

章节 1

背景综述

章节 2

课题陈述

本文所要研究的金融问题是基于欧洲式期权的离散时间对冲策略的风险控制。欧式期权(European Option) 是期权的一种类型。另外两种期权类型是美式期权 (American Option)和亚洲期权(Asian Option)。期权本质上是一种金融合约，它授予期权的拥有者（买方）一种在某一特定时间点或之前，以某一个在合约中给出的价格 (strike price) 购买或者出售某一种金融产品(underlying asset) 的权利。买家在执行自己的权利时，卖家必须按照合约规定的价格进行交易，但买家同时需要付给买家一定的额外交易手续费(premium)。期权的研究主要集中在对其价值的评估。期权的价值一般可以分为两个部分。一是期权的内部价值 (intrinsic value)，主要由该期权基于的金融产品在交易时的市场价格和合约定价的差值决定。二是期权的时间价值 (Time Value)，这一部分取决于交易前的其他因素如 利息，增值等。期权的研究从19世纪就开始了，但目前的研究一般都是基于1973年提出的Black-Scholes模型。我们的课题也是基于Black-Scholes模型，并对欧式期权 对冲策略的风险控制进行研究。

2.1 欧式期权

欧式期权的特点在于买方必须在合约规定的时间到期时才能行使权力。通常将欧式期权的合理价格价定为其期望价值。相比与美式期权，二者在执行时间上有所不同。美式期权合同在到期日前的任何时候或在到期日都可以执行合同，结算日则是在履约日之后的一天或两天，大多数的美式期权合同允许持有人在交易日到履约日之间随时履约，但也有一些合同规定一段比较短的时间可以履约，如“到期日前两周”。欧式期权合同要求其持有者只能在到期日履行合同，结算日是履约后的一天或两天。目前国内的外汇期权交易都是采用的欧式期权合同方式。通过比较，结论是：欧式期权本少利大，但在获利的的时间上不具灵活性；美式期权虽然灵活，但付费十分昂贵。因此，国际上大部分的期权交易都是欧式期权。中国国内银行设立的外汇期权业务都是采用的欧式期权交易。

2.2 Black-Scholes模型

Black-Scholes 是一个70年代由Fischer Black和Myron Scholes 提出的用于金融衍生品交易的数学模型。这个模型可以推出Black-Scholes公式，并用于预测欧式期权定价的理论值。70年代至今在众多的实际交易中，该公式的预测估值被认为是非常接近实际观测到的价值。这个模型的成功也直接导致了期权交易市场的繁荣，并能使芝加哥期权交易市场等大的期权交易场所得到科学的有效的监管。Black-Scholes 公式是一个偏微分方程，用于预测期权价格随时间的变化。而该模型背后的策略则是通过合理地买卖期权来对冲风险，这种对冲策略被 成为"Delta Hedging"，这也是许多更复杂的对冲机构策略的基础。此后Robert C. Merton 进一步发展了该模型，并和Schole分享了1997年的诺贝尔经济学奖。

2.3 课题表述

本课题研究的对象就是欧式期权利用对冲策略来降低交易的风险。我们采用了经典的Black-Scholes模型，同时将预测时间 T 进行了离散化处理

$$0 = T_0 < T_i < T_N = T \quad (2.1)$$

由于每一个时间点上 T_i 上的对冲交易都要支付一笔手续费，所以在一定的交易风险范围内，如果能找到一个最小的 N ，就可以最优化交易成本。假设在 $t = T$ 时，交易的风险可表示为 $|P(X) - P_{T_N}(X)|$ ，而可接受的最大误差为 ϵ ，我们将采用蒙特卡洛模拟算法，通过 M 次模拟，计算出 $|P(X) - P_{T_N}(X)| < \epsilon$ 的概率 $Prob(M, N)$ ，当 $Prob(M, N)$ 的值大于一个预先设定的概率值时，我们认为 N 是一个可接受的抽样次数。算法1给出了我们计算验证一个 N 值的串行算法伪代码。

在获得一个可接受的 N 值后，我们可以继续寻找满足条件的更小的 N 值，一般初始的 N 值取的较大，所以这种最优化的寻找可以通过一个二分法的搜寻算法来实现。在算法2中我们设置的收敛条件为 $|nU - nL| \leq \lambda$ 这是因为蒙特卡洛模拟本身具有一定的随机性。在 N 的收敛过程中会出现波动，导致无法收敛到一个单一的值上，在 $BSERROR$ 中给定了置信概率 $Prob$ 的情况下，我们很自然地就能利用置信区间这一概念来表述我们取到的最优值 N 。在实际应用中， N 的初始值通常需要取得很大，然后通过二分法来逐步缩小范围取得最优值。此外蒙特卡洛模拟得次数 M 也需要取一个很大的值来保证其精确性。如果 N 和 M 的取值都较大，就会极大地增加计算时间和内存空间需求。这时串行程序在普通的计算机上就难以满足问题的需求，我们将需要转而开发并行程序并使其运行在超级计算机上。

2.4 BSERROR算法的并行化

Algorithm 1 欧式期权对冲策略误差控制的串行算法

```

1: 参数  $X_0$                                 ▷ 金融产品在  $t = 0$  时的初始价格
2: 参数  $\sigma$                                 ▷ 市场波动性
3: 参数  $K$                                 ▷ 期权合约价格
4: 参数  $T$                                 ▷ 期权到期时间
5: 参数  $\epsilon$                                 ▷ 可接受的误差上限
6: 参数  $Prob$                             ▷ 离散策略误差可接受概率下限
7: procedure BSERROR( $M, N$ )                ▷  $M$  是蒙特卡洛模拟次数,  $N$  是离散抽样次数
8:    $\delta t \leftarrow T/N$                                 ▷ 时间间隔
9:    $count \leftarrow 0$                                 ▷ 在  $M$  次模拟中  $N$  被接受的次数
10:  for all  $m = 1 : M$  do                            ▷ 外部循环实现多次蒙特卡洛模拟
11:     $NRV[N]$                                 ▷ 一组高斯分布的随机数
12:    for all  $NRV[j]$  do  $NRV[j] \leftarrow \text{GaussienNumGenerator}()$ 
13:  end for
14:   $BM[N]$                                 ▷ 一个数组模拟布朗随机过程
15:   $BM[0] \leftarrow NRV[0] \cdot \sqrt{\delta t}$ 
16:  for all  $BM[j]$  do  $BM[j] \leftarrow BM[j-1] + NRV[j] \cdot \sqrt{\delta t}$ 
17:  end for
18:   $PX[N+1]$                                 ▷ 离散化的期权估价
19:   $PX[0] \leftarrow X_0$ 
20:  for all  $j = 1 : N+1$  do
21:     $PX[j] \leftarrow X_0 \cdot \exp(-0.5 \cdot \sigma^2 \cdot j \cdot \delta t + \sigma \cdot BM[j-1])$ 
22:  end for
23:   $error \leftarrow 0$                                 ▷ 离散估价和连续估价的误差
24:  for all  $j = 0 : N$  do
25:     $Upper \leftarrow (\log(PX[j]/K) + 0.5 \cdot \sigma^2 \cdot (T - T_j)) / (\sigma \cdot \sqrt{T - T_j})$     ▷ 积分上界
26:     $error \leftarrow error - 1/(\sqrt{2\pi}) \cdot (PX[j+1] - PX[j]) \cdot \int_{-\infty}^{Upper} e^{-t^2/2} dt$ 
27:  end for
28:  if  $PX[N] > K$  then                            ▷ 只考虑在合约到期时间买入的情况
29:     $error \leftarrow error + (PX[N] - K)$ 
30:  end if
31:   $Upper1 \leftarrow (\log(X_0/K) + 0.5 \cdot \sigma^2 \cdot T) / (\sigma \sqrt{T})$ 
32:   $Upper2 \leftarrow (\log(X_0/K) - 0.5 \cdot \sigma^2 \cdot T) / (\sigma \sqrt{T})$ 
33:   $error = error + K / (\sqrt{2\pi}) \cdot \int_{-\infty}^{Upper2} e^{-t^2/2} dt - X_0 / (\sqrt{2\pi}) \cdot \int_{-\infty}^{Upper1} e^{-t^2/2} dt$ 
34:  if  $error < \epsilon$  then
35:     $count \leftarrow count + 1$ 
36:  end if
37: end for
38:  $prob \leftarrow count/M$ 
39: if  $prob < Prob$  then
40:    $N \leftarrow N + 1$ 
41: else
42:    $N \leftarrow N - 1$ 
43: end if
44: return  $N$ 
45: end procedure

```

Algorithm 2 最优 N 值的搜索算法

```

1: 参数  $M0$                                 ▷ 蒙特卡洛模拟的次数,  $M0$ 越大模拟结果可信度越高
2: procedure NBSECT( $N0, \lambda$ )                ▷  $N0$ 是初始化抽样次数,  $\lambda$ 是最优解的置信区间
3:    $nL \leftarrow 1$                                 ▷ 设置 $n$ 值的初始下界
4:    $nU \leftarrow N0$                                 ▷ 设置 $n$ 值的初始上界
5:    $n \leftarrow 0.5 \cdot (nL + nU)$ 
6:   while  $|nU - nL| > \lambda$  do
7:     if  $BSERROR(M0, n) < n$  then                                ▷ 缩小搜索范围
8:        $nU \leftarrow n$ 
9:        $n \leftarrow 0.5 \cdot (nL + nU)$ 
10:    else                                ▷ 扩大搜索范围
11:       $nL \leftarrow n$ 
12:       $n \leftarrow 0.5 \cdot (nL + nU)$ 
13:    end if
14:  end while
15:  return  $n, nU, nL$ 
16: end procedure

```
