电子科技大学 经济与管理 学院

标 准 实 验 报 告

（实验）课程名称 金融衍生工具

电子科技大学教务处制表

电 子 科 技 大 学

实 验 报 告

学生姓名：于汇洋　　学 号：2018150801020　　指导教师： 夏 晖

实验地点： 经济与管理学院 A405　　实验时间：2 学时

一、实验室名称：　　A405

二、实验项目名称： 用隐式差分求欧式期权的价格

三、实验学时：2

四、实验原理：隐式差分方法（见附页）

五、实验目的：用隐式差分求欧式期权的价格

六、实验内容：见附页

七、实验器材（设备、元器件）：pc 一台，matlab 软件

八、实验步骤：见附页实验内容

九、实验数据及结果分析：见附页

十、实验结论：见附页

十一、总结及心得体会：见附页

十二、对本实验过程及方法、手段的改进建议：见附页

报告评分：

指导教师签字：

四、实验原理：隐式差分方法

在求解矩阵中的各个点的价格的过程中会用到以下方法：

根据式 $f(i+1,j)=a_j f(i,j-1)+b_j f(i,j)+c_j f(i,j+1)$

其中：

$$\begin{cases} a_j=\frac{1}{2}rj\Delta t-\frac{1}{2}\sigma^2 j^2\Delta t \\ b_j=1+r\Delta t+\sigma^2 j^2\Delta t \\ c_j=-\frac{1}{2}rj\Delta t+\frac{1}{2}\sigma^2 j^2\Delta t \end{cases}$$

对于所有的 i 时刻的 $f(i,j)$ 和所有的 i+1 时刻的 $f(i+1,j)$ 就会有如下关系：

$$\begin{pmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & a_3 & b_3 & c_3 & & \\ & & \cdots & \cdots & \cdots & \\ & & & a_{M-2} & b_{M-2} & c_{M-2} \\ & & & & a_{M-1} & b_{M-1} \end{pmatrix} \bullet \begin{pmatrix} f(i,1) \\ f(i,2) \\ f(i,3) \\ \cdots \\ f(i,M-2) \\ f(i,M-1) \end{pmatrix} = \begin{pmatrix} f(i+1,1)-a_1 f(i,0) \\ f(i+1,2) \\ f(i+1,3) \\ \cdots \\ f(i+1,M-2) \\ f(i+1,M-1)-c_{M-1} f(i,M) \end{pmatrix}$$

将上式写为：

$$M \bullet F_i = F_{i+1}$$

其中 M 可以构造出来，是已知的，F_{i+1} 可以由右侧的边界给出并依次求出，也是已知的。因此可以求得：

$$F_i = M^{-1} \bullet F_{i+1}$$

上下边界 $f(i,M)$，$f(i,0)$ 可以得到，这样重复进行到初始时刻，即可得到隐式差分法计算的美式看跌期权价格。

六、实验内容：

1. 题目：

隐式差分求期权：考虑标的物资产为某股票的欧式期权，股票的当前价格为 50，波动率为 40%，无风险利率为 5%，期权到期期限为一年。

到期日期权的现金流如下：

$$f_\tau = \begin{cases} 5, & S_T > 60 \\ \min\{\max(S_T - 50, 0).2\}, & 40 \le S_T \le 60 \\ 2, & S_T < 40 \end{cases}$$

求该欧式期权的价格，通过增加时间的阶段数 N 和股价的阶段数 M 来提高计算精度，并分析计算记过可能不收敛的原因，尝试画出初始时刻（t = 0）该期权价格的 Delta 随股票价格变化的图形。

八、实验步骤：

1. Matlab 代码如下：

一些函数和变量的声明如下：

```
function [f_mat,test1_price]=EurImp_Op(S_0,r,sigma,T,S_top,N,M)
%Function to calculate the price of a European option;
%Put or Call option using the implicit finite difference method;
%
%Inputs:__
%   :S_0 - stock price;
%   :r - risk free interest rate;
%   :sigma - volatility;
%   :T - date;
%   :S_top - the maxmum stock price we use;
%   :M - Stock price S divided into M segments;
%   :N - Time T divided into N segments.
%Outputs:__
%   :test1_price - Implicit european option price;
%   :f_mat - the Matrix generated by Implicit Difference Method.
```

设置 deltaS 和 deltaT，声明向量：

```
dS = S_top / M;
dT = T / N;
S = (0:dS:S_top)';
j_vector = 0:M;
i_vector = 0:N;
```

设置参数 abc：

```
%First we calculate the coefficients(a,b,c):
%in this case there is no dividend.
sigma2 = sigma * sigma;
a_j = (dT*j_vector/2).*(r - sigma2*j_vector);
b_j = 1 + dT*(sigma2*(j_vector.^2) + r);
c_j = -(dT*j_vector/2).*(r + sigma2*j_vector);
```

设置零矩阵：

```
%We Pre-set a zero matrix:
```

```
f_mat = zeros(M + 1,N + 1);
```

定义边界条件：

```
% Specify the boundary conditions:
% According to the question
price(S>60) = 5;
price(S<40) = 2;
price(S<=60&S>=40) = min(max(S(S>=40&S<=60)-50,0),2);
f_mat (:,end) = price;
f_mat (1,:) = 2*exp(-r*(N-i_vector)*dT);
f_mat (end,:) = 5*exp(-r*(N-i_vector)*dT);
```

执行实验内容中提到的步骤，设置一个三对角矩阵：

```
% Form the tridiagonal matrix
tri_mat = diag(a_j(3:M),-1) + diag(b_j(2:M)) + diag(c_j(2:M-1),1);
[L,U] = lu(tri_mat);
```

求解欧式期权的价格：

```
% Solve at each node
offset = zeros(size(tri_mat,2),1);
for idx = N:-1:1
    offset(1) = a_j(2)*f_mat(1,idx);
    offset(end) = c_j(end)*f_mat(end,idx);
    f_mat(2:M,idx) = U\(L\(f_mat(2:M,idx+1) - offset));
end
%then we calculate the option price we want:
test1_price =  interp1(S,f_mat(:,1),S_0);
```

画图

```
%plot a figure
delta = diff(f_mat(:,1)) / dS;
h = figure;
set(h,'color','w');
plot(S(1:end-1),delta)
```

九、实验数据及结果分析：

执行代码：

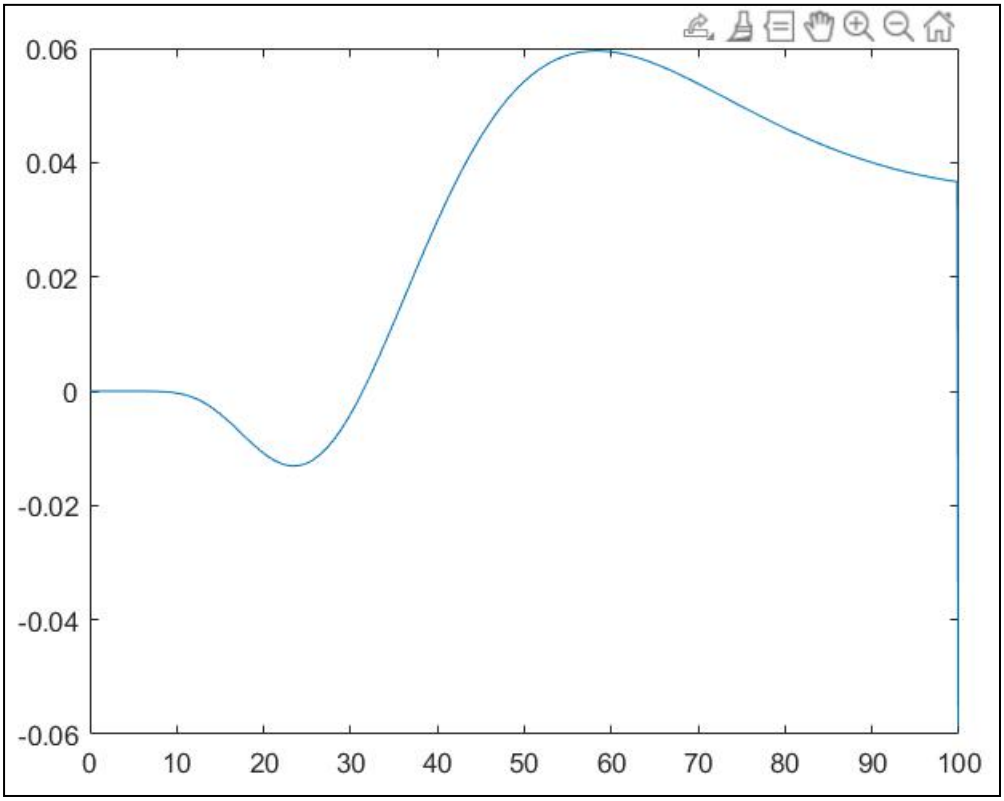
```
>> [test1_price,f_mat] = EurImp_Op(50,0.05,0.4,1,100,1000,1000)
```

结果如下：

```
f_mat =

    2.3059
```

结果生成的矩阵数据量很大无法呈现，但是在 1001 列符合题目中的到期股票价格的要求。

初始时刻（t = 0）该期权价格的 Delta 随股票价格变化的图形如下：



十、实验结论：

隐式差分方法可以求期权价格，是一种求解期权价格的方法。

十一、总结及心得体会：

隐式差分虽然在执行过程中要求解很多方程，但是避免了显示差分方法求解过程中股票和时间进行划分的区间个数对结果产生的影响。

十二、对本实验过程及方法、手段的改进建议：

1. 在实验过程中发现 delta 的图像在最后一个位置发生了突变，感觉很奇特。于是在网上找了一篇用 Python 求隐式差分法美式期权的方法，在上面进行更改。

2. 代码如下：

```
# Created with Python AI

# -*- coding: utf-8 -*-
"""
Created on Tue Dec 29 22:21:09 2020

@author: yuyuyu
"""
```

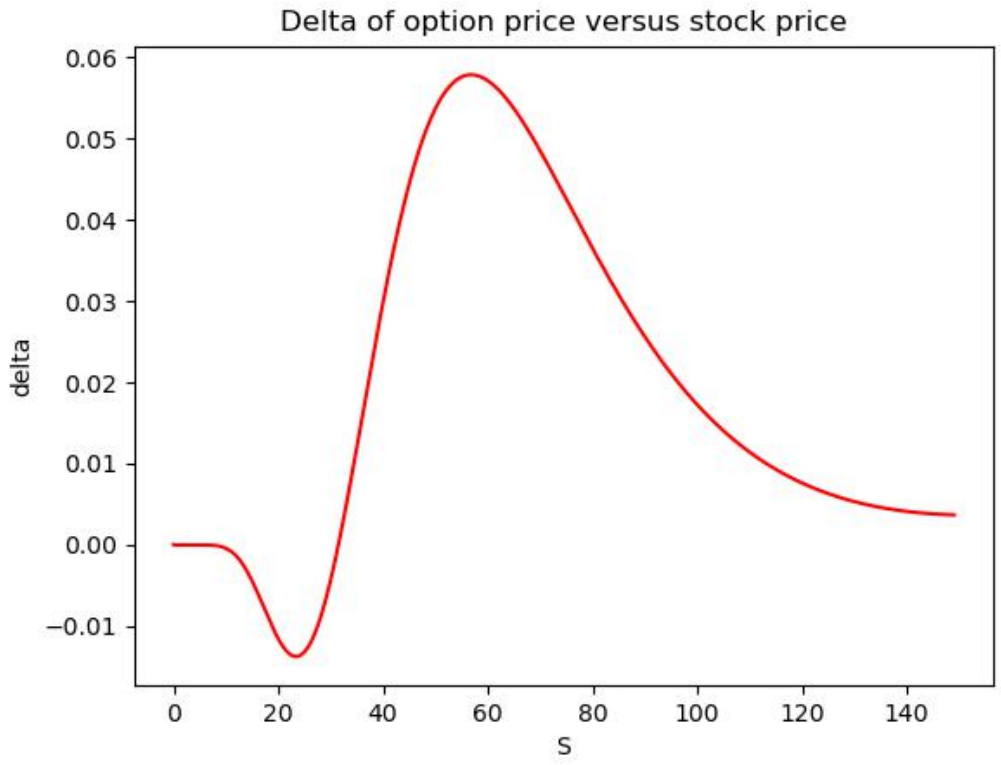
```
import numpy as np
import math
import matplotlib.pyplot as plt

def European_implicit(r, sigma, S_0,top, T, M, N):
    ds = top/M
    dt = T/N

    # 将 a_j, b_j, c_j 写为 3 个函数。
    a = lambda j: 0.5*r*j*dt-0.5*sigma*sigma*j*j*dt
    b = lambda j: 1+r*dt+sigma*sigma*j*j*dt
    c = lambda j: -0.5*r*j*dt-0.5*sigma*sigma*j*j*dt

    # f1 和 f2 为两列用来迭代计算的期权价格。
    f1 = [0]*(M+1);
    for i in range(M+1):
        if (i*ds < 40):
            f1[i] = 2
        elif (i*ds <= 60 and i*ds >=40):
            f1[i] = min(max(i*ds-50,0),2)
        elif (i*ds > 60):
            f1[i] = 5
    #print(f1)
    #print(len(f1))
    f2 = [None for i in range(M+1)]
    # coeffs 为上文中的 M 系数矩阵。
    coeffs = np.zeros((M-1, M-1))

    flag = 1
    for i in range(N-1, -1, -1):
        f2 = list(f1)
        coeffs[0][0] = b(1)
        coeffs[0][1] = c(1)
        coeffs[M-2][M-2] = b(M-1)
        coeffs[M-2][M-3] = a(M-1)
        for j in range(2, M-1, 1):
            coeffs[j-1][j-2] = a(j)
            coeffs[j-1][j-1] = b(j)
            coeffs[j-1][j] = c(j)
        # 参数矩阵求逆。
        coeffs_inv = np.linalg.inv(coeffs)
        F2 = f2[1:-1]
        #print(f2)
        #print(len(f2))
```

```
            #print(F2)
            #print(len(F2))
            F2[0] -= a(1)*f1[0]
            F2[M-2] -= c(M-2)*f1[M]
            F1 = np.matmul(coeffs_inv, F2)
            f1[1:M] = F1
            f1[0] = 2*math.e**(-r/N*flag)
            f1[M] = 5*math.e**(-r/N*flag)
            flag += 1
            # 判断是否执行美式看跌期权。
            #f1 = np.maximum(f1, K-np.linspace(0, M, M+1)*dS)
            print(f1)
        pos = int(S_0/ds)
        put_price = f1[pos] + (f1[pos+1]-f1[pos])/ds*(S_0-pos*ds)

        return put_price,f1

def plot_delta(f1,M,top):
    ds = top/M
    delta = [0]*(len(f1)-2)
    S = [0]*(M-1)
    for i in range(len(f1)-2):
        delta[i] = (f1[i+1]-f1[i])/ds
        S[i] = i*ds
    #print(f1)
    #print(S)
    #print(delta)
    #print(len(S))
    #print(len(delta))
    plt.figure()
    plt.xlabel('S')
    plt.ylabel('delta')
    plt.title('Delta of option price versus stock price')
    plt.plot(S,delta,color='r')
    plt.show()
    plt.savefig(addr)


# 计算例子。
if __name__ == "__main__":
    put_price,f1 = European_implicit(0.05, 0.4, 50, 150, 1.0, 300,300)
    print("European option price: {0:0.5f}".format(put_price))
    addr = ('C:/Users/yuyuyu/Desktop/option.png')
    plot_delta(f1,300,150,addr)
```

3. 结果如下

```
In [6]: runfile('C:/Users/yuyuyu/Desktop/option.py', wdir='C:/Users/yuyuyu/Desktop')
European option price: 2.28333
```

价格是 2.8333 元



4. 经过对比发现图像形状其实差不多，但是后面图像有差别，可能是 matlab 的方法有些问题。

5. 所以综上所述，不足之处是代码的处理部分有些问题，如果时间允许，希望可以给他解决出来。