ISSN 2616-7433 Vol. 2, Issue 11: 08-13, DOI: 10.25236/FSST.2020.021102

Reduction of the Hedging Error in the Trinomial Tree Model

Hantao Zhou[1], Jiong Huang[2], Qianwei Yin[3]

1 School of Politics and Economics, King's College London, London WC2R 2LS, United Kingdom
2 School of Business, Macau University of Science and Technology, Macau, China
3 School of Surveying and Mapping, Wuhan University, Wuhan 430072, China

ABSTRACT. *The research problem is how large the replication error is if an investor Delta hedges an European option over many short periods. In the project, the trinomial stock price tree is firstly built to simulate stock price changes. Then, Monte-Carlo approximation is used to put many short-period trinomial tree models together and derive the average hedging errors. The hedging errors become significantly smaller as periods involved in the trinomial tree models pass a certain number. This method can be used to create portfolios with European options.*

KEYWORDS: *Option pricing, Trinomial tree model, Delta hedging, Replication error*

1. Introduction

In non-arbitrage option pricing, both trinomial and binomial tree models are often used to derive option prices. Unlike binomial tree model, trinomial tree model is more intricate in terms of the situations involved. In the binomial tree model, one can easily deduce the hedging strategies by solving for two simultaneous equations. However, in the trinomial tree model, three situations are present and there may or may not be a solution. Thus, an investor cannot derive the perfect strategy in this model by solving three equations. Instead, the strategies are uncertain. As a result, there are constant debates of scholars over the best hedging strategies in the trinomial tree model. Therefore, in this research project, the best hedging strategies are attempted to work out and hence benefit investors in the financial market.

For binomial tree models, stock prices can only go up and down. However, stock prices can also remain unchanged in the trinomial model. These situations are represented by the three simultaneous equations below:

$$\begin{cases} N^B R + N^S S_0 u = F(S_0 u) \\ N^B R + N^S S_0 m = F(S_0 m), \\ N^B R + N^S S_0 d = F(S_0 d) \end{cases}$$

in which F represents the payoffs of the option when the stock price goes up, down, or remains the same. N^B and N^S stands for the trading strategies in the replicating portfolio. u, m and d represents by how much the stock price changes. It is assumed that $u = \frac{1}{d}$ for simplicity in this model. Otherwise there will be too many branches in the model. The simultaneous equations show that the value of the portfolio should be equal to the option's payoff at the next time period regardless to how stock price changes, so that there is no arbitrage opportunities. With an additional situation m, the trinomial tree model can approximate continuous stock price changes better than the binomial model, making its results closer to the Black-Scholes model.

Since there are three simultaneous equations in this system but only two unknown variables, namely N^B and N^S, the solution is rather uncertain. Hence, the hedging strategies from binomial tree model is used in this research. The hypothesis is that if it is delta hedged over many periods, the replication error will become smaller. The hypothesis will be addressed throughout this research using the Monte-Carlo method.

2. Modelling the Trinomial Tree Model and Hedging Errors

2.1 Monte-Carlo

Monte-Carlo simulation is a method that generates artificial random variables 'in order to solve purely deterministic problems'. In this context, millions of random variables are generated and a certain process to

ISSN 2616-7433 Vol. 2, Issue 11: 08-13, DOI: 10.25236/FSST.2020.021102

derive the option prices as well as the errors are repeated millions of times. Thus, the average of errors can be calculated and compared using Python.

Firstly, the parameters are defined below:

```
T = 1
u = 2
m = 1
d = 1/2
R = 1
S0 = 200
K = 150
```

Then, the stock price tree is modeled using Python:

```
import numpy as np
stock = np.zeros([T * 2 + 1, T + 1])
for t in range(T + 1):
for i in range(t * 2 + 1):
stock[i, t] = S0 * (u ** np.maximum(t - i, 0)) * (m ** i) * (d ** np.maximum(i - t, 0))
with np.printoptions(precision=2, suppress=True):
print(stock)
```

The output in red shows the trinomial tree model in the form of a matrix, with the initial situation in the first column and three situations in the second column:

```
[[200. 400.]
[  0. 200.]
[  0. 100.]]
```

After that, the Monte-Carlo approximation is carried out by repeating the process below 1000000 times:

```
pu = 1/3
pm = 1/3
pd = 1/3
N = 1000000
import numpy as np
x = np.random.multinomial(n=1, pvals=[pu,pm,pd], size = N)
Y1 = (u ** x[:, 0]) * (m ** x[:, 1]) * (d ** x[:, 2])
S1 = S0 * Y1
def put_payoff(stock, strike):
pyff = np.maximum(strike - stock, 0)
return pyff
f = put_payoff(S1, K)
```

Then the hedging strategies in the binomial tree model is computed and the mean of its errors is derived as a reference:

```
N_B = (1 / R) * (u * put_payoff(S0 * d, K) – d * put_payoff(S0 * u, K)) / (u – d)
N_S = (put_payoff(S0 * u, K) – put_payoff(S0 * d, K)) / (S0 * u – S0 * d)
```

ISSN 2616-7433 Vol. 2, Issue 11: 08-13, DOI: 10.25236/FSST.2020.021102

```
portfolio = N_B * R + N_S * S1

error = f – portfolio

error.mean()

-11.11060000000001
```

The absolute value of the error is 11.1 in this case.

2.2 Strategy to Reduce Error

From the valuation of portfolio in the section above, it can be inferred that stock price is the only independent variable that is volatile in the trinomial tree model. By contrast, the value of risk-free asset has only a constant growth rate. By solving the three simultaneous equations two at a time, the solutions vary only because of the stock price changes. Using this observation, the strategy to reduce error is derived. The hedging strategy for the risk-free asset, N^B, remains constant as it is in the binomial tree model. However, the strategy for stock, N^S, is taken average for all its three values from the three sets of two simultaneous equations. The detailed procedure for the strategy is calculated below in a *one-period* trinomial tree model. Firstly, the three sets of equations are solved using NumPy:

```
A1 = [[R, S0 * u], [R, S0 * m]]

B1 = [put_payoff(S0 * u, K), put_payoff(S0 * m, K)]

N1 = np.linalg.solve(A1, B1) # This contains both N_B and N_S

A2 = [[R, S0 * m], [R, S0 * d]]

B2 = [put_payoff(S0 * m, K), put_payoff(S0 * d, K)]

N2 = np.linalg.solve(A2, B2)

# Check the solution procedure is correct by solving for N3 – exact
hedging strategy from binomial tree model:

A3 = [[R, S0 * u], [R, S0 * d]]

B3 = [put_payoff(S0 * u, K), put_payoff(S0 * d, K)]

N3 = np.linalg.solve(A3, B3)

print(N3)

[66.66666667 -0.16666667]
```

Then take average of N^S from all three values and valuate the mean error:

```
N_Save = (N1[1] + N2[1] + N3[1]) / 3

n_portfolio = N_B * R + N_Save * S1

n_error = f - n_portfolio

n_error.mean()
```

2.3 The error becomes much smaller than its previous absolute value.

2.4 Multi-Period Model

One period in previous steps refers to one day. It means the stock will go up or down only after a whole day. But now one may know the variation after one hour, one minute or even one second. And since the stock moves a little in a day, it only moves much less than a little after a very short period.

2 parameters related to the expectation and volatility of the multi-period stock is involved, namely $E^P[\log(S_n)]$ and $E^P[\log(S_n)^2]$. Since $Y_n = \frac{S_{n+1}}{S_n}$ indicates variation and the mathematical logarithm will decrease the error of an unknown strategy because its value will level off when independent variable increases, the variation of the average of $\log(S_n)$ and $\log(S_n)^2$ reveal the volatility how a stock moves 'continuously'.

ISSN 2616-7433 Vol. 2, Issue 11: 08-13, DOI: 10.25236/FSST.2020.021102

The two figures can be realized by coding. It is assumed that $u = e^{\frac{1.5}{n^{0.05}}}$, beacuse as n increases, the length of each period becomes shorter and shorter. So the variation in stock price becomes less and less. And when n gets very large towards infinity, u gets a little bit more than 1, and d gets a little bit less than 1.

```
import numpy as np
n = 400
u = np.exp(1.5 / (n ** 0.5))
m = 1
d = 1 / u
R = 1
S0 = 1
K = 0.8
pu = 2 / 9
pm = 5 / 9
pd = 2 / 9
N = 10
x = np.random.multinomial(n=1, pvals=[pu,pm,pd], size = [N, n])
```

Here, *np.random.multinomial( )* is used to extract examples from polynomial distribution. It means making one argument among u, m and d be 1 and the other two be 0. This process will be done for 10×400 times, as there are 10 independent experiences (set for ease of looping through large number of codes) and each has 400 periods. After that, individual situations are summed together to form a *random* multi-period stock price model:

```
a = x.sum(axis = 1)
Yn = (u ** a[:, 0]) * (m ** a[:, 1]) * (d ** a[:, 2])
Sn = S0 * Yn
```

Y_n means how much S_0 will multiply according to the random sequences above. The sum of the frequency when stock goes up, goes down and keeps stay equals to n, which is 400. S_n is the stock price when it comes to maturity time. There are 10 values since 10 random sequences are considered. Finally, $E^P[\log(S_n)]$ and $E^P[\log(S_n)^2]$ are computed:

```
log_Sn = np.log(Sn)
log_Sn2 = np.log(Sn ** 2)
EP_log_Sn = log_Sn.mean()
EP_log_Sn2 = log_Sn2.mean()
print("EP_log_Sn =", EP_log_Sn, "EP_log_Sn2 =", EP_log_Sn2)
```

EP_log_Sn = -0.2925000000000044 EP_log_Sn2 = -0.5850000000000091

The output above shows that after many times of trials, the expectation of a stock price change is approximately zero, since three situations can all happen with roughly equal probability. Its variance is also smaller since many trials are involved. Therefore, multi-period trinomial tree model simulates stock prices with higher certainty, consequently making the hedging strategies closer to perfect.

2.5 Deriving the New Hedging Error in the Multi-Period Model

In Section 2.3, it can be concluded that the multi-period trinomial stock price tree model can significantly reduce the uncertainty in stock price changes. Thus, it may be able to yield a better hedging strategy with a lower error.

First, define a function to get the payoff from a put option:

ISSN 2616-7433 Vol. 2, Issue 11: 08-13, DOI: 10.25236/FSST.2020.021102

```
def put_payoff(stock, strike):
pyff = np.maximum(strike - stock, 0)
return pyff
f = put_payoff(Sn, K)
```

Then, construct S_n in n periods:

```
b = x.cumsum(axis = 1)
S_matrix = np.zeros([N, n])
for nx in range(n):
S_matrix[:, nx] = S0 * (u ** b[:, nx][:, 0]) * (m ** b[:, nx][:, 1]) * (d ** b[:, nx][:, 2])
```

Finally, compute hedging strategies N^B, N^S and the mean error:

```
N_Blast = (1 / R) * (u * put_payoff(S_matrix[:, n - 2] * d, K) - d * put_payoff(S_matrix[:, n - 2] * u, K)) / (u - d)
N_Slast = (put_payoff(S_matrix[:, n - 2] * u, K) - put_payoff(S_matrix[:, n - 2] * d, K)) / (S_matrix[:, n - 2] * u - S_matrix[:, n - 2] * d)
portfolio = N_Blast * R + N_Slast * Sn
error = f - portfolio
error.mean()
-6.106226635438361e-17
```

Indeed, the error is much smaller than the previous absolute value of 11.1. Also, as n and N get larger, the error would probably fall. This is because larger n and N make the process more random and unpredictable. And u or d, being close to 1, make the variation flow slightly. So it won't surge or decrease drastically.

2.6 Plot of Error Against n

To visualize the change in error against the number of periods in the model, n, a graph is plotted using Python and Matplotlib. The process is presented below:

```
import matplotlib.pyplot as plt
%matplotlib inline
# Construct an isochromatic series: From 1 to 401, and 9 intervals in total:
n_x = np.linspace(1, 401, 9)
# Record the errors when changing 'n' for 9 times:
errors = [-0.24311404952653365, -0.0003960939712604028, -0.005616339608761273, 1.1102230246251566e-17, 6.661338147750939e-17, 1.1102230246251566e-17, -0.0034619381741944734, -0.0024881370217119973, 3.3306690738754695e-17]
# Plot the graph:
plt.plot(n_x, errors)
plt.xlabel(“$n$”)
plt.ylabel(“Errors”)
plt.title(“Errors against $n$”)
```

ISSN 2616-7433 Vol. 2, Issue 11: 08-13, DOI: 10.25236/FSST.2020.021102

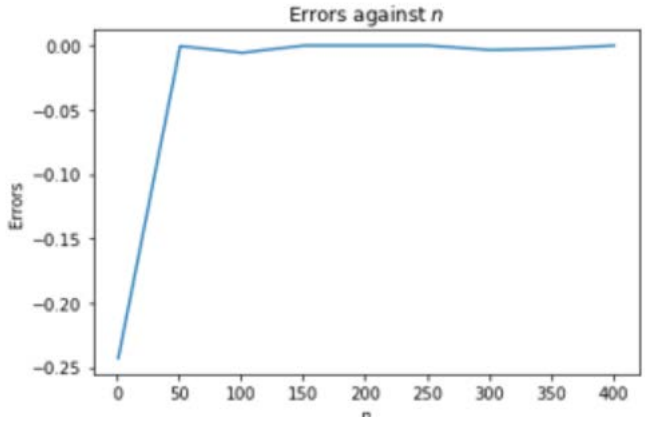

Fig.1 Errors against n

The image output is shown on the right as Figure 1. And as Figure 1 illustrates, errors become significantly smaller after reaching a certain number of n and have very small fluctuations. However, since the interval for n is set 50 for the ease of looping through the entire file and derive the 9 errors, it is uncertain where the cut-off point is. But it must lie between 1 and 51.

3. Conclusion

From the research above, it can be inferred that the hedging error becomes smaller as more periods are involved in the trinomial tree model. Firstly, in Section 2.1, the stock price tree is modelled to derive stock prices as well as their corresponding payoffs. Then, the hedging strategies N^B and N^S from the binomial tree model is used. They are calculated using the formulae $N^B = R^{-1}\frac{uF(S_0d)-dF(S_0u)}{u-d}$ and $N^S = \frac{F(S_0u)-F(S_0d)}{S_0(u-d)}$. The strategies are substituted into the trinomial tree model and yield error. The error is the difference between true payoffs and portfolios valued using the strategies. After that, a Monte-Carlo study is conducted with $P(u) = P(m) = P(d) = \frac{1}{3}$. This computes the errors many times and the mean of errors is -11.1.

Inspired by the variation in stock prices, a strategy to reduce error is computed in Section 2.2. In the strategy, average error in one period is reduced when the average of N^S from solving three systems of two simultaneous equations is used. While N^B is calculated using the previous formula.

For different numbers of periods n, two parameters, $E^P[\log(S_n)]$ and $E^P[\log(S_n)^2]$ are calculated in Section 2.3 using many numbers of trials. The probabilities of changes are altered as well. The result indicates that the expectation of a stock price change is approximately zero, as all three situations can happen with roughly equal probability. The variance also becomes smaller. Hence, multi-period trinomial tree model simulates stock prices with higher certainty, consequently making the hedging strategies closer to perfect.

In Section 2.4, the mean of hedging errors is calculated in order to prove that trinomial model does improve certainty of stock price changes. And consequently, the errors are plotted against the number of periods in Section 2.5. The graph shows that hedging errors become smaller as n increases, especially after n passes a particular value. This proves an important finding that the error can be made smaller when many trinomial tree models are used consecutively to approximate a continuous stock price change.

References

[1] Hull, J, Options (2015). Futures, and Other Derivatives. 9th ed., New Jersey, Pearson.
[2] Deutsch, H (2002). Binomial and Trinomial Trees'. Derivatives and Internal Models.
[3] Kroese, D, T Brereton, T Taimre, Z Botev (2014). Why the Monte Carlo Method is So Important Today. Wiley Interdisciplinary Reviews: Computational Statistics, no.6. pp.12-13.