CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

METHODS TO SOLVE ASSET BUBBLE IN FINANCE

A thesis submitted in partial fulfillment of the requirements
For the degree of Master of Science in
Applied Mathematics

by

Jaspreet Kaur

August 2013

The thesis of Jaspreet Kaur is approved:

Dr. Stephen Breen	Date
Dr. Vladislav Panferov	Date
Dr. Jorge Balbás, Chair	Date

California State University, Northridge

Dedication

Jas' dedication

Acknowledgements

I would like to thank my advisor Dr. Jorge Balbas

Table of Contents

ABSTRACT

METHODS TO SOLVE ASSET BUBBLE IN FINANCE

By

Jaspreet Kaur

Master of Science in Applied Mathematics

Financial Market is very attracting topic in finance and mathematics world. Recently we have heard a lot about Gold Prices inflations. It is the the hot topic in today's finance market. So how will be combine mathematics with today's asset changes like gold? How can we determine the tale of asset's volatility for future? These are the questions which we will consider in this thesis. We will study non parametric estimator Floren Zmirou in local real time on compact domain with stochastic differential equation which has unknown drift and diffusion coeificents. Once we will have volatility from floren zmirou then we will able to use RKHS to estimates function which will extrapolate the tale of function.

Chapter 2

0.1 Theoretical Background

First of all we will introduce Stochastic Differential Equations SDE. SDE are used in various fields like biology, physics, mathematics and of course finance. We will focus in finance , SDE is used to model asset price with Brownian motion. In this thesis work, we will use constant parameters drift μ and diffusion coeficients σ. Let's suppose we have μ and σ With these assumptions, we will use euler muruyama method to model asset price.
In this chapter, we will focus on numerical solution of stochastic differential equations (SDE). It will give us better understanding toward the theory behind SDE. We will also study Brownian motion and compute Brownian paths with different methods for example Euler -Maruyama method, strong and weak convergence , milstein method are being used to show solutions of SDE.
Now let's start with finance knowledge, Suppose the market price of an asset increases significantly. How can one determine if the market price is inflated above the actual price of an asset? This price behavior is know as a bubble.
To model price bubbles, we want to consider the following:

- What is an asset price bubble?
- How does one determine if an asset price is experiencing a bubble?

- Market Price - The current price of an asset.
- Fundamental Price - The actual value of an asset based on an underlying perception of its *true value*.
- Risk - Variance of return on an asset
- Portfolio - Set of Assets.
- Asset Bubble- The difference between the market and fundamental price, if any, is a price bubble.
- Strike price -The strike price or exercise price of an option is the fixed price at which the owner of the option can buy(in the case of call) or sell (in the case of a put) the underlying security or commodity.
- Volatility-Rate at which the price of security moves up and down.

Lets's consider mathematical definations :

0.1.1 Probability Space

$(\Omega, \mathcal{F}, P)$ where Ω is a set (sample space), $\mathcal{F}$ is a sigma algebra of subsets (events) of Ω, and P is a Probability Measure.

0.1.2 Random Variable

- Measurable functions of real analysis. $X : \Omega \to \mathcal{R}$ map $X : (\Omega, \mathcal{F}) \mapsto (\mathcal{R}, \mathcal{B})$ and X is random variable if
$X^{-1}(A)\epsilon\mathcal{F}, \forall A\epsilon\mathcal{B})$,
where $X^{-1}(A) := \omega\epsilon\Omega \mid X(\omega)\epsilon A$

0.1.3 Stochastic Differential Equations

We treat the asset price as a stochastic process:

0.1.3.1 Stochastic Process

Given a probability space $(\Omega, \mathcal{F}, P)$, a stochastic process with state space X is a collection of X-valued random variables, S_t, on Ω indexed by a set T (e.g. time).

$$S = \{S_t : t \in T\} \tag{1}$$

One can think of S_t as a asset price at time t.

0.1.3.2 Stochastic Differential Equation

A differential equation with one or more terms is a stochastic process.

0.1.4 Brownian Motion

$\{S_t : 0 \leq t \leq T\}$:

$$\begin{aligned} dS_t &= \sigma(S_t)dW_t + \mu(S_t)dt \\ S_0 &= 0 \end{aligned} \tag{2}$$

- W_t denotes the standard Brownian Motion.
- $\mu(S_t)$ called the drift coefficient.
- $\sigma(S_t)$ called the volatility coefficient.

0.1.4.1 Defination of Brownian motion

A continuous-time stochastic process $\{S_t : 0 \leq t \leq T\}$ is called a *Standard Brownian Motion* on $[0, T]$ if it has the following four properties:

(i) $S_0 = 0$

(ii) The increment of S_t are independent; given

$$0 \leq t_1 < t_2 < t_3 < \cdots < t_n \leq T$$

the random variables $(S_{t_2} - S_{t_1})$, $(S_{t_3} - S_{t_2})$, $\cdots$, $\left(S_{t_n} - S_{t_{n-1}}\right)$ are independent.

(iii) $(S_t - S_s)$, $0 \leq s \leq t \leq T$ has the Gaussian distribution with mean zero and variance $(t - s)$

(iv) $S_t(W)$ is a continuous function of t, where $W \in \Omega$.

0.1.5 Martingales

(a) $E[\mid S_n \mid] < +\infty$, for all n.

(b) S_n is said to be *adapted* if and only if S_n is $\mathcal{F}_n$-measurable.

The stochastic process $S = \{S_n\}_{n=0}^{\infty}$ is a *martingale* with respect to $(\{\mathcal{F}_n\}, P)$ if $E[S_{n+1} \mid \mathcal{F}_n] = S_n$, for all n, almost surely and:

- S satifies (a) and (b).

0.1.6 Supermartingale

The stochastic process $S = \{S_n\}_{n=0}^{\infty}$ is a *supermartingale* with respect to $(\{\mathcal{F}_n\}, P)$ if $E[S_{n+1} \mid \mathcal{F}_n] \leq S_n$, for all n, almost surely and:

- S satifies (a) and (b).

0.1.7 Local Martingale

If $\{S_n\}$ is adapted to the filtration $\{\mathcal{F}_n\}$, for all $0 \leq t \leq \infty$, then $\{S_n : 0 \leq t \leq \infty\}$ is called a *local martingle* provided that there is nondecreasing sequence $\{\tau_k\}$ of stopping times with the property that $\tau_k \to \infty$ with probability one as $k \to \infty$ and such that for each k, the process defined by

$$S_t^{(k)} = S_{t \wedge \tau_k} - S_0$$

for $t \in [0, \infty)$ is a martingale with respect to the filtration

$$\{\mathcal{F}_n : 0 \leq t < \infty\}$$

0.1.8 Remark

A strict local martingale is a non-negative local martingale.

0.1.9 Theorem

If for any strict local martingale

$$\{S_t : 0 \leq t \leq T\}$$

with $E\left[|S_0|\right] < \infty$ is also a supermartingale and $E\left[S_T\right] = E\left[S_0\right]$, then $\{S_t : 0 \leq t \leq T\}$ is in fact a martingale.

0.1.10 Remark

- $\{S_t : 0 \leq t \leq T\}$ is a supermartingale and a martingale if and only if it has constant expectation.
- For a strict local martingale its expectation decreases with time.

0.1.11 Relating Martingales and Bubbles

0.1.12 theorem

$\{S_t : 0 \leq t \leq T\}$ is a strict local martingale if and only if

$$\int_{\alpha}^{\infty} \frac{x}{\sigma^2(x)} dx < \infty \tag{3}$$

for all $\alpha > 0$.

- A bubble exists if and only if (3) is finite.
- We shall call (3) the volatility of asset return.
- In this scope, the difference between a martingale and a strict local martingale is whether the volatility of asset return, (3), is finite or not finite.

0.1.13 Numerical Methods of SDE

For $t \in [0, T]$, (2) can be represented in an integral form in the following way:

$$\begin{aligned} dS_t &= \sigma(t)dW_t + \mu(t)dt \\ \int_0^t dS_t &= \int_0^t \sigma\left(S_t\right) dW_t + \int_0^t \underbrace{\mu\left(S_t\right)}_{\in \mathcal{R}^+} dt \\ S_t - S_0 &= \int_0^t \sigma\left(S_t\right) dW_t + \left(\underbrace{\mu\left(S_t\right) \cdot t}_{x_0} - \mu\left(S_t\right) \cdot 0 \right) \\ S_t &= x_0 + \int_0^t \sigma\left(S_t\right) dW_t \end{aligned}$$

0.1.13.1 What is $S_t = x_0 + \int_0^t \sigma\left(S_t\right) dW_t$?

The price model is

$$S_t = x_0 + \int_0^t \sigma\left(S_t\right) dW_t \tag{4}$$

$$\begin{aligned} dS_t &= \mu(S_t)dt + \sigma(S_t)dW_t \\ S_0 &\in \mathcal{R} \end{aligned} \tag{5}$$

0.1.13.2 The Euler-Maruyama Method

Equation (4) can be written into integral form as:

$$S_t = S_0 + \int_0^t f\left(S_s\right) ds + \int_0^t g\left(S_s\right) dW(s), t \in [0, T] \tag{6}$$

f, g are scaler function with $S_0 = x_0$**â** random variable

$$\begin{cases} dS_t = \mu(S_t)dt + \sigma(S_t)dW_t \\ S(0) = S_0 \end{cases}$$

Using Euler Maruyama method:
$w_0 = S_0$

$w_{i+1} = w_i + a(t_i, w_i) \triangle t_{i+1} + b(t_i, w_i) \triangle W_{i+1}$

$w_{i+1} = w_i + \mu w_i \triangle t_i + \sigma w_i \triangle W_i$

$\triangle t_{i+1} = t_{i+1} - t_i$

$\triangle W_{i+1} = W(t_{i+1} - W(t_i)$
Now drift coefficient μ and diffusion coefficient σ are constants, the SDE has an exact

solution:

$$S(t) = S_0 \cdot Exp\left(\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W(t)\right) \tag{7}$$

0.1.14 What is $S_t = x_0 + \int_0^t \sigma\left(S_t\right) dW_t$?

$$S(t) = S_0 \cdot Exp\left(\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W(t)\right) \tag{8}$$

For an example, we use the Euler-Maruyama Approximation Method on the SDE where the constants $\mu = 2$, $\sigma = 1$, and $S_0 = 1$ are given.

There are other methods such as Strong and weak convergence of the Euler Muruyama method, Milstein's Higher Order Method, Linear Stability and Stochastic Chain Rule are also used for numerical solutions for SDE.

- From this, we will focus on real time stock data. We will have couple estimators to determine volatility function. For instance,
- We will assume that σ is not constant. We will approximate σ with non parametric estimator method on local time.

0.1.15 Methods used to solve the problem

In this classical setting, Jarrow, Protter, and Shimbo [19], [20] show that there are three types of asset price bubbles possible. Two of these price bubbles exist only in infinite horizon economies, the thirdÑcalled type 3 bubblesÑexist in finite horizon settings. Consequently, type 3 bubbles are those most rele- vant to actual market experiences. For this type of bubble, saying whether or not a bubble exists amounts to determining whether the price process under a risk neutral measure is a martingale or a strict local martingale: if it is a strict local martingale, there is a bubble.

Stock price is strict local martingale if and only if

$$\int_\alpha^\infty \frac{x}{\sigma\left(x\right)} dx < \infty forall \alpha > 0 \tag{9}$$

First We will obtain data from Google Finance and Yahoo Finance. Algorithms 1, 2 and 3 shows the process how to get Stock Prices. Floren Zmirou's non parametric estimator is based on the local time of the Diffusion Process.

- Diffusion Process

In probability theory, a branch of mathematics, a diffusion process is a solution to a stochastic differential equation. It is a continuous-time Markov process with almost surely continuous sample paths.

0.1.16 Theorem1

Theorem 0.1.1 *If σ is bounded above and below from zero, has three continuous and bounded derivatives, and if $(h_n)_n{}^3 1$satisfies $nh_n \to \infty$ and $nh_4 n \to 0$, then $S_n(x)$ is a consistent estimator of $\sigma^2(x)$.*

In Theorem 0.1.1 we say XYZ....

0.1.17 Theorem2

If $nh_n \to 0$, then $\sqrt{N_x^n}((\frac{S_n(x)}{\sigma^2(x)}) - 1)$ converges in distribution to $\sqrt{2}Z$ where Z is a standard normal ransom variable and N_x^n = $1_{\{|S_{ti}-x|<hn\}}$

Floren Zmirou estimator requires a grid step which is $h_n \to 0$ such that $nh_n \to \infty$ and $nh_n^4 \to 0$. We will choose step size $h_n = \frac{1}{n^{\frac{1}{3}}}$ so that all the conditions will be satisfied.

0.1.18 Floren Zmirou

Lets consider following equation:

$$S_t = S_0 + \int_0^t \sigma\left(S_t\right) dW_t \tag{10}$$

In Floren Zmirou, the drift coefficient $\mu(S_t)$ is null which is ignored without loss of generality. It is not involved in our problem. $\sigma(S_t)$ the volatility coefficient is unknown. We will follow steps for Floren Zmirou method:

- $(S_{t1} \dots S_{tn})$ are the stock prices in the interval $t_1 \dots t_n \in [0, T]$
- Without loss of generality, we assume T = 1, therefore $t_i = i/n$

Estimator as follows:
Local time = $l_T(x) = lim_{\epsilon \longrightarrow 0} \frac{1}{2\epsilon} \int_0^T 1_{\{|S_s - x| < \epsilon\}} d\langle S, S\rangle_s$

where $d\langle S, S\rangle_s = \sigma^2(S_s)$

$L_T(x) = lim_{\epsilon \longrightarrow 0} \frac{1}{2\epsilon} \int_0^T 1_{\{|S_s - x| < \epsilon\}} dS$

$\implies l_T(x) = \sigma^2(S_s) L_T(x)$

$\implies l_T(x)_T(x) = \sigma^2(S_s)$

Now we will define local time of S_s in x during [0,t].

Let's assume that $nh_n \to \infty$ and $h_n \to 0$

$\implies L_T^n(x) = \frac{T}{2nh_n}\Sigma_{i=1}^n 1_{\{|S_{t_i}-x|<h_n\}}$

estimator of $\sigma^2(S_x)$ as follows:

$$S_n(x) = \frac{\Sigma_{i=1}^n 1_{\{|S_{t_i}-x|<h_n\}} n(S_{t_i+1} - S_{t_i})^2}{\Sigma_{i=1}^n 1_{\{|S_{t_i}-x|<h_n\}}} \tag{11}$$

$S_n(x) = \frac{\Sigma_{i=1}^n 1_{\{|S_{t_i}-x|<h_n\}} n(S_{t_i+1}-S_{t_i})^2}{\Sigma_{i=1}^n 1_{\{|S_{t_i}-x|<h_n\}}}$

Now we have $\sigma^2(x)$ and we will use interpolation methods to see function's behaviour.

0.1.19 Interpolation

Interpolation is a method of constructing new data points within the range of a discrete set of known data points. There are many methods to do interpolation. For example Linear Interpolation,polynomial interpolation, piecewise constant interpolation, spline interpolation. Here we will interpolate an estimate of $\sigma^2(x_i)$ where $i \in [1.M]$ within the bounded finite interval D where we have observations. We used cubic spline interpolation and Reproducing Kernal Hilbert Spaces to get interpolation function.

0.1.19.1 Cubic Spline Interpolation

- **Piecewise-polynomial approximation** : An alternative approach is to divide the interval into a collection of subintervals and construct a different approximating polynomial on each subinterval. Approximation by functions of this type is called piecewise-polynomial approximation.
- The most common piecewise-polynomial approximation uses cubic polynomial between each sucessive pair of nodes and is called **cubic spline interpolation**.
- **Cubic Spline Interpolation**
  Given a function f defined on [a,b] and a set of nodes $a = x_o < x_1 < \ldots < x_n = b$ a **cubic spline interpolant S** for f is a function that satisfies the following conditions:

  a) $S(x)$ is a cubic polynomial, denoted $S_j(x)$ on the subinterval $[x_j, x_{j+1}]$ for each $j = 0, 1, 2, \ldots, n-1$;

  b) $S_j(x_j) = f(x_j)$ and $S_j(x_{j+1}) = f(x_{j+1})$ for each $j = 0, 1, 2, \ldots, n-1$;

  c) $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ for each $j = 0, 1, 2, \ldots, n-2$;

d) $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ for each $j = 0, 1, 2, \ldots, n-2$;

e) $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ for each $j = 0, 1, 2, \ldots, n-2$;

f) One of the following sets of boundry conditions is satisfied;

i) $S''(x_0) = S''x_n) = 0$;

ii) $S'(x_0) = f'x_0)$ and $S'(x_n) = f'x_n)$

- **Natural Spline** When the free boundry conditions occur, the spline is called **natural spline**.

0.1.19.2 Interpolation is seen as inverse problem.

0.1.19.3 We will have two types of solutions for inverse problem.

0.1.19.4 Normal Solution

It will allows an exact interpolation with minimal squared norm.

0.1.19.5 Regularized Solution

it will yields quasi interpolative results, accompained by an error bound analysis with Tikhonov Regularization produces an approximate solution f_α which belongs to $H(D)$and that can be obtained via the minimization of the regularization functional. Regularized solution with Kernal function For interpolation, we denote kernal function $K^{a,b}_{n,\tau}$ for n =1 and n = 2. of $H^n(a, d)$ where $D = (a, b)$ Now one we have interpolation function and extended form of the estimated $\sigma^2(x)$ call this extended function $\sigma^b(x)$, we can decide if there is need of extrapolation.

- if the volatility $\sigma^2(x_i)$ doesn not diverge to ∞ when $x \to \infty$ and it remains bounded on $\mathcal{R}^+$.

  - No extrapolation is required.

  - $\int_\epsilon^\infty \frac{x}{\sigma^2(x)}$ is infinite.

  - The process is true martingale.

- If the volatility diverges to ∞ when $x \to \infty$ then we will extrapolate.

0.1.20 Extrapolation ,Optimization and Minimization

Now since we have extended function $\sigma^b(x)$, we will extrapolate this over $\mathcal{R}^+$. Let's consider following optimization with minimazation problem:

$\bar{m}$ = argminm≥ 0 $\sqrt{\int_{[a,\infty[\cap D} |\sigma_m - \sigma^b|^2 dS}$

σ_m will interpolate the input data points when n = 2.

$\sigma_{\bar{m}}$ has the asymptotic behavior that beast matches our function on the estimation interval. We will construct extrapolation by choosing the asymptotic weighting function parameter m such that $f_m = \frac{1}{\sigma^2 m}$.

a = max S - $\frac{1}{3}$(maxS-min S)

$H_{2,m}$ allows the best interpolation of M which is estimated points such that the extrapolation function remains as close as possible to $\sigma^b(x)$.

We will plot function with different asymptotic weighting parameter m which is obtained from RKHS extrapolation method.

The asymptotic weighting function's parameter mbtained by optimization and minimization is the most consistent which exactly match the input data within all the function in $H_{1,m}$

Chapter 1

Implementation of an Asset Bubble Problem

1.1 Implementation

We used sage python to solve this problem. We orgnize our data and process in six classes. Each class is designed to store the information about Stock, Floren Zmirou AssetBubble, AssetBubbleDetection, Approximation, and Run. First we will work with Stock Class. First We will start with time and stock prices for some stock symbol. First question which will come in mind that How to download and use historic stock prices? So to answer that question, we initialize Stock Class which will explain step by step process how to download, open, read and save historical stock prices.

1.1.1 Stock Class

Methods of Stock Class We obtain stock data by two types of methods. First, we will get it through Google Finance and second, we will get it from CSV file which we download and saved from yahoo finance. There are four functions IsNumber,GetGoogleData, GetStockPrice,and init in Stock Class.

1. `IsNumber()`: Input for this function is rowValue. This function determine if rows of stock data is a numerical string or not.

2. `GetGoogleData()`: Input for this function are Ticker,days and period. This function obtains data for any stock from Google finance.

3. `GetStockPrices()`: Input for this function is csv filename. This function obtains stock prices which are stored in csv file by filename. It will read the yahoo API based minute to minute data.

4. `__init__()`: Input for this function is keyward Usage.This function will analyzie wheather csv filename is used or Ticker parameter for Google finance.

1.1.2 Floren Zmirou Class

This implements equation (11)

Process to solve Floren Zmirou Now we have Stock Prices from Stock class. We will obtain list of sigma values from list of Stock Prices.We will explain this class in following algorithms:

T = 60*n where T is the minute to minute time period [0,T].
Now we will derive hn, x-step_size, and x values which will be used in Floren Zmirou Estimator.

Algorithm 1 GetGoogleData

INPUT: Ticker, days, period
OUTPUT: Obtain S (Stock Prices) in either NASD or NYSE.

if the length of the Ticker ≤ 3 **then**
    exchange it with New York Stock Exchange (NYSE).
else
    exchange it with Natinal Association of Securities Dealers (NASD)
end if
We initialize current time in integer.
We will open Google Finance link.
DataList = read each line from Google Finance link.
We initialize tickerData to be the list of array of dataList.
We will put stockPrices in list.
We initialize minuteData.

for minuteData in tickerData **do**
    We split minuteData in commas and put in list.
    We append row one in float and store it in S list.
return S

Algorithm 2 GetStockPrices

INPUT: filename
OUTPUT: Obtain S (Stock Prices) from CSV file.

cr = open and read the csv filename.
We skip the header.
c1 = the empty list.
for row in cr **do**
    if the second row is numberical string **then**
        we add second row to c1.
return c1

Algorithm 3 –init–

INPUT: Keyword Usage
OUTPUT:

if filename is in keyward Usage **then**
    We will obtain Stock Prices from GetStockPrices in list of filename Ticker is used in keyward Usage
    We will obtain Stock Prices from GetGoogleData in a list of Ticker
elseWe will give exception of bad parameters

5. `(Derive_hn)`: Input for this function is Stock Prices which we obtained from Stock class. This function will derive hn which will be used in Floren Zmirou Estimator.

Algorithm 4 Derive hn

INPUT: Stock Prices S
OUTPUT: hn

1: n = length of Stock Prices S.
2: hn = 1.0/n**(1.0/3.0)
3: return hn =0

6. `(x_step_size)`: Input for this function is Stock Prices which we obtained from Stock class. This function will derive x_step_size which will be used to create step size to generate x.

Algorithm 5 x_hn = x step size

INPUT: Stock Prices S
OUTPUT: $\frac{x}{hn}$

1: We obtained hn from Derive hn function.
2: Double hn = 2 **hn
3: Difference = max S-min S
4: x_hn = Difference * Double hn
5: Return x_hn

7. `(Derive_x_values)`: Input for this function is Stock Prices which we obtained from Stock class. This function will derive x_values which will be used in Floren Zmirou Estimator. Now we have all the ingridents which will be

Algorithm 6 Derive x Values

INPUT: Stock Prices S
OUTPUT: x

1: halfh_n = x_hn /2.0
2: x = empty list.
3: Append x with min S + halfh_n
4: We initilaize ex to be first element of x's list.
5: **while** ex < max S **do**
6: ex = ex+x_hn
7: **end while**
8: We will append ex into x's list
9: Return x

usefull to solve floren zmirou estimator. We will implement floren Zmirou on the following functions. Floren Zmirou has Sublocal time, Local time, volatility Estimator and Indicator function.

8. `(Sublocal_Time)`: Input for this function are T,S,x,n,hn Output for this function will be $L_T^n(x) = (\frac{T}{2nhn}) \sum_{i=1}^{n} 1_{|S\_t(i)-x)|<hn}$

Algorithm 7 Sublocal Time

INPUT: T=Time,S= Stock Prices,x= grid points,n= number of total Stock Prices,hn=Step Size
OUTPUT: $L_T^n(x) = (\frac{T}{2nhn}) \sum_{i=1}^{n} 1_{|S\_t(i)-x)|}$

```
sum = 0.0
scalar = T/(2.0*n*hn)
for i in range of the length of Stock Prices do
    We initialize Sti to be the ith element of the list of Stock prices
    absoluteValue = abs(Sti-x)
     We initilize indicatorValue to pass through indicator function
    if absoluteValue is less than hn then
        sum = sum+indicatorValue return scalar*sum
```

9. `(Local_Time)`: Input for this function are T,S,x,n,hn. Output for this function will be $L_T^n(x) = (\frac{T}{2nhn}) \sum_{i=1}^{n} 1_{|S\_t(i)-x)|<hn} * n(S(t(i+1)) - S(t(i))^2$

Algorithm 8 Local Time

Inputs: T=Time,S= Stock Prices,x= grid points,n= number of total Stock Prices,hn=Step Size
Steps

```
sum = 0.0
scalar = T/(2.0*n*hn)
for for i in range of the length of Stock prices - 1 do
    We initialize Sti to be the ith element of the list of Stock prices
    We initialize Stj to be the jth element of the list of Stock prices
    absoluteValue = abs(Sti-x)
    Difference = (Stj-Sti)**2
     We initilize indicatorValue to pass through indicator function
    if  absoluteValue is less than hn then
        sum = sum+indicatorValue*n*Difference return scalar*sum
```

10. `(Volatility_Estimator Sn(x))`: Volatility Estimator is $\sigma^2(x)$.Input for this function are T,S,x,n,hn. Output for this function will be Local Time/Sublocal Time. We have $\sigma^2(x)$ for Stock Prices. Now we want to see how many stock

Algorithm 9 Volatility Estimator

Inputs: T=Time,S= Stock Prices,x= grid points,n= number of total Stock Prices,hn=Step Size

Steps

1: We will initialize ratio to be Local Time / Sublocal Time.
2: Return ratio. =0

prices are in each grid point. If there are less than 0 or 1 percent of stock prices in grid then we will exclude that grid point from our calculation process.

11. `(DoGridAnalysis)` :Input for this function are T,S,x,n,hn. Output for this function give us the list of usable grid points and for each usable grid point, the list of Stock prices.

Algorithm 10 DoGridAnalysis

INPUT: T=Time,S= Stock Prices,x= grid points,n= number of total Stock Prices,hn=Step Size,Y = Y percent of total data points.
OUTPUT:

1: x = useableGridPoints.
2: d = empty dictionary
3: Si = length of stock prices.
4: **for** grid Points in x **do**
5: create a dictionary where the keys are the grid points and the value is a list for each grid point
6: **for** stockPrice in S **do**
7: **if** |grid Points - stockPriceCount|$\leq$ hn **then**
8: Add x value to corresponding Si
9: **for** grid Points in x **do**
10: **if** if the list of data points corresponding to x values than Y percent of total grid points **then**
11: add it to the list of usable grid points
12: L = dictionary with key values of grid points.
13: N = Length of L in float.
14: Percent of Stock Prices = N/S
15: **if** Percent of Stock prices < Y **then**
16: Remove grid points from x
17: Delete L
18: **return** x = usable grid points and d

12. `(GetGridVariance`:Input for this function is self.

13. `(GetGridInverseStandardDeviation)` :Input for this function is self.

Algorithm 11 GetGridVariance

INPUT:
OUTPUT: Standard Deviation of usable grid points

1: Points = Empty list
2: **for** x in usable grid points **do**
3: y =Sn(x))
4: Put (x,y) in Points's list
5: **return** Points

Algorithm 12 GetGridInverseStandardDeviation

INPUT:
OUTPUT: Standard Deviation of usable grid points

1: Points = Empty list
2: **for** x in usable grid points **do**
3: y = 1/sqrt(Sn(x))
4: Put (x,y) in Points's list
5: **return** Points

Natural Cubic Spline- To construct the cubic spline interpolant S for the function f, defined at the numbers $x_0 < x_1 < \ldots < x_n$ satisfying $S''(x_0) = S''(x_n) = 0$:

14. `(Natural Cubic Spline)`

Algorithm 13 Natural Cubic Spline

INPUT: $n, x_0, x_1, \ldots, x_n; a_0 = f(x_0), a_1 = f(x1), \ldots, a_n = f(x_n)$
OUTPUT: a_j, b_j, c_j, d_j for $j = 0, 1, \ldots, n-1$.
(note: $S_(x) = S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$ for $x_j \leq x \leq x_{j+1}$)

for $i = 0, 1, \ldots, n-1$ **do**
    $h_i = x_{i+1} - x_i$
end for
for $i = 1, 2, \ldots, n-1$ **do**:
    $\alpha_i = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1})$
end for
Set $l_0 = 1$;
$\mu = 0$;
$z_0 = 0$.
for $i = 1, 2, \ldots, n-1$ **do**
    $l_1 = 2(x_{i+1} - x_{i-1}) - h_{i-1}\mu_{i-1}$;
$\mu_1 = \frac{h_1}{l_1}$;
$z_1 = \frac{(\alpha_1 - h_{i-1}z_{i-1})}{l_1}$.
Set $l_n = 1$;
$z_n = 0$;
$c_n = 0$.
    for $j = n-1, n-2, \ldots, 0$ **do**
        $c_j = z_j - \mu_j c_{j+1}$;
        $b_j = (a_{j+1} - a_j)/h_j - h_j(c_{j+1} + 2c_j)/3$;
        $d_j = (c_{j+1} - c_j)/(3h_j)$
Return (a$_j, b_j, c_j, d_j for j = 0, 1, \ldots, n-1$);