1. Introduction

In nowadays stock market, option pricing has been a demanding research field due to dynamic features of stock price change. A number of mathematicians and economists have put great effort on numerical methods for option pricing since early 20th. This report begins with introducing basic concepts in stochastic process such as Brownian motion and stochastic differential equation then commonly used financial models for option pricing.

The concept of **Brownian motion** was first discovered in physics, which originally describes random motion of particles in a medium such as a liquid or a gas. It is named after Scottish botanist Robert Brown. Due to its stochastic properties, the name of Brownian motion is also used in mathematics, which is called Wiener process in honor of American mathematician Norbert Wiener.

Developed in the Brownian motion theory, a **stochastic differential equation**(SDE) is a differential equation in which the values of one or more variable are stochastic, so that the value of solution is also stochastic. The general form of SDE is:

$$dY(t) = aY(t)dt + bY(t)dB(t)$$

where $B(t)$ is Brownian motion.

To understand the mathematical properties of SDE, Japanese mathematician Kiyosi Itô introduced the concept of stochastic integral in 1940s which first proposed the definition of SDE. Thus, for a non-differentiable function at any point, the integral can be defined as long as the integrand is adapted.

In Itô's calculus, **Itô's lemma** is used to find differential of time-dependent stochastic function. For example, consider geometric Brownian motion in SDE form with drift μ and diffusion σ, taking log of $Y(t)$ gives:

$$d\log Y(t) = \sigma dB(t) + (\mu - \frac{\sigma^2}{2})dt$$

$$\log Y(t) = \log Y(0) + \sigma B(t) + (\mu - \frac{\sigma^2}{2})t$$

So that the expression of $Y(t)$ can be found as:

$$Y(t) = Y(0)e^{\sigma B(t)+(\mu-\frac{\sigma^2}{2})t}$$

which is considered as exact solution for $Y(t)$.

Based on mathematical properties of stochastic process, **Black–Scholes model** is developed in financial mathematics to estimate price of European options. It is named after American economists Fischer Black and Myron Scholes who first published it in 1973. The estimate of stock price with Brownian motion in Black–Scholes model can be expressed as:

$$S(T) = S(0)e^{\sigma B(T)+(r-\frac{\sigma^2}{2})T}$$

2. Methodology

2.1 Black–Scholes formula

The Black–Scholes equation from Black–Scholes model is a partial differential equation which describes stock price over time:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

Notation:

$S(t)$: stock price at time t

$V(S(t), t)$: option price corresponding to stock price at time t

r: annual risk-free interest rate

σ: volatility or standard deviation of stock price change

t: time in years

From the Black–Scholes equation, the Black–Scholes formula can be deduced to calculate the price of call and put options:

$$C(S(t), t) = S(t)N(d_1) - Ke^{-r(T-t)}N(d_2)$$

$$P(S(t), t) = Ke^{-r(T-t)} - S(t) + C(S(t), t) = Ke^{-r(T-t)}N(-d_2) - S(t)N(-d_1)$$

$$\text{where } d_1 = \frac{\ln\frac{S(t)}{K} + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}, d_2 = \frac{\ln\frac{S(t)}{K} + (r - \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}} = d_1 - \sigma\sqrt{T-t}$$

Notation:

$C(S(t), t)$: call option price corresponding to stock price at time t

$P(S(t), t)$: put option price corresponding to stock price at time t

$N(\cdot)$: CDF of $N(0,1)$

T: expiry time in years

K: strike price

For example, to predict Zoom's stock price at 60 days after start date in Python, the stock price data is extracted from Yahoo database:

```
stock = dr.DataReader('ZM','yahoo',start='2020-09-10')['Adj Close'][:60]
stock[:20]
```

```
Date
2020-09-10    384.480011
2020-09-11    383.000000
2020-09-14    403.510010
2020-09-15    410.940002
2020-09-16    408.170013
2020-09-17    413.125000
2020-09-18    438.730011
2020-09-21    468.470001
2020-09-22    492.600006
2020-09-23    500.529999
2020-09-24    464.980011
2020-09-25    496.500000
2020-09-28    487.660004
2020-09-29    465.500000
2020-09-30    470.109985
2020-10-01    482.989990
2020-10-02    482.230011
2020-10-05    485.369995
2020-10-06    478.250000
2020-10-07    480.609985
Name: Adj Close, dtype: float64
```

From the stock price data above, the parameters for Black–Scholes model can be obtained. Note that K, r and σ are computed from historical stock price change from 2020-09-10 to 2020-12-03.

```
# parameter
S0 = stock[0]   # stock price at start time = 384.48
K = S0*np.exp(stock.pct_change().mean())    # strike price = 385.45
T = 1/6   # expiry time in years
r = stock.pct_change().mean()*60    # annual risk-free interest rate = 0.151
sigma = stock.pct_change().std()*np.sqrt(60)    # volatility = 0.387
```

Then the call option and put option can be computed by Black–Scholes formula, and based on the definition of call option and put option:

$$C = \max(S - K,0),\ P = \max(K - S,0)$$

Since the call option is always positive during this period, the stock price at expiry time can be computed by:

$$S(T) = K + C(S(T), T)$$

```
# Black–Scholes formula
d1 = (np.log(S0/K) + (r+sigma**2/2)*(T-0)) / (sigma*np.sqrt(T-0))
d2 = d1 - sigma*np.sqrt(T-0)

C = S0*norm.cdf(d1) - K*np.exp(-r*(T-0))*norm.cdf(d2)   # call option
P = K*np.exp(-r*(T-0))*norm.cdf(-d2) - S0*norm.cdf(-d1)   # put option

ST = K + C
ST
```

```
413.48833436881137
```

The stock price of Zoom on 2020-12-03 computed by Black–Scholes formula is $413.49, which is very close to the actual price $413.54. In other words, Black–Scholes model is sufficient to predict future stock price as long as the risk-free interest rate is accurate and the volatility for the price change trend is obtained.

2.2 Euler-Maruyama method

Euler-Maruyama method(Euler method) is a basic method for numerical approximation of SDE solution. It is named after 18th Swiss scientist Leonhard Euler and Japanese mathematician Gisiro Maruyama.

To solve an SDE on time interval $[0,T]$, the approximate solution to Y can be seen as a Markov chain defined below:

(1) Partition $[0,T]$ into N equal subintervals with width $\Delta t > 0$:

$$\{\hat{Y}(0), \hat{Y}(\Delta t), \hat{Y}(2\Delta t), \cdots, \hat{Y}(N\Delta t)\} \text{ where } \Delta t = T/N$$

(2) For $0 \le n \le N + 1$, the approximation of $Y(t_n)$ is defined as:

$$\hat{Y}(t_n) = \hat{Y}(t_{n-1}) + a\hat{Y}(t_{n-1})\Delta t + b\hat{Y}(t_{n-1})\Delta B(t_{n-1})$$

where $\Delta B(t_{n-1}) = B(t_n) - B(t_{n-1})$ with i.i.d. N(0, Δt)

(3) Hence the approximation of $Y(T)$ is:

$$\hat{Y}(T) = \hat{Y}(t_N)$$

To predict Zoom's stock price in Black–Scholes model above, the stock price at expiry time can be computed as $395.36 by iteration in Python:

```
# Euler method
I = 1000   # number of iterations
N = 1000
dt = T/N

ST_E = []
for i in range(I):
  Sn_E = S0
  dB = np.random.normal(0,np.sqrt(dt),N)  # create N random numbers
  for j in dB:
    Sn_E += r*Sn_E*dt + sigma*Sn_E*j
  ST_E.append(Sn_E)

ST_E = sum(ST_E)/I
ST_E
```

```
395.3633780291864
```

2.3 Richardson extrapolation

In Euler scheme, Richardson extrapolation is an alternative method to reduce the convergence error and improve the convergence rate of the approximations. Based on the iterations by Euler method, Richardson extrapolation has the following improvement:

(1) Estimate $Y_1(T)$ with Δt

(2) Estimate $Y_2(T)$ with $2\Delta t$

(3) The approximation equation is: $\hat{Y}(T) = 2\hat{Y}_1(T) - \hat{Y}_2(T)$

Note that ΔB for the two approximations should have common random numbers in order to capture sufficient improvement of the approximation performance.

The Richardson extrapolation estimate for Zoom's stock price at expiry time is $392.69.

```
# Richardson extrapolation
I = 1000   # number of iterations
N1 = 1000
dt1 = T/N1
ST1, ST2 = [], []

for i in range(I):
  dB = np.random.normal(0,np.sqrt(dt1),N1)  # create N1 random numbers

  Sn1 = S0
  for j in dB:
    Sn1 += r*Sn1*dt1 + sigma*Sn1*j
  ST1.append(Sn1)

  Sn2 = S0
  for k in dB[1::2]:    # use same random numbers
    Sn2 += r*Sn2*dt1*2 + sigma*Sn2*k
  ST2.append(Sn2)

ST_R = 2*sum(ST1)/I - sum(ST2)/I
ST_R
```

```
392.6918092999584
```

2.4 Milstein method

Milstein method increases the accuracy of the Euler approximation by adding a second-order correction term, which is derived from the stochastic Taylor series expansion of $Y(t)$ by applying Itô's lemma to $a(\cdot)$ and $b(\cdot)$.

Recall the Euler approximation of $Y(n)$, Milstein method constructs a superior approximation for b on time interval $[(n-1)\Delta t, n\Delta t]$:

$$\hat{Y}(t_n) = \hat{Y}(t_{n-1}) + a\hat{Y}(t_{n-1})\Delta t + b\hat{Y}(t_{n-1})\Delta B(t_{n-1}) + \frac{1}{2}b'\hat{Y}(t_{n-1})b(\hat{Y}(t_{n-1}))((\Delta B(t_{n-1}))^2 - \Delta t)$$

Note that when $b' = 0$, this method is equivalent to Euler method.

For the Zoom stock price example, the approximation equation becomes:

$$\hat{S}(t_n) = \hat{S}(t_{n-1}) + r\hat{S}(t_{n-1})\Delta t + \sigma\hat{S}(t_{n-1})\Delta B(t_{n-1}) + \frac{1}{2}\sigma^2\hat{S}(t_{n-1})((\Delta B(t_{n-1}))^2 - \Delta t)$$

and the stock price at expiry time is $391.35 by iteration in Python:

```
# Milstein method
I = 1000   # number of iterations
N = 1000
dt = T/N

ST_M = []
for i in range(I):
  Sn_M = S0
  dB = np.random.normal(0,np.sqrt(dt),N)  # create N random numbers
  for j in dB:
    Sn_M += r*Sn_M*dt + sigma*Sn_M*j + 0.5*sigma**2*Sn_M*(j**2-dt)
  ST_M.append(Sn_M)

ST_M = sum(ST_M)/I
ST_M
```

```
391.3451171149855
```

3. Numerical results

3.1 Convergence error

In stochastic process, the concept of convergence defines error of two stochastic processes as Δt is reduced to zero. There are two common convergence definitions: weak convergence and strong convergence. The error terms for weak and strong convergences are defined below:

$$e_w(\Delta t) = \sup_{t_n} |E(X(t_n)) - E(Y(t_n))|$$

$$e_s(\Delta t) = \sup_{t_n} E(|X(t_n) - Y(t_n)|)$$

From the error terms, weak convergence computes the error between expected values of two stochastic processes at time t_n, which represents average performance of the numerical approximation for stock price on given date; strong convergence computes the MSE between the

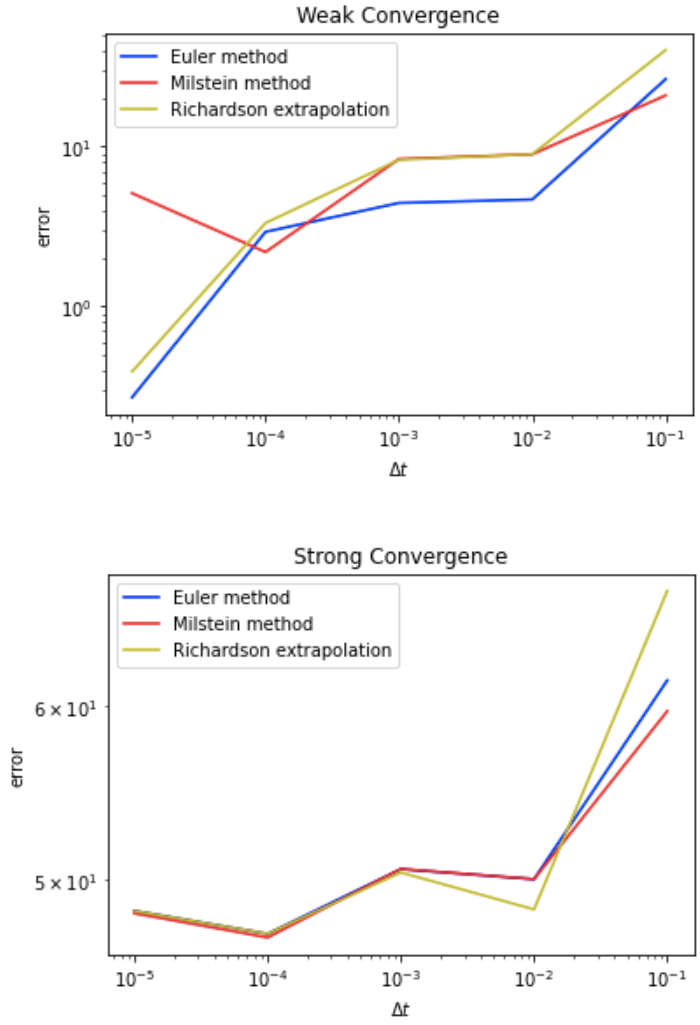
approximation and exact solution for each sample path, which is more appropriate to assess the performance of numerical methods in SDE.

In any case, $X(t)$ exhibits weak or strong convergence to $Y(t)$ if the error equals to zero as Δt tends to zero:

$$\lim_{\Delta t \to 0} e(\Delta t) = 0$$

To evaluate the performance of the three approximation methods above, the weak and strong convergence errors are computed by iterations over random sample paths and simulation on future stock price in Python. The modeling parameters are from the Zoom's stock price example. Due to computational limits, the number of sample paths taken for approximation is set to 1000, and Δt ranges from 0.00001 to 0.1.

```
# Convergence error

I = 1000   # number of iterations
ew_E, ew_M, ew_R = [], [], []   # weak error by Euler, Milstein, Richardson extrapolation
es_E, es_M, es_R = [], [], []   # strong error by Euler, Milstein, Richardson extrapolation

for dt in np.logspace(-5,-1,5):   # set a range of delta t: (0.00001, 0.0001, 0.001, 0.01, 0.1)
  tn = np.arange(dt,T+dt,dt)   # create time intervals: (dt, 2dt, ... , T)
  N = len(tn)

  Xn_Esum, Xn_Msum, Xn_Rsum, Yn_sum = np.zeros(N), np.zeros(N), np.zeros(int(N/2)), np.zeros(N)
  error_E, error_M, error_R = np.zeros(N), np.zeros(N), np.zeros(int(N/2))

  for i in range(I):
    dB = np.random.normal(0,np.sqrt(dt),N)   # create N random numbers

    Yn = S0*np.exp(sigma*dB+(r-sigma**2/2)*tn)   # exact solution

    X_E, Xn_E, X_M, Xn_M = S0, [], S0, []
    for j in dB:
      # Euler method
      X_E += r*X_E*dt + sigma*X_E*j
      Xn_E.append(X_E)
      # Milstein method
      X_M += r*X_M*dt + sigma*X_M*j + 0.5*sigma**2*X_M*(j**2-dt)
      Xn_M.append(X_M)

    # Euler method with Richardson extrapolation
    X_E2, Xn_E2 = S0, []
    for k in dB[1::2]:
      X_E2 += r*X_E2*dt*2 + sigma*X_E2*k
      Xn_E2.append(X_E2)
    Xn_R = 2*np.array(Xn_E[1::2]) - np.array(Xn_E2)
    error_E += abs(Yn-Xn_E)
    error_M += abs(Yn-Xn_M)
    error_R += abs(Yn[1::2]-Xn_R)

    Yn_sum += Yn
    Xn_Esum += Xn_E
    Xn_Msum += Xn_M
    Xn_Rsum += Xn_R

  ew_E.append(max(abs(Yn_sum-Xn_Esum)/I))
  ew_M.append(max(abs(Yn_sum-Xn_Msum)/I))
  ew_R.append(max(abs(Yn_sum[1::2]-Xn_Rsum)/I))

  es_E.append(max(error_E/I))
  es_M.append(max(error_M/I))
  es_R.append(max(error_R/I))
```



From the weak and strong convergence errors plots by Euler, Milstein and Richardson extrapolation methods above, the convergence errors tend to decrease as Δt is reduced to zero, which indicates the approximation accuracy can be improved by increasing the number of time intervals. In the Weak Convergence plot, the lines of Euler method and Richardson extrapolation have similar positive slope since the basic approximation equation is the same, while the line of Milstein method has no specific slope. In the Strong Convergence plot, the convergence errors decrease as Δt decreases in general, while the lines of all three methods tend to be similar.

In conclusion, although the approximation performance can be assessed intuitively by convergence error plot, it is hard to compare with other numerical methods. Besides, reducing Δt to zero is costly in the simulation process for increasing computation time and reduction limit of Δt since the simulation program can only use finite number. In order to generate a precise assessment for approximation methods, the concept of convergence rate is introduced in next section.

3.2 Convergence rate

For numerical approximations, convergence order and convergence rate measure how fast an approximation converges as Δt is reduced to zero. An approximation has convergence rate γ if for some constant C:

$$e(\Delta t) \leq C(\Delta t)^{\gamma}$$

Convergence order measures the rate at which the error decreases. If a numerical approximation has a higher convergence order, it typically needs fewer iterations to yield a sufficient estimate. For example, if an approximation is convergent with order γ, the approximation error will decrease by k^{γ} times with Δt reduced by k times. Hence a larger value of γ is preferable for an approximation method indicating a faster convergence of the approximation error to zero.

To compute convergence rate, taking log of Markov's inequality gives:

$$\log(e(\Delta t)) = \log(C) + \gamma \log(\Delta t)$$

Hence the value of γ can be estimated by the slope of error and Δt on logarithmic scale.

For the Zoom's stock price example, the statsmodels library is imported in Python to generate ordinary least squares(OLS) model, so that the convergence rates of the three methods can be computed based on the convergence errors above.

```
# Convergence rate
log_dt = sm.add_constant(np.log(np.logspace(-5,-1,5)))
print('weak convergence rate by Euler =', sm.OLS(np.log(ew_E), log_dt).fit().params[1])
print('weak convergence rate by Milstein =', sm.OLS(np.log(ew_M), log_dt).fit().params[1])
print('weak convergence rate by Richardson extrapolation =', sm.OLS(np.log(ew_R), log_dt).fit().params[1])

print('\n'+'strong convergence rate by Euler =', sm.OLS(np.log(es_E), log_dt).fit().params[1])
print('strong convergence rate by Milstein =', sm.OLS(np.log(es_M), log_dt).fit().params[1])
print('strong convergence rate by Richardson extrapolation =', sm.OLS(np.log(es_R), log_dt).fit().params[1])
```

```
weak convergence rate by Euler = 0.4185129826386479
weak convergence rate by Milstein = 0.18359359811590759
weak convergence rate by Richardson extrapolation = 0.4453835910372216

strong convergence rate by Euler = 0.02361784326068528
strong convergence rate by Milstein = 0.02119530306583406
strong convergence rate by Richardson extrapolation = 0.03044298949927543
```

4. Conclusion

From the output of convergence rates above, it can be concluded that:

(1) The strong convergence rates are all lower than the weak convergence rates, which is often the case in practice;

(2) Richardson extrapolation has slightly higher convergence rates than basic Euler method indicating sufficient improvement, which is hard to interpret from the error plots;

(3) The appropriate method for Zoom's stock price simulation is Euler method with Richardson extrapolation, and Milstein method has inferior approximation performance in this case.

Future improvements:

(1) Increase the number of sample paths

(2) Reduce Δt close to zero

(3) Consider change of volatility by stochastic volatility model. For example, the SDE for variance $V(t)$ in Heston model is:

$$dV(t) = \theta(\omega - V(t))dt + \xi\sqrt{V(t)}dW(t)$$

where ω is mean long-term variance, θ is rate at which $V(t)$ reverts to ω, ξ is volatility of $V(t)$, and $dW(t)$ has N(0, Δt) distribution correlated with $dB(t)$ in the SDE for $S(t)$

5. References

[1] Ross, S. M. (2010). *Introduction to probability models.* Amsterdam: Academic Press.

[2] Sauer, Timothy. (2012). *Numerical Solution of Stochastic Differential Equations in Finance.*

[3] Wang, Liguo. (2016). *Numerical Solutions of Stochastic Differential Equations.* University of Tennessee.

[4] Higham, Desmond. (2001). *An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations.* SIAM Review.

[5] Martin Haugh. (2017). *Simulating Stochastic Differential Equations.* Columbia University

[6] Reaz Chowdhury, M.R.C. Mahdy, Tanisha Nourin Alam. (2020). *Predicting the stock price of frontier markets using machine learning and modified Black–Scholes Option pricing model.* Physica A: Statistical Mechanics and its Applications.

[7] Ruye Wang. (2015). *Order and rate of convergence.* http://fourier.eng.hmc.edu/e176/lectures/NM/node3.html.

[8] *Ordinary Least Squares.* (2020). https://www.statsmodels.org/dev/examples/notebooks/generated/ols.html.