CISC 5352 FINANCIAL PROGRAMMING AND DATA ANALYTICS LECTURE NOTE (4)

Henry Han Ph.D.
Department of Computer and Information Science
Fordham University, New York NY 10023

## Last class review

## Last class review

① Python's data analytics package: Pandas

② Two basic data structures of Pandas: series and dataframe

③ Stock data retrieval using Pandas

④ Volatility computing via Pandas/How to compute volatility

**Volatility reflects the spread degree of underlying asset price instead of the moving direction.**

Up to now, we only use historical data to compute Volatility

Such a volatility can not predict an asset's future spread degree

The future spread degree needs implied volatility

Compute implied volatility is an essential problem in Finance

Robert Engle at NYU was awarded 2003 Nobel prize for its theory in volatility.

**Q1: How to compute volatility?**

**Q1: How to compute volatility?**

① Collect n+1 number of observations (e.g., daily observation): $0, 1, 2, \ldots, n$;

## Q1: How to compute volatility?

① Collect n+1 number of observations (e.g., daily observation): 0, 1,2,…,n;

② Use $S_i$ to represent the stock price (close price) in the ith interval: [i-1, i]

## Q1: How to compute volatility?

① Collect n+1 number of observations (e.g., daily observation): 0, 1,2,…,n;

② Use $S_i$ to represent the stock price (close price) in the ith interval: [i-1, i]

③ Compute a U sequence: : $u_i=\ln(S_i/S_{i-1})$, which is also called **log return** for (stock) data

## Q1: How to compute volatility?

① Collect n+1 number of observations (e.g., daily observation): 0, 1,2,…,n;

② Use $S_i$ to represent the stock price (close price) in the ith interval: [i-1, i]

③ Compute a U sequence: : $u_i=\ln(S_i/S_{i-1})$, which is also called **log return** for (stock) data

④ The volatility can be estimated as ratio of the standard deviation of the log return and the square root of the length of time interval

$$\hat{\sigma} = \frac{s}{\sqrt{\tau}}$$

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} u_i^2 - \frac{1}{n(n-1)}\left(\sum_{i=1}^{n} u_i\right)^2}$$

① s is the standard deviation of a U sequence

② $\tau$ : the length of time interval in years (e.g.,1/252)

## Q2: How to compute volatility under high frequency trading?

① Unlike traditional trading stock data, HFT data are usually recorded in 1 minute interval.

② Suppose there are 10000 HFT stock prices records (one stock but in a transaction period of10000 minutes), how to compute its volatility in a month?

③ How to compute volatility in a year?

$$\hat{\sigma} = \frac{s}{\sqrt{\tau}}$$

## Q2: How to compute volatility under high frequency trading?

① Unlike traditional trading stock data, HFT data are usually recorded in 1 minute interval.

② Suppose there are 10000 HFT stock prices records (one stock but in a transaction period of10000 minutes), how to compute its volatility in a month?

③ How to compute volatility in a year?

$$\hat{\sigma} = \frac{s}{\sqrt{\tau}}$$

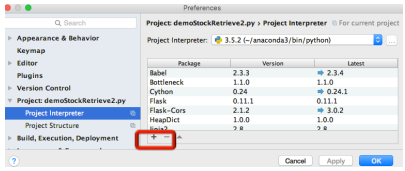
**That's one of your project 1 problems!**

## pandas.io.data is old! Its updated package is: pandas-datareader

If you use Ipython: add package pandas-datareader by typing the following commands in your terminal

pip install pandas-datareader

If you use PyCharm: go to Preference->Project Interpreter , Click '+' to add pandas-datareader package




## More information about datareader(Remote data access)

http://pandas-datareader.readthedocs.io/en/latest/remote_data.html

**Data sources**

- Yahoo! Finance
- Google Finance
- St.Louis FED (FRED)
- Kenneth French's data library
- World Bank
- OECD
- Eurostat

```
import numpy              as np
import pandas             as pd
import pandas_datareader.data as web_pdr

# Retrive all google stock from the s_date to the end_date

s_date='6/1/2008'  # start date
e_date='6/1/2016'  # end date

# Retrieve google stock ( 'GOOG') from google finance

goog2 = web_pdr.DataReader('GOOG', data_source='google',
                start=s_date, end=e_date)

# name the output file according to start and end date
filename= ("google_stock_data_"+"2" + ".xlsx")

# write data into a Excel file
writer  = pd.ExcelWriter(filename)
goog2.to_excel(writer, 'Sheet1')
writer.save()
```

```
goog.to_excel(filename, sheet_name='sheet1', index=False)
```

**An alternative Excel writing**
**Writer acts as a 'pointer'**

```
## check if the file exists

import os.path
if (os.path.isfile(filename)==True):
   print("\n "+filename + " is saved!\n" )
else:
   print("\n file not existed!\n")

# compute volitality of google

# 1. bookkeeping
S_i       = goog['Close']
S_i_minus_1 = goog['Close'].shift(1)

# 2. Create U sequence
goog['U_sequence']=np.log(S_i/S_i_minus_1)
```

rolling_std is deprecated for Series

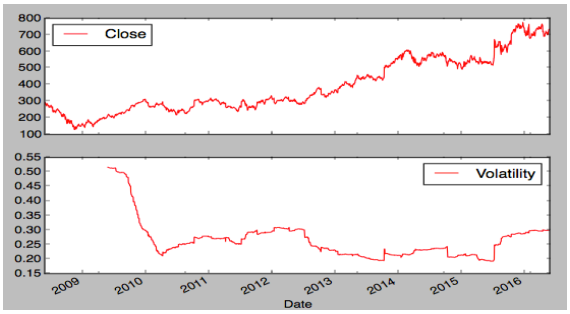
s=pd.rolling_std(goog['U_sequence'], window=252)

Replaced by a more OOP based update function

s=goog['U_sequence'].rolling(window=252,center=False).std()

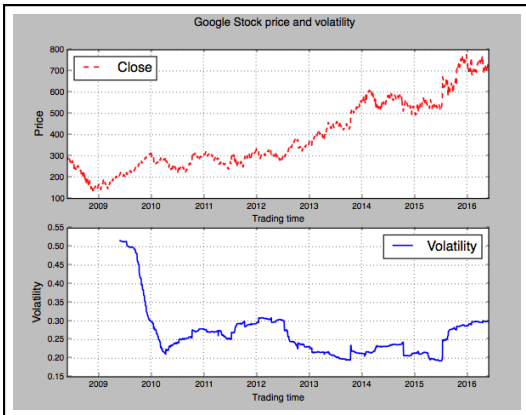
Series.rolling(window, min_periods=None, freq=None, center=False, win_type=None, axis=0)

```
# 3. Compute s
s=goog['U_sequence'].rolling(window=252,center=False).std()

# 4. Compute volatility
goog['Volatility'] = s* np.sqrt(252)


# 5. Invoke matplotlib to plot (only works for Ipython)
matplotlib
goog[['Close', 'Volatility']].plot(subplots=True, color='red',figsize=(8, 6));
```



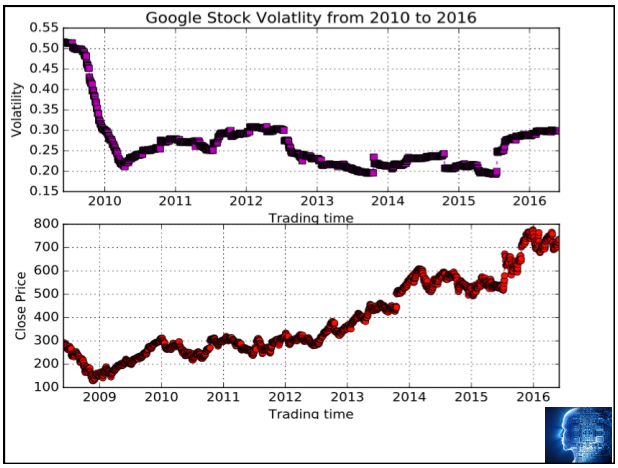

**Can we make the visualization better?**

```
gplot=goog[['Close', 'Volatility']].plot(subplots=True,
      title    = ' Google Stock price and volatility',
      sharex   = False,   # don't share x axis
      grid     ='on',
      fontsize  = 9,
      color    = ['red', 'blue'],
      style    = ['--', '-'],
      linewidth = 1.5,
      figsize   = (8, 6))


gplot[0].set_xlabel('Trading time', fontsize=10)
gplot[1].set_xlabel('Trading time', fontsize=10)
gplot[0].set_ylabel('Price', color='k')
gplot[1].set_ylabel('Volatility', color='k')
```




```
import matplotlib.pylab  as pylab

goog_volatlity=goog['Volatility']
pylab.subplot(2,1,1)
pylab.plot(goog_volatlity, 'm-.s')
pylab.xlabel('Trading time')
pylab.ylabel('Volatility')
pylab.title('Google Stock Volatlity from 2010 to 2016')
pylab.grid('on')


goog_close = goog['Close']
pylab.subplot(2,1,2)
pylab.plot(goog_close, 'r-.o')
pylab.xlabel('Trading time')
pylab.ylabel('Close Price')
pylab.grid('on')

filename='goog.eps'
pylab.savefig(filename)


import os.path
if (os.path.isfile(filename)==True):
   print("\n "+filename + " is saved!\n" )
else:
   print("\n file not existed!\n")
```

**Visualization done by pylab package**




## How about options?

Can we retrieve option data from Web automatically?

We still need to use Pandas' class Options to get Option data from Yahoo! Finance

from pandas.io.data import Options

We also need to a module html5lib and lxml

You can install html5lib and lxml by doing

- pip install html5lib
- Or install it from PyCharm
- Mac user: make sure your Xcode matches your OS!
- If your Xcode is old, lxml cannot be installed

## Get all AAPL 's options

```
from pandas.io.data import Options
import html5lib
aapl = Options('aapl', 'yahoo')
data = aapl.get_all_data()
```

**Type: 'whos' see the following items in your workspace**

```
In [5]: whos
Variable   Type        Data/Info
-------------------------------
Options    type        <class 'pandas.io.data.Options'>
aapl       Options     <pandas.io.data.Options object at 0x106a68b50>
data       DataFrame                        <...>n[1041 rows x 13 columns]
html5lib   module      <module 'html5lib' from 'html5lib/__init__.pyc'>
```

NOTE: it seems Pandas has an issue to retrieve option data from Yahoo finance due to Yahoo Finance's update and other package's update

## Check the first 5 row data

data.iloc[0:5, 0:5]

```
In [17]: data.iloc[0:5, 0:5]
Out[17]:
                                                 Last    Bid    Ask   Chg   PctChg
Strike Expiry     Type Symbol
34.29  2016-01-15 call AAPL160115C00034290    97.55  92.10  94.30  0.00    0.00%
                  put  AAPL160115P00034290     0.02   0.01   0.03  0.01  100.00%
35.71  2016-01-15 call AAPL160115C00035710    88.30  90.80  91.95  0.00    0.00%
                  put  AAPL160115P00035710     0.01   0.01   0.04  0.00    0.00%
37.14  2016-01-15 call AAPL160115C00037140    90.70  88.50  90.55  0.60    0.67%
```

## BID and ASK PRICE

Bid: The price a buyer is willing to pay for a security

Ask price is the price a seller is looking to get for his or her shares.

## List all column (attributes/header) names for this data frame

list(data.columns.values)

```
['Last',
 'Bid',
 'Ask',
 'Chg',
 'PctChg',
 'Vol',
 'Open_Int',
 'IV',
 'Root',
 'IsNonstandard',
 'Underlying',
 'Underlying_Price',
 'Quote_Time']
```

## Save it in an excel file for later usage

filename = 'aapl_option_data.xlsx'
data.to_excel(filename, sheet_name='sheet1', index=False)

Sure, you can also initialize an excel writer to finish this.

| Last | Bid | Ask | Chg | PctChg | Vol | Open_Int | IV | Root | Nonstanda | Underlying | Derlying_Pi | Quote_Time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 97.55 | 92.10 | 94.30 | 0 | 0.00% | 1 | 46 | 84.42% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 0.02 | 0.01 | 0.03 | 0.01 | 100.00% | 173 | 1254 | 60.16% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 88.3 | 80.80 | 91.95 | 0 | 0.00% | 809 | 7 | 97.70% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 0.01 | 0.01 | 0.04 | 0 | 0.00% | 15 | 1115 | 59.38% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 90.7 | 88.50 | 90.55 | 0.6 | 0.67% | 1 | | 85.99% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 0.02 | 0.00 | 0.04 | 0 | 0.00% | 5 | 815 | 56.25% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 86.45 | 87.40 | 89.95 | 0 | 0.00% | 5 | | 84.26% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 0.02 | 0.00 | 0.04 | 0 | 0.00% | 300 | 370 | 54.69% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 89.4 | 86.00 | 87.75 | 0 | 0.00% | 2 | 5 | 82.62% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 0.03 | 0.00 | 0.04 | 0 | 0.00% | 5 | 679 | 53.13% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 84.71 | 84.55 | 86.80 | 0 | 0.00% | 100 | | 80.11% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 0.04 | 0.00 | 0.05 | 0 | 0.00% | 2 | 586 | 52.73% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 87.6 | 83.80 | 84.80 | 0 | 0.00% | 1 | 112 | 75.78% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 0.03 | 0.00 | 0.04 | 0 | 0.00% | 140 | 1844 | 50.00% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 79.95 | 82.30 | 83.35 | 0 | 0.00% | 1601 | 8 | 73.10% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 0.03 | 0.02 | 0.05 | -0.02 | -40.00% | 225 | 722 | 51.56% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |
| 82.79 | 80.95 | 82.15 | 0 | 0.00% | 14 | 7 | 56.54% | AAPL | 0 | AAPL | 127.17 | 2015-06-14 16:00:00 |

## It is same under windows

But you need to install html5lib and lxml

To achieve this, you need to download cygwin, then doing the following start for your program

```
from pandas.io.data import Options
import lxml
import html5lib
```

## Open_Int: open interest

This value indicates the total number of contracts of a particular option that have been opened.

## What's "implied" volatility (IV)?

**Implied volatility is a "future volatility":** **market's opinion of the stock's potential move!**

① It is volatility of the stock in the future based on its option price in the market (it is also a ratio value).

**Implied volatility is a "future volatility": market's opinion of the stock's potential move!**

① It is volatility of the stock in the future based on its option price in the market (it is also a ratio value).

② We can't observe it directly from the historical pricing data!

**Implied volatility is a "future volatility": market's opinion of the stock's potential move!**

① It is volatility of the stock in the future based on its option price in the market (it is also a ratio value).

② We can't observe it directly from the historical pricing data!

③ It is a decision factor about an option's price!

**Implied volatility is a "future volatility": market's opinion of the stock's potential move!**

① It is volatility of the stock in the future based on its option price in the market (it is also a ratio value).

② We can't observe it directly from the historical pricing data!

③ It is a decision factor about an option's price!

④ It is the value that makes the theoretical price of an option under an option pricing model equal to its current market price.

## Volatility tells us what already happened: what's the standard deviation of the return of the underlying asset

① It is calculated by using the history data!

② Implied volatility will tell us what will happen in the future!

③ That is what's the future tendency of the underlying asset!

④ Implied volatility is a forward-looking measure

## The most popular index for implied volatility

SPX VIX is an index of the implied volatility of 30-days options on the S&P 500 calculated from a wide range of calls and puts

Want to know more about VIX: check their white paper

https://www.cboe.com/micro/vix/vixwhite.pdf

Example: Which option will be chapter?

Two call options which will expire one year later

| Option | Option Price | Stock Price | Implied Volatility |
|---|---|---|---|
| 1 | $1.50 | $42.05 | 28.0% |
| 2 | $2.10 | $43.34 | 17.2% |

**The second one for its low implied volatility (more predictable return), even if the option's price is higher.**

| Option | Option Price | Stock Price | Implied Volatility |
|---|---|---|---|
| 1 | $1.50 | $50.00 | 20.0% |
| 2 | $2.20 | $55.00 | 10.0% |

**Accurate implied volatility estimation is essential for option pricing and the investment/trading risk control!**

## How to estimate implied volatility?

### Traditional Model driven approach ()

- Implied volatility is the value that makes the theoretical price of an option under an **option pricing model** equal to its current market price.
- **It means implied volatility is the solution to this equation**
  **ModelOptionPrice=OptionMarketPrice**
- The most successful option pricing model is **Black–Scholes (BS) model**
- **BS model is a model published in 1973 for option pricing. Black and Scholes were awarded 1997 Nobel economics prize for this model!**

## Model driven approach (BS model)

Use **Black–Scholes (BS) model to compute implied volatility by solving a nonlinear equation numerically:** the implied **volatility is the root of this equation**

**BSModelOptionPrice(S,K,r, T, $\sigma_{imp}$)=OptionMarketPrice**

**S: stock price (known)**
**K strike price (known)**
**t: interest rate (known)**
**T: maturity time (known)**
**OptionMarketPrice: (known)**
$\sigma_{imp}$ : implied volatility (unknown)

## Model driven approach (BS model)

Use Black–Scholes (BS) model to compute implied volatility by solving a nonlinear equation numerically: the implied volatility is the root of this equation

BSModelOptionPrice(S,K,r, T, $\sigma$imp)=OptionMarketPrice

S: stock price (known)

K strike price (known)

t: interest rate (known)

T: maturity time (known)

OptionMarketPrice: (known)

$\sigma$imp : implied volatility (unknown)

## We still assume we have a European call option)

We can view the BSM model as an option pricing formula

$$C(S_t,K,t,T,r,\sigma) = S_t \cdot \mathbf{N}(d_1) - e^{-r(T-t)} \cdot K \cdot \mathbf{N}(d_2)$$

$$\mathbf{N}(d) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{d} e^{-\frac{1}{2}x^2} dx$$

$$d_1 = \frac{\log \frac{S_t}{K} + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}}$$

$$d_2 = \frac{\log \frac{S_t}{K} + \left(r - \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}}$$

**Different?**

## Suppose an option price is C*

Then the implied volatility $\sigma_{imp}$ is the quantity that solve the equation

$$C\left(S_t,K,t,T,r,\sigma^{imp}\right) = C^*$$

Implied volatility is a solution to the equation that makes the theoretical value equal to market value

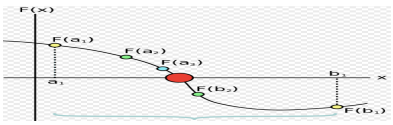
## How to solve this equation?

## How to solve this equation?

There is no analytic solution→ we have to use numerical method to solve it→Find the root of an equation!

## Bisection Method: the basic method to estimate the implied volatility

① A numerical method to find the root of a function
② Straightforward and 'intuitive'
③ Easily coding




## The target function of the Bisection method

f(x) = BSMPrice(S,K,r, T, x) – MarketPrice

x is just $\sigma_{imp}$ → the implied volatility

BSMPrice means the option price calculated by BSM model

MarketPrice means the current option price from market

## The mathematical details of Bisection methods

To find the root of equation f(x)=0, we need an interval [a,b], where f(a) has opposite sign with f(b).

There must be a value "c" between "a" and "b", that can make f(c)=0.

"c" is the root of the equation f(x)=0.

## Bisection Algorithm

① First, input initial interval [a,b], where f(a)*f(b)<0.

② Second, calculate the midpoint c between a and b, subject to c=(a+b)/2.

③ Third, calculate function value at c, which is f(c).

## Bisection Algorithm Cont'd

Fourth, compare the sign of f(c) to that of f(a) and f(b)

If f(a)*f(c)<0, set [a,c] as the subinterval

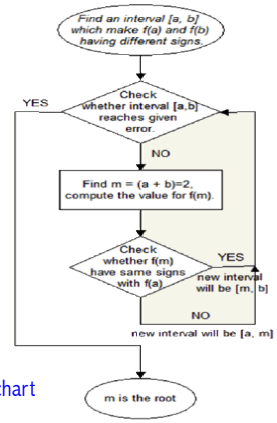
If f(b)*f(c)<0, set [c,b] as the subinterval

Fifth, calculate $c_2$ as the midpoint of subinterval and repeat step 2~4 for n times, until $f(c_n)=0$ or $f(c_n)<\varepsilon$, where $\varepsilon$ stands for tolerant error.

small ←→a
big ←→ b
middle ←→ c

## Bisection Algorithm Code segment

```
①  while(fsmall*fbig<0){
②      middle=(small+big)/2.0;
③          fmiddle=calculateValue(middle);

④                  if(fsmall*fmiddle<0){
⑤                   big=middle;
⑥                   fbig=calculateValue(big);
⑦          }

⑧          if(fmiddle*fbig<0){
⑨                  small=middle;
⑩                  fsmall=calculateValue(small);
11          }
12  }
```




Bisection method flowchart

## Pros and cons of Bisection

## Pros and cons of Bisection

This method is simple and robust.

What you need is only to set the initial interval, making sure f(a)*f(b)<0.

It is globally convergent. It will always converge to a real solution.

## Pros and cons of Bisection Cont'd

The disadvantage is that it is relatively slow.

Each time of iteration reduces the width of subinterval by 50%

**The order of convergence is 1 (linear),** and the number of iterations is $\log_2((b-a)/\varepsilon)$, where $\varepsilon$ is the tolerant error.

## An example

Data: Apple call & put option prices on July 25, 2013

Stock price: 438.5

Risk-free rate: 0.02 (3-month T-bill rate)

Time to expiration: 19/365=0.0521

Example: August 13 call option

| K | C | IV |
|---|---|---|
| 425 | 16.67 | 0.198474 |
| 425 | 18.35 | 0.250799 |
| 425 | 20.2 | 0.304535 |
| 430 | 11.05 | 0.137883 |
| 430 | 11.25 | 0.144102 |

## Last question for Bisection: what will be a and b for call and put options

You can search in any internal you like. But the implied volatility is always between 0 and 1.

## Next class: Newton-Raphson (Newton) method

It is a root-finding method faster than bisection method with higher convergent rate (convergence order: quadratic convergence: 2)

## Newton-Raphson (Newton) method cont'd

It assumes f(x) is differentiable

It starts with a first guess $x_0$ for the root

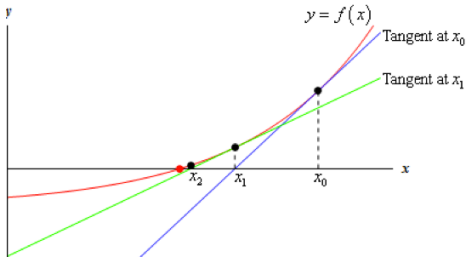
It moves to find a possible root

$x_1 = x_0 - f(x_0)/f'(x_0)$

Keep going

$x_n = x_{n-1} - f(x_{n-1})/f'(x_{n-1})$ until $f(x_n)=0$ or <tolerance

## Newton-Raphson (Newton) method cont'd




## Pros and cons of Newton method

① Newton's method may not converge if started too far away from a root.

② When it does converge, it is faster than the bisection method, and is usually quadratic.

③ For some functions, it is difficult to calculate the derivative.

## Come back our "fancy version": C →f(x)→ can we use newton method? If so, how?

$$C\left(S_t, K, t, T, r, \sigma^{imp}\right) = C^*$$