宋运翔　计算金融　2016141223037

对算法进行改进，使用马尔科夫，得到的计算结果相同，但是两次计算时间的对比如下：

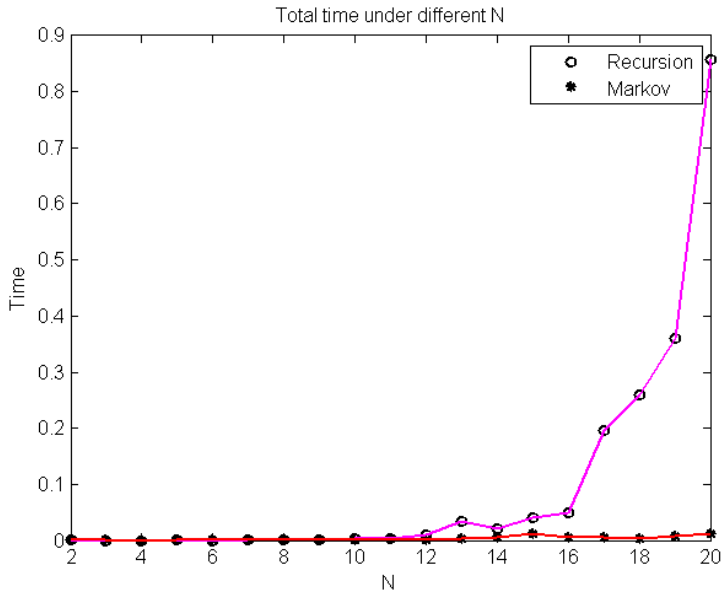

图 1.1 不同 N 时两种算法的时间对比

算法实现代码如下：

```
import java.util.*;
import java.io.*;
import java.lang.*;
//状态矩阵类
class Statematrix {
int row;//行
int maxrow;//最大行数
double [][] Data;
public Statematrix(int row, int maxrow) {
this.row = row;
this.maxrow=maxrow;
this.Data=new double[maxrow][3];//每个状态矩阵列数都为 3
}
public int getindex(double NewMax,double NewState){
int index=0;//初始化索引
for(int i=0;i<this.row;i++)
if(this.Data[i][0]==NewMax && this.Data[i][1]==NewState)
index=i;
return index;
}
//将状态添加入矩阵,[最大值,下一个状态值,期权价格]
```

```
public void addmatrix(double NewMax,double NewState){
//利用变量判断该状态序列是否已经在状态矩阵中
int flag=0;
if(this.row!=0)
flag=getindex(NewMax,NewState);
//将状态向量添加入矩阵
if(flag==0){
this.Data[this.row][0]=NewMax;
this.Data[this.row][1]=NewState;
this.Data[this.row][2]=NewMax-NewState;
//添加向量后，行数增加
row++;
}
}
}
class Markov{
public static void main(String[] args) {
//输入参数为现价 S0，连续利率 R，连续标准差 sigma，总步数 N，时间跨度 t
double S0 = 1, R = 0.05, sigma = 0.2, t = 0.0833;
int N;
//定义输出字符串向量
ArrayList<String>plotdata=new ArrayList<>();
for(N=2;N<=20;N++) {
//二叉树参数
double TimeA = System.currentTimeMillis();
double sigmaN = Math.sqrt((sigma * sigma * t) / N);
double rN = Math.pow((1 + R), (t / N)) - 1;
double u = Math.exp(sigmaN);
double d = 1 / u;
double p = (1 + rN - d) / (u - d);
Statematrix[] States = new Statematrix[N + 1];
//初始化状态向量[S0,S0,0]
States[0] = new Statematrix(1, 1);
States[0].Data[0][0] = S0;
States[0].Data[0][1] = S0;
States[0].Data[0][2] = 0;
//forward，产生状态(Sn;MN)
for (int i = 0; i < N; i++) {
States[i + 1] = new Statematrix(0,1000);//初始化下一阶状态矩阵
for (int j = 0; j < States[i].row; j++) {
double OldMax = States[i].Data[j][0];
double OldState = States[i].Data[j][1];
//上升分支
double NewupMax = Math.max(OldMax, u * OldState);
```

```
double NewupState = u * OldState;
States[i + 1].addmatrix(NewupMax, NewupState);
//下降分支
double NewdownMax = Math.max(OldMax, d * OldState);
double NewdownState = d * OldState;
States[i + 1].addmatrix(NewdownMax, NewdownState);
}
}
//backward,计算价格
for (int i = N - 1; i >= 0; i--)
for (int j = 0; j < States[i].row; j++) {
double OldMax = States[i].Data[j][0];
double OldState = States[i].Data[j][1];
double NewupMax = Math.max(OldMax, u * OldState);
double NewupState = u * OldState;
double v1, v2;
v1 = States[i + 1].Data[States[i + 1].getindex(NewupMax, NewupState)][2];
double NewdownMax = Math.max(OldMax, d * OldState);
double NewdownState = d * OldState;
v2 = States[i + 1].Data[States[i + 1].getindex(NewdownMax, NewdownState)][2];
States[i].Data[j][2] = (p * v1 + (1-p) * v2) / (1 + rN);
}
double TimeB=System.currentTimeMillis();
double Time=(TimeB-TimeA)/1000;
//输出结果
System.out.println("N="+N+"时，期权价格为："+States[0].Data[0][2]+",运算时间为：
"+Time+"s。");
//将结果添加到输出序列
plotdata.add(N+" "+States[0].Data[0][2]+" "+Time);
}
//考虑到 java 绘图并不美观，将计算数据输出到文本"plotdata.txt"中，用 matlab 绘图
FileWriter fileWriter = null;
try {
fileWriter = new
FileWriter("D:/Documents/GitHub/First/Financial-stochastic-analysis/Marplotdata.txt");
int i=0;
while(i<19){//循环写入
fileWriter.write(plotdata.get(i)+"\r\n");//写入 \r\n 换行
i++;
}
fileWriter.flush();
fileWriter.close();
} catch (IOException e) {
e.printStackTrace();} } }
```