

Numerical Finance Reading Course

Sheet 12 - Sample Solution

Exercise 1: Projected SOR methods for American Put

Implement the Crank-Nicolson scheme, where you solve for each time step k the problem

$$\begin{cases} (Au^{k+1} - b^k)^T (u^{k+1} - g^{k+1}) = 0 \\ Au^{k+1} - b^k \geq 0 \\ u^{k+1} \geq g^{k+1} \end{cases}$$

using the projected Gauß-Seidel method to obtain the prices $V(S, t)$ of an American Put with $K = 12$, $r = 0.04$, $\sigma = 0.4$ and $S \in (0.00001, 20)$ for $t \in [0, T]$. Plot the solution and compare with the values of the corresponding European Put (exercise 2 on sheet 6).

Solution:

```
#include <iostream>
#include <cmath>

using namespace std;

int main(){

    // Problem Definition:
    double K = 12;
    double sigma = 0.4;
    double r = 0.04;
    double T = 1;

    // Grid Sizes:
    // Time:
    int M = 50;
    double deltaTau = 0.5*sigma*sigma*T / (double)M;
    // Space:
    int N = 200;
    // Calculate V(S, t) for S in (0, 20]
    double xmin = log(0.00001/K);
    double xmax = log(20/K);
    double h = (xmax-xmin) / (double)N;

    // Variables for Crank-Nicholson & SOR
    double gamma = deltaTau/(h*h);
    double tau;
    double omega = 1;
    double eps = 0.0000000001;
    double z[N+1];
    double b[N+1], g[N+1];
    double tmp[N+1];
    double err;
    double aii_inv = 1./(1+gamma);

    // Initialization:
    double u_k[N+1];
    double q = (2*r)/(sigma*sigma);
```

```

double x;
for(int i = 0; i <= N; i++){
    x = xmin + i*h;
    u_k[i] = max(exp(0.5*x*(q-1)) - exp(0.5*x*(q+1)) , 0.0);
    //Print: T S_T V(S_T)
    cout << T << "┌" << K*exp(x) << "┐" << K*exp(-0.5*(q-1)*x)*u_k[i] << endl;
}
cout << endl;

// Crank-Nicholson
for(int j = 1; j <= M; j++){
    tau = j*deltaTau;

    // Calculate b_k, g_k
    g[0] = exp(0.5*(q-1)*xmin + (0.25*(q-1)*(q-1) + q)*tau)*max(1-exp(xmin), 0.0);
    for(int i = 1; i < N; i++){
        x = xmin + i*h;
        b[i] = (1-gamma)*u_k[i] + 0.5*gamma*(u_k[i-1] + u_k[i+1]);
        g[i] = exp(0.5*(q-1)*x + (0.25*(q-1)*(q-1) + q)*tau)*max(1-exp(x), 0.0);
    }
    g[N] = 0;
    b[1] += 0.5*gamma*g[0];
    b[N-1] += 0.5*gamma*g[N];

    // Projected SOR (Gau-Seidel)
    do{
        // z_{k+1} und u_{k+1}(tmp)
        z[1] = (0.5*gamma*u_k[2] + b[1])*aai_inv;
        tmp[1] = max(u_k[1] + omega*(z[1] - u_k[1]) , g[1]);
        for(int i = 2; i < N-1; i++){
            z[i] = aai_inv*(0.5*gamma*(tmp[i-1] + u_k[i+1]) + b[i]);
            tmp[i] = max(u_k[i] + omega*(z[i] - u_k[i]) , g[i]);
        }
        z[N-1] = aai_inv*(0.5*gamma*tmp[N-2] + b[N-1]);
        tmp[N-1] = max(u_k[N-1] + omega*(z[N-1] - u_k[N-1]) , g[N-1]);

        // error
        err = 0;
        for(int i = 1; i < N; i++){
            err += (tmp[i] - u_k[i])*(tmp[i] - u_k[i]);
        }
        err = sqrt(err);

        for(int i = 1; i < N; i++){
            u_k[i] = tmp[i];
        }
    }while(err > eps);

    // Set boundary values
    u_k[0] = g[0];
    u_k[N] = g[N];

    //Print
    for(int i = 0; i <= N; i++){
        //Print: t S_t V(S_t)
        x = xmin + i*h;
        cout << T - tau/(0.5*sigma*sigma) << "┌" << K*exp(x) << "┐"
            << K*exp(-0.5*(q-1)*x - (0.25*(q-1)*(q-1) + q)*tau)*u_k[i] << endl;
    }
    cout << endl;
}

return 0;
}

```

Put: $K=12$, $r=0.04$, Symbol $s = 0.4$

