EA MAP 511
High Order Discretization Schemes For The CIR Process

Siguang QI, Jiale NING

December 6, 2021

1 Introduction

The Cox-Ingersoll-Ross (CIR for short) process is defined by the following stocastic integral equation

$$\begin{cases} X_t = x + \int_0^t (a - kX_s)ds + \sigma \int_0^t \sqrt{X_s} dW_s \\ X_0 = x > 0 \end{cases} \tag{1}$$

or in a differential way

$$dX_t = (a - kX_t)dt + \sigma\sqrt{X_t}dW_t$$

with $(a, k, \sigma) \in \mathbb{R}_+^* \times \mathbb{R} \times \mathbb{R}_+$ and W a Brownian motion.

With the parameters satisfying the above condition, CIR process is always non-negative, and thus the square roots in the above equations are well-defined. However, when doing simulations, if we simply use the Euler scheme, since the Gauss distribution is not bounded, we might get a negative value during the simulation, which will certainly be a problem. What's more, in order to accelerate the convergence of the Monte Carlo simulation, we need more effective discretization schemes.

2 Theoretical Notions and Methodologies

2.1 Associated Differential Operator

In order to apply the arguments to the more general solution, we will use in the following part several notions. For a $\mathbb{R}^d$-valued SDE

$$X_t^x = x + \int_0^t b(X_s^x)ds + \int_0^t \sigma(X_s^x)dW_s,$$

with W_t a d_W-dimension Brownian motion, b valued in $\mathbb{R}^d$ and σ valued in $\mathcal{M}_{d\times d_W}(\mathbb{R})$ satisfying certain restriction on the growing speed of its derivatives, define the differential operator associated to it L as (f valued in $\mathbb{R}$)

$$Lf(x) = \sum_{i=1} b_i(x)\partial_i f(x) + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d_W}\sigma_{i,k}(x)\sigma_{j,k}(x)\partial_i\partial_j f(x)$$

which implies by Ito's formula $\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}[f(X_t^x)] = \mathbb{E}[Lf(X_t^x)]$.

2.2 General Discretization Scheme

Define $t_i^n = T\frac{i}{n}$, $\Delta T = T/n$ where n is the number of the sub-interval in the scheme and $X_{t_i^n}^n$ the simulated $X_{t_i^n}$. Since the solution of the equation satisfies the strong Markov property, the value and the dynamics at time t_{i+1}^n depend only on the value at t_i^n. As a result, we can represent any scheme of discretization in the form of a family of transition probability $\left(\hat{p}_x(t)(dz)\right)_{t,x}$: given the value of $\hat{X}_{t_i^n}^n$, we choose the value of $\hat{X}_{t_{i+1}^n}^n$ according to the principle

$$\mathbb{E}[f(\hat{X}_{t_{i+1}^n}^n)|\hat{X}_{t_i^n}^n] = \int f(z)\hat{p}_{\hat{X}_{t_i^n}^n}(\Delta T)(dz),$$

which means the distribution of $\hat{X}_{t_{i+1}^n}^n$ is $\hat{p}_{\hat{X}_{t_i^n}^n}(\Delta T)$. Thus, this family of transition probability can be also represented by a group of random variable $\left(\hat{X}_t^x\right)_{t,x}$ of distribution $\left(\hat{p}_x(t)(dz)\right)_{t,x}$.

2.3 Order of a scheme

For any scheme $\left(\hat{X}_t^x\right)_{t,x}$, we know that it's a weak ν-order scheme if using this scheme we obtain $\left(\hat{X}_{t_i^n}^n\right)_{0\leq i\leq n}$ and $|\mathbb{E}[f(X_T^x)] - \mathbb{E}[f(\hat{X}_{t_n^n}^n)]| \leq \frac{K}{n^\nu}$. In order to analysis more directly the performance of a scheme, a equivalent definition of a weak ν-order scheme, under certain regularity constraints, is for all t small enough and x

$$\left|\mathbb{E}[f(\hat{X}_t^x)] - \left[f(x) + \sum_{k=1}^{\nu} \frac{1}{k!} t^k L^k f(x)\right]\right| \leq Ct^{\nu+1}(1 + \|x\|^E).$$

This is because by the definition of L we know

$$\begin{aligned}\mathbb{E}[f(X_t^x)] = f(x) + \int_0^t \mathbb{E}[Lf(X_{t_1}^x)]\mathrm{d}t_1 = f(x) + \int_0^t Lf(x) + (\int_0^{t_1} \mathbb{E}[Lf(X_{t_2}^x)]\mathrm{d}t_2)\mathrm{d}t_1 \\ = f(x) + \sum_{k=1}^{\nu} \frac{1}{k!} t^k L^k f(x) + \int_0^t \int_0^{t_1} \cdots \int_0^{t_\nu} \mathbb{E}[L^{\nu+1} f(X_{t_{\nu+1}}^x)]\mathrm{d}t_{\nu+1} \cdots \mathrm{d}t_2 \mathrm{d}t_1\end{aligned}$$

and after taking the regularity constraints into account, the given condition implies $|\mathbb{E}[f(X_t^x)] - \mathbb{E}[f(\hat{X}_t^x)]| = O(t^{\nu+1})$ and thus $|\mathbb{E}[f(X_T^x)] - \mathbb{E}[f(\hat{X}_{t_n^n}^n)]| = O(n^{-\nu})$.

2.4 Combination of operators and Composition of schemes

The composition of two transition probabilities $\hat{p}_x^1(t)(dz)$ and $\hat{p}_x^2(t)(dz)$ is defined as

$$\hat{p}^2(t_2) \circ \hat{p}_x^1(t_1)(dz) := \hat{p}^2_{\hat{p}_x^1(t_1)}(t_2)(dz) = \int \hat{p}_y^2(t_2)(dz)\hat{p}_x^1(t_1)(dy).$$

and in sense of random variables: $\hat{X}_{t_2,t_1}^{2\circ 1,x} := \hat{X}_{t_2}^{2,\hat{X}_{t_1}^{1,x}}$. For simplicity, given two operators L_1 and L_2 as well as two second order schemes of these operator $\hat{p}_x^1(t)$ and $\hat{p}_x^2(t)$, the scheme $\hat{p}^1(\frac{t}{2}) \circ \hat{p}^2(t) \circ \hat{p}_x^1(\frac{t}{2})$ is a second order scheme for $L_1 + L_2$ since formally

$$(I + \frac{t}{2}L_1 + \frac{t^2}{8}L_1^2 + O(t^3))(I + tL_2 + \frac{t^2}{2}L_2^2 + O(t^3))(I + \frac{t}{2}L_1 + \frac{t^2}{8}L_1^2 + O(t^3)) = I + t(L1 + L2) + \frac{t^2}{2}(L_1 + L_2)^2 + O(t^3)$$

and

$$\mathbb{E}[f(\hat{X}_t^x)] = \mathbb{E}\left[\mathbb{E}\left[\mathbb{E}\left[f\left(\hat{X}_{t/2}^{1,\hat{X}_t^{2,\hat{X}_{t/2}^{1,x}}}\right)|\hat{X}_t^{2,\hat{X}_{t/2}^{1,x}}\right]|\hat{X}_{t/2}^{1,x}\right]\right],$$

we therefore see the order of this scheme is 2 by using to the equivalent definition in the former part after some calculations using the above formal equality (noticing that each term on the left corresponds to $\mathbb{E}[f(\hat{X}_t^{i,x})]$ for $\hat{p}_x^i(t)$). And by similar arguments we can produce many high order schemes for more complicated operators by composing several schemes of the same order for some less complicated ones.

3 The Euler scheme for the CIR process

Firstly, we have implemented the Euler discretization method to simulate the CIR process. That is :

$$\begin{cases} \hat{X}_{t_{i+1}^n}^n = \hat{X}_{t_i^n}^n + (a - k\hat{X}_{t_i^n}^n)\Delta t + \sigma\sqrt{(\hat{X}_{t_i^n}^n)_+}(W_{t_{i+1}^n} - W_{t_i^n}), t \in [0,T], \\ \hat{X}_0^n = x \end{cases} \tag{2}$$

with parameters $(a, k, \sigma) \in \mathbb{R}_+^* \times \mathbb{R} \times \mathbb{R}_+$.

We have fixed T=1 and using different values of parameters, we obtained the following results.

The figure on the right illustrates $\mathbb{E}(\exp(-\hat{X}_{t_n^n}^n))$ as a function of 1/n where n is the number of steps. The blue horizontal line is the theoretical value of this term. The figure on the left illustrates the logarithm of the error as a function of $log(n)$ where error is the difference between $\mathbb{E}(exp(-\hat{X}_{t_n^n}^n))$ and its exact value. The yellow line and the green line are respectively upper bound and lower bound of the 0.95 confidence interval. Through this figure, we can estimate then the order of the scheme.

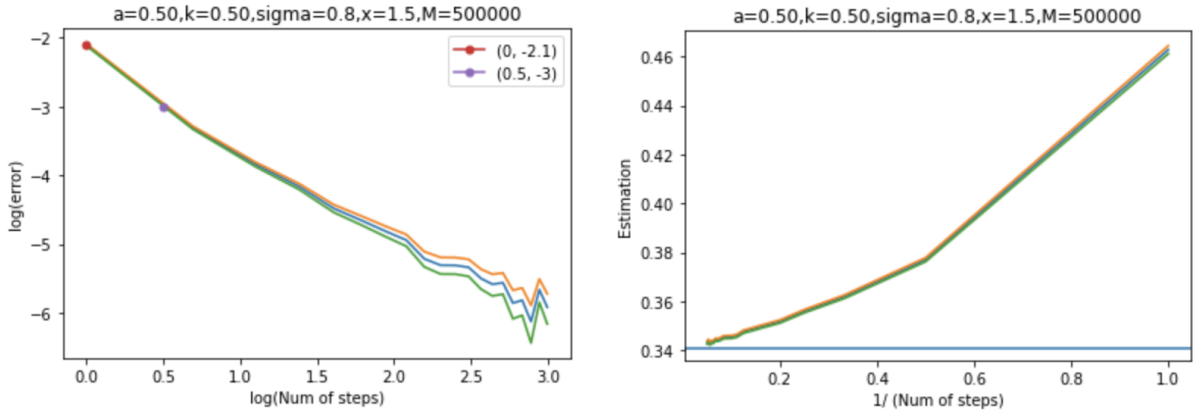

Figure 1: Euler scheme: $\sigma^2 < 4a$

We see that when $\sigma^2 < 4a$, the Euler scheme is close to a second order scheme. Then we look at the case where $\sigma^2 > 4a$. In this case, the Euler scheme is generally not well defined.

When we take a=0.04 and σ=2, through the usual discretization we still have a first order scheme to simulate the process which is however degraded.

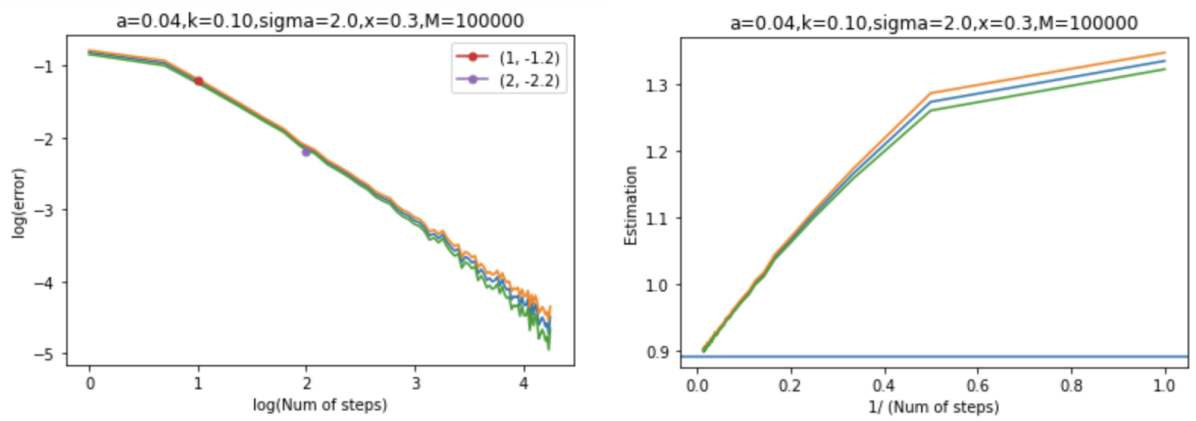

Figure 2: Euler scheme: $\sigma^2 > 4a$

But when we take σ=5, the Euler scheme does not work any more, as showed below. Therefore, we would need other schemes to support large values of σ.

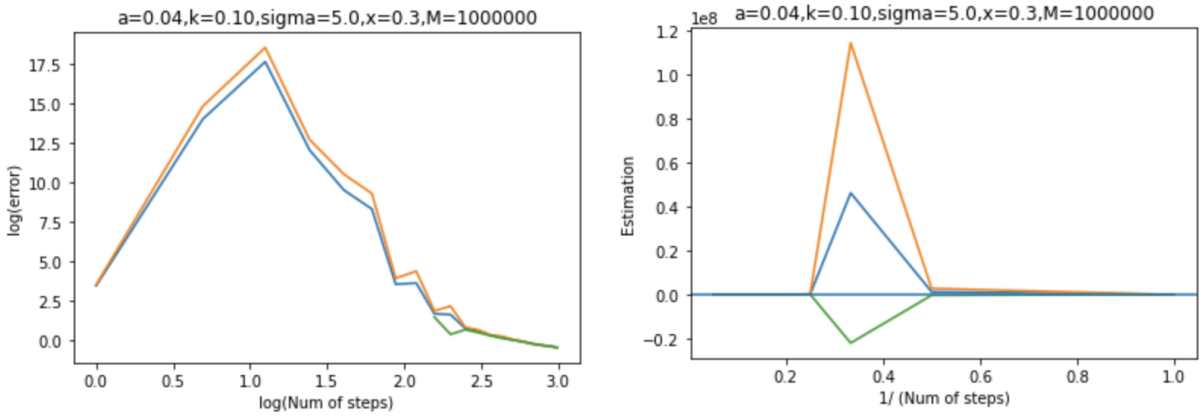

Figure 3: Euler scheme: $\sigma^2 >> 4a$

4 A second order scheme for the CIR process

In this part, we have implemented a second order scheme without restrictions on the CIR parameters (a,k,σ). This scheme consists in the scheme of Ninomiya and Victoir when the value of the process is large enough (depending on σ and a) and switching to different schemes otherwise when the process is in a neighborhood of 0.

With $\psi_k(t) = \frac{1-e^{-kt}}{k}$ and $\psi_0(t) = t$, the scheme of Ninomiya and Victoir yields that if we define

$$X_0^{\text{CIR}}(t,x) = xe^{-kt} + (a - \frac{\sigma^2}{4})\psi_k(t) \quad X_1^{\text{CIR}} = ((\sqrt{x} + \frac{\sigma}{2}t)_+)^2,$$

the composition $X_0^{\text{CIR}}(\frac{t}{2}, X_1^{\text{CIR}}(\sqrt{t}N, X_0^{\text{CIR}}(\frac{t}{2}, x)))$ is a second order scheme and in this case, this scheme can be written as $\hat{X}_t^x = \varphi(x,t,\sqrt{t}N)$ with N a standard normal distribution where φ is written as

$$\varphi(x,t,w) = e^{-kt/2}(\sqrt{(a-\sigma^2/4)\psi_k(t/2) + e^{-kt/2}x} + \sigma w/2)^2 + (a-\sigma^2/4)\psi_k(t/2).$$

However, this scheme is only well-defined when $\sigma^2 \leq 4a$. When $\sigma^2 > 4a$, $\hat{X}_t^x$ might be negative if x is not large enough. In order to solve this problem, it's first useful to observe that the order of the scheme will preserve if we replace N by another random variable Y whose five first moments meet those of standard Gaussian variable, which allows us to use a bounded variable so as to give a alternative solution. Therefore, if we use Y a such random variable defined by $\mathbb{P}(Y = \sqrt{3}) = 1/6, \mathbb{P}(Y = -\sqrt{3}) = 1/6$ and $\mathbb{P}(Y = 0) = 2/3$, simple calculation shows that this scheme still gives positive value as long as

$$x > K_2(t) := 1_{\sigma^2 > 4a} e^{\frac{kt}{2}} \left((\frac{\sigma^2}{4} - a)\psi_k\left(\frac{t}{2}\right) + \left[\sqrt{e^{\frac{kt}{2}}(\frac{\sigma^2}{4} - a)\psi_k\left(\frac{t}{2}\right)} + \frac{\sigma}{2}\sqrt{3t} \right]^2 \right).$$

We then use this scheme for x large enough.

For $x < K_2(t)$, in order to keep the stimulated process positive, we use the scheme $\hat{X}_t^x = \mathbf{1}_{\{U \leq \pi(t,x)\}} \frac{\tilde{u}_1(t,x)}{2\pi(t,x)} + \mathbf{1}_{\{U > \pi(t,x)\}} \frac{\tilde{u}_1(t,x)}{2(1-\pi(t,x))}$ where $U \sim \mathcal{U}([0,1])$, $\tilde{u}_q(t,x) = E((X_t^x)^q)$ for $q \in \mathbb{N}$ and $\pi(t,x) = \frac{1-\sqrt{1-\tilde{u}_1(t,x)^2/\tilde{u}_2(t,x)}}{2}$. This scheme is chosen in order to satisfy the non-negativity and meet the first two moment of the process. As a result, it's also a second order scheme.

Since the scheme is of order 2, we know that the weak error is $O(\frac{1}{n^2})$. And since the error of Monte Carlo is always $O(\frac{1}{\sqrt{N}})$ with N the number of simulation, in order to illustrate the real convergence rate, we need to eliminate the error from Monte Carlo and that is to say, $\frac{1}{\sqrt{N}} \ll \frac{1}{n^2}$ or $N \gg n^4$. Here we choose $n = 10$ and we should take $N \gg 1 \times 10^4$, for example, $N = 1 \times 10^6$.

Once the scheme correctly implemented, we obtain the following results. As we can see when the error of Monte Carlo is not too significant with respect to the weak error, the scheme is of order 2.

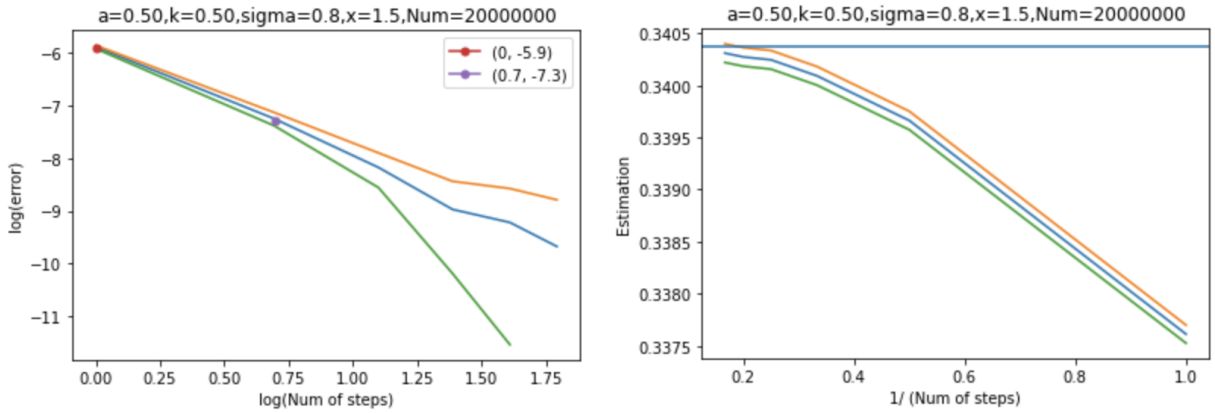

Figure 4: Second order scheme: $\sigma^2 < 4a$

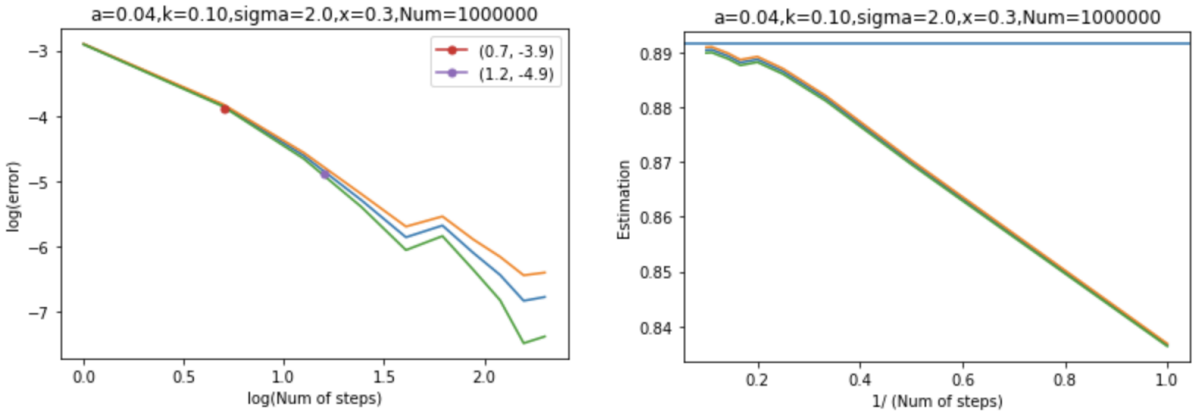

Figure 5: Second order scheme: $\sigma^2 > 4a$

5 A third order scheme for the CIR process

Using the same technique as for the second order scheme, there is a third order scheme for the CIR diffusion.

With the argument in 2.4, we can notice that

$$\frac{1}{6}\Big(\sum_{\epsilon\in\{-1,1\}} \hat{p}^3(\epsilon t)\circ\hat{p}^1(t)\circ\hat{p}^2_x(t) + \hat{p}^1(t)\circ\hat{p}^3(\epsilon t)\circ\hat{p}^2_x(t) + \hat{p}^1(t)\circ\hat{p}^2(t)\circ\hat{p}^3_x(\epsilon t)\Big)$$

corresponds to a third order scheme for L_1+L_2 with $L_3^2 = L_1L_2 - L_2L_1$ and $\hat{p}^i_x(t)$ $(i\in\{1,2,3\})$ the corresponding 3-order transition probability for L_i. Here in order to simulate the CIR process, we consider the case $k=0$ and take $\hat{p}^i_x(t)(dz) \sim X_i^{\mathrm{CIR}}(t,\cdot)$ with $(i\in\{1,2\})$. And $\frac{1}{2}(L_1L_2 - L_2L_1) = \frac{\sigma^2}{2}\left(a-\frac{\sigma^2}{4}\right)\partial_x^2$. Thus with switching the definition of L_1 and L_2 when $4a<\sigma^2$, we may define $\hat{p}^3_x(t)(dz)\sim \tilde{X}(t,x) := x + t\sigma\sqrt{|a-\sigma^2/4|/2}$.

In order to implement this scheme, let ϵ and ζ be respectively independent uniform r.v. on $\{-1,1\}$ and $\{1,2,3\}$, Y a r.v. such that $\mathbb{P}(Y=\sqrt{3+\sqrt{6}}) = \mathbb{P}(Y=-\sqrt{3+\sqrt{6}}) = \frac{\sqrt{6}-2}{4\sqrt{6}}$ and $\mathbb{P}(Y=\sqrt{3-\sqrt{6}}) = \mathbb{P}(Y=-\sqrt{3-\sqrt{6}}) = \frac{1}{2}-\frac{\sqrt{6}-2}{4\sqrt{6}}$ so that the seven first moments of Y fit those of a standard normal distribution. And we can also define $K_3(t)$ so that for $x>K_3(t)$ the non-negativity of the third order scheme is preserved. Thus, when $x>K_3(t)$ and $k=0$, the scheme

$$\hat{X}^x_t := \hat{X}^{x,k=0}_t = \begin{cases} \tilde{X}(\epsilon t, X_0^{CIR}(t, X_1^{CIR}(\sqrt{t}Y,x))) \text{ (resp. } \tilde{X}(\epsilon t, X_1^{CIR}(\sqrt{t}Y, X_0^{CIR}(t,x))) \text{)} & \text{if } \zeta=1, \\ X_0^{CIR}(t,\tilde{X}(\epsilon t, X_1^{CIR}(\sqrt{t}Y,x))) \text{ (resp. } X_1^{CIR}(\sqrt{t}Y,\tilde{X}(\epsilon t, X_0^{CIR}(t,x))) \text{)} & \text{if } \zeta=2, \\ X_0^{CIR}(t, X_1^{CIR}(\sqrt{t}Y,\tilde{X}(\epsilon t,x))) \text{ (resp. } X_1^{CIR}(\sqrt{t}Y, X_0^{CIR}(t,\tilde{X}(\epsilon t,x))) \text{)} & \text{if } \zeta=3 \end{cases} \tag{3}$$

is a potential third order scheme where for $\sigma^2 \le 4a$ (resp. $\sigma^2>4a$). For the general case, by the identity $(X_t, t\ge 0) \sim (e^{-kt}X^{x,k=1}_{\psi_{-k}(t)}, t\ge 0)$, we know when $k\neq 0$, $\hat{X}^x_t := e^{-kt}\hat{X}^{x,k=0}_t$ is a third order scheme.

When $x\in[0,K_3(t)]$, we have a potential third order scheme defined by $\tilde{X}^x_t = \mathbf{1}_{U\le\pi(t,x)}x_+(t,x) + \mathbf{1}_{U>\pi(t,x)}x_-(t,x)$ where $x_\pm = \frac{s\pm\sqrt{s^2-4p}}{2}$ and $\pi = \frac{\tilde{u}_1(t,x)-x_-}{x_+-x_-}$ with $s=\frac{\tilde{u}_3-\tilde{u}_1\tilde{u}_2}{\tilde{u}_2-\tilde{u}_1^2}$ and $p=\frac{\tilde{u}_1\tilde{u}_3-\tilde{u}_2^2}{\tilde{u}_2-\tilde{u}_1^2}$.

As we can see in the pictures below, when the number of steps (n) becomes larger, the scheme tends to be a third order scheme. In fact, to obtain such result by which we can estimate the order of the scheme, especially in the case $\sigma^2<4a$, the number of samples (M) of Monte Carlo estimation has to be very large to limit the MC noise so that we can see the error of the third order scheme.

Limited by the calculation capacity of our laptop, for $\sigma^2<4a$, we have calculated for n=1,2,3,4 and M=3×10^8. We see that in this case n=4 is sufficient to include the exact value in the confidence interval and that it tends to be a third order scheme but the confidence interval is still large. To reduce the interval we can increase M but it becomes too hard for our computer to calculate. Also, for $\sigma^2>4a$, we have just calculated n=1,...,10 and M=10^7. We see that towards the end when n goes up to 10, the scheme is getting closer to a third order scheme.

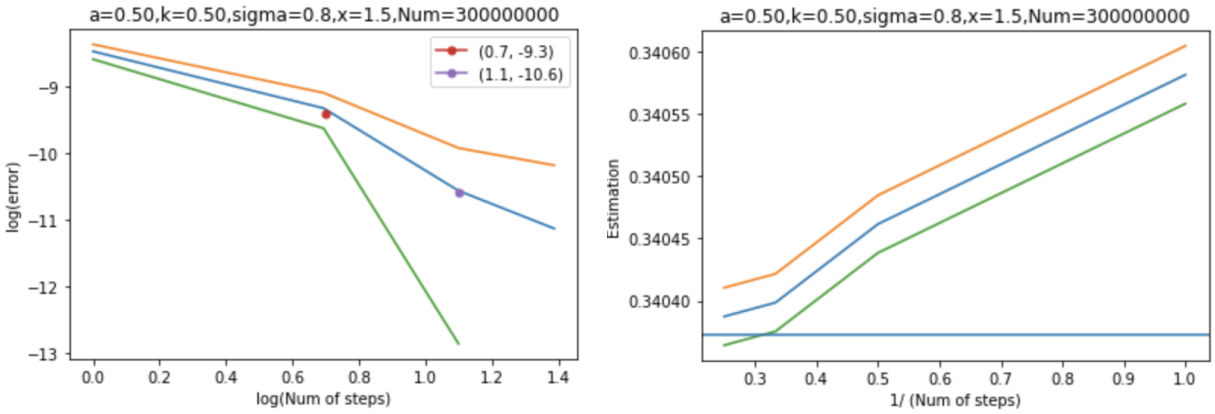

Figure 6: Third order scheme: $\sigma^2 < 4a$

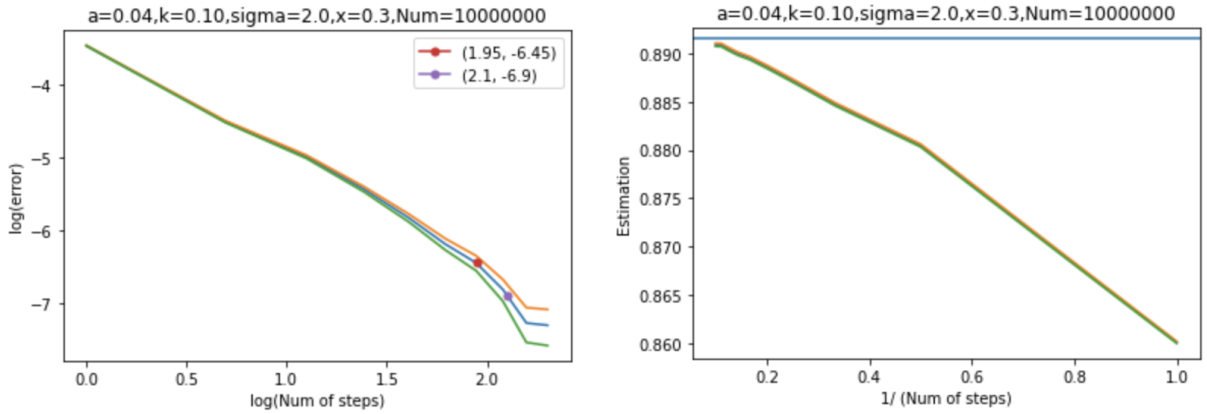

Figure 7: Third order scheme: $\sigma^2 > 4a$

6 Application to the Heston Model

Let W and Z be two independent Brownian motions, we would like to simulate the following SDE which is called Heston Model:

$$\begin{cases} X_t^1 = X_0^1 + \int_0^t (a - kX_s^1)ds + \sigma \int_0^t \sqrt{X_s^1} sW_s \\ X_t^2 = \int_0^t X_s^1 ds \\ X_t^3 = X_0^3 + \int_0^t rX_s^3 ds + \int_0^t \sqrt{X_s^1} X_s^3 (\rho dW_s + \sqrt{1-\rho^2} dZ_s) \\ X_t^4 = \int_0^t X_s^3 ds \end{cases} \tag{4}$$

To approximate this diffusion, like in mentioned in 2, we can separate equation into two part and these two part correspond to the two following transition probability respectively: for x=$(x_1, x_2, x_3, x_4)^{'} \in \mathbb{R}^4$,

$$\hat{p}_x^Z(x) \sim (x_1, x_2, x_3 exp(\sqrt{tx_1(1-\rho^2)}N), x_4)^{'}, \quad N \sim \mathcal{N}(0,1)$$

$$\hat{p}_x^W(dz) \sim \begin{pmatrix} \hat{X}_{x_1}^{CIR}(t) \\ x_2 + \frac{x_1 + \hat{X}_{x_1}^{CIR}(t)}{2} t \\ x_3 \xi_{x_1}(t) \\ x_4 + x_3 \frac{1+\xi_{x_1}(t)}{2} t \end{pmatrix}.$$

where $\xi_{x_1}(t) = exp[(r - \rho a/\sigma)t + (\rho k/\sigma - 1/2)\frac{x_1 + \hat{X}_{x_1}^{CIR}(t)}{2} t + \frac{\rho}{\sigma}(\hat{X}_{x_1}^{CIR}(t) - x_1)]$. And at last we consider the scheme

$$\hat{p}_x(t) = \frac{1}{2}(\hat{p}^W(t) \circ \hat{p}_x^Z(t) + \hat{p}^Z(t) \circ \hat{p}_x^W(t)),$$

with which we can simulate the Heston model.

The figure below shows $\mathbb{E}[e^{-r}(S-(\hat{X}^n_{t^n_n})_3)^+]$ as a function of 1/n. Because of lack of time, we didn't find how to calculate the exact value of this term and we have used the same parameters as set in the reference article and we estimate that the exact value is about 19.006. We have used the second order scheme to simulate the Heston model and obtained the figure below which is similar to the result in the reference article. We see that with the increase of the number of steps, the estimation converges to the exact value and it is in the confidence interval.

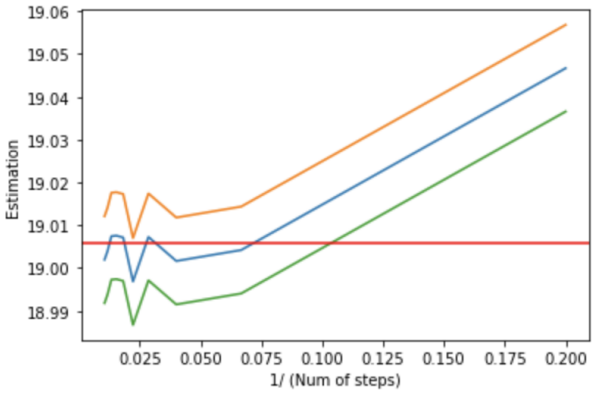

Figure 8: Heston Model: $X^1_0 = 0.04, k = 0.5, a = 0.02, \sigma = 0.4, r = 0.02, X^3_0 = 100, \rho = -0.5$ and $S = 120$.

7 Conclusion

In this EA, we have learned a lot in many aspects.

Firstly, this EA provides us with an precise example to understand what we have learned in stochastic calculus. In the theoretical part where we have seen how the Ito's formula plays an fundamental role in the stochastic analysis and we have been more familiar with the notions such as conditional expectation and transition probability of continuous time Markov chains.

Secondly, we have learned much on how to simulate a stochastic process. In the beginning the representation of discretization scheme with transition probability was really difficult for us to understand since we had only learned the Euler scheme before. But later when trying to implement the algorithm in the article, we saw how powerful this representing method is in analysing the order. What's more, it's very interesting to see that sometimes we can replace some random variables with the other ones of the same first several moments like wen dealing with the problem of negativity when $\sigma^2 > 4a$ in the second order scheme case because it is really a basic but insightful observation and it helps us understand better how we can solve such problems in the future. And we also learned that the decomposition of differential operator in the article is also a powerful way to simulate a complicated process, despite its intuitiveness.

At last, the analysis of the error in this article makes the connection between the deterministic numerical analysis and simulation of stochastic process more clear to us. Above all, the ideas and the methods analysis of error of a scheme in this EA is really close to those in numerical analysis where we apply similar argument to analysis the performance of a scheme. In addition, the illustration of the convergence speed is a very inspiring process even though only some simple calculations are involved: the error of a scheme is hidden behind the random error, and thus in order to see the convergence we need to reduce randomness to such extent that the systematical error is noticeable in the graph. All these observations make us understand the main idea of the paper much more easily.

8 Reference

[1] A. Alfonsi. High order discretization schemes for the CIR process: application to Affine Term Structure and Heston models. Mathematics of Computation, American Mathematical Society, 2010, 79 (269), pp.209-237.

9 Annex: Codes

(1) The function implementing the Euler Scheme

```
def schema_naif(a,k,sigma,x,T,N,Num):
    dt=T/N
    X=np.zeros(Num)+x
    for i in range(N):
        W=np.random.randn(Num)*np.sqrt(dt)
        X += (a-k*X)*dt+sigma*np.sqrt(np.maximum(X,0))*W
    return X
```

(2) The functions for calculating the exact value

```
def laplace(v,x,alpha,b,a,t):
    m = np.exp(b*t)
    q = a**2*(np.exp(2*b*t)-1)/(2*b)
    return np.exp(m**2*x*v/(1-2*q*v))/(1-2*q*v)**(alpha/2)

def exact_value(a,k,sigma,x,T):
    return laplace(-1,x,4*a/sigma**2,-k/2,sigma/2,T)
```

(3) The functions for implementing the second order scheme

```
def psi(k,t):
    if k==0:
        return t
    return (1-np.exp(-k*t))/k

def phi(a,k,sigma,x,t,w):
    p1=np.exp(-k*t/2)*np.power(np.sqrt((a-sigma**2/4)*psi(k,t/2)+np.exp(-k*t/2)*x)+sigma*w/2,2)
    return p1+(a-sigma**2/4)*psi(k,t/2)

def u1(a,k,t,x):
    return x*np.exp(-k*t)+a*psi(k,t)

def u2(a,k,sigma,t,x):
    return np.power(u1(a,k,t,x),2)+(sigma**2)*psi(k,t)*(a*psi(k,t)/2+x*np.exp(-k*t))

def pi(a,k,sigma,t,x):
    delta = 1-np.power(u1(a,k,t,x),2)/u2(a,k,sigma,t,x)
    return (1-np.sqrt(delta))/2

def K2(a,k,sigma,t):
    if sigma**2>4*a:
        p1=np.power(np.sqrt(np.exp(k*t/2)*(sigma**2/4-a)*psi(k,t/2))+sigma*np.sqrt(3*t)/2,2)
        return ((sigma**2/4-a)*psi(k,t/2)+p1)*np.exp(k*t/2)
    else:
        return 0

def Y_norm(Num):
    return np.random.choice(np.array([-1,0,1]),size=Num,p=np.array([1./6.,2./3.,1./6.]))*np.sqrt(3)

def CIR_order2(a,k,sigma,x,T,N,Num):
    X=np.zeros(Num)+x
    dt = T/N
    K =K2(a,k,sigma,dt)
    for i in range(N):
        ind1=(X>K)
        ind2=(X<=K)
        Norm=np.sqrt(dt)*Y_norm(int(np.sum(ind1)))
        X[ind1] = phi(a,k,sigma,X[ind1],dt,Norm)
```

```
        pi_ = pi(a,k,sigma,dt,X[ind2])
        u1_ = u1(a,k,dt,X[ind2])
        U = np.random.uniform(0,1,len(pi_))
        X[ind2] = (U<=pi_)*u1_/(2.*pi_)+(U>pi_)*u1_/(2.*(1.-pi_))
    return X
```

(4) The functions for implementing the third order scheme

```
def X_0(a,k,sigma,t,x):
    return x*np.exp(-k*t)+(a-sigma**2/4)*psi(k,t)

def X_1(sigma,t,x):
    return np.power((np.maximum(np.sqrt(x)+sigma*t/2,0)),2)

def X_tld(a,sigma,t,x):
    return x+t*sigma*np.sqrt(np.abs(a-sigma**2/4))/np.sqrt(2)

def K3(a,k,sigma,t):
    p=psi(-k,t)
    a_s2=a-np.power(sigma,2)/4
    c=np.sqrt(3+np.sqrt(6))
    if a_s2<=0:
        return (-a_s2+np.power(np.sqrt(sigma*np.sqrt(-a_s2/2))+sigma*c/2,2))*p
    elif a<= 1.5*a_s2:
        return sigma*np.sqrt(a_s2/2)*p
    else :
        return p*np.power(np.sqrt(-a_s2+sigma*np.sqrt(a_s2/2))+sigma*c/2,2)

def Y2(N):
    s=np.sqrt(6)
    ran=np.array([np.sqrt(3+s),-np.sqrt(3+s),np.sqrt(3-s),-np.sqrt(3-s)])
    p_=np.array([(s-2)/(4*s),(s-2)/(4*s),0.5-(s-2)/(4*s),0.5-(s-2)/(4*s)])
    return np.random.choice(ran,size=N,p=p_)

def scheme_order3_1(a,k,sigma,t,x):
    Num=len(x)
    x1=x.copy()
    eps=np.random.choice(np.array([-1,1]),size=Num)
    zet=np.random.choice(np.array([1,2,3]),size=Num)
    pt=psi(-k,t)
    Y=Y2(Num)
    if sigma*sigma<=4*a:
        x1[zet==1]=X_tld(a,sigma,eps[zet==1]*pt,X_0(a,0,sigma,pt,X_1(sigma,np.sqrt(pt)*Y[zet==1],x1[zet==1])))
        x1[zet==2]=X_0(a,0,sigma,pt,X_tld(a,sigma,eps[zet==2]*pt,X_1(sigma,np.sqrt(pt)*Y[zet==2],x1[zet==2])))
        x1[zet==3]=X_0(a,0,sigma,pt,X_1(sigma,np.sqrt(pt)*Y[zet==3],X_tld(a,sigma,eps[zet==3]*pt,x1[zet==3])))
    else :
        x1[zet==1]=X_tld(a,sigma,eps[zet==1]*pt,X_1(sigma,np.sqrt(pt)*Y[zet==1],X_0(a,0,sigma,pt,x1[zet==1])))
        x1[zet==2]=X_1(sigma,np.sqrt(pt)*Y[zet==2],X_tld(a,sigma,eps[zet==2]*pt,X_0(a,0,sigma,pt,x1[zet==2])))
        x1[zet==3]=X_1(sigma,np.sqrt(pt)*Y[zet==3],X_0(a,0,sigma,pt,X_tld(a,sigma,eps[zet==3]*pt,x1[zet==3])))
    return x1*np.exp(-k*t)

def u3(a,k,sigma,t,x):
    return u1(a,k,t,x)*u2(a,k,sigma,t,x)+sigma**2*psi(k,t)*(
        2*x*x*np.exp(-2*k*t)+psi(k,t)*(a+sigma**2/2)*(3*x*np.exp(-k*t)+a*psi(k,t)) )

def s(a,k,sigma,t,x):
    return (u3(a,k,sigma,t,x)-u1(a,k,t,x)*u2(a,k,sigma,t,x))/(u2(a,k,sigma,t,x)-u1(a,k,t,x)**2)

def p(a,k,sigma,t,x):
    return (u1(a,k,t,x)*u3(a,k,sigma,t,x)-u2(a,k,sigma,t,x)**2)/(u2(a,k,sigma,t,x)-u1(a,k,t,x)**2)
```

```
def delta(a,k,sigma,t,x):
    return s(a,k,sigma,t,x)**2-4*p(a,k,sigma,t,x)

def x_pos(a,k,sigma,t,x):
    return (s(a,k,sigma,t,x)+np.sqrt(delta(a,k,sigma,t,x)))/2

def x_neg(a,k,sigma,t,x):
    return (s(a,k,sigma,t,x)-np.sqrt(delta(a,k,sigma,t,x)))/2

def pi2(a,k,sigma,t,x):
    return (u1(a,k,t,x)-x_neg(a,k,sigma,t,x))/np.sqrt(delta(a,k,sigma,t,x))

def scheme_order3_2(a,k,sigma,t,x):
    U = np.random.uniform(0,1,len(x))
    pi = pi2(a,k,sigma,t,x)
    return (U<=pi)*x_pos(a,k,sigma,t,x)+(U>pi)*x_neg(a,k,sigma,t,x)

def CIR_order3(a,k,sigma,x,T,N,Num):
    X=np.zeros(Num)+x
    t = T/N
    K = K3(a,k,sigma,t)
    for i in range(N):
        ind1=(X>=K)
        ind2=(X<K)
        X[ind1] = scheme_order3_1(a,k,sigma,t,X[ind1])
        X[ind2] = scheme_order3_2(a,k,sigma,t,X[ind2])
    return X
```

(5) The functions for implementing the Heston Model

```
def CIR_od2_1pas(a,k,sigma,x,dt):
    X=x.copy()
    K=K2(a,k,sigma,dt)
    ind1=(X>K)
    ind2=(X<=K)
    Norm=np.sqrt(dt)*Y_norm(int(np.sum(ind1)))
    X[ind1] = phi(a,k,sigma,X[ind1],dt,Norm)
    pi_ = pi(a,k,sigma,dt,X[ind2])
    u1_ = u1(a,k,dt,X[ind2])
    U = np.random.uniform(0,1,len(pi_))
    X[ind2] = (U<=pi_)*u1_/(2.*pi_)+(U>pi_)*u1_/(2.*(1.-pi_))
    return X

def pZ(x,rho,t):
    y=x.copy()
    y[:,2]=y[:,2]*np.exp(np.sqrt(t*x[:,0]*(1-rho**2))*np.random.randn(len(x)))
    return y

def kesi(x,a,k,sigma,r,rho,t,CIR):
    ks=(r-rho*a/sigma)*t+(rho*k/sigma-0.5)*(x+CIR)*t/2+rho*(CIR-x)/sigma
    return np.exp(ks)

def pW(x,a,k,sigma,r,rho,t,CIR):
    y=CIR(a,k,sigma,x[:,0],t)
    z=kesi(x[:,0],a,k,sigma,r,rho,t,y)
    return np.array([y,x[:,1]+(x[:,0]+y)*t/2,x[:,2]*z,x[:,3]+x[:,2]*(1+z)*t/2]).T

def p(x,a,k,sigma,r,rho,t,CIR):
    u=np.random.uniform(0,1,len(x))
    WZ=pW(pZ(x,rho,t),a,k,sigma,r,rho,t,CIR)
    ZW=pZ(pW(x,a,k,sigma,r,rho,t,CIR),rho,t)
```

```
    return np.append(WZ[u>0.5],ZW[u<=0.5],axis=0)

def Heston(x,a,k,sigma,r,rho,CIR,T,N,Num):
    X=np.array([x]*Num)
    t=T/N
    for i in range(N):
        X=p(X,a,k,sigma,r,rho,t,CIR)
    return X
```