**MATLAB科学计算HW\_3**

13328157 骆锎源 岭南学院13级8班

【1】

**The complexity of “Fib\_Re.m” is Θ[(n/2-2)2^(n/2)](for even terms)**. The reason is as follow:

For simplicity, suppose we always want to calculate an even term of the progression.

Also, we define t(n) as the time needed to complete the calculation of the nth term in the progression, terminal nodes as the nodes where recursion goes to Fib\_Re(2) or Fib\_Re(1), T(n) as the number of terminal nodes at layer n, N(n) as the number of nodes at layer n.

**Then as the recursion goes on, N(n) would be like a tower:**

2N n=1

2N-1 2N-2 n=2

……

2N-(N-1) …… 2N-(2N-2) n=N

……

…

2 1 n=2N-1

There are 2^(n-1) terms on each of the first n layers of the tower,

i.e., **N(n)=2^(n-1) when 1<=n<=N**

And there are [2^n-(2^1+2^2+…+2^i)] on layer (N+i), 1<=i<=n-1,

i.e., **N(N+i)= [2^N-(2^1+2^2+…+2^i)] , 1<=i<=N-1**

Terminal nodes only exist when n>=N, and the relation between T and N is as follow:

T(2N-1)=N(2N-1)=2

T(2N-2)=N(2N-2)-1/2N(2N-1)

T(2N-3)=N(2N-3)-1/2N(2N-2)

……

T(2N-N)=N(2N-N)-1/2N(2N-N+1)

Apparently **Sum(T(N:2N-1))**=N(N)+1/2Sum(N(N+1:2N-1))

=2N-1+1/2[]

=2N-1+1/2[(N-1)2N-]

=N\*2N-1-2(2N-1-1)+(N-1)

=(N-2)2N-1+N+1

Then, **t(2N)**=t(2N-1)+t(2N-2)=…=t(2N-N+1)+t(2N-N)+…+t(3)+t(2)=…**=nt(2)+mt(1)**

**=2\*Sum(T(N:2N-1))**

= (N-2)2N+2N+2

That is, t(2N) isΘ{(N-2)2^N}.

The complexity of calculating odd terms(2N-1) are similar, also beingΘ{(N-2)2^N}.

【2】

**2a.** For European put option: t(T)=2\*t(T-1)+1=2\*[2\*t(T-2)+1]=…=2^T\*t(0)+2^(T)-1∈Θ（2^T）

For American put option: t(T)=2\*t(T-1)+3=2\*[2\*t(T-2)+3]=…=2^T\*t(0)+3\*2^(T)-3∈Θ（2^T）

**2b.**

1) As for European put option, it is possible to directly **calculate the price based on the statistical distribution of payoff.**

Define N as the times of stock price going up before T==0 and p as the probability of price going up at a specific point of time. Apparently **N is a random variable which obeys binomial distribution**. P(N=i)=nchoosek(T,i)\*p^i\*(1-p)^(T-i), and the terminal stock price when N=i equals to S\*u^i\*d^(T-i). Commensurately, the terminal option price should be max(0, K-terminal stock price). With these numbers available, the expected option price at T can be calculated accordingly. Discount the expected terminal option price and we will get the price of the put option today.

The *script file “EuroPut1.m”* is the function calculating option price this way.

**The complexity of this algorithm should be Θ(T).**

2) As for American put option, once you know the **option price (or stock price) of the next term** and the **stock price at this term**, you can **calculate the option price of this term**.

First of all, we calculate the terminal stock price (t=T) and next to terminal stock price (t=T-1) with methodology in “EuroPut1.m”. Then we can calculate the option price at terminal points and next terminal points accordingly. After that, we store option price at (t=T-1) and stock price at (t=T-2) in separated arrays. With these, we can calculate the option price at (t=T-2). Repeat the process until reaching the start point.

The *script file “AmeriPut1.m”* is the function calculating option price this way.

**The complexity of this algorithm should be Θ(T^2).**

【3】

The files *“EuroCall.m”* and *“AmeriCall.m”* are functions that can be used to calculate European call option price and American call option price respectively. The file *“ValueTrial.m”* can be used to compare price of European and American option price with randomly chosen input variables. As for the conclusion, **American call option is generally more expensive than European call option.**

Specific outcomes are as follows:

|  |  |  |
| --- | --- | --- |
| [S K T u d q R] | AmeriCall | EuroCall |
| 54 45 3 1.2092 0.8270 0.5097 1.1823 | 10.3597 | 8.7378 |
| 53 48 9 1.2069 0.8286 0.5439 1.1382 | 11.5303 | 8.8654 |
| 55 52 2 1.2980 0.7704 0.5446 1.1646 | 9.9861 | 9.9861 |
| 54 52 3 1.3359 0.7485 0.5655 1.1163 | 16.0158 | 16.0158 |
| 42 49 9 1.2681 0.7886 0.5585 1.1224 | 10.4659 | 9.8402 |
| 55 47 5 1.3398 0.7464 0.5891 1.1959 | 18.8250 | 17.9141 |
| 56 47 9 1.2700 0.7874 0.5197 1.1251 | 16.3827 | 12.8957 |
| 52 49 4 1.3662 0.7320 0.5585 1.1550 | 16.2996 | 16.1502 |
| 58 47 7 1.3507 0.7403 0.5380 1.1568 | 20.9559 | 18.3348 |
| 50 46 2 1.2515 0.7990 0.5841 1.1254 | 10.2375 | 10.2375 |