JPMC Quant Challenge’22

Derivative Modelling

Vignesh Kumar S

B.Tech in Chemical Engineering, M.Tech in Data Science (Dual Degree)

IIT Madras

Overview:

1. OLS Estimates
   a. Euler Discretization
   b. Parameter Estimation
   c. Error normality test
2. ML Estimates
   a. Likelihood function
   b. Optimization

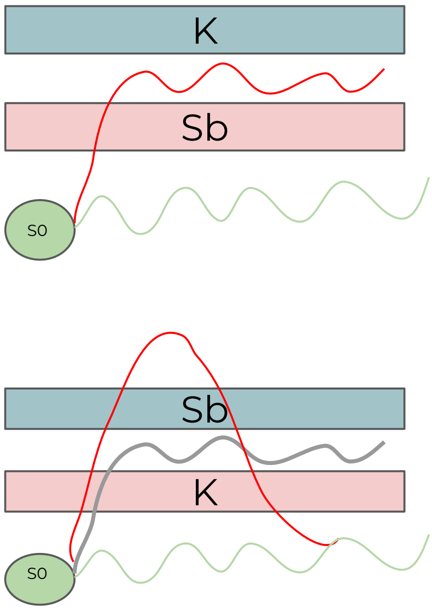
Calibrating Data

1. Discretization
   a. Euler
   b. Milstein
   c. Predictor Corrector
2. Option Pricing using Monte Carlo Simulation
   a. Up and out put option

Option Pricing

Problem Statement:

Up and Out Put Barrier Option:



CIR Model:

- $dX_t = a(b-X_t) + \sigma(\sqrt{X_t})dW_t$
  - First term: Drift
  - b is long run eq. Value
  - a is strength of reversion
  - Second term: Diffusion
  - Sqrt prevents X from becoming negative (2ab>sig2)
  - $dW_t \sim N(0,dt)$
- Use this to calculate expected payoff discounted back to present value to price option

Overview:

1. Ordinary Least Squares Estimates
    a. Euler Discretization
    b. Parameter Estimation
    c. Error normality test
2. ML Estimates
    a. Likelihood function
    b. Optimization

Calibrating Data

1. Discretization
    a. Euler
    b. Milstein
    c. Predictor Corrector
2. Option Pricing using Monte Carlo Simulation
    a. Up and out put option

Option Pricing

Calibrating Data: (Discretization and OLS)

- Continuous: $dX_t = a(b-X_t) + \sigma(\sqrt{X_t})dW_t$
- Discrete (Euler): $X_{t_{i+1}} - X_{t_i} = a(b - X_{t_i}) + \sigma\sqrt{|X_{t_i}|\Delta T}\,\varepsilon$ [Errors are **Heteroskedastic**]
- $\dfrac{X_{t_{i+1}} - X_{t_i}}{\sqrt{X_{t_i}}} = \dfrac{ab\Delta t}{\sqrt{X_{t_i}}} - a\sqrt{X_{t_i}}\Delta t + \sigma\sqrt{\Delta T}\,\varepsilon$ [Errors are **Homoskedastic**]
- $Y = Z\beta + \epsilon$, where
- $Y = \begin{bmatrix} \frac{X_{t_2}-X_{t_1}}{\sqrt{X_{t_1}}} \\ \frac{X_{t_3}-X_{t_2}}{\sqrt{X_{t_2}}} \\ .. \end{bmatrix};\ Z = \begin{bmatrix} \frac{\Delta T}{\sqrt{X_1}} & -\sqrt{X_1}\Delta T \\ \frac{\Delta T}{\sqrt{X_2}} & -\sqrt{X_2}\Delta T \\ .. & .. \end{bmatrix};\ \beta = \begin{bmatrix} ab \\ a \end{bmatrix}$
- $\hat{\beta} = \arg\min ||Y - Z\beta||^2$
  $\hat{\beta} = \left(Z^T Z\right)^{-1} Z^T Y$
- $\hat{\sigma^2} = \dfrac{1}{\Delta T(N-2)} \left|\left|Y - \hat{Y}\right|\right|^2$

Results from OLS:

Params	LS estimates
a	0.909846889
b	135.0939703
sigma	0.399912442

Parameter estimates obtained by averaging across all runs through OLS

- Most of the runs pass AD test or Shapiro test for normality on error terms
- Mean of errors is on order of 1e-6 indicating strongly that the error term population mean is likely to be **zero**
- P value indicating that the model coefficients are **statistically significant**
- 95 percent confidence interval for *b* (for each run) has a width of around 20,and a width of 0.25 for *a*
- Covariance for parameters (in each run) is also obtained $\hat{\sigma}^2 (X^T X)^{-1}$

- Additional tests are in appendix of the python notebook

Overview:

1. OLS Estimates
    a. Euler Discretization
    b. Parameter Estimation
    c. Error normality test
2. ML Estimates
    a. Likelihood function
    b. Optimization

Calibrating Data

1. Discretization
    a. Euler
    b. Milstein
    c. Predictor Corrector
2. Option Pricing using Monte Carlo Simulation
    a. Up and out put option

Option Pricing

Calibrating Data: ML estimates

$$L(\theta\,|\,y) = f(\,y_1, y_2, .\,.\,, y_N\,|\,\theta\,)$$
$$= p(x_{t_1})\,\Pi\, p(\,x_{t_{i+1}}\,|\,x_{t_i}, \theta)$$

$$\ln L(\theta\,|\,y) = \ln p(x_{t_1}) \;+\; \Sigma\,\ln p(x_{t_{i+1}}\,|\,x_{t_i}\,,\,\theta)$$

$$p(x_t\,|\,x_s) = c\,\exp\left(-(u+v)\right)\left(\frac{v}{u}\right)^{\frac{q}{2}} I_q(2\sqrt{uv})$$

$where$

$$c = \frac{2a}{\sigma^2(1-\exp\left(-a\Delta t\right))};$$
$$u = c\,x_s\,\exp\left(-a\Delta t\right);$$
$$v = c\,x_{t;}$$
$$q = \frac{2ab}{\sigma^2} - 1;$$
$$\Delta t = t - s$$

Objective: $\arg\max\;\ln L(\theta)$

$$\ln L = (N-1)\ln c + \sum_{i=1}^{N-1}\left\{-u_{t_i} - v_{t_{i+1}} + 0.5q\ln\left(\frac{v_{t_{i+1}}}{u_{t_i}}\right) + \ln\{I_q^1(2\sqrt{u_{t_i}v_{t_{i+1}}})\} + 2\sqrt{u_{t_i}v_{t_{i+1}}}\right\}$$

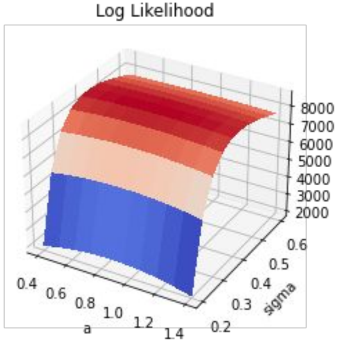
Implementation:

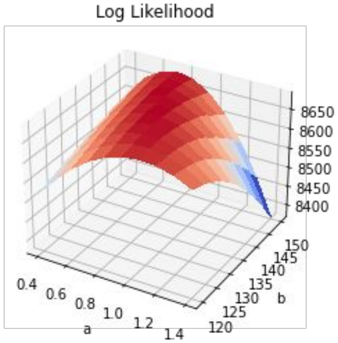
- Solved by minimising negative loglikelihood using scipy optimise minimisation algorithm Nelder-Mead
- Used **OLS** estimates on discretized version as **initial starting points**
- Modified Bessel function of first kind, I_q tends quickly to infinity leading to numerical instability. Instead exponentially scaled bessel function, I_q^1, is used to solve this problem. The objective function is adjusted accordingly.

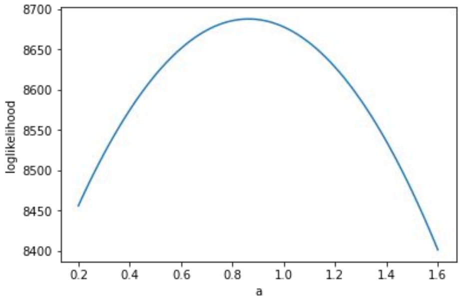
Results from ML estimates

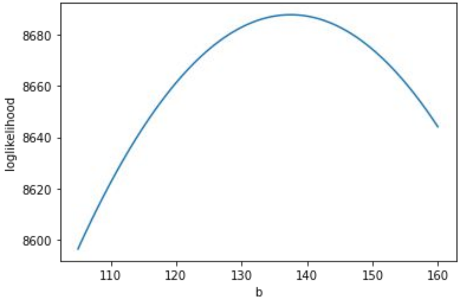
Params	LS estimates	ML estimates
a	0.9098468	0.91001019
b	135.093970	135.09418157
sigma	0.39991244	0.399907344

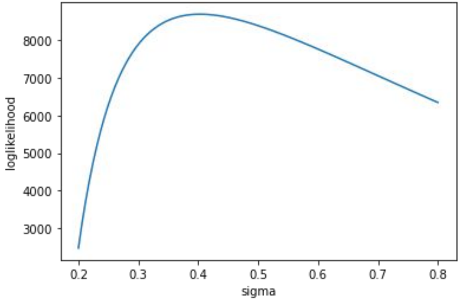
- Estimates are obtained with **Nelder Mead** method
- By fixing one/two parameters, and varying the rest, objective function seems concave near our search region











Overview:

1. OLS Estimates
    a. Euler Discretization
    b. Parameter Estimation
    c. Error normality test
2. ML Estimates
    a. Likelihood function
    b. Optimization

Calibrating Data

1. Discretization
    a. Euler
    b. Milstein
    c. Predictor Corrector
2. Option Pricing using Monte Carlo Simulation
    a. Up and out put option

Option Pricing

Option Pricing

- From calibration:
  a, b, sigma = 0.91001, 135.09418, 0.39990

- From question/data:
  S_0, tau, r = 20, 4, 0.1

- Monitoring frequency = 1/12

- Discretization Method: Predictor-Corrector Method

$$S^*_{t+1} = S_t + a(b - S_t)\Delta t + \sigma\sqrt{S_t\Delta t}\,\varepsilon$$
$$S_{t+1} = S^*_{t+1} + \frac{1}{2}\big(a\big(b - S^*_{t+1}\big) - a(b - S_t)\big)\Delta t$$

- Realizations = 100,000

Sb	K	UO Put	UI Put	Put
140	145	8.606491	0.064784	8.671275
140	150	11.892999	0.129109	12.022108
137	137	3.325484	0.065847	3.391331
139	139	4.654781	0.022847	4.677628
150	200	45.547866	0.0	45.547866
145	145	8.673678	0.000178	8.673855
132	132	0.699778	0.172378	0.872155

Option Pricing

- From calibration:
  a, b, sigma = 0.91001, 135.09418, 0.39990

- From question/data:
  S_0, tau, r = 20, 4, 0.1

- Monitoring frequency = 1/12

- Discretization Method: Predictor-Corrector Method

$$S^*_{t+1} = S_t + a(b - S_t)\Delta t + \sigma\sqrt{S_t \Delta t}\,\varepsilon$$

$$S_{t+1} = S^*_{t+1} + \frac{1}{2}\left(a\left(b - S^*_{t+1}\right) - a(b - S_t)\right)\Delta t$$

- Realizations = 100,000

Sb	K	UO Put	UI Put	Put
140	145	8.606491	0.064784	8.671275
140	150	11.892999	0.129109	12.022108
137	137	3.325484	0.065847	3.391331
139	139	4.654781	0.022847	4.677628
150	200	45.547866	0.0	45.547866
145	145	8.673678	0.000178	8.673855
132	132	0.699778	0.172378	0.872155

Option Pricing

- From calibration:
  a, b, sigma = 0.91001, 135.09418, 0.39990

- From question/data:
  S_0, tau, r = 20, 4, 0.1

- Monitoring frequency = 1/12

- Discretization Method: Predictor-Corrector Method

$$S^*_{t+1} = S_t + a(b - S_t)\Delta t + \sigma\sqrt{S_t \Delta t}\,\varepsilon$$
$$S_{t+1} = S^*_{t+1} + \frac{1}{2}\left(a\left(b - S^*_{t+1}\right) - a(b - S_t)\right)\Delta t$$

- Realizations = 100,000

Sb	K	UO Put	UI Put	Put
140	145	8.606491	0.064784	8.671275
140	150	11.892999	0.129109	12.022108
137	137	3.325484	0.065847	3.391331
139	139	4.654781	0.022847	4.677628
150	200	45.547866	0.0	45.547866
145	145	8.673678	0.000178	8.673855
132	132	0.699778	0.172378	0.872155

UO Put + UI Put = Put

Overview:

1. OLS Estimates
    a. Euler Discretization
    b. Parameter Estimation
    c. Error normality test
2. ML Estimates
    a. Likelihood function
    b. Optimization

Calibrating Data

1. Discretization
    a. Euler
    b. Milstein
    c. Predictor Corrector
2. Option Pricing using Monte Carlo Simulation
    a. Up and out put option

Option Pricing

Discretization Methods

SDE: $dX_t = a(X,t)dt + b(X,t)dW_t$

- Euler: $X_{i+1} = X_i + a(X_i,t)\Delta t + b(X_i,t_i)\Delta w_i$
- Milstein: $X_{i+1} = X_i + a(X_i,t)\Delta t + b(X_i,t_i)\Delta w_i + \frac{1}{2}b(X_i,t_i)\frac{\partial b(X_i,t_i)}{\partial x}(\Delta w_i^2 - \Delta t_i)$
- Predictor Corrector: $X^*_{t_{i+1}} = X_{t_i} + a(t_i,X_{t_i})\Delta t + b(t_i,X_{t_i})\Delta w_{t_i}$

$$X_{t_{i+1}} = X_{t_i} + \frac{1}{2}\left(a(t_i,X_{t_i}) + a\left(t_{i+1},X^*_{t_{i+1}}\right)\right)\Delta t + b(t_i,X_{t_i})\Delta w_{t_i}$$

Discretization Methods

SDE: $dS_t = a(b-S_t) + \sigma(\sqrt{S_t})dW_t$

- Euler: $S_{t+1} = S_t + a(b-S_t)\Delta t + \sigma\sqrt{S_t \Delta t}\,\varepsilon$
- Milstein: $S_{t+1} = S_t + a(b-S_t)\Delta t + \sigma\sqrt{S_t \Delta t}\,\varepsilon + \frac{1}{4}\sigma^2\Delta t(\varepsilon^2 - 1)$
- Predictor Corrector: $S^*_{t+1} = S_t + a(b-S_t)\Delta t + \sigma\sqrt{S_t \Delta t}\,\varepsilon$
  $S_{t+1} = S^*_{t+1} + \frac{1}{2}\left(a\left(b - S^*_{t+1}\right) - a(b-S_t)\right)\Delta t$

Discretization Methods

SDE: $dX_t = a(b-X_t) + \sigma(\sqrt{X_t})dW_t$

- Euler: $S_{t+1} = S_t + a(b - S_t)\Delta t + \sigma\sqrt{S_t \Delta t}\,\varepsilon$
- Milstein: $S_{t+1} = S_t + a(b - S_t)\Delta t + \sigma\sqrt{S_t \Delta t}\,\varepsilon + \frac{1}{4}\sigma^2 \Delta t(\varepsilon^2 - 1)$
- Predictor Corrector: $S^*_{t+1} = S_t + a(b - S_t)\Delta t + \sigma\sqrt{S_t \Delta t}\,\varepsilon$
  $S_{t+1} = S^*_{t+1} + \frac{1}{2}\left(a\left(b - S^*_{t+1}\right) - a(b - S_t)\right)\Delta t$

$$E[\,x_t \mid x_0] = x_0 \exp(-at) + b(1 - \exp(-at))$$

Euler	Milstein	Predictor Corrector	Theoretical
133.192083	133.154320	132.805721	132.86035

Thank you

Comparison between different solvers

Optimization Methods	a	b	c	Mean loglikelihood
Nelder Mead	0.910010	135.094181	0.399907	12080.1074
BFGS	0.909854	135.093968	0.399909	12080.2126
L-BFGS-B	0.906971	135.069208	0.342597	13096.1080

L-BFGS-B was the fastest, followed by Nelder Mean and BFGS

*additional/helper slides