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

Table of Contents

ABSTRACT

METHODS TO SOLVE ASSET BUBBLE IN FINANCE

By

Jaspreet Kaur

Master of Science in Applied Mathematics

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. We will obtain volatility funtion then we will interpolate with cubic spline to see the behavior of the function.

Chapter 1

Implementation of an Asset Bubble

In this chapter. we will provide valueable information about the methods used in our problem. We used sage python to solve this problem. We orgnize data and process in four classes. Each class is designed to store the information about Stock, Floren Zmirou ,Cubic Spline,and Run.

1.1 Stock

Let's consider following questions:

- What kind of Data will be used?
- How can someone obtain accurate historic data information?
- Will it be Mintue to mintue data information?
- How we will store and use data information?
- Can we get data information for any stock symbol in market?

Following Stock Class which will provide answers to all the above questions.

1.1.1 Stock Class

We used Google Finance and CSV file from Yahoo Finance methods to download stock data information. There are GetGoogleData, GetStockPrice and init are the four algorithms used in Stock Class which will decribe the process.

- S Stock Price.
- Smax Maximum Stock Price.
- Smin Minimum Stock Price.
- Row of Stock prices is numberical string.
- Ticker is Ticker symbol name.
- D Number of days.
- T time in seconds in intergers.

We will read, download and save stock data by following methods.

1.1.1.1 Google Finance Stock Prices

1. Get Google Data: This function obtains mintue to mintue stock prices in numberical string for any ticker from Google Finance and save in array.

Algorithm 1 Get Google Data

INPUT: Ticker, D ,T
OUTPUT: $S = (S_{t_1} \dots S_{t_n})$ in [0,T] where $t_i = \frac{i}{n}T$.

if Ticker length $\leq$ 3 **then**
    exchange Ticker with New York Stock Exchange (NYSE).
else
    exchange Ticker with Natinal Association of Securities Dealers (NASD)
end if
Open Google Finance link.
DataList = reads each line from Google Finance link.
TickerData = list of array of DataList.
Put stock Prices in a list.
for MinuteData in TickerData **do**
    Seperate MintueData with commas and store in a list S.
    Append S as float and save it.
return S = Stock prices

1.1.1.2 Yahoo Finance Stock Prices

2. Get Yahoo Data : This function obtains the stock prices which are downloaded from yahoo finance and stored in csv file by filename. It will provide the yahoo API based minute to minute data.

Algorithm 2 Get Yahoo Data

INPUT: Filename
OUTPUT: S from CSV file.

Open Yahoo Finace link.
Open and read the csv Filename as universal.
Skip the header.
Create a empty list.
for row in csv Filename **do**
    if the second row is numberical string **then**
        Add second row to empty list
Name this row S.
return S

1.1.1.3 Choosing either Google Ticker Or Yahoo CSV file

3. Class Constructor: This function will analyzie wheather to use csv filename for Yahoo or Ticker for Google finance.

Algorithm 3 –init–

INPUT: User Keyword
OUTPUT: S

if Filename is in keyward Usage **then**
    Obtain Stock Prices from Get Stock Prices. Ticker is used in keyward Usage
    Obtain Stock Prices from Get Google Data.
else Bad parameters
return S

1.1.2 Example

1.2 Floren Zmirou

This class will implement the following equation:

$$S_n(x) = \frac{\Sigma_{i=1}^{n} 1_{\{|S_{t_i} - x| < h_n\}} n(S_{t_i+1} - S_{t_i})^2}{\Sigma_{i=1}^{n} 1_{\{|S_{t_i} - x| < h_n\}}} \tag{1.1}$$

Now we have Stock Prices from Stock Class. Our goal is to get $\sigma(x)$ values from list of Stock Prices S. We will import sage and stock class to Floren Zmirou class. In order to use Floren Zmirou Estimator, we will need following:

- n = Number of stock prices.
- T = Time in seconds
- hn = $1/(n)^{(}1/3)$
- x-stepsize
- x values

- We will implement hn, x-stepsize, and x values so we can use the terms in Floren zmirou estimator equation (1.1).

1.2.1 Floren Zmirou Class

Algorithm 4 Derive hn

INPUT: Stock Prices S
OUTPUT: hn

```
n = length of Stock Prices S.
hn = 1.0/n**(1.0/3.0)
return hn =0
```

4. (x step size): This function will be used to create step size to generate x.

Algorithm 5 x_hn = x step size

INPUT: Stock Prices S
OUTPUT: x_hn

```
Double hn = 2 **hn
Difference = max S-min S
x_hn = Difference * Double hn
return x_hn
```

Algorithm 6 x Values

INPUT: Stock Prices S
OUTPUT: x

half h_n = x_hn /2.0
x = empty list.
Append x with Smin + half h_n
FirstElement = first element of x's list.
while FirstElement < Smax **do**
    FirstElement = FirstElement + x_hn
end while
Append FirstElement into x's list.
return x

- Now we will implement floren Zmirou Estimator. Floren Zmirou has Sublocal time, Local time, volatility Estimator and Indicator function.

5. Sublocal Time:This function will give $L_T^n(x) = (\frac{T}{2nhn}) \sum_{i=1}^{n} 1_{\left|S_{t_i}-x\right|<h_n}$

Algorithm 7 Sublocal Time

INPUT: T, S, x, n, hn.
OUTPUT: $L_T^n(x) = (\frac{T}{2nhn}) \sum_{i=1}^{n} 1_{\left|S_{t_i}-x\right|}$

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

6. Local Time: $L_T^n(x) = (\frac{T}{2nh_n}) \sum_{i=1}^{n} 1_{\left|S_{t_i}-x\right|<hn} * n(S(t(i+1)) - S(t(i))^2$
7. Volatility Estimator Sn(x): Volatility Estimator is $\sigma^2(x)$. This function will give Local Time/Sublocal Time.

Algorithm 8 Local Time

INPUT: T, S, x, n, hn.
OUTPUT:

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

Algorithm 9 Volatility Estimator

INPUT: T, S, x, n, hn.
OUTPUT:

```
Ratio = Local Time / Sublocal Time.
return Ratio =0
```

- We have $\sigma^2(x)$ for Stock Prices. Now we want to see how many stock prices are in each grid point. If there are less than 0 or 1 percent of stock prices in each grid then we will exclude that grid point from our calculation process.

8. (DoGridAnalysis): This function give us the list of usable grid points and for each usable grid point, the list of Stock prices.

1.3 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 $S''(x_0) = S''(x_n) = 0$:
(Natural Cubic Spline)

1.4 Run the Program

Algorithm 10 DoGridAnalysis

INPUT: T,S,x,n,hn.
OUTPUT:

```
x = useableGridPoints.
d = empty dictionary
Si = length of stock prices.
for grid Points in x do
    create a dictionary where the keys are the grid points and the value is a list for each grid point
    for stockPrice in S do
        if |grid Points - stockPriceCount| ≤ hn then
            Add x value to corresponding Si
            for grid Points in x do
                if if the list of data points corresponding to x values than Y percent of total grid points then
                    add it to the list of usable grid points
                    L = dictionary with key values of grid points.
                    N = Length of L in float.
                    Percent of Stock Prices = N/S
                    if Percent of Stock prices < Y then
                        Remove grid points from x
                        Delete L
return x = usable grid points and d
```

Algorithm 11 Get Grid Variance

INPUT: S,x_hn, x values
OUTPUT: Floren Zmirou estimation over usable grid points.

```
Points = Empty list.
half x_hn = x step size*(Stock Prices)/2.0
for x in usable grid points do
    y =Sn(x))
    Append (x,y)
return (x,y)
```

Algorithm 12 GetGridInverseStandardDeviation

INPUT:S,x_hn,x values.
OUTPUT: Standard Deviation of usable grid points.

Points = Empty list
for x in usable grid points **do**
    y = 1/sqrt(Sn(x))
    Append (x,y) in Points's list
return (x,y)

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 $(a_j, b_j, c_j, d_j for j = 0, 1, \ldots, n-1)$;