J. Korean Soc. Ind. Appl. Math. Vol.23, No.3, 237–251, 2019 http://doi.org/10.12941/jksiam.2019.23.237

ANDROID APPLICATION FOR PRICING TWO-AND THREE-ASSET EQUITY-LINKED SECURITIES

HANBYEOL JANG[1], HYUNSOO HAN[1], HAYEON PARK[1], WONJIN LEE[1], JISANG LYU[2], JINTAE PARK[2], HYUNDONG KIM[2], CHAEYOUNG LEE[2], SANGKWON KIM[2], YONGHO CHOI[3], AND JUNSEOK KIM[2†]

[1]DEPARTMENT OF FINANCIAL ENGINEERING, KOREA UNIVERSITY, SEOUL 02841, REPUBLIC OF KOREA

[2]DEPARTMENT OF MATHEMATICS, KOREA UNIVERSITY, SEOUL 02841, REPUBLIC OF KOREA
Email address: `cfdkim@korea.ac.kr`

[3]DEPARTMENT OF MATHEMATICS AND BIG DATA, DAEGU UNIVERSITY, GYEONGSAN-SI, GYEONGSANBUK-DO 38453, REPUBLIC OF KOREA

ABSTRACT. We extend the previous work [J. Korean Soc. Ind. Appl. Math. 21(3) 181] to two-and three-asset equity-linked securities (ELS). In the real finance market, two-or three-asset ELS is more popular than one-asset ELS. Therefore, we need to develop mobile platform for pricing the two-and three-asset ELS. The mobile implementation of the ELS pricing will be very useful in practice.

1. INTRODUCTION

Fintech has become an important field in the financial industry [1, 2, 3, 4]. For this reason, we are interested in mobile app that can compute the prices of financial derivatives. In this paper, we extend the previous work [5] of mobile platform for pricing of one-asset equity-linked securities (ELS) to two-and three-asset ELS. In the real finance market, two-or three-asset ELS is more popular than one-asset ELS. As listed in Table 1, the percentages of two-or three-asset ELS are significantly higher than the one-asset ELS. Here, we show four representative financial companies for issuing ELS from August 2, 2018 to September 2, 2019. The portion of two-and three-asset is almost 96.8% among surveyed ELS products. For this reason, it is very important to calculate the fair value of two-and three-asset ELS products over one-asset ELS product. Several models and numerical methods have been studied to compute fair prices for financial derivatives [6, 7, 8, 9, 10, 11, 12, 13]. We compute the price of ELS using the Monte Carlo simulation (MCS). The MCS is a popular method for pricing of financial derivatives [14, 15, 16, 17, 18].

Received by the editors September 5 2019; Accepted September 14 2019; Published online September 25 2019.
2000 *Mathematics Subject Classification.* 91G60, 91G80, 97N40.
Key words and phrases. Mobile platform, MCS, ELS.
† Corresponding author.

TABLE 1. Data for ELS. The values inside parentheses indicate percentages.

Case	One-asset ELS	Two-asset ELS	Three-asset ELS
Company 1	10 (1.33 %)	41 (5.48 %)	697 (93.18 %)
Company 2	1 (0.28 %)	56 (15.86 %)	296 (83.95 %)
Company 3	45 (5.62 %)	61 (7.62 %)	695 (86.77 %)
Company 4	19 (4.24 %)	80 (17.85 %)	349 (77.90 %)

Figure 1 shows the numerical Black–Scholes (BS) model input/out tab after computation of put option (left) and display of computational result (right) [19]. It is a simple implementation of European put option. So far, there is only one-asset ELS mobile implement [5]. To the authors' knowledge this is the first time to implement a mobile platform of two- and three-asset ELS.

FIGURE 1. Left and right are numerical BS model input/out tab after computation of put option and display of computational result, respectively. Reprinted from [19] with permission from IEEE.

Therefore, we need to develop mobile platform for pricing the two-and three-asset ELS. The mobile implementation of the ELS pricing will be very useful in practice. This article is organized as follows. In Section 2, numerical solution algorithms are given. In Section 3, numerical experiments are presented. Conclusions are discussed in Section 4.

2. NUMERICAL SOLUTION ALGORITHMS

The parameters we used for pricing the step-down ELS are the ELS early redemption date (T), strike percentages (K), coupon rates (c), face value $(F = 100)$, knock-in barrier $(KI = 65)$, dummy rate $(d = 0.15)$, the risk-free interest rate $(r = 0.0139)$, volatility $(\sigma = 0.2085)$, and size of time-step $(\Delta t = 1/365)$. Some other parameter values are listed in Table 2.

TABLE 2. Early redemption dates, strike percentages, and coupon rates for the step-down ELS.

Redemption date	$T_1 = 0.5$	$T_2 = 1$	$T_3 = 1.5$	$T_4 = 2$	$T_5 = 2.5$	$T_6 = 3$
Strike percentage	$K_1 = 95$	$K_2 = 95$	$K_3 = 95$	$K_4 = 90$	$K_5 = 90$	$K_6 = 90$
Coupon rate	$c_1 = 0.025$	$c_2 = 0.05$	$c_3 = 0.075$	$c_4 = 0.1$	$c_5 = 0.125$	$c_6 = 0.15$

The MCS algorithm for two-asset ELS is described in Algorithm 1.

Algorithm 1 MCS algorithm for two-asset ELS

Require: Set initial price $(S_{1_0},\ S_{2_0})$,time-step t, maturity T, the number of checking days N_c, the number of sample paths N_m, the number of total time steps N_T, time-step size $\Delta t = T/N_T$, face value F, volatility of underlying assets (σ_1, σ_2), covariance matrix Σ, risk-free interest rate r, early redemption dates T_i, coupon rates c_i for early and final redemptions, strike percentages K_i, dummy d, and knock-in barrier KI. Set payoff $M_i = 0$, $X(t) = S_1(t)/S_{1_0}$ and $Y(t) = S_2(t)/S_{2_0}$. Here $0 \le j \le N_T - 1$, $1 \le i \le N_c$ and $T_0 = 0$.

▷ Cholesky decomposition computation of (2×2) covariance matrix Σ, C is upper triangular matrix
$C = chol(\Sigma)$
for $k = 1$ to N_m **do**
  ▷ $(N_T, 2) \times (2, 2)$ matrix multiplication
  $\begin{pmatrix} W_1, & W_2 \end{pmatrix} = \begin{pmatrix} Z_1, & Z_2 \end{pmatrix} \times C,\ Z_1, Z_2 \sim N(0,1)$
  ▷ Generate stock path for t_j
  for $j = 0$ to $N_T - 1$ **do**
    $X(t_{j+1}) = X(t_j)\exp((r - 0.5\sigma_1^2)\Delta t + \sigma_1\sqrt{\Delta t}W_{j_1}),\ W_{j_1} \sim N(0,1)$
    $Y(t_{j+1}) = Y(t_j)\exp((r - 0.5\sigma_2^2)\Delta t + \sigma_2\sqrt{\Delta t}W_{j_2}),\ W_{j_2} \sim N(0,1)$
  end for
  ▷ Check minimum
  $R = \min(X, Y)$
  ▷ Check the value of the stock path at checking days
  if $R(T_1) \ge K_1$ **then** $M_1 = M_1 + (1 + c_1)F$
  else if $R(T_2) \ge K_2$ **then** $M_2 = M_2 + (1 + c_2)F$
  ⋮
  else if $R(T_{N_c}) \ge K_{N_c}$ **then** $M_{N_c} = M_{N_c} + (1 + c_{N_c})F$
  else if $\min_{1 \le j \le N_T} \{R(t_j)\} \le KI$ **then** $M_{N_c} = M_{N_c} + R(t_{N_T})$
  else
    $M_{N_c} = M_{N_c} + (1 + d)F$
  end if
end for
▷ Take average and discount to present value.
$V^0 = \sum_{i=1}^{N_c} e^{-rT_i} M_i / N_m$

Figure 2 shows an Android application screen for pricing 2-asset ELS using MCS. Please refer to the previous reference [5] for a detailed implementation of mobile platform.

The MCS algorithm for three-asset ELS is described in Algorithm 2.

Figure 3 shows an Android application screen for pricing 3-asset ELS using MCS.

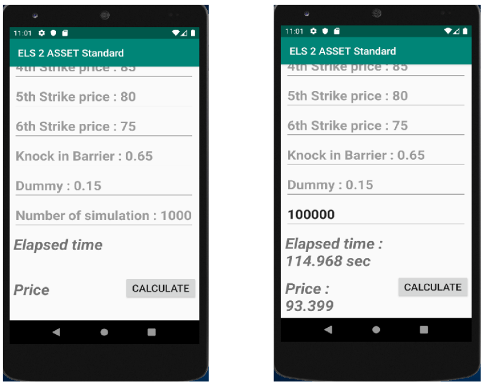

FIGURE 2. Android application screen for pricing 2-asset ELS using MCS. Left and right display before and after the computation, respectively.

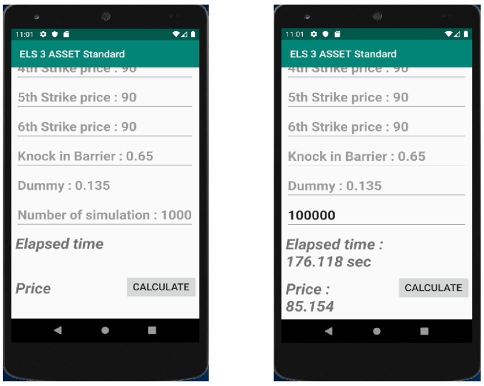

FIGURE 3. Android application screen for pricing 3-asset ELS using MCS. Left and right display before and after the computation, respectively.

3. NUMERICAL EXPERIMENT

We present computational tests such as the convergence test and option pricing for two-and three-asset ELS. All computations are run in Samsung Galaxy S9 on a quad 2.7GHz Octa-core with 4 GB RAM.

3.1. **ELS with two underlying assets.** Because most of the ELS products traded in Korea have two or three underlying assets, first, we use the proposed algorithm to calculate the price of ELS product with two underlying assets. To calculate the price of an ELS product made up of two underlying assets, we need to use a Cholesky factorization to generate a random number with correlation. We generate correlated random numbers Z_1^*, Z_2^* from a standard bivariate normal distribution using Cholesky factorization [14, 20]:

$$Z_1^* = Z_1,\ Z_2^* = \rho Z_1 + \sqrt{1-\rho^2} Z_2,$$

Algorithm 2 MCS algorithm for three-asset ELS

Require: Set initial price (S_{1_0}, S_{2_0}, S_{3_0}) ,time-step t, maturity T, the number of checking days N_c, the number of sample paths N_m, the number of total time steps N_T, time-step size $\Delta t = T/N_T$, face value F, volatility of underlying assets (σ_1, σ_2, σ_3), covariance matrix Σ, risk-free interest rate r, early redemption dates T_i, coupon rates c_i for early and final redemptions, strike percentages K_i, dummy d, and knock-in barrier KI. Set payoff $M_i = 0$, $X(t) = S_1(t)/S_{1_0}$, $Y(t) = S_2(t)/S_{2_0}$ and $Z(t) = S_3(t)/S_{3_0}$. Here $0 \le j \le N_T - 1$, $1 \le i \le N_c$ and $T_0 = 0$.

▷ Cholesky decomposition computation of (3×3) covariance matrix Σ, C is upper triangular matrix
$C = chol(\Sigma)$
for $k = 1$ to N_m **do**
  ▷ $(N_T, 3) \times (3, 3)$ matrix multiplication
  $\begin{pmatrix} W_1, & W_2, & W_3 \end{pmatrix} = \begin{pmatrix} Z_1, & Z_2, & Z_3 \end{pmatrix} \times C,\ Z_1, Z_2, Z_3 \sim N(0,1)$
  ▷ Generate stock path for t_j
  for $j = 0$ to $N_T - 1$ **do**
    $X(t_{j+1}) = X(t_j)\exp((r - 0.5\sigma_1^2)\Delta t + \sigma_1\sqrt{\Delta t}W_{j_1}),\ W_{j_1} \sim N(0,1)$
    $Y(t_{j+1}) = Y(t_j)\exp((r - 0.5\sigma_2^2)\Delta t + \sigma_2\sqrt{\Delta t}W_{j_2}),\ W_{j_2} \sim N(0,1)$
    $Z(t_{j+1}) = Z(t_j)\exp((r - 0.5\sigma_3^2)\Delta t + \sigma_3\sqrt{\Delta t}W_{j_3}),\ W_{j_3} \sim N(0,1)$
  end for
  ▷ Check minimum
  $R = \min(X, Y, Z)$
  ▷ Check the value of the stock path at checking days
  if $R(T_1) \ge K_1$ **then** $M_1 = M_1 + (1 + c_1)F$
  else if $R(T_2) \ge K_2$ **then** $M_2 = M_2 + (1 + c_2)F$
  ⋮
  else if $R(T_{N_c}) \ge K_{N_c}$ **then** $M_{N_c} = M_{N_c} + (1 + c_{N_c})F$
  else if $\min_{1 \le j \le N_T} \{R(t_j)\} \le KI$ **then** $M_{N_c} = M_{N_c} + R(t_{N_T})$
  else
    $M_{N_c} = M_{N_c} + (1 + d)F$
  end if
end for
▷ Take average and discount to present value.
$V^0 = \sum_{i=1}^{N_c} e^{-rT_i} M_i / N_m$

where Z_1 and Z_2 are independent standard normal distribution. Here, ρ is the correlation coefficient between the two underlying assets. We generate the two correlated asset paths using the following formulas:

$$
\begin{aligned}
X_1(t_{i+1}) &= X_1(t_i)e^{(r-0.5\sigma_1^2)\Delta t + \sigma_1\sqrt{\Delta t}Z_{1i}^*}, \\
X_2(t_{i+1}) &= X_2(t_i)e^{(r-0.5\sigma_2^2)\Delta t + \sigma_2\sqrt{\Delta t}Z_{2i}^*}.
\end{aligned}
$$

Next, we define the worst performer ($WP(t_i)$) of the two asset paths:

$$
WP(t_i) = \min(X_1(t_i), X_2(t_i)) \tag{3.1}
$$

Figure 4 shows that ELS price converges with the number of samples in two-asset ELS. For each number of samples, we plot 100 simulation results. As the number of samples increases, we can observe the convergence.

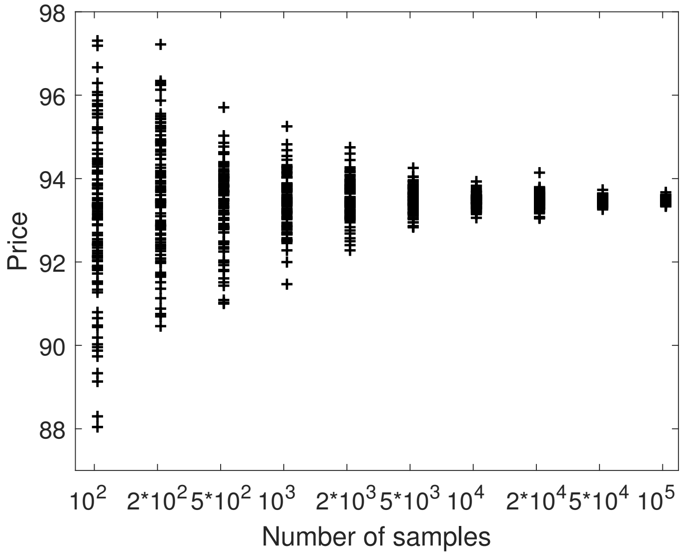

FIGURE 4. ELS price versus the number of samples. Here, we plot 100 simulation results for each case.

Figure 5 presents that elapsed time of pricing ELS increases in proportion to the number of samples. Table 3 shows ELS price and elapsed time for two-asset ELS.

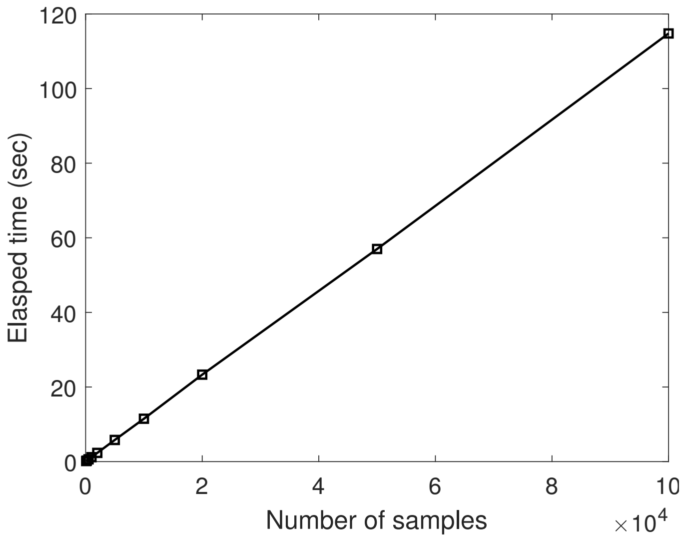

FIGURE 5. Elapsed time of pricing ELS versus the number of samples.

TABLE 3. Comparison of the elapsed time (in seconds) for two-asset with strike prices $K_1 = 90,\ K_2 = 90,\ K_3 = 85,\ K_4 = 85,\ K_5 = 80,\ K_6 = 75$, knock-in barrier $KI = 65$, volatilities $\sigma_1 = 0.2490,\ \sigma_2 = 0.2182$, the correlation coefficient $\rho = 0.0981$, and the risk-free interest free $r = 0.0165$.

M	10^2	10^3	10^4	10^5
MCS	93.1256	93.4251	93.4644	93.4757
Elapsed time	0.1579	1.1903	11.4999	114.7761

3.2. **ELS with three underlying assets.** Next, we calculate an ELS product with three underlying assets. An ELS product consisting of three underlying assets can be calculated by using Cholesky factorization. We can generate correlated random numbers Z_1^*, Z_2^*, Z_3^* from a standard multivariate normal distribution using Cholesky factorization [14, 20]:

$$
\begin{aligned}
Z_1^* &= Z_1,\ Z_2^* = \rho_{12} Z_1 + \sqrt{1-\rho_{12}^2} Z_2, \\
Z_3^* &= \rho_{13} Z_1 + \frac{(\rho_{23} - \rho_{12}\rho_{13})}{\sqrt{1-\rho_{12}^2}} Z_2 + \sqrt{1 - \rho_{13}^2 - \frac{(\rho_{23} - \rho_{12}\rho_{13})^2}{1-\rho_{12}^2}} Z_3,
\end{aligned}
$$

where Z_1,Z_2,Z_3 are independent standard normal distribution. Here, ρ_{12}, ρ_{13}, and ρ_{23} are the correlation coefficients among the three underlying assets. We create the three correlated asset paths using the following formulas:

$$
\begin{aligned}
X_1(t_{i+1}) &= X_1(t_i) e^{(r-0.5\sigma_1^2)\Delta t + \sigma_1 \sqrt{\Delta t} Z_{1i}^*}, \\
X_2(t_{i+1}) &= X_2(t_i) e^{(r-0.5\sigma_2^2)\Delta t + \sigma_2 \sqrt{\Delta t} Z_{2i}^*}, \\
X_3(t_{i+1}) &= X_3(t_i) e^{(r-0.5\sigma_3^2)\Delta t + \sigma_3 \sqrt{\Delta t} Z_{3i}^*}.
\end{aligned}
$$

Then, we define the worst performer($WP(t_i)$) among three asset paths:

$$
WP(t_i) = \min(X_1(t_i), X_2(t_i), X_3(t_i)). \tag{3.2}
$$

Figure 6 shows that ELS price converges with the number of samples in three-asset. For each number of samples, we plot 100 simulation results. As the number of samples increases, we can observe the convergence.

Figure 7 presents that elapsed time of pricing ELS increases in proportion to the number of samples. Table 4 shows ELS price and elapsed time for three-asset ELS.

3.3. **Greeks.** In this section, we calculate Greeks of *delta* ($\Delta_1 = \partial V^0/\partial S_1$, $\Delta_2 = \partial V^0/\partial S_2$) of the two-asset ELS. To compute these Greeks, we apply the central finite difference approximation, i.e., $\Delta_1 \approx [V^0(S_1 + \Delta S, S_2) - V^0(S_1 - \Delta S, S_2)]/(2\Delta S)$ and $\Delta_2 \approx [V^0(S_1, S_2 + \Delta S) - V^0(S_1, S_2 - \Delta S)]/(2\Delta S)$, where V^0 is the ELS price, S_1 and S_2 are the underlying assets, and $\Delta S = 5$. Figure 8(a), (b), and (c) show the option price Δ_1, and Δ_2 of the ELS. The number of samples is $M = 2 \times 10^4$.

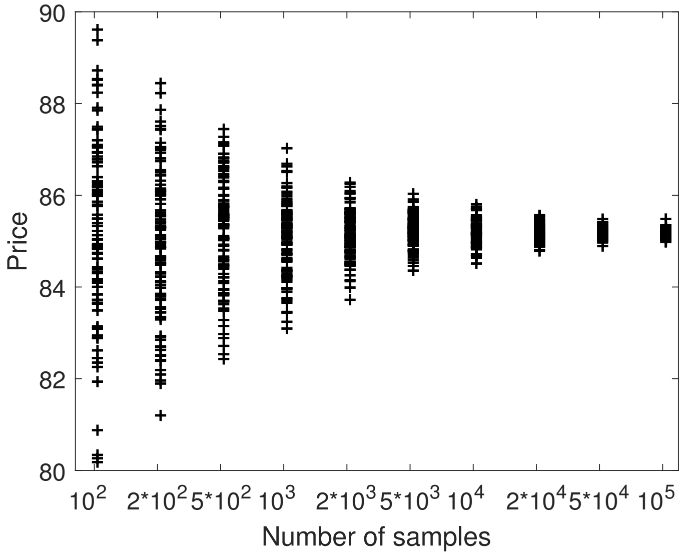

FIGURE 6. ELS price versus the number of samples. Here, we plot 100 simulation results for each case.

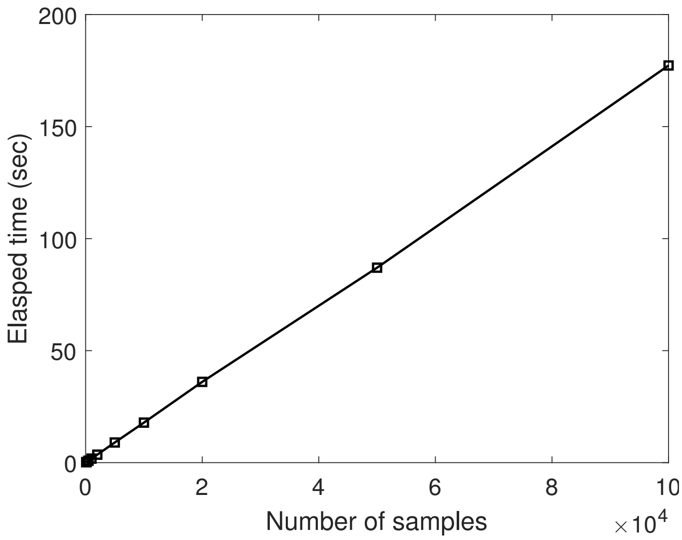

FIGURE 7. Elapsed time of pricing ELS versus the number of samples.

4. CONCLUSION

This paper presented extensions of mobile implementation for the previous one-asset ELS pricing to two-and three-asset ELS pricing because two-or three-asset ELS is more popular

TABLE 4. Comparison of the elapsed time (in seconds) for three-asset with strike prices $K_1 = 95,\ K_2 = 95,\ K_3 = 90,\ K_4 = 90,\ K_5 = 90,\ K_6 = 90$, knock-in barrier $KI = 65$, volatilities $\sigma_1 = 0.26620,\ \sigma_2 = 0.2105,\ \sigma_3 = 0.2111$, the correlation coefficient $\rho_{12} = 0.279,\ \rho_{13} = 0.2895,\ \rho_{23} = 0.5295$, and the risk-free interest free $r = 0.0139$.

M	10^2	10^3	10^4	10^5
MCS	85.2299	85.0656	85.1863	85.1816
Elapsed time	0.1758	1.7988	17.8713	177.2818

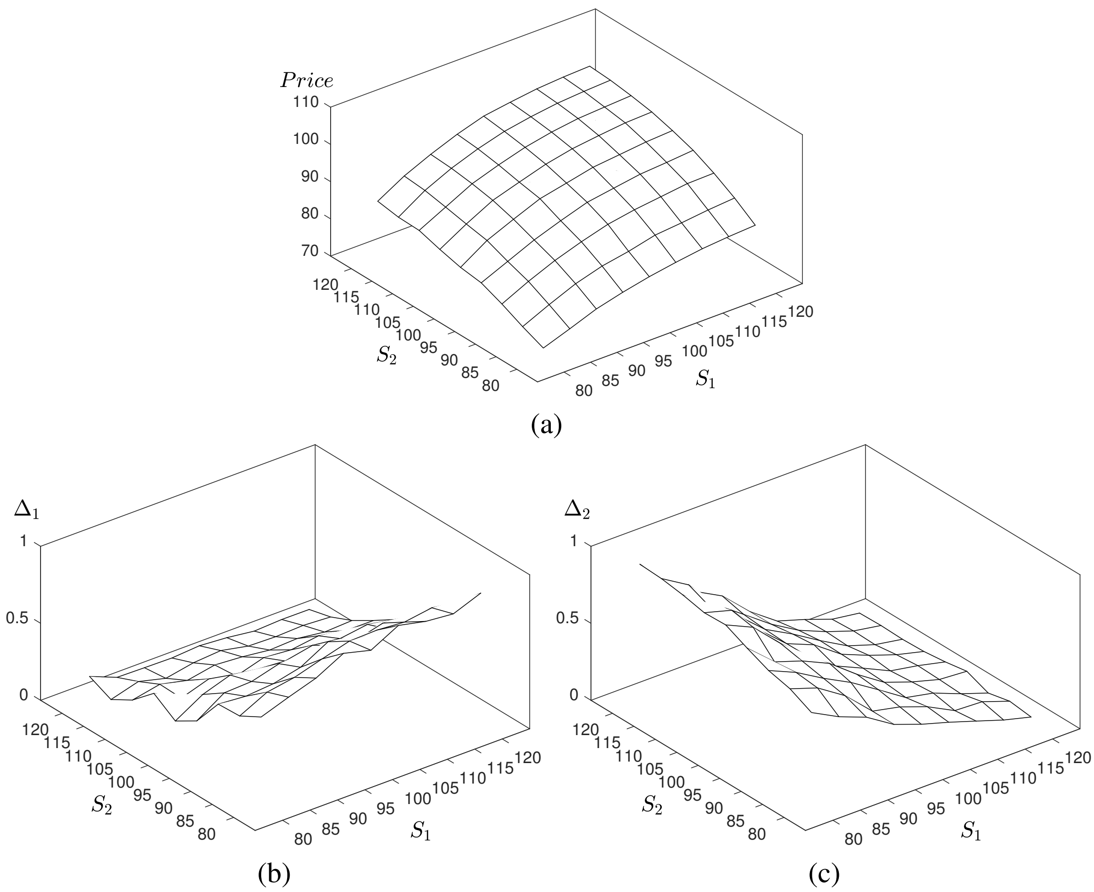

FIGURE 8. (a), (b), and (c) are the option price, Δ_1, and Δ_2, respectively.

than one-asset ELS in the real finance market. We performed standard convergence tests and option pricing for two-and three-asset ELS pricing. The computational results demonstrated the convergence of the mobile implementation. The mobile implementation of the multi-asset ELS pricing will be very useful in real financial market.

ACKNOWLEDGMENT

We are grateful to the reviewers for the constructive and helpful comments on the revision of this article. The corresponding author (J.S. Kim) expresses thanks for the support from the BK21 PLUS program.

APPENDIX

In this appendix, we provide an Android Studio source code for two-asset ELS pricing.

```
package com.example.els2assetmc;
import android.os.Bundle;
import android.os.Environment;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Random;
public class ELS2assetMC extends AppCompatActivity {
Random random = new Random();
Button Calculate_button;
int len = 6;
EditText[] Date_Edit = new EditText[len + 1];
EditText[] Coupon_Edit = new EditText[len];
EditText[] Strike_Edit = new EditText[len];
EditText Risk_free_rate_Edit;
EditText Volatility1_Edit;
EditText Volatility2_Edit;
EditText Correlation_Edit;
EditText Knock_in_Barrier_Edit;
EditText Dummy_Edit;
EditText NoSimulation_Edit;
TextView ElapsedTime_Result1;
TextView ElapsedTime_Result2;
TextView ElapsedTime_Result3;
TextView Price_Result;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
setTitle("ELS 2 ASSET Standard");
```

```
Date_Edit[0] = findViewById(R.id.Basedate);
Date_Edit[1] = findViewById(R.id.Redemption1);
Date_Edit[2] = findViewById(R.id.Redemption2);
Date_Edit[3] = findViewById(R.id.Redemption3);
Date_Edit[4] = findViewById(R.id.Redemption4);
Date_Edit[5] = findViewById(R.id.Redemption5);
Date_Edit[6] = findViewById(R.id.Maturity);
Coupon_Edit[0] = findViewById(R.id.Coupon1);
Coupon_Edit[1] = findViewById(R.id.Coupon2);
Coupon_Edit[2] = findViewById(R.id.Coupon3);
Coupon_Edit[3] = findViewById(R.id.Coupon4);
Coupon_Edit[4] = findViewById(R.id.Coupon5);
Coupon_Edit[5] = findViewById(R.id.Coupon6);
Strike_Edit[0] = findViewById(R.id.Strike1);
Strike_Edit[1] = findViewById(R.id.Strike2);
Strike_Edit[2] = findViewById(R.id.Strike3);
Strike_Edit[3] = findViewById(R.id.Strike4);
Strike_Edit[4] = findViewById(R.id.Strike5);
Strike_Edit[5] = findViewById(R.id.Strike6);
Risk_free_rate_Edit = findViewById(R.id.Risk_free_rate);
Volatility1_Edit = findViewById(R.id.Volatility1);
Volatility2_Edit = findViewById(R.id.Volatility2);
Correlation_Edit = findViewById(R.id.Correlation);
Knock_in_Barrier_Edit = findViewById(R.id.Knock_in_Barrier);
Dummy_Edit = findViewById(R.id.Dummy);
NoSimulation_Edit = findViewById(R.id.NoSimulation);
ElapsedTime_Result3 = findViewById(R.id.ElapsedTime3);
Price_Result = findViewById(R.id.Price);
Calculate_button = findViewById(R.id.Calculate);
Calculate_button.setOnClickListener(new View.OnClickListener() {
public void onClick(View arg0) {
long start1 = System.currentTimeMillis();
long start3 = System.currentTimeMillis();
int M, tot_date;
int check_dayInt[] = new int[len];
long[] check_day = new long[len];
double r, vol1, vol2, Knock_in_Barrier, dummy, ran1, ran2, corr;
double dt = 1.0 / 365;
double sum = 0.0;
double[] coupon_rate = new double[len];
double[] strike_price = new double[len];
double[] S1;
double[] S2;
double[] indexs = new double[len];
double[] payoff;
double[] tot_payoff = new double[len];
double[] disc_payoff = new double[len];
double[] payment = new double[len];
String[] Date = new String[len + 1];
String[] Coupon_String = new String[len];
String[] Strike_String = new String[len];
String Risk_free_rate_String, Knock_in_Barrier_String;
```

```
String Volatility1_String, Volatility2_String;
String Correlation_String;
String Dummy_String, NoSimulation_String;
for (int i = 0; i < len + 1; i++) {
Date[i] = Date_Edit[i].getText().toString(); }
for (int i = 0; i < len; i++) {
Coupon_String[i] = Coupon_Edit[i].getText().toString();
Strike_String[i] = Strike_Edit[i].getText().toString();}
Risk_free_rate_String = Risk_free_rate_Edit.getText().toString();
Volatility1_String = Volatility1_Edit.getText().toString();
Volatility2_String = Volatility2_Edit.getText().toString();
Correlation_String = Correlation_Edit.getText().toString();
Knock_in_Barrier_String=Knock_in_Barrier_Edit.getText().toString();
Dummy_String = Dummy_Edit.getText().toString();
NoSimulation_String = NoSimulation_Edit.getText().toString();
if (Date[0].trim().equals("")) { Date[0] = "20180629"; }
if (Date[1].trim().equals("")) { Date[1] = "20181221"; }
if (Date[2].trim().equals("")) { Date[2] = "20190625"; }
if (Date[3].trim().equals("")) { Date[3] = "20191223"; }
if (Date[4].trim().equals("")) { Date[4] = "20200624"; }
if (Date[5].trim().equals("")) { Date[5] = "20201223"; }
if (Date[6].trim().equals("")) { Date[6] = "20210624"; }
if (Coupon_String[0].trim().equals("")) { coupon_rate[0] = 0.025; }
else { coupon_rate[0] = Double.parseDouble(Coupon_String[0]); }
if (Coupon_String[1].trim().equals("")) { coupon_rate[1] = 0.050; }
else { coupon_rate[1] = Double.parseDouble(Coupon_String[1]); }
if (Coupon_String[2].trim().equals("")) { coupon_rate[2] = 0.075; }
else { coupon_rate[2] = Double.parseDouble(Coupon_String[2]); }
if (Coupon_String[3].trim().equals("")) { coupon_rate[3] = 0.100; }
else { coupon_rate[3] = Double.parseDouble(Coupon_String[3]); }
if (Coupon_String[4].trim().equals("")) { coupon_rate[4] = 0.125; }
else { coupon_rate[4] = Double.parseDouble(Coupon_String[4]); }
if (Coupon_String[5].trim().equals("")) { coupon_rate[5] = 0.150; }
else { coupon_rate[5] = Double.parseDouble(Coupon_String[5]); }
if (Strike_String[0].trim().equals("")) { strike_price[0] = 90; }
else { strike_price[0] = Double.parseDouble(Strike_String[5]); }
if (Strike_String[1].trim().equals("")) { strike_price[1] = 90; }
else { strike_price[1] = Double.parseDouble(Strike_String[1]); }
if (Strike_String[2].trim().equals("")) { strike_price[2] = 85; }
else { strike_price[2] = Double.parseDouble(Strike_String[2]); }
if (Strike_String[3].trim().equals("")) { strike_price[3] = 85; }
else { strike_price[3] = Double.parseDouble(Strike_String[3]); }
if (Strike_String[4].trim().equals("")) { strike_price[4] = 80; }
else { strike_price[4] = Double.parseDouble(Strike_String[4]); }
if (Strike_String[5].trim().equals("")) { strike_price[5] = 75; }
else { strike_price[5] = Double.parseDouble(Strike_String[5]); }
if (Risk_free_rate_String.trim().equals("")) { r = 0.0165; }
else { r = Double.parseDouble(Risk_free_rate_String); }
if (Volatility1_String.trim().equals("")) { vol1 = 0.2490; }
else { vol1 = Double.parseDouble(Volatility1_String); }
if (Volatility2_String.trim().equals("")) { vol2 = 0.2182; }
else { vol2 = Double.parseDouble(Volatility2_String); }
```

```
if (Correlation_String.trim().equals("")) { corr = 0.0981; }
else { corr = Double.parseDouble(Correlation_String); }
if (Knock_in_Barrier_String.trim().equals("")) {
Knock_in_Barrier = 0.65 * 100; }
else { Knock_in_Barrier = Double.parseDouble(
Knock_in_Barrier_String) * 100; }
if (Dummy_String.trim().equals("")) { dummy = 0.150; }
else { dummy = Double.parseDouble(Dummy_String); }
if (NoSimulation_String.trim().equals("")) { M = 1000; }
else { M = Integer.parseInt(NoSimulation_String); }
try {for (int i = 0; i < len; i++) {
check_day[i] = diffOfDate(Date[0], Date[i+1]);
check_dayInt[i] = (int) check_day[i];}
} catch (Exception z) {
z.printStackTrace();}
tot_date = check_dayInt[len - 1];
S1 = new double[tot_date + 1];
S2 = new double[tot_date + 1];
S1[0] = 100.0;
S2[0] = 100.0;
double[] arr_ran1 = new double[tot_date];
double[] arr_ran2 = new double[tot_date];
double coef11 = (r - Math.pow(vol1, 2) / 2) * dt;
double coef12 = vol1 * Math.sqrt(dt);
double coef21 = (r - Math.pow(vol2, 2) / 2) * dt;
double coef22 = vol2 * Math.sqrt(dt);
for (int i = 0; i < len; i++) {
payment[i] = S1[0] * (1.0 + coupon_rate[i]); }
long end1 = System.currentTimeMillis();
long start2 = System.currentTimeMillis();
for (int j = 0; j < M; j++) {
int repay_event = 0;
double minn = 100.0;
for (int i = 0; i < arr_ran1.length; i++) {
ran1 = random.nextGaussian();
ran2 = random.nextGaussian();
arr_ran1[i] = ran1;
arr_ran2[i]=corr * ran1 + Math.sqrt(1 - Math.pow(corr, 2)) * ran2;
S1[i + 1] = S1[i] * Math.exp(coef11 + coef12 * arr_ran1[i]);
S2[i + 1] = S2[i] * Math.exp(coef21 + coef22 * arr_ran2[i]);
if (Math.min(S1[i + 1], S2[i + 1]) < minn) {
minn = Math.min(S1[i + 1], S2[i + 1]); }}
for (int h = 0; h < len; h++) {
indexs[h] = Math.min(S1[check_dayInt[h]], S2[check_dayInt[h]]); }
payoff = new double[len];
for (int n = 0; n < len; n++) {
if (indexs[n] >= strike_price[n]) {
payoff[n] = payment[n];
repay_event = 1;
break;}}
if (repay_event == 0) {
if (minn > Knock_in_Barrier) {
```

```
payoff[payoff.length - 1] = S1[0] * (1 + dummy); }
else { payoff[payoff.length - 1] = indexs[len - 1]; }}
for (int k = 0; k < len; k++) {
tot_payoff[k] = tot_payoff[k] + payoff[k]; }}
for (int i = 0; i < len; i++) {
tot_payoff[i] = tot_payoff[i] / M;
disc_payoff[i]=tot_payoff[i] * Math.exp(-r * check_day[i] / 365.0);
sum += disc_payoff[i];}
long end2 = System.currentTimeMillis();
long end3 = System.currentTimeMillis();
ElapsedTime_Result3.setText
("Elapsed time:\n"+(end3-start3)/1000.0+"sec");
Price_Result.setText("Price:\n"+Math.round(sum*1000.0)/1000.0);}});}
public long diffOfDate(String begin, String end) throws Exception {
SimpleDateFormat formatter = new SimpleDateFormat("yyyyMMdd");
Date beginDate = formatter.parse(begin);
Date endDate = formatter.parse(end);
long diff = endDate.getTime() - beginDate.getTime();
long diffDays = diff / (24 * 60 * 60 * 1000);
return diffDays;}}
```

REFERENCES

[1] D. Gabor and S. Brooks, *The digital revolution in financial inclusion: international development in the fintech era*, New Polit. Econ., **22**(4) (2017), 423–436.

[2] P. Gomber, J.A. Koch, and M. Siering, *Digital Finance and FinTech: current research and future research directions*, J. Bus. Econ., **87**(5) (2017), 537–580.

[3] P. Gomber, R.J. Kauffman, C. Parker, and B.W. Weber, *On the fintech revolution: interpreting the forces of innovation, disruption, and transformation in financial services*, J. Manag. Inf. Syst., **35**(1) (2018), 220–265.

[4] K. Gai, M. Qiu, and X. Sun, *A survey on FinTech*, J. Netw. Comp. Appl., **103** (2018), 262–273.

[5] W. Jian, J. Ban, J. Han, S. Lee, and D. Jeong, *Mobile platform for pricing of Equity-Linked Securities*, J. Korean Soc. Ind. Appl. Math., **21**(3) (2017), 181–202.

[6] S.G. Kou, *A jump-diffusion model for option pricing*, Manag. Sci., **48**(8) (2002), 1086–1101.

[7] J.D. Hamilton, *A new approach to the economic analysis of nonstationary time series and the business cycle*, Econometrica, **57**(2) (1989), 357–384.

[8] S.S. Clift and P.A. Forsyth, *Numerical solution of two asset jump diffusion models for option valuation*, Appl. Numer. Math., **58**(6) (2008), 743–782.

[9] D.J. Duffy. Finite Difference methods in financial engineering: a Partial Differential Equation approach, *John Wiley and Sons*, 2013.

[10] A.Q.M. Khaliq, D.A. Voss, and K. Kazmi , *Adaptive θ-methods for pricing American options*, J. Comput. Appl. Math., **222**(1) (2008), 210–227.

[11] Z. Cen, A. Le, and A. Xu, *Finite difference scheme with a moving mesh for pricing Asian options*, App.l Math. Comput., **219**(16) (2013), 8667–8675.

[12] D. Jeong, J. Kim, and I.S. Wee, *An accurate and efficient numerical method for Black-Scholes equations*, Commun. Korean Math. Soc., **24**(4) (2009), 617–628.

[13] S. Foulon, *ADI finite difference schemes for option pricing in the Heston model with correlation*, Int. J. Numer. Anal. Model., **7**(2) (2010), 303–320.

[14] P. Glasserman, Monte Carlo methods in financial engineering. *Springer Science & Business Media*, 2013.

[15] P.P. Boyle, *Options: A monte carlo approach*, J. Fin. Econ., **4**(3) (1977), 323–338.

[16] P. Boyle, M. Broadie, and P. Glasserman, *Monte Carlo methods for security pricing*, J. Econ. Dyn. Control, **21** (1997), 1267–1321.
[17] C.P. Fries and M.S. Joshi. Conditional analytic Monte-Carlo pricing scheme of auto-callable products, *Centre for Actuarial Studies, Department of Economics, University of Melbourne*, 2008.
[18] J.E. Handschin, *Monte Carlo techniques for prediction and filtering of non-linear stochastic processes*, Automatica, **6**(4) (1970), 555–563.
[19] H.L. Koh and S.Y. Teh, *Learning Black Scholes option pricing the fun way via mobile apps*, Proc. 2013 IEEE Int. Conf. Teaching, Assessm. Learn. Eng. (TALE), 192–195.
[20] J. Wang and C. Liu, *Generating multivariate mixture of normal distributions using a modified Cholesky decomposition*, Proc. 38th Conf. Winter Simul., 2006.