Backward Stochastic Differential Equation

Derivative Method : Analysis

Majdi Rabia

1 Introduction

Simulations on American Options Pricing in high dimensions work well as they do not involve a noisy process in regression step. This is the case in our bid-ask problem, following the BSDE:

$$-dY_t = f(t, Y_t, Z_t)dt - Z_t dW_t \tag{1}$$

where

$$\begin{cases} f(t, Y_t, Z_t) = -\theta Z_t - rY + (R - r)(Y - \frac{Z_t}{\sigma})^- \\ Z_t = \sigma \sum_{i=1}^{p} \Phi_t^i \\ \theta = \frac{\mu - r}{\sigma} \end{cases}$$

Let's compare our results to Gobet's simulation with the following parameter :

T	S_0	K	r	R	μ	σ
0.5	100	100	0.04	0.06	0.06	0.2

2 Regression

$$Z_{t_i} = \frac{1}{\Delta t_i}\mathbb{E}[Y_{t_{i+1}}\Delta B_{t_i}|\mathcal{F}_{t_i}]$$

$$Y_{t_i} = \mathbb{E}[Y_{t_{i+1}}|\mathcal{F}_{t_i}] + f(t_i, S_{t_i}, Y_{t_{i+1}}, Z_{t_i})\Delta t_i$$

3 Derivative Iteration

This method uses the fact $Z_t = \sigma S_t \frac{\partial w}{\partial S}$ where $Y_t = w(t, S_t)$

- $\forall t_i$ we compute once the regression using for example a RandomForest (in high dimensions), or a least square estimator on polynomial basis (one dimension).
- Assumption : $Y_{t_i} = \hat{f}(S_{t_i})$

  Gaussian approximation : $\hat{f}(x) = \sum_{i=1}^{N} \frac{\omega(x,x_i)}{\sum_{j=1}^{N}\omega(x,x_j)} Y_i$ where i denotes the i'th simulation

  where the weight $\omega(x, x_i) = e^{-\frac{(x-x_i)^2}{2l^2}}$

  l being a parameter depending on the scale of the given data (the smaller, the better)
- Once $\hat{f}$ is computed, as it was chosen derivable on purpose, we compute the derivative $\hat{f}'$
- We set $Z_{t_i} = \sigma S_{t_i}\hat{f}'(S_{t_i})$

We iterate this method at every step, to get the smoother Z possible.

4 Simulations

4.1 1st : Price convergence

N_particles	m_discretization	N_run	n_picard	nearest_neighbors	l
1000	12	20	3	100	1.0

Theoretical Price : 7.15

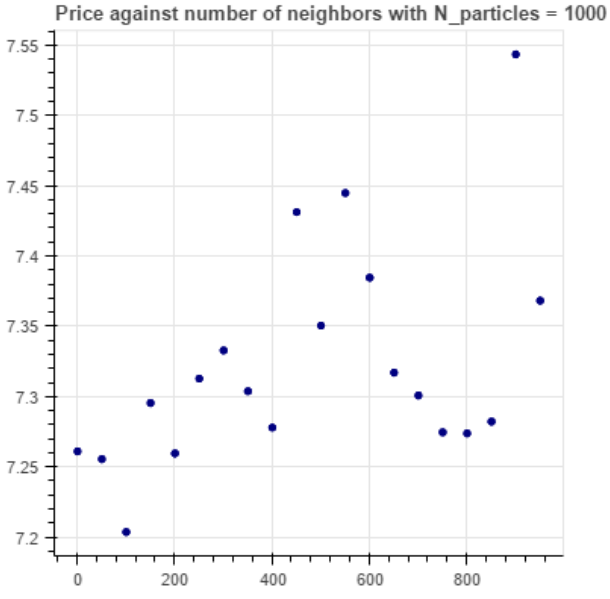
Table 1: Simulation

Run Number	Time in seconds	Price
run 1	3.591	7.428653384
run 2	4.751	7.042321827
run 3	3.932	7.257010564
run 4	3.618	6.917118799
run 5	3.223	7.171610637
run 6	4.307	7.090800489
run 7	3.75	7.206036474
run 8	3.97	6.952620543
run 9	3.494	7.648404958
run 10	4.907	6.741787516
run 11	4.204	7.696553542
run 12	4.136	6.818502178
run 13	4.403	7.775232407
run 14	4.081	6.780866829
run 15	4.013	7.180213783
run 16	3.572	7.485186614
run 17	4.204	7.258745764
run 18	4.473	7.270984402
run 19	3.673	6.800149302
run 20	3.942	7.035487849

mean	std	95% confidence interval	min	max
7.1779	0.3088	[7.1476, 7.2082]	6.7418	7.7752

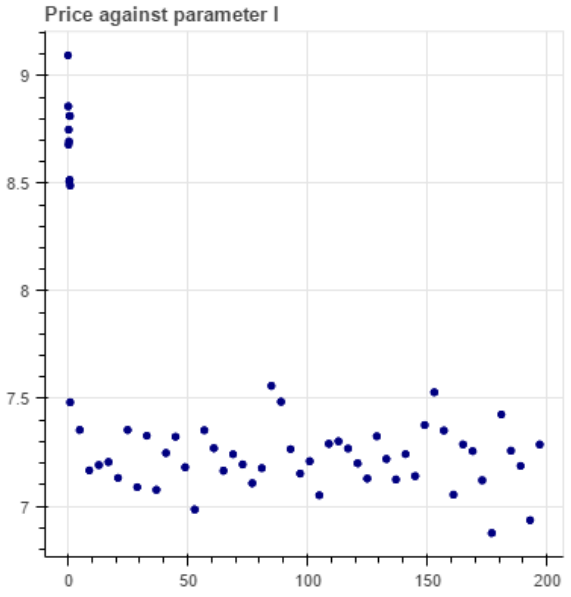
Remark Even with a smoother Z, we can notice a high variance. Of course, the number of 1000 particles being limited, it might be a cause. I am working on getting the same result with a bigger N_particles

4.1.1 Analysis of the number of neighbors on the Price



Remarks Given the variance of one run with n_neighbors, this simulation has been made with several runs (5) for each point ... The results are not the one we would have expected, as a number of neighbours increasing seem to be less precise. My assumption is that given the small difference between r and R $(0.06 - 0.04)$, Z_t might not be so important in this computation.

4.2 Analysis of the parameter l

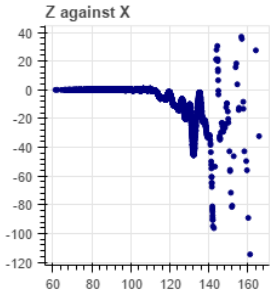


Remarks As before, given the variance of one run , this simulation has been made wwith 5 runs for each point. A value near 0 for l parameter gives for every point i (at each step t) a very small weight. That explains the values we have around 0.

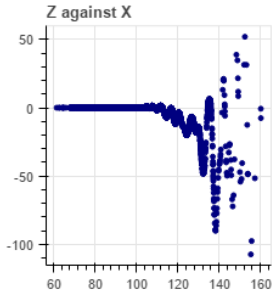
We should observe for big values of l a bigger error (as for l near 0), but we do not. Indeed, when l is bigger than $(x - x_i)$, the weight ω should be almost equal to 1! Giving then the same weight for each point Y_i and we would have a simple Monte-Carlo.

My assumption is that we get $\mathbb{E}[Y_{t_i}] \forall t_i$ instead of Y_{t_i}, which does not seem to be much different, and would explain why bigger l do not give worse values.

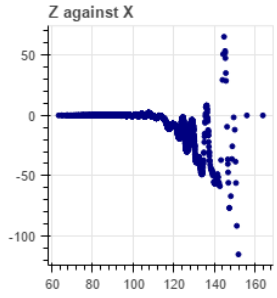
4.3 Plotting the noisy Process Z against the Stock Price X



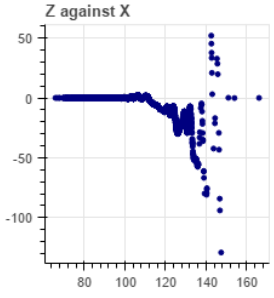
(a) Figure A



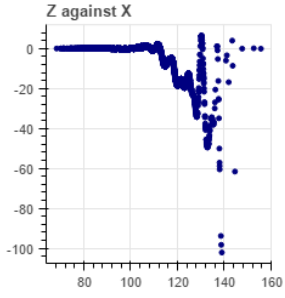
(b) Figure B



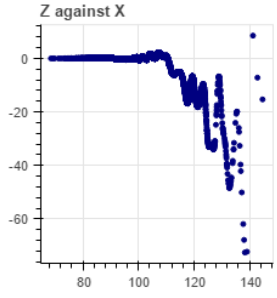
(c) Figure C



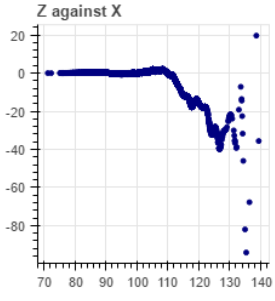
(d) Figure D



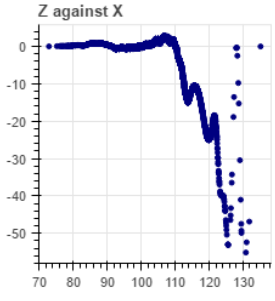
(e) Figure E



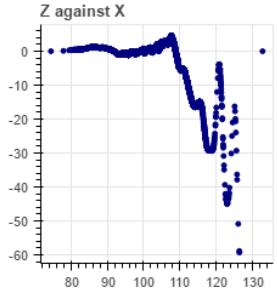
(f) Figure F



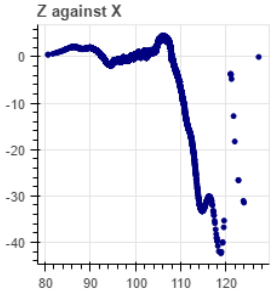
(g) Figure G



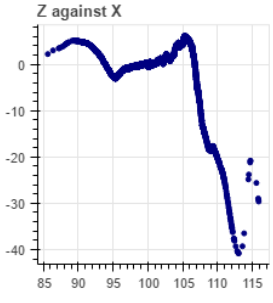
(h) Figure H



(i) Figure I



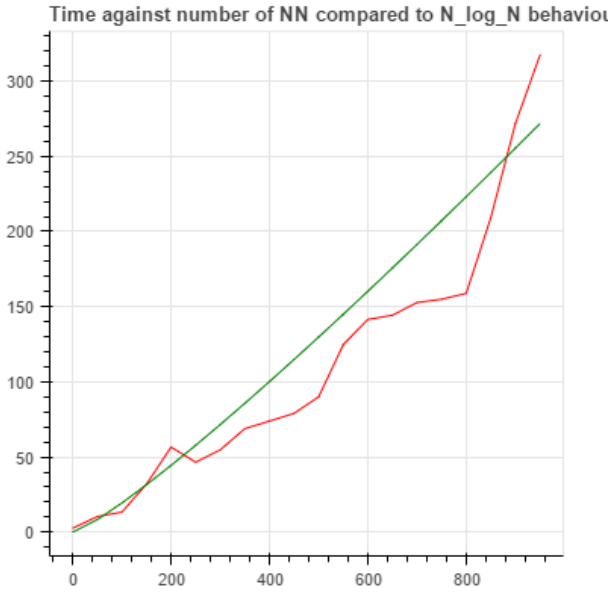
(j) Figure J



(k) Figure K

Figure 1: Z against X for a 12 steps discretization

4.4 Time analysis



Remark I plotted here an $Nlog(N)$ behaviour against a number of neighbours increasing, along with the time taken for each number of NN. We can notice how the two functions seem to behave samely here .