# A model of Accelerated Share Repurchase contract

Olivier Guéant, Jiang Pu and Guillaume Royer in '*Accelerated Share Repurchase: pricing and execution strategy*' (available at https://arxiv.org/pdf/1312.5617.pdf – the paper is denoted as *the paper* below) developed a model of an accelerated share repurchase contract (ASR). The file attached implements this model within a Python class object *ASR*.

**ASR Object Description**

Initialization of an *ASR* object:

***ASR(****S0, sigma, T, N, V, Q, eta, fi, gamma****)***

*int* ***S0*** – the initial price of a share,

*float* ***sigma*** – the annual volatility,

*int* ***T*** – the maturity of a contract,

*list([int, int])* ***N*** – the lower and upper boundary of time, in which the bank can choose whether it delivers the shares,

*int* ***V*** – the market volume over the unit of time,

*int* ***Q*** – the number of shares to buy,

*float* ***eta****, float* ***fi*** – the parameters of the function $L(\rho) = eta|\rho|^{1+fi}$, which models the execution costs,

*float* ***gamma*** – the risk aversion.

Methods of an *ASR* object:

***ASR.initialize(****NQ, INF****)***

Set the *q*-grid, infinity value, and performs all necessary methods to start working with an object.

*int* ***NQ*** – the number of cells in the *q*-grid (*q*-grid builds in the range of [0, Q]),

*float* ***INF*** – the value for infinity (by default, it is $10^9$).

***ASR. get_TETAs()***

Calculate all values of $\Theta_n(q, \zeta)$ according to Proposition 8 in the paper (see page 16).

***ASR.save_TETAs()***

Save all values of $\Theta_n(q, \zeta)$ to a text file '*teta_qgrid_{NQ}_gamma_{gamma}.txt*'

***ASR.save_gzip_TETAs()***

Save all values of $\Theta_n(q,\zeta)$ to a compressed *gzip* file '*teta_qgrid_{NQ}_gamma_{gamma}.gzip*'

***ASR.read_TETAs(****filename****)***

Read all values of $\Theta_n(q,\zeta)$ from a text file.

*str* ***filename*** – the name of a file to read.

***ASR.read_gzip_TETAs(****filename****)***

Read all values of $\Theta_n(q,\zeta)$ from a compressed *gzip* file.

*str* ***filename*** – the name of a file to read.

***ASR.get_PI()***

Calculate the indifference price of the contract, according to Proposition 3 in the paper (see page 11). The result is stored in the property *float* ***ASR.PI***.

***ASR.set_example_S(****i****)***

Set the price trajectory of the shares to one of the given in the paper (see 5.2.1). The result is stored in the property *list(int)* ***ASR.s***.

*int* ***i*** – the number of a price trajectory:

> *i* = 1 for the first price trajectory having an upward trend (see Figure 1),
>
> *i* = 2 for the second price trajectory having a downward trend (see Figure 3),
>
> *i* = 3 for the third price trajectory oscillating around *S0* (see Figure 5).

***ASR.set_S()***

Set the price trajectory of the shares to a randomly generated one according to the pentanomial rule shown on the page 16:

$$\epsilon_n = \begin{cases} +2 \quad with\ probability\ \dfrac{1}{12} \\ +1 \quad with\ probability\ \dfrac{1}{6} \\ 0 \quad with\ probability\ \dfrac{1}{2} \\ -1 \quad with\ probability\ \dfrac{1}{6} \\ -2 \quad with\ probability\ \dfrac{1}{12} \end{cases}$$

The result is stored in the property *list(int)* ***ASR.s***.

***ASR.get_A()***

Calculate the average prices. The result is stored in the property *list(int)* ***ASR.a***.

***ASR.get_Z()***

Calculate Z spread. The result is stored in the property *list(int)* ***ASR.z***.

***ASR.get_q()***

Calculate the number of shares remained to buy for the optimal buy-only strategy. The result is stored in the properties *list(int)* ***ASR.q*** for remains and *list(int)* ***ASR.v*** for shares bought.

***ASR.save_results()***

Save the results related to a specific price trajectory to a txt file:
'*data_qgrid_{NQ}_gamma_{gamma}.txt*'.
The output file contains 6 tab-separated fields which correspond, respectively, to time, price *ASR.s*, bought shares *ASR.v*, shares remained to buy *ASR.q*, average price *ASR.a*, and Z *ASR.z*.

***ASR.plot_trajectory()***

Plot the graph of the price trajectory, containing the price spread, the average price, and Z spread, like Figure 1 in the paper.

***ASR.plot_otimal_strategy()***

Plot the graph of the optimal strategy, containing the buy-only strategy and Z spread, like Figure 2 in the paper.

**Required Modules**

The program uses the following Python modules:

*gzip*

*numpy*

*matplotlib.pyplot*

**Configuration of the Program**

By default, the program configured as follow:

- the values of the variables are set according to the reference scenario, shown on page 18 of the paper,
- the number of *q*-grid elements is 10 (to change it use line: NQ = 10 # the computational grid for q),
- the program calculates all values of $\Theta_n(q,\zeta)$ and saves them to a *gzip* file, see lines:

  *# uncomment 2 of the 3 following lines to calculate and save TETAs*
  *# - use 'save_TETAs()' to save results to a text file*
  *# - use 'save_ip_TETAs()' to save results to a gzip file*
  *scenario.get_TETAs()*
  *#scenario.save_TETAs()*
  *scenario.save_gzip_TETAs()*

- the block which reads of $\Theta_n(q,\zeta)$ from a file is commented, see lines:

  *# uncoment the 2 of the 4 following lines to read TETAs from a file:*
  *# - use 'read_TETAs' to read from a text file*
  *# - use 'read_gzip_TETAs' to read from a gzip file*
  *#filename = 'teta_qgrid_50_gamma_2.5e-07.txt' # define a filename to read TETAs*
  *#scenario.read_TETAs(filename)*
  *#filename = 'teta_qgrid_50_gamma_2.5e-07.gzip' # define a filename to read TETAs*
  *#scenario.read_gzip_TETAs(filename)*

*NOTE:*

One set of $\Theta_n(q,\zeta)$ is used for a price trajectory of a specific scenario and *q*-grid. One can calculate and save $\Theta_n(q,\zeta)$ for the given scenario and chosen *q*-grid (for example, *NQ* = 20). Then one can use these $\Theta_n(q,\zeta)$ to simulate any price trajectory for the given scenario.

The higher value of *NQ* gives more accurate results but uses more computer resources (runtime and memory space).

NQ = 20 gives results which deviate by ~10% from the results shown in the paper.

- it is chosen the first example trajectory from the paper, see lines:

  *# ucomment the following line to choose an example price trajectory*
  *scenario.set_example_S(1) # specify the number of the example price trajectory: 1, 2 or 3*

- the block which generates a new price trajectory is commented, see lines:

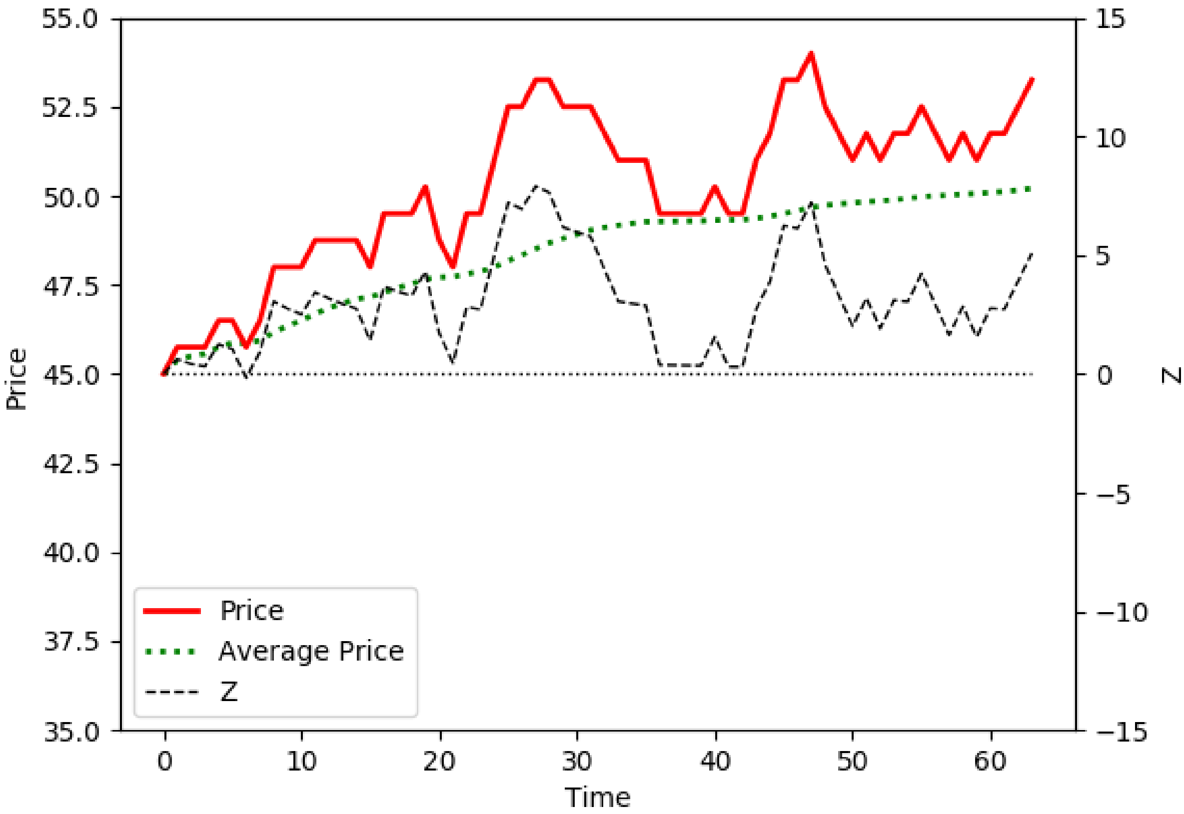
  *# ucomment the following line to generate a new price trajectory*
  *# scenario.set_S()*

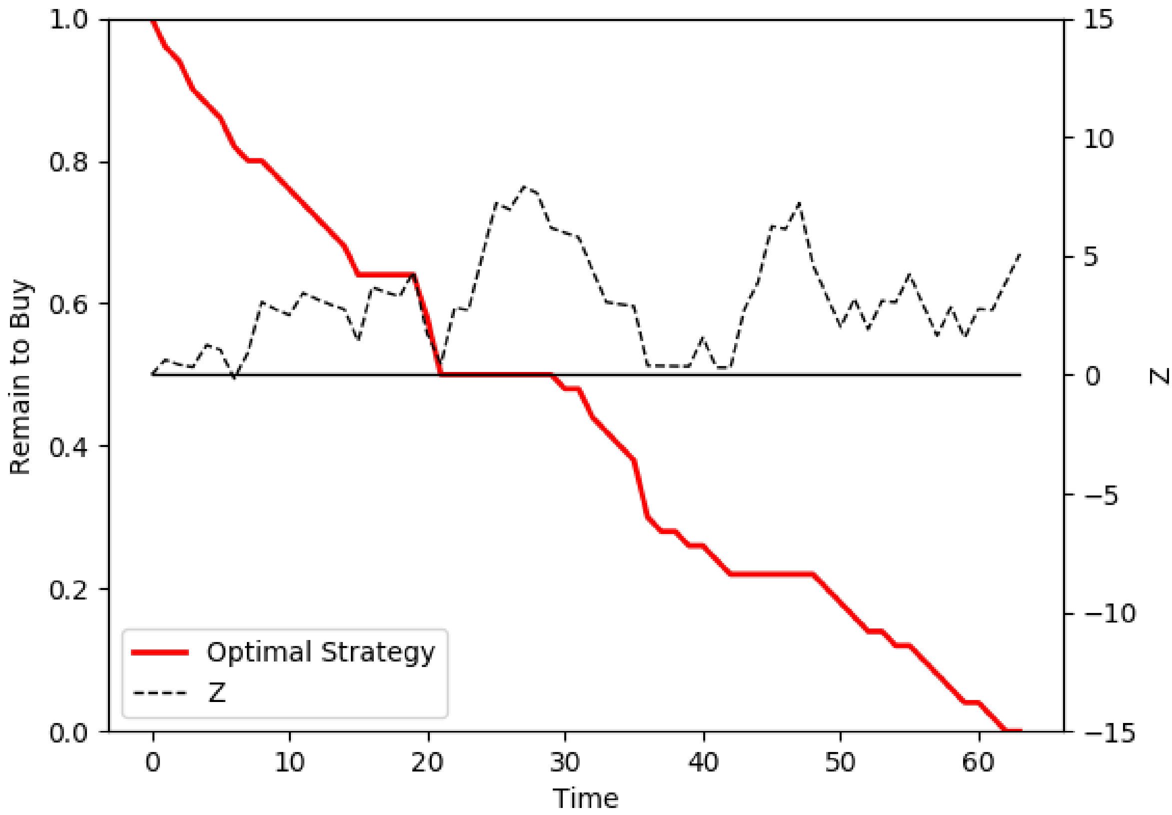
**Sample Output**

The program outputs in the standard output the value of the indifference price of the ASR contract as shown in the Figure below.

```
------------ OUTPUT ----------------
Price of the ASR contract: PI/Q = -0.5008116089062428
```

The program builds two graphs as shown below.






The program output to a txt file '*data_qgrid_{NQ}_gamma_{gamma}.txt*' is as shown below:

| Time | Price | Bought | Remains | Average | Z |
|---|---|---|---|---|---|
| 0 | 45 | 0 | 1 | 45.0000 | 0.0000 |
| 1 | 45.75 | 2000000 | 0.9 | 45.3750 | 0.6250 |
| 2 | 45.75 | 0 | 0.9 | 45.5000 | 0.4167 |
| 3 | 45.75 | 0 | 0.9 | 45.5625 | 0.3125 |
| 4 | 46.5 | 0 | 0.9 | 45.7500 | 1.2500 |
| 5 | 46.5 | 2000000 | 0.8 | 45.8750 | 1.0417 |
| 6 | 45.75 | 0 | 0.8 | 45.8571 | -0.1786 |
| 7 | 46.5 | 0 | 0.8 | 45.9375 | 0.9375 |
| 8 | 48.0 | 0 | 0.8 | 46.1667 | 3.0556 |
| 9 | 48.0 | 0 | 0.8 | 46.3500 | 2.7500 |
| 10 | 48.0 | 2000000 | 0.7 | 46.5000 | 2.5000 |
| 11 | 48.75 | 0 | 0.7 | 46.6875 | 3.4375 |
| 12 | 48.75 | 0 | 0.7 | 46.8462 | 3.1731 |
| 13 | 48.75 | 0 | 0.7 | 46.9821 | 2.9464 |
| 14 | 48.75 | 0 | 0.7 | 47.1000 | 2.7500 |
| 15 | 48.0 | 2000000 | 0.6 | 47.1562 | 1.4062 |