2n-1} + \frac{1}{\pi}[tan^{-1}(\Delta_n) - tan^{-1}(\Delta_{n-1})] \tag{20}$$

$$P_{2n+1} = P_{2n} + \frac{1}{\pi}[tan^{-1}(\Delta_n) - tan^{-1}(\Delta_{n-1})] \tag{21}$$

where $\tilde{P}_k = [P_{k-1}, P_k)$, $\tilde{P}_1 = [0, \frac{2}{\pi}tan^{-1}(\Delta_0))$. We will use these partitions to break up the interval $[0, 1]$ in a way where if a random number $q \in [0, 1]$ is generated, then if $q \in [P_{k-1}, P_k)$ we will use the associated interval. The associated interval has $k = 2n$ or $k = 2n+1$ where n is an integer. if $k = 2n$ then the interval chosen is $[\Delta_{n-1}, \Delta_n)$. If $k = 2n+1$ then the interval chosen is $(-\Delta_n, -\Delta_{n-1}]$. These associated intervals are designed by equation (19) where the probability distribution is nearly uniform. We can use a random number generator that treats the probability as uniform and pick $\Delta \in [\Delta_{n-1}, \Delta_n)$ or $\Delta \in (-\Delta_n, -\Delta_{n-1}]$ accordingly. Even though this methodology can be expanded to infinity, a computer can only use a finite number of partitions. Finite partitions will never cover the space $[0, 1]$, we need a method to deal with tail ends.

3.5 Dealing with Tail ends of the distribution

3.5.1 Defining the tail end

To deal with the tail end, we would need a way to define our maximum k value for our $\{\Delta_k\}$ sequence. To do so we will define P_{max} to be an estimate of the maximum range that our calculated intervals cover. Our partitions would have a resulting k_{max} associated with the maximum probability range where $P_{max} = \frac{2}{\pi}tan^{-1}(\Delta_{k_{max}})$. As $P_{max} \to 1$ we can use an approximation around the pole to estimate k_{max}. First start with $1 - P_{max} = \frac{2}{\pi}(\frac{\pi}{2} - tan^{-1}(\Delta_{k_{max}}))$. Letting $\theta = \frac{\pi}{2} - tan^{-1}(\Delta_{k_{max}})$ we can easily show that $\Delta_{k_{max}} = tan(\frac{\pi}{2} - \theta) = cot(\theta) = \frac{1}{\theta} + O(\theta)$. Hence we can calculate $\Delta_{k_{max}} = \frac{\pi}{2(1-P_{max})} + O(\frac{2(1-P_{max})}{\pi})$ where the first order correction would be $O(\frac{2(1-P_{max})}{\pi}) = -\frac{2(1-P_{max})}{3\pi} + O((\frac{2(1-P_{max})}{\pi})^3)$. The linear correction being negative means that our original equation will choose a smaller k_{max} than our approximation would. Hence using this estimate you will contain a probability region slightly larger than P_{max} by setting $\Delta_{k_{max}} = \frac{\pi}{2(1-P_{max})}$. Combining this with equation (19) we will get $h^{\frac{1}{2}}(1 + \frac{h}{2})^{k_{max}} = \frac{\pi}{2(1-P_{max})}$. From here we can to define an integer $\kappa \geq k_{max}$ that covers the probability region with a length greater than P_{max} with the following equation.

$$\kappa = \text{Ceiling}[\frac{Log(\frac{4}{\pi^2(1-P_{max})^2 h})}{2Log(1 + \frac{h}{2})}] \tag{22}$$

This generates our last partitions created for finite regions of Δ be $\tilde{P}_{2\kappa}$,$\tilde{P}_{2\kappa+1}$. where $\tilde{P}_{2\kappa+2} = [P_{2\kappa+1}, 1]$ makes the region for the rest of the possible Δ values where $|\Delta| \geq \Delta_\kappa$.

3.5.2 Choosing sign for tail end value

If the random number lands in the $\tilde{P}_{2\kappa+2}$ region, then there are two more random numbers that need to be chosen. They can be done in either order with the easier of the two being used to pick if the number is negative or positive. The even symmetry of the Lorentzian makes it equally likely for Δ being positive as well as being negative. Let my random number s be generated in an interval of $[0, 1]$. we can define a function where if $s < \frac{1}{2}$ then we have a positive sign where if $s > \frac{1}{2}$ we have a negative sign. On the condition that $s = \frac{1}{2}$ it is best to regenerate a new s from the random number generator due to lack of preference.

3.5.3 Breaking down the tail region

Let us calculate the survival function of our distribution.

$$F(\Delta) = \int_{\Delta}^{\infty} \frac{d\Delta'}{\pi(1+\Delta'^2)} \tag{23}$$

Note that our survival function is equivalent to our cummulative distribution function in the negative region, $F(\Delta) = \int_{-\infty}^{-\Delta} \frac{d\Delta'}{\pi(1+\Delta'^2)}$. The exact analytical solution for this equation is defined below.

$$F(\Delta) = \frac{1}{\pi}(\frac{\pi}{2} - tan^{-1}(\Delta)) \tag{24}$$

If we presume Δ to be a large number, the function will be computationally more difficult for the computer to calculate. Using a geometric expansion we can approximate this equation exactly as

$$F(\Delta) = \frac{1}{\pi} \sum_{n=0}^{\infty} \frac{1}{(2n+1)\Delta^{2n+1}} \tag{25}$$

If we sufficiently define a P_{max} where the resulting κ from equation (22) creates a $\Delta_\kappa >> 1$ then we can estimate $F(\Delta) \approx \frac{1}{\pi\Delta}$ with an error of $O(\Delta^{-3})$. With this estimate we can calculate the conditional probability of a subinterval $[\Delta_a, \Delta_b)$ of the interval $[\Delta_\kappa, 0)$. First note that the exact conditional probability is

$$P(\Delta \in [\Delta_a, \Delta_b) | [\Delta_\kappa, \infty)) = \frac{F(\Delta_a) - F(\Delta_b)}{F(\Delta_\kappa)} \tag{26}$$

Using the zero order estimate, we find that our probability can be approximated as

$$P(\Delta \in [\Delta_a, \Delta_b) | [\Delta_\kappa, \infty)) \approx \frac{\Delta_a^{-1} - \Delta_b^{-1}}{\Delta_\kappa^{-1}} \tag{27}$$

which equation (27) has an error of $O(\Delta_\kappa^{-3})$ for large $\Delta_\kappa >> 1$. Using our definition of (19), we can define an interval $[\Delta_{\kappa+n-1}, \Delta_{\kappa+n})$ where if we plug in the formula into (27) we will get

$$P(\Delta \in [\Delta_{\kappa+n-1}, \Delta_{\kappa+n}) | [\Delta_\kappa, \infty)) = \frac{h}{2}(1 + \frac{h}{2})^{-n} \tag{28}$$

And the survival conditional probability would be

$$P(\Delta \in [\Delta_{\kappa+n}, \infty)|[\Delta_\kappa, \infty)) = (1 + \frac{h}{2})^{-n} \tag{29}$$

The implication of these conditional probabilities is that the probability space that is in the interval $[\Delta_\kappa, \infty)$ can be broken down into geometric regions where the relative probability density difference in intervals $[\Delta_{\kappa+n-1}, \Delta_{\kappa+n})$ still remain less than h.

Analogously this defines partitions for the region $[\Delta_\kappa, \infty)$ similarly to the ones defined in 3.2. However since the region still has infinite subintervals what we can do instead is use a random number to determine our subinterval choice. Let us define these conditional intervals $[0, 1]$ partitions as $\tilde{P}_{\kappa,n}$. where

$$P_{\kappa,0} := 0, P_{\kappa,n} := P([\Delta_{\kappa+n-1}, \Delta_{\kappa+n})|[\Delta_\kappa, \infty)) + P_{\kappa,n-1} \tag{30}$$

Making the partition interval $\tilde{P}_{\kappa,n} = [P_{\kappa,n-1}, P_{\kappa,n})$. Since $[0, 1]$ is now partitioned by these $\tilde{P}_{\kappa,n}$ intervals, then we can use the random number generator once again to find the interval $[\Delta_{\kappa+n-1}, \Delta_{\kappa+n})$ with using a random number $r' \in [0, 1]$, we can estimate n using the geometric partitioning. If $r' \in \tilde{P}_{\kappa,n}$ then $P_{\kappa,n-1} \leq r' < P_{\kappa,n}$ which we can use $r' = P_{\kappa,n'}$ where $n - 1 \leq n' < n$. By the equation in (30) and using (29) we know that $r' = 1 - (1 + \frac{h}{2})^{-n'}$. Since the random number $r = 1 - r'$ is just as likely as r' then we can calculate n in the following equation

$$n = \text{Ceiling}[-\frac{Log(r)}{Log(1 + \frac{h}{2})}] \tag{31}$$

This randomly generates the interval $[\Delta_{\kappa+n-1}, \Delta_{\kappa+n})$ with the probability given in equation (28). from there we can randomly generate $\Delta \in [\Delta_{\kappa+n-1}, \Delta_{\kappa+n})$ knowing that the probability density varies at most by a factor of h. I would like to note there is also an exact methodology that can be used to estimate in general by defining

$$n = \text{Ceiling}[\frac{Log(\frac{2}{h}cot(r(\frac{\pi}{2} - tan^{-1}\Delta_\kappa))}{Log(1 + \frac{h}{2})}] - \kappa \tag{32}$$

However I used the prior condition for (31) since it is computationally less expensive and the error isn't dominated by the underlying method errors of the computational framework.

4 Analysis

So far we have came up with a theoretical framework that can be used to generate random variables behaving like a Lorentzian. I have put that to the test and used that to generate a histogram. After we have confirmed we have a Lorentzian with a generated histogram, we can adjust the Lorentzian with a mean μ and quasi variance γ to calculate what an accumulation function may look like.

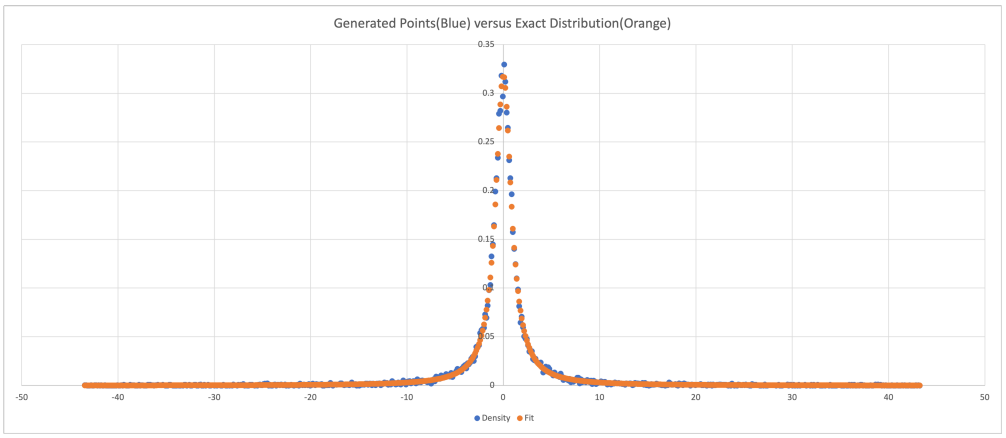

Figure 3: This is a scatter plot histogram that has the theoretical fit being the orange points while the blue points are generated from the function described in the theory section.

4.1 Confirming Background Lorentzian

To show that a Lorentzian can be generated by the methodology discussed in this paper, take a look at figure 3. In figure 3 I graphed the generated points in blue and the expected value in orange. In total there were 20,000 points generated where I set $P_{max} = 0.999$ and our step size to $h = 0.01$. This however has a large domain of values that had a minima of -13897.59081 and a maximum of 8013.970027 generated. This is due to fact that the variance of a Lorentzian distribution is infinite. The generated data set was for Δ values making it centered around 0 and have an effective $\gamma = 1$. due to the large variance of the data, we find that our bin size was about 0.130727097 which is $\frac{1}{1000}$th of the standard deviation $\sigma = 130.7$. The data set here is a sample of points ranging from $\Delta \in [-43.45993376, 43.21213127]$. As you can see from the graph 3, there are differences between the orange exact curve and the blue simulated points. Out of the 20,000 points, this interval consists of 19,741 points in total which is 98.705% of the graph. Using this we can do a comparison with the binomial standard deviation like what was done for the points collected from the S&P 500.

The plot in figure 4 describes the likelyhood of how different the point on the theoretical graph is from the generated graph. The standard deviation is calculated like the previous graph 2 where the standard deviation is calculated from a binonomial distribution as if a random number making the bin has a probabilty P and not making the bin is $1 - P$. P is calculated from the theoretical curves density times the bin size. From the figure 4, we can see that the distribution behaves in a statistical matter. We would expect if the binomial distribution is near Gaussian that 95% of data should be within 2 standard deviations. The figure 4 does show that the vast majority of data points are under two standard deviations making the data differences look like pure white noise. Other standard values calculate a $\chi^2 = 0.002598568$ which considering that $\langle \text{Density}|\text{Density} \rangle = 1.232326228$, $\langle \text{Density}|\text{Fit} \rangle = 1.223593571$ and $\langle \text{Fit}|\text{Fit} \rangle = 1.217459481$ is really small by the innerproduct standard discussed in (6). Using equation (7), we get that the effective Euclidean angle is $\theta = 0.045662208$ and $cos^2(\theta) = 0.997916411$. With the Born interpretation, our generated histogram matches the theoretical probability distribution function by 99.79%. One thing that maybe of relevance from the formulation of the generated data is that

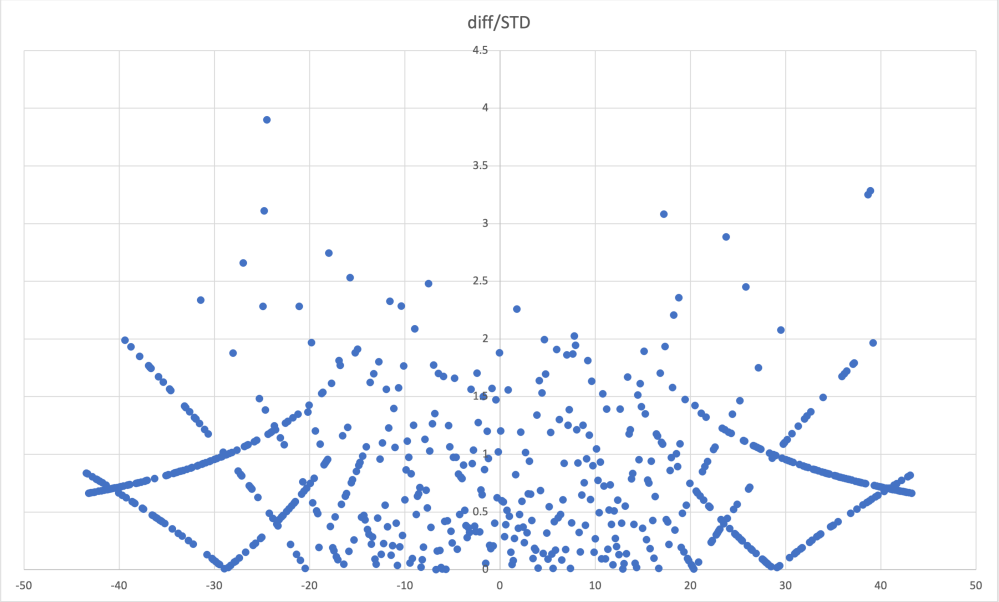

Figure 4: This scatter plot measures the difference in frequency between the generated points and the frequency expected from the fit compared it to the standard deviation of a binomial. the y-axis is the number of standard deviations different, and the x-axis is bin number using $\Delta \in [-43.45993376, 43.21213127]$

$\langle \text{Density}|\text{Density}\rangle \geq \langle \text{Fit}|\text{Fit}\rangle$. One possible reason why the norm of the density is slightly larger than fit may be due to the fact that generated points assume that the probability is flat in regions where the probability density varies by $h = 0.01$, this may cause more numbers to be slightly more probable than they actually are making the graph a tiny amount more spread out.

4.2 Generated accumulation function data

We have been able to observe that our random number generator does indeed produce a Lorentzian distribution for the probability density function defined as $\rho(\Delta) = \frac{1}{\pi(1+\Delta^2)}$. Thus what we can do is use this random number generator to generate millions of points for this distribution. With those millions of Δ values saved to memory, we can randomly select a point among those millions of points and calculate our continual interest rate as $\delta = \gamma\Delta + \mu$ where δ defined in equation (2) where δ replaces x used in the definition of a lorentzian in equation (1). Our accumulation function can be calculated from each value of δ. For a time step $\partial t = t_i - t_{i-1}$, the accumulation function will have a continuous interest rate δ_i randomly selected. Our accumulation function can be generated from $a(t_i) = a(t_{i-1})e^{\delta_i \partial t}$. Using this definition in a recursive fashion would define $a(t_i) = a(0)e^{\sum_{j=1}^{i} \delta_j \partial t}$. Conventionally an accumulation function is defined to have $a(0) = 1$ making our generated accumulation function be

$$a(t_i) = e^{\partial t \sum_{j=1}^{i} \delta_j} \tag{33}$$

This equation can be reformulated in a matter that makes it easier to see what is going on among various accumulation functions by taking a logarithm of the data.

$$Log(a(t_i)) = \partial t \sum_{j=1}^{i} \delta_j \tag{34}$$

$$Log_{10}(a(t_i)) = \frac{\partial t}{Log_{10}(e)} \sum_{j=1}^{i} \delta_j \tag{35}$$

where both equations are valid for graphing $a(t)$ due to the change of base formula. However it is a lot more clear for a person to work in base-10. The data I have provided of generated accumulated functions are $Log_{10}(a(t))$ on the y-axis and time on the x-axis in years in figures 5 and 6. Both of these figures use the μ and γ values calculated for the S&P 500. The graphs have 20 simulations in each graph representing the various possibilities over the next 20 years. For the most part the graphs show that the accumulation function almost works in a step like manner where there are random instances where the accumulation function jumps in value significantly or decreases in value significantly. For the most part what these simulations tend to say that over long periods of times there will be a general trend where the accumulation function does not change a lot but every once in a while there is a spike/drop due to random conditions from tail interest rates. These swift changes in the market maybe due to an event that can cause a company implode or expand in a ridiculous manor. The S&P 500 being a mutual fund of 500 companies make can swiftly change if one of the companies just spikes in value dominating the portfolio, or when a dominating company tanks causing the value of the S&P 500 to drop significantly.

One must note though that even though these type of accumulation functions do somewhat simulate what goes on the market, the underlying assumption that the market behaves with no correlation is obviously wrong. Short term traders tend to invest in things that are seem to have upward trends and sell assets from things that have a downward trend. This would mean that there should be expected upward trajectories correlated over a time frame of several days, to months to years of correlation. The model here does not assume that hence has a lot more random fluctuation causing the interest rate to be nearly flat for most lengths of time with a slight bias towards increasing.

5 Conclusions

We have covered from our findings that the S&P 500 continual interest rates vary like a Lorentzian, how to simulate a Lorentzian in a computer and how to generate accumulation functions for a fund. Even though the contents of this paper are simple, there is an implication that we could generalize a method of how to create simulations for general probability density functions. As an extra application, we could utilize this method of generating Lorentzian distributions for simulations of emission/absorption spectra in physics. There are also hints from this of what could be done to improve this model. I believe the next step to make the model more sophisticated is to find the auto-correlation of the data with respect to differences in time. We could generate a probability distribution, where we compare an

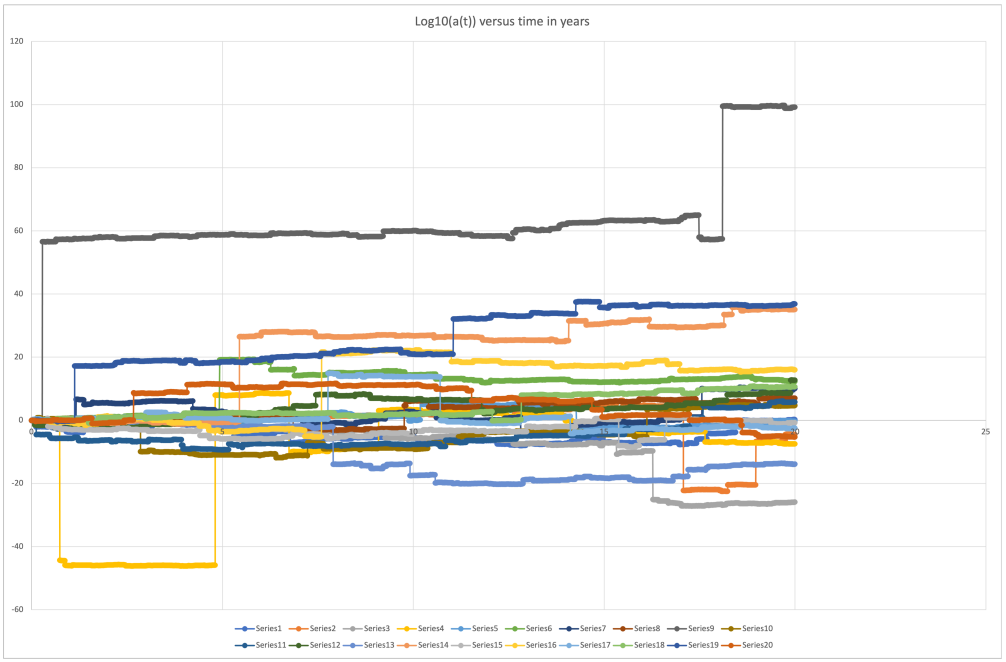

Figure 5: This figure is a set of generated accumulation functions that have $\gamma = 1.268911575$ and $\mu = 0.21691887$ to simulate the S&P 500. The y-axis is the $Log_{10}(a(t))$ while x-axis is the number of years passed where $\partial t = \frac{1}{365}$ years and $t \in [0, 20]$.

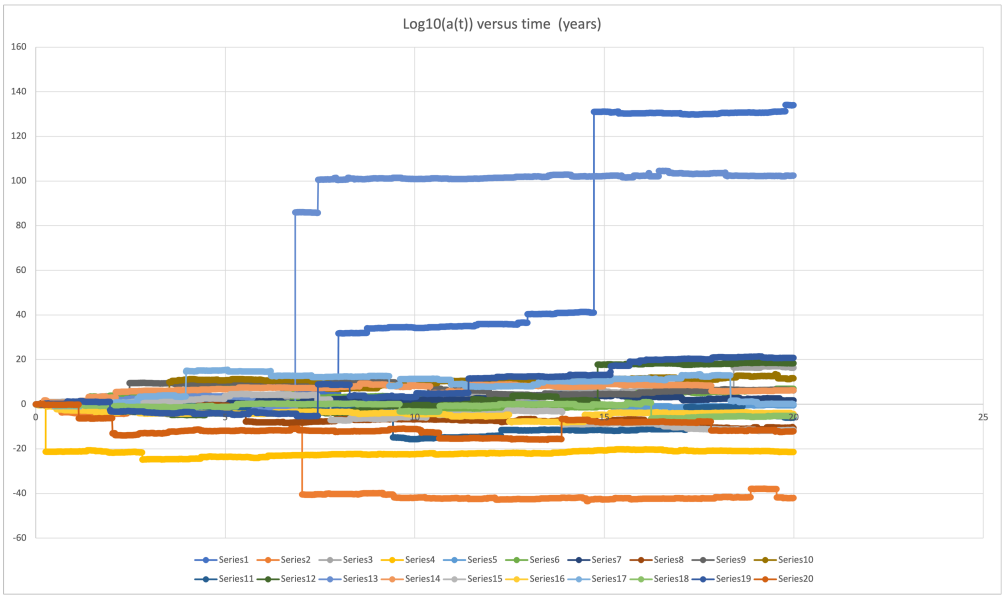

Figure 6: This figure is a set of generated accumulation functions that have $\gamma = 1.268911575$ and $\mu = 0.21691887$ to simulate the S&P 500. The y-axis is the $Log_{10}(a(t))$ while x-axis is the number of years passed where $\partial t = \frac{1}{365}$ years and $t \in [0, 20]$.

interest rate δ_1 occurring at some time t and another interest rate δ_2 at some time $t+\tau$. These two methods can help predict what time correlated factors produce the general trends that we see in the actual market unlike the near flat trends we see from the generated accumulation functions in this paper. One trivial idea that I have thought about is an idea of interest opacity. Just like a random walk in Brownian motion, where a particle has a constant trajectory until it collides, we can also assert that there is a time frame where the interest rate can be near constant until it collides with a market shift. My expectation for this time frame should follow a probability density function that is exponential $\rho(\tau) = \lambda e^{-\lambda\tau}$. Another avenue that would be worthwhile is to find a methodology to model the relative noise of the data with respect to the Lorentzian. Considering that figure 2 shows that difference is not within statistical reason means that 8.8% of noise is impacting the underlying probability distribution significantly. A methodology that I would like to investigate is to generate an orthogonal set of functions to the Lorentzian that we can project the noise onto. We could do this by creating a Hamiltonian where the Loretzian is a solution such as $H = -\frac{\partial^2}{\partial x^2} + V(x)$, we can define $V(x)$ in a way where that if $\rho(x)$ is a Lorentzian then $V(x) = E_0 + \frac{1}{\rho}\frac{\partial^2 \rho}{\partial x^2}$ which would be a real function hence making H hermitian, and the eigenvalue for the Lorentzian would be E_0. We could utilize SU decomposition to find eigenvalues/eigenvectors of the hermitian operator to generate our orthogonal basis.

References

[1] R.N. Mantegna and H.E. Stanley, *Introdouction to Econophysics:Correlations and Complexity in Finance*, (Cambridge University Press; 1st edition, 2007).

[2] *Market Data*, available at `https://www.marketwatch.com/investing/index/spx/download-data`.