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

HANBYEOL JANG¹, HYUNSOO HAN¹, HAYEON PARK¹, WONJIN LEE¹, JISANG LYU², JINTAE PARK², HYUNDONG KIM², CHAEYOUNG LEE², SANGKWON KIM², YONGHO CHO¹, AND JUNSEOK KIM²†

¹DEPARTMENT OF FINANCIAL ENGINEERING, KOREA UNIVERSITY, SEOUL 02841, REPUBLIC OF KOREA
²DEPARTMENT OF MATHEMATICS, KOREA UNIVERSITY, SEOUL 02841, REPUBLIC OF KOREA
Email address: cfdkim@korea.ac.kr
³DEPARTMENT 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].
TABLE 1. Data for ELS. The values inside parentheses indicate percentages.

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

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.

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.

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

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

**Algorithm 1** MCS algorithm for two-asset ELS

**Require:** Set initial price $(S_{10}, S_{20})$, 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 ($\sigma_1$, $\sigma_2$), covariance matrix $\Sigma$, 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 $M_0 = 0$, $X(t) = S_1(t)/S_{20}$ and $Y(t) = S_2(t)/S_{20}$. Here $0 \leq j \leq N_T - 1, 1 \leq i \leq N_c$ and $T_0 = 0$.  

▷ Cholesky decomposition computation of $(2 \times 2)$ covariance matrix $\Sigma$, $C$ is upper triangular matrix $C = \text{chol}(\Sigma)$

for $k = 1$ to $N_m$ do 
  ▷ $(N_T, 2) \times (2, 2)$ matrix multiplication
  $(W_1, \ W_2) = (Z_1, \ Z_2) \times C, \ Z_1, Z_2 \sim N(0, 1)$
  ▷ Generate stock path for $t_j$
  for $j = 0$ to $N_T - 1$ do
    $X(t_{j+1}) = X(t_j) \exp((r - 0.5\sigma_1^2)\Delta t + \sigma_1\sqrt{\Delta t}W_{j1}), \ W_{j1} \sim N(0, 1)$
    $Y(t_{j+1}) = Y(t_j) \exp((r - 0.5\sigma_2^2)\Delta t + \sigma_2\sqrt{\Delta t}W_{j2}), \ W_{j2} \sim N(0, 1)$
  end for
  ▷ Check minimum
  $R = \min(X, Y)$
  ▷ Check the value of the stock path at checking days
  if $R(T_1) \geq K_1$ then $M_1 = M_1 + (1 + c_1)F$
  else if $R(T_2) \geq K_2$ then $M_2 = M_2 + (1 + c_2)F$
  :...
  else if $R(T_{N_c}) \geq K_{N_c}$ then $M_{N_c} = M_{N_c} + (1 + c_{N_c})F$
  else if $\min_{1 \leq j \leq N_T} \{R(t_j)\} \leq KI$ 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.
$V^0 = \sum_{i=1}^{N_r} e^{-rT_i} M_i/N_m$

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.
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]:

$$Z_1^* = Z_1, \quad Z_2^* = \rho Z_1 + \sqrt{1 - \rho^2} Z_2,$$
Algorithm 2 MCS algorithm for three-asset ELS

Require: Set initial price \((S_{10}, S_{20}, S_{30})\), 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 rate \(F\), volatility of underlying assets \((\sigma_1, \sigma_2, \sigma_3)\), covariance matrix \(\Sigma\), 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 \(M_t = 0\), \(X(t) = S_1(t)/S_{10}, Y(t) = S_2(t)/S_{20}\) and \(Z(t) = S_3(t)/S_{30}\). Here \(0 \leq j \leq N_T - 1\), \(1 \leq i \leq N_c\) and \(T_0 = 0\).

▷ Cholesky decomposition computation of \((3 \times 3)\) covariance matrix \(\Sigma\), \(C\) is upper triangular matrix \(C = \text{cho}(\Sigma)\)

\[
\text{for } k = 1 \text{ to } N_m \text{ do}
\]

▷ \((N_T, 3) \times (3, 3)\) matrix multiplication
\[
(W_1, \ W_2, \ W_3) = (Z_1, \ Z_2, \ Z_3) \times C, \ Z_1, Z_2, Z_3 \sim N(0, 1)
\]

▷ Generate stock path for \(t_j\)

\[
\text{for } j = 0 \text{ to } N_T - 1 \text{ do}
\]

\[
X(t_{j+1}) = X(t_j) \exp((r - 0.5\sigma_1^2)\Delta t + \sigma_1\sqrt{\Delta t}W_{j1}), \ W_{j1} \sim N(0, 1)
\]

\[
Y(t_{j+1}) = Y(t_j) \exp((r - 0.5\sigma_2^2)\Delta t + \sigma_2\sqrt{\Delta t}W_{j2}), \ W_{j2} \sim N(0, 1)
\]

\[
Z(t_{j+1}) = Z(t_j) \exp((r - 0.5\sigma_3^2)\Delta t + \sigma_3\sqrt{\Delta t}W_{j3}), \ W_{j3} \sim N(0, 1)
\]

end for

▷ Check minimum
\[
R = \min(X, Y, Z)
\]

▷ Check the value of the stock path at checking days
\[
\text{if } R(T_1) \geq K_1 \text{ then } M_1 = M_1 + (1 + c_1)F
\]
\[
\text{else if } R(T_2) \geq K_2 \text{ then } M_2 = M_2 + (1 + c_2)F
\]
\[
\vdots
\]
\[
\text{else if } R(T_{N_c}) \geq K_N \text{ then } M_{N_c} = M_{N_c} + (1 + c_{N_c})F
\]
\[
\text{else if } \min_{1 \leq j \leq N_T} \{R(t_j)\} \leq K_I \text{ then } M_{N_c} = M_{N_c} + R(t_{N_T})
\]
\[
\text{else}
\]
\[
M_{N_c} = M_{N_c} + (1 + d)F
\]

end if

end for

▷ Take average and discount to present value.
\[
V^0 = \sum_{i=1}^{N_c} e^{-rT_i} M_i/N_m
\]

\[
\text{where } Z_1 \text{ and } Z_2 \text{ are independent standard normal distribution. Here, } \rho \text{ is the correlation coefficient between the two underlying assets. We generate the two correlated asset paths using the following formulas:}
\]

\[
X_1(t_{i+1}) = X_1(t_i)e^{(r-0.5\sigma_1^2)\Delta t + \sigma_1\sqrt{\Delta t}Z_{1i}},
\]

\[
X_2(t_{i+1}) = X_2(t_i)e^{(r-0.5\sigma_2^2)\Delta t + \sigma_2\sqrt{\Delta t}Z_{2i}}.
\]

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

\[
WP(t_i) = \min(X_1(t_i), X_2(t_i))
\]

(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.
paths using the following formulas:

\[ \text{correlation coefficients among the three underlying assets.} \]

We create the three correlated asset

\[ \text{number of samples, we plot} \]

\[ \text{100} \]

\[ \text{Figure 6 shows that ELS price converges with the number of samples in three-asset. For each} \]

\[ \text{Greeks.} \]

\[ \text{3.3.} \]

\[ \text{Table 4 shows ELS price and elapsed time for three-asset ELS.} \]

\[ \text{can observe the convergence.} \]

\[ \Delta \]

\[ \partial V / \partial S_1, \Delta_2 = \partial V / \partial S_2 \]

**Table 3. Comparison of the elapsed time (in seconds) for two-asset with**

<table>
<thead>
<tr>
<th>( M )</th>
<th>( 10^2 )</th>
<th>( 10^4 )</th>
<th>( 10^4 )</th>
<th>( 10^5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>93.1256</td>
<td>93.4251</td>
<td>93.4644</td>
<td>93.4757</td>
</tr>
<tr>
<td>Elapsed time</td>
<td>0.1579</td>
<td>1.1903</td>
<td>11.4999</td>
<td>114.7761</td>
</tr>
</tbody>
</table>

\[ \text{3.2. ELS with three underlying assets.} \]

\[ \text{Next, we calculate an ELS product with three underlying assets.} \]

\[ \text{An ELS product consisting of three underlying assets can be calculated by} \]

\[ \text{using Cholesky factorization. We can generate correlated random numbers} \]

| \( Z_1^* = Z_1, Z_2^* = \rho_{12} Z_1 + \sqrt{1 - \rho_{12}^2} Z_2, \) |
| \( Z_3^* = \rho_{13} Z_1 + \sqrt{1 - \rho_{12}^2} \left( \frac{\rho_{23} - \rho_{12} \rho_{13}}{\sqrt{1 - \rho_{12}^2}} \right) Z_2 + \sqrt{1 - \rho_{12}^2 - \left( \frac{\rho_{23} - \rho_{12} \rho_{13}}{\sqrt{1 - \rho_{12}^2}} \right)^2} Z_3, \) |

\[ \text{where} \]

\[ \text{Z}_1,Z_2,Z_3 \]

\[ \text{are independent standard normal distribution. Here,} \]

\[ \text{ρ}_{12}, \text{ρ}_{13}, \text{and} \]

\[ \text{ρ}_{23} \]

\[ \text{are the correlation coefficients among the three underlying assets.} \]

\[ \text{We create the three correlated asset paths using the following formulas:} \]

\[ \text{X}_1(t_{i+1}) = X_1(t_i) e^{(r - 0.5 \sigma_1^2) \Delta t + \sigma_1 \sqrt{\Delta t} Z_1^*}, \]

\[ \text{X}_2(t_{i+1}) = X_2(t_i) e^{(r - 0.5 \sigma_2^2) \Delta t + \sigma_2 \sqrt{\Delta t} Z_2^*}, \]

\[ \text{X}_3(t_{i+1}) = X_3(t_i) e^{(r - 0.5 \sigma_3^2) \Delta t + \sigma_3 \sqrt{\Delta t} Z_3^*}. \]

\[ \text{Then, we define the worst performer(WP(t_i)) among three asset paths:} \]

\[ \text{WP(t_i) = min(X_1(t_i), X_2(t_i), X_3(t_i)).} \]

\[ (3.2) \]

\[ \text{Figure 6 shows that ELS price converges with the number of samples in three-asset. For each} \]

\[ \text{number of samples, we plot 100 simulation results. As the number of samples increases, we} \]

\[ \text{can observe the convergence.} \]

\[ \text{Figure 7 presents that elapsed time of pricing ELS increases in proportion to the number of} \]

\[ \text{samples. Table 4 shows ELS price and elapsed time for three-asset ELS.} \]

\[ \text{3.3. Greeks.} \]

\[ \text{In this section, we calculate Greeks of delta} \]

\[ \text{(Δ1 = ∂V/∂S_1, Δ2 = ∂V/∂S_2)} \]

\[ \text{of the two-asset ELS. To compute these Greeks, we apply the central finite difference approximation, i.e.,} \]

\[ \text{Δ1 ≈ [V^0(S_1 + ΔS, S_2) - V^0(S_1 - ΔS, S_2)]/(2ΔS) and Δ2 ≈ [V^0(S_1, S_2 +} \]

\[ \text{ΔS) - V^0(S_1, S_2 - ΔS)]/(2ΔS),} \]

\[ \text{where V^0 is the ELS price, S_1 and S_2 are the underlying assets, and ΔS = 5. Figure 8(a), (b), and (c) show the option price Δ1,} \]

\[ \text{and Δ2 of the ELS. The number of samples is} \]

\[ M = 2 \times 10^4. \]
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 $r = 0.0139$.

<table>
<thead>
<tr>
<th>$M$</th>
<th>$10^2$</th>
<th>$10^3$</th>
<th>$10^4$</th>
<th>$10^5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>85.2299</td>
<td>85.0656</td>
<td>85.1863</td>
<td>85.1816</td>
</tr>
<tr>
<td>Elapsed time</td>
<td>0.1758</td>
<td>1.7988</td>
<td>17.8713</td>
<td>177.2818</td>
</tr>
</tbody>
</table>

FIGURE 8. (a), (b), and (c) are the option price, $\Delta_1$, and $\Delta_2$, 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.

```java
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 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.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, v01, v02, 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[] indexes = 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[0]); }
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_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[1], S2[1]) < minn) {
        minn = Math.min(S1[1], S2[1]);
    }
    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_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[1], S2[1]) < minn) {
            minn = Math.min(S1[1], S2[1]);
        }
        for (int n = 0; n < len; n++) {
            if (indexes[n] >= strike_price[n]) {
                payoff = new double[len];
                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:
  +" + (end3-start3)/1000.0 + "sec");
Price_Result.setText("Price:
   +" + 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


