A Linear Algorithm for Black Scholes Economic Model

Ion SMEUREANU
Department of Economic Informatics, Bucharest Academy of Economic Studies
Dumitru FANACHE
Mathematics Department, Valahia University Târgovişte

The pricing of options is a very important problem encountered in financial domain. The famous Black-Scholes model provides explicit closed form solution for the values of certain (European style) call and put options. But for many other options, either there are no closed form solution, or if such closed form solutions exist, the formulas exhibiting them are complicated and difficult to evaluate accurately by conventional methods. The aim of this paper is to study the possibility of obtaining the numerical solution for the Black-Scholes equation in parallel, by means of several processors, using the finite difference method. A comparison between the complexity of the parallel algorithm and the serial one is given.
Keywords*: algorithm, model, Black-Scholes, price, evaluation.*

1 Introduction

It is well-known that the Black-Scholes equation is used in computing the value of an option. In sume cases, e.g. a European options, it gives exact solutions, but for other, more complex, numerical attempts are made in order to obtain an approximation of the solution. Several numerical methods are used for solving the Black-Scholes equation.

A European call option is a contract such that the owner may (without obligation) buy some prescribed asset (called the underlying) S at a prescribed time (expiry date) T at a prescribed price (exercice or strike price) K, the risk-free interest rate r (is an idealized interest rate). A European put option is the same as call option, except that "buy" is replaced by "sell".

2. Black-Scholes Model for evaluating an option price

Black-Scholes model for a European call option can be described ([7]) or [5] by the following (diffusion-type) partial differential equation (PDE) for this value:

$$\frac{\partial f}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} + rS\frac{\partial f}{\partial S} - rf = 0 \quad (1)$$

with final condition

$$f(S,T) = max(S-K,0) \quad (2)$$

and boundary conditions

$$f(0,t) = 0,\ f(S,t) \sim S \quad as\ S \to \infty \quad (3)$$

The European put option satisfies the same equation as (2), but with final condition

$$f(S,T) = max(K-S,0) \quad (4)$$

and boundary conditions

$$f(0,t) = Ke^{-r(T-t)},\ f(S,t) \sim 0\ as\ S \to \infty \quad (5)$$

In both cases, there are explicit closed form solution. For the call option, the solution is

$$f(S,t) = C(S,t) = SN(d_1) - Ke^{-r(T-t)}N(d_2) \quad (6)$$

with

$$d_1 = \frac{\ln\frac{S}{K} + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}},\quad d_2 = d_1 - \sigma\sqrt{T-t} \quad (7)$$

and $N(z)$ is the cumulative distribution function of the standard normal distribution. For the put option,

$$f(S,t) = P(S,t) = Ke^{-r(T-t)}N-(d_2) - SN(-d_1) \quad (8)$$

with the same d_1, d_2, and $N(z)$. For most other style option, however, there are no known closed form solution. Thus, approximate method and numerical methods, such as lattice methods ([3], [4]) and finite difference methods ([6]) are used estimate their values.

3. Models by using finite difference methods

The finite difference method consists of discretizing the partial differential pricing equation and the boundary conditions using a forward or a backward difference approximation.

We discretize the equation with respect to time and to the underlying asset price. Divide the (*S*, *t*) plane into a sufficiently dense grid or mesh, and approximate the infinitesimal steps ΔS and Δt by some small fixed finite steps. Further, define an array of $N+1$ equally spaced grid points $t_0, t_1, \ldots, t_N$ to discretize the time derivative with $\Delta t_{n+1} - t_n = \Delta t$ and $\Delta t = T / N$.

We know that the stock price cannot go below 0 and we have assumed that $S_{\max} = 2S_0$. We have $M+1$ equally spaced grid points $S_0, S_1, \ldots, S_M$ to discretize the stock price derivative with $S_{m+1} - S_m = \Delta S$ and $\Delta S = S_{\max} / M$.

This gives us a rectangular region on the (*S*, *t*) plane with sides $(0, S_{\max})$ and $(0, T)$. The grid coordinates (*n*, *m*) enables us to compute the solution at discrete points.

The time and stock price points define a grid consisting of a total of $(M+1)\times(N+1)$ points. The (n,m) point on the grid is the point that corresponds to time $n\Delta t$ for $n = \overline{0,N}$, and stock price $m\Delta S$ for $m = \overline{0,M}$. We will denote the value of derivative at time step t_n when the underlying asset has value S_m as

$$f_{n,m} = f(n\Delta t, m\Delta S) = f(t_n, S_m) = f(t,S) \quad (9)$$

where *n* and *m* are the number of discrete increments in the time to maturity and stock price respectively. The discrete increments in the time to maturity and the stock price are given by Δt and ΔS, respectively.

Let $f_n = f_{n,0}, f_{n1}, \ldots, f_{n,M}$ for $n = \overline{0,N}$. Then, the quantities $f_{0,m}$ and $f_{N,m}$ for $m = \overline{0,M}$ are referred to as the boundary values which may or may not be known ahead of time but in our PDE they are known. The quantities $f_{n,m}$ for $n = \overline{1,(N-1)}$ and $m = \overline{0,M}$ are referred to as interior points or values.

3.1 The Implicit finite difference method.

We express $f_{n+1,m}$ implicitly in-terms of the unknowns $f_{n,m-1}, f_{n,m}$ and $f_{n,m+1}$. We discretize the Black Scholes PDE in (1) using the forward difference for time and central difference for stock price to have:

$$\frac{f_{n+1,m} - f_{n,m}}{\Delta t} + rm\Delta S\left[\frac{f_{n,m+1} - f_{n,m-1}}{2\Delta S}\right] + \frac{1}{2}\sigma^2 m^2 \Delta S^2\left[\frac{f_{n,m+1} - 2f_{n,m} + f_{n,m-1}}{\Delta S^2}\right] = rf_{n+1,m} \quad (10)$$

Rearranging, we get

$$f_{n+1,m} = \frac{1}{1 - r\Delta t}\left[\alpha_{1m} f_{n,m-1} + \alpha_{2m} f_{n,m} + \alpha_{3m} f_{n,m+1}\right] \quad (11)$$

for $n = \overline{0, N-1}$ and $m = \overline{1, M-1}$. The implicit method is accurate to $O(\Delta t, \Delta S^2)$, the parameters α_{km}'s for $k = 1,2,3$ are given as:

$$\alpha_{1m} = \frac{1}{2} rm\Delta t - \frac{1}{2}\sigma^2 m^2 \Delta t,\ \alpha_{2m} = 1 + \sigma^2 m^2 \Delta t,\ \alpha_{3m} = -\frac{1}{2} rm\Delta t - \frac{1}{2}\sigma^2 m^2 \Delta t \quad (12)$$

The system of equations can be expressed as a tridiagonal system([1])

$$\begin{bmatrix} f_{n+1,0} \\ f_{n+1,1} \\ \dots \\ f_{n+1,M-1} \\ f_{n+1,M} \end{bmatrix} = \begin{bmatrix} \alpha_{20} & \alpha_{30} & 0 & \dots & 0 & 0 & 0 \\ \alpha_{11} & \alpha_{21} & \alpha_{31} & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \alpha_{1M-1} & \alpha_{2M-1} & \alpha_{3M-1} \\ 0 & 0 & 0 & \dots & 0 & \alpha_{1M} & \alpha_{2M} \end{bmatrix} \begin{bmatrix} f_{n,0} \\ f_{n,1} \\ \dots \\ f_{n,M-1} \\ f_{n,M} \end{bmatrix} \quad (13)$$

which can be written as: $Af_{n,m} = f_{n+1,m}$ for $m = \overline{0,M}$ (14)
Let $f_n = f_{n,m}$ and $f_{n+1} = f_{n+1,m}$, then we need to solve for f_n given matrix A and column vector f_{n+1} and this implies that

$$f_n = A^{-1} f_{n+1} \quad (15)$$

We can deduce:

$$f_{n-1} = A^{-1} f_n = \left(A^{-1}\right)^2 f_{n+1}, \dots, f_0 = \left(A^{-1}\right)^{n+1} f_{n+1}$$

.
The matrix A has $\alpha_{2m} = 1 + \sigma^2 m^2 \Delta t > 0, m = \overline{0,M}$, $\prod_{m=0}^{M} \alpha_{2m} \neq 0$, and therefore the matrix is nonsingular. We can solve the system by finding the inverse matrix A^{-1}.
When we apply the boundary conditions together with (11), this gives rise to some changes in the elements of matrix A with

$$\begin{cases} \alpha_{20}, \alpha_{2M} = 1 \\ \alpha_{30}, \alpha_{1M} = 0 \end{cases} \quad (16)$$

Our initial condition give values for N^{th} time step, and we solve for f_n at t_n in terms of f_{n+1} at t_{n+1}. We set the right hand side of the system to our initial condition and solve the system to produce a solution to the equation for time step *N*-1. By repeatedly iterating in such a manner, we can obtain the value of *f* at any time step $0,1,\dots,N-1$.

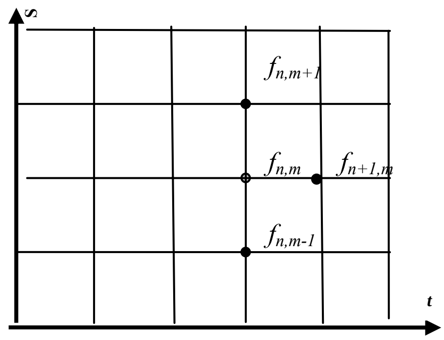

Fig.1. Trinomial tree of implicit finite difference discretization

3.2. The stability of implicit method. The eigenvalues λ_n are given by

$$\lambda_n = \alpha_{2m} + 2\left[\alpha_{1m}\alpha_{3m}\right]^{1/2} \cos\frac{n\pi}{N} \quad \text{for } n = \overline{1,(N-1)} \quad (17)$$

Substituting the values $\alpha_{1m}, \alpha_{2m}, \alpha_{3m}$ with values from (14), we have

$$\lambda_n = 1 + \sigma^2 m^2 \Delta t + \sigma^2 m^2 \Delta t \left[1 - \frac{r^2}{\sigma^4 m^2}\right]^{1/2} \left[1 - 2\sin^2\frac{n\pi}{2N}\right] \text{ for } n = \overline{1,(N-1)} \quad (18)$$

Furthermore, applying the binomial expansion on the square root part and re-arranging we have

$$\lambda_n \approx 1 + 2\sigma^2 m^2 \Delta t - 2\sigma^2 m^2 \Delta t \sin^2\frac{n\pi}{2N}$$

where there is change of sign due to the truncation of the binomial expansion. Therefore the equation are stable when

$$\|A\|_2 = \max\left|1 + 2\sigma^2 m^2 \Delta t - 2\sigma^2 m^2 \Delta t \sin^2\frac{n\pi}{2N}\right| \le 1$$

that is,

$$-1 \le 1 + 2\sigma^2 m^2 \Delta t - 2\sigma^2 m^2 \Delta t \sin^2 \frac{n\pi}{2N} \le 1 \quad \text{for } n = \overline{1,(N-1)} \quad (19)$$

As $\Delta t \to 0, N \to \infty$ and $\sin^2 \frac{(N-1)\pi}{2N} \to 1$, (19) reduces to $|1| \le 1$.

Alternatively, $1 + \sigma^2 m^2 \Delta t \ge 0$ şi $\|A\|_\infty = 1$

Therefore by *Lax's equivalence theorem*([2], [6]), the scheme is unconditionaly stable, convergent and consistent.

3.3. The results concerning convergence speed of implicit method. For a European put option when: S = 20, K= 22, r = 0.1, T = 0.5 şi σ = 0.25, the results content in table **Table 1** shows that when N and M are different, the finite difference methods converges faster than N and M are the same.

Table 1. The comparison of the convergence of implicit method for increase N and M

N=M	Implicit Method	N	M	Implicit Method
10	2,0574	10	20	2,1326
20	2,1546	20	40	2,2091
30	2.2204	30	60	2,2234
40	2,2177	40	80	2,2287
50	2,2286	50	100	2,2328
60	2,2317	60	120	2,2352
70	2,2342	70	140	2,2366
80	2,2352	80	160	2,2377
90	2,2379	90	180	2,2387
100	2,2374	100	200	2,2393

```
1  function[P]=impl_method(S,K,r,sigma,T,N,M);
2  dt=T/N;ds=2*S/M;A=sparse(M+1,M+1);
3  f=max(K-(0:M)*ds,0);// final conditions
4  for m=1:M-1
5   x=1/(1-r*dt);
6   A(m+1,m)=x*(r*m*dt-sigma*sigma*m*m*dt)/2;
7   A(m+1,m+1)=x*(1+sigma*sigma*m*m*dt);
8   A(m+1,m+2)=x*(-r*m*dt-sigma*sigma*m*m*dt)/2;
9  end
10 A(1,1)=1;A(M+1,M+1)=1;
11 for i=N:-1:1
12  f=A\f'; f=max(f,(K-(0:M)*ds)');
13 end
14 P=f(round((M+1)/2));
```

The 11-13[th] lines of program from Table 1 are large consumption of computation time. In practice, there are far more efficient solution techniques than matrix inversion, due to the propriety of A being tridiagonal. Then, methods like *LU* decomposition or *SOR* are applied directly to (10), and the execution time is $O(N)$ per solution. In order to compute A^{-1}, one needs (N^2) operation and others $O(M^2)$ to find $(A^{-1})^m$, using one processor, so in a serial manner. But with several processors under a convenient network, we show in what follows that we can obtain a time of execution $O(N)$, to compute the inverse A^{-1}.

4. Parallel algorithm for calculating the numerical solution

4. 1 The Gauss Jordan method for solving a inverse of matrix. If $N = M$ then A is a $N\times N$-square matrix again f_n and f_{n+1} are N-dimensional vectors. We use the method of elementary transformation to compute the inverse matrix, A^{-1} ([6]). In few words, we start from the matrix A^1, which is obtained from A and a unit matrix, written on the right side of A, as follows:

$$A^0 = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & 1 & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & a_{2N} & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Note. For the sake of the clearness, we denote by $a_{ij}, i, j = \overline{1,N}$ all the elements of matrix A, it means $\alpha_{1m}, \alpha_{2m}, \alpha_{3m}$ and 0. Further, making elementary transformation only on the lines of A^0, after several steps, we bring it to the form A^N, where

$$A^N = \begin{pmatrix} 1 & 0 & \cdots & 0 & a_{1,N+1} & a_{1,N+2} & \cdots & a_{1,2N} \\ 0 & 1 & \cdots & 0 & a_{2,N+1} & a_{2,N+2} & \cdots & a_{2,2N} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & a_{N,N+1} & a_{N,N+2} & \cdots & a_{N,2N} \end{pmatrix}$$

The part $\left(a_{i,j}\right)_{i=\overline{1,N}, j=\overline{N+1,2N}}$ represents A^{-1}.

The computation is made in the following manner:

Step 1.

$$A^1 = \begin{pmatrix} 1 & a_{12}^1 & \cdots & a_{1N}^1 & a_{1,N+1}^1 & a_{1,N+2}^1 & \cdots & a_{1,2N}^1 \\ 0 & a_{22}^1 & \cdots & a_{2N}^1 & a_{2,N+1}^1 & a_{2,N+2}^1 & \cdots & a_{2,2N}^1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & a_{N2}^1 & \cdots & a_{NN}^1 & a_{N,N+1}^1 & a_{N,N+2}^1 & \cdots & a_{N,2N}^1 \end{pmatrix}$$

where $a_{1j}^1 = a_{1j} / a_{11},\ j = \overline{1,2N}$

$$a_{ij}^1 = a_{ij} - a_{1j}^1 a_{i1},\ i = \overline{2,N},\ j = \overline{1,2N} \quad (20)$$

Step 2.

$$A^2 = \begin{pmatrix} 1 & 0 & a_{13}^2 & \cdots & a_{1N}^2 & a_{1,N+1}^2 & a_{1,N+2}^2 & \cdots & a_{1,2N}^2 \\ 0 & 1 & a_{23}^2 & \cdots & a_{2N}^2 & a_{2,N+1}^2 & a_{2,N+2}^2 & \cdots & a_{2,2N}^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & a_{N3}^2 & \cdots & a_{NN}^2 & a_{N,N+1}^2 & a_{N,N+2}^2 & \cdots & a_{N,2N}^2 \end{pmatrix}$$

where $a_{2j}^2 = a_{2j}^1 / a_{22}^1,\ j = \overline{1,2N}$

$$a_{ij}^2 = a_{ij}^1 - a_{2j}^2 a_{i2}^1, \quad i = \overline{1,N}, i \neq 2, j = \overline{1,2N} \quad (21)$$

and so on, till the matrix has the final form

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & a_{1,N+1}^N & a_{1,N+2}^N & \cdots & a_{1,2N}^N \\ 0 & 1 & \cdots & 0 & a_{2,N+1}^N & a_{2,N+2}^N & \cdots & a_{2,2N}^N \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & a_{N,N+1}^N & a_{N,N+2}^N & \cdots & a_{N,2N}^N \end{pmatrix}$$

and A^{-1} is read from the second part of this matrix.

4.2. Analysis of sequential algorithm. From (15) and previous section need first decrease the computing time of matrix A. The number of operations, n_{GJ}, through Gauss Jordan method is computing remarking a each step s, we calculating N-1 multiplicators. Then([6])

$$n_{GJ} = \sum_{s=1}^{n} \left[(N-1) + (N-1)(N+1-s) \right] = \frac{N^3}{2} + N^2 - \frac{3N}{2} \approx \frac{N^3}{2} + N^2 \quad (22)$$

```
function x=
gaussjordan(S,K,r,sigma,T,N,M)
dt=T/N; ds=2*S/M; A=sparse(M+1,M+1);
%boundary conditions
A(1,1)=1;A(M+1,M+1)=1;
A(1,2)=0;A(M+1,M)=0;
% tridiagonal matrix form
for m=1:M-1
  A(m+1,m)=0.5*r*m*dt-0.5*
```

```
for i=2:M+1
   for j=1:2*(M+1)
     D(i,j)=C(i,j)- D(1,j)*C(i,1);
    end
end
C=D;
for pas=2:M+1
 for j=1:2*(M+1)
    D(pas,j)=C(pas,j)/C(pas,pas);
```

```
          sigma*sigma*m*m*dt;
  A(m+1,m+1)=(1+sigma*sigma*m*m*dt);
  A(m+1,m+2)=-0.5*r*m*dt-
           0.5*sigma*sigma*m*m*dt;
end
B=[A eye(size(A))]; % matrix [A I]
C=B;% Gauss Jordan Algorithm
for j=1:2*(M+1)
    D(1,j)=C(1,j)/C(1,1);
end
  end
  for i=1:M+1
   for j=1:2*(M+1)
   if i~=pas
   D(i,j)=C(i,j)-D(pas,j)*C(i,pas);
   end
  end
 end
 C=D;
 end
```

Here an example of execution for $M = N = 4$:

The initial matrix									
1.0000	0	0	0	0	1.0000	0	0	0	0
0.0009	1.0025	-0.0034		0	0	1.0000	0	0	0
0	-0.006	1.0100	-0.0094	0	0	0	1.0000	0	0
0	0	-0.0047	1.0225	-0.0178	0	0	0	1.0000	0
0	0	0	0	1.0000	0	0	0	0	1.0000
The Gauss Jordan final matrix is identical with Matlab call: inv(A)									
1.0000	0	0	0	0	1.0000	0	0	0	0
0	1.0000	0	0	0	-0.0009	0.9975	0.0034	0.0000	0.0000
0	0	1.0000		0	-0.0000	0.0006	0.9901	0.0091	0.0002
0	0	0	1.0000	0	-0.0000	0.0000	0.0045	0.9780	0.0174
0	0	0	0	1.0000	0	0	0	0	0

It is clear that, using only one processor to make all computations, the time of execution is $O(N^3)$, because we have N steps and every step needs $O(N^2)$ operations to be computed. In order to reduce the execution time, we can use the parallel calculus.
Having in mind the previous method, we come back to the solving of system (1), using more than one processor. This can be with $N \times 2N$ processors connected under a lattice network. In every node of the network there is a processor. According with [1], under this connectivity, every processor P_{ij} is connected and may transfer information with its four neighbourhood $P_{i-1,j}$, $P_{i+1,j}$, $P_{i,j-1}$, $P_{i,j+1}$, $i, j = \overline{1, N-1}$. The computation of the inverse matrix A^{-1} can be made in the following manner:

Step 0. (*Initialization*)

$P_{ij} \leftarrow A^0, i = \overline{1,N}, j = \overline{1,2N}$ (each processor save A^0 matrix)

Step 1. **In parallel do**:

$$P_{1j} \leftarrow a_{1j}^1 = a_{1j} / a_{11}, \quad j = \overline{1,2N}$$

$$P_{ij} \leftarrow a_{ij}^1 = a_{ij} - a_{1j}^1 a_{i1}, \quad i = \overline{2,N}, j = \overline{1,2N}$$

Step 2

In parallel do:

$$P_{2j} \leftarrow a_{2j}^2 = a_{2j}^1 / a_{22}^1, \quad j = \overline{1,2N}$$

$$P_{ij} \leftarrow a_{ij}^2 = a_{ij}^1 - a_{2j}^2 a_{i2}^1, \quad i = \overline{1,N}, j = \overline{1,2N}, i \neq 2$$

and so on, till step N, when the matrix in final form is obtained and the inverse matrix A^{-1} can be read. The effort of computation is of order $O(N)$, because we still have N steps, but in parallel, every step takes the time for doing a division, a multiplication and a substraction.

Note. Due to the fact that at step i, the line of processor $P_{ij}, j = \overline{1,2N}$ executes a division and all the other processors executes a subtraction and a multiplication, the problem of their synchronization has be taken into account.

4.3. Solving the final system in parallel. In the previous paragraph we show how the inverse matrix A^{-1} can be computed in parallel, with an execution time of order $O(N)$. In order to solve the system (11), which gives the final numerical solution for the Black-Scholes equation, we have to compute the power m of matrix A^{-1}. According with [2] and [4], this can be done in a logarithmic time, $O(\log_2 N)$ using a binary-tree connectivity among processors, like in Figure 2.

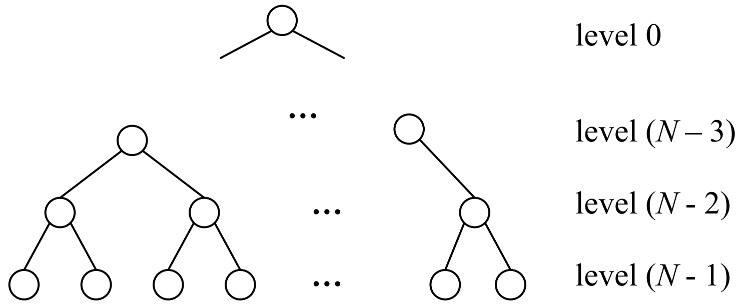

Fig.2. The binary-tree network

Note. In every node of this network there is a processor. The idea of computation is the following:

Step 1. (Initialization)

Every processor leaf (at level (*N*-1)) memorizes the matrix A^{-1}.

Step 2. Every processor at level (*N*-2) computes $\left(A^{-1}\right)^2 = A^{-1} \cdot A^{-1}$.

Step 3. Every processor at level (*N*-3) computes $\left(A^{-1}\right)^4 = \left(A^{-1}\right)^2 \cdot \left(A^{-1}\right)^2$ and so on.

After $\log_2 N$ steps, the final results $\left(A^{-1}\right)^N$ will be computed by the processor root.

5. Conclusion

We presented an algorithm which generates the numerical solution of the Black-Scholes equation for European option in an execution time of order $O(N \cdot \log_2 N)$, using parallel calculus. The binary-tree network can be included in the lattice network, in order to use the same processors.

6. References

[1] **Gerbessiotis**, **Alexandros V**., *Trinomial-tree parallel option price valuation*, New Jersey Institute of Technology, Newark, NJ 07102, USA, June 25, 2002

[2] **White**, **R.F**. *Computational Modeling with Methods and Analysis,* Department of Mathematics North Carolina State University, CRC Press , 2003 (format electronic)

[3] **Duffy, Daniel J**.,*Finance Difference Methods in Financial Engineering*, *A Partial Differential Equation Approach*, John Wiley and Sons, Chichester, UK, 2004 (format electronic)

[4] **Kazuhiro Iwa awa**, (*Analitic Formula for the European Normal Black Scholes Formula*), New York University, Department of Matematics, December, 2, 2001

[5] **M.B. Voc**, **I.Boztosun**, **D.Boztosun**, *On the Numerical Solution of Black-Scholes Equation*, proc. Of Int. Workshop on Mesh Free Methods, 2003

[6] **Brebente Corneliu, Mitran Sorin, Zancu Silviu**, *Metode numerice*, Editura Tehnică, Bucureşti, 1997,(versiune electronică)

[7] **Moisă, Altăr**, *Inginerie financiară*, (2007, Note de curs format electronic, ASE Bucureşti)