Purdue University

Computational Finance Program

Stat 541 Project

Implementing a computationally efficient method for the pricing of European and American contingent claims

by Nik Tuzov

www.ntuzov.com

Spring 2006

1. Problem statement

The goal of this project is to implement the algorithms for the pricing of simple European and American contingent claims described in Chapter 5 of [1] (the page references below refer to that source). The word "simple" means that, if exercised at time t, the claim pays off a value of $f(S_t)$.

The underlying S_t is assumed to follow a standard GBM with constant drift and diffusion, $t \in [0,T]$, where T is the time of maturity. The algorithm should be implemented for a general payoff function *f* and for both American and European exercise styles.

The solution is obtained by localization (restricting the state space) and applying the finite difference method to the original parabolic problem. To that end, the goal is to select reasonable localization bounds for the space and to show how the space-time grid parameters influence the convergence and pricing error. Also, the algorithm should be implemented with a maximal computational efficiency.

2. Localization and space grid

As shown on page 102, in the case of simple contingent claim, we have to solve the parabolic problem:

$$\frac{\partial u}{\partial t} + \tilde{A}u(t,x) = 0 \quad \text{on } [0, \mathrm{T}] \times \mathrm{R}$$
$$u(T,x) = f(x), \quad \forall x \in R,$$

where *u(t, x)* is a pricing function, *f* is a payoff function, *x* – a state variable (log of the underlying price), $\tilde{A}$ - differential operator.

The first step is to localize the space by introducing a constant L: $x \in [-L, L]$. The formula (5.11) on p 105 gives us a clue, but it requires that the payoff *f(.)* be bounded by a constant, which is not always true. A simpler way to chose L which doesn't depend on *f* is to say that for practical purposes, we may think that a normal random variable is within four standard deviations of its mean. Therefore, we may say that

$$S(T) = S_0 \exp(\ (r - \sigma^2/2)T \ \pm \ 4\sigma\sqrt{\mathrm{T}} \ )$$

From which we can find a reasonable L as:

$$L = \max\{ \ |\log(S_0) + (r - \sigma^2/2) + 4\sigma\sqrt{T}|, \ |\log(S_0) + (r - \sigma^2/2) - 4\sigma\sqrt{T}| \ \ \}$$

The number of nodes on the space interval [-L, L] is equal to (N + 2), and the number of nodes on the time axis is equal to (M + 1). The implemented pricing function *claim* still allows the user to specify his own values for L, M, N (see below).

3. Computational efficiency

The implementation of finite difference method is fairly straightforward and described in Ch 5. It works fine for small values of N and M. However, if we try to set N = 10000 or more, MatLab runs out of memory because it has to handle a couple of matrices whose size is N by N. At the same time, these matrices are very sparse (see p 107), so I decided to re-write the computation formulas so that they don't involve any N*N matrices. Thus, the computational time needed to get a pricing function for a fixed *t* became *O(N)* as opposed to $O(N^3)$.

4. Empirical results: European and American options with an arbitrary contract function.

4.0 MatLab implementation

All Matlab files are supplied with comments for better understanding. Matlab implementation consists of the following files:

1) Finite_Difference_Pricing.fig, Finite_Difference_Pricing.m implement graphical user interface to show plots. To run GUI, run Finite_Difference_Pricing.m.

2) get_plot_data.m – creates dataset plotdata.mat to be used in 1)

3) claim.m – the core file that contains pricing function for a simple claim. The function *claim* has a number of parameters: the name of contract function, the indicator of whether the contract is European or American, and so on.

4) call.m – contract function for a regular call option
5) put.m – contract function for a regular put option
6) root.m – contract function for an exotic "root" call option
7) sine_opt.m – contract function for an exotic "sine" contract

Contract functions take one argument – the underlying price. In particular, graphs in Part 4.1, 4.2, 4.3, 4.4 were created using contract functions defined in the files call.m, root.m, put.m, sine_opt.m above, respectively. You are encouraged to create your own contract functions and, with a small modification of get_plot_data.m, you can price the claim and create graphs similar to those shown below.

4.1 Vanilla European Call Option

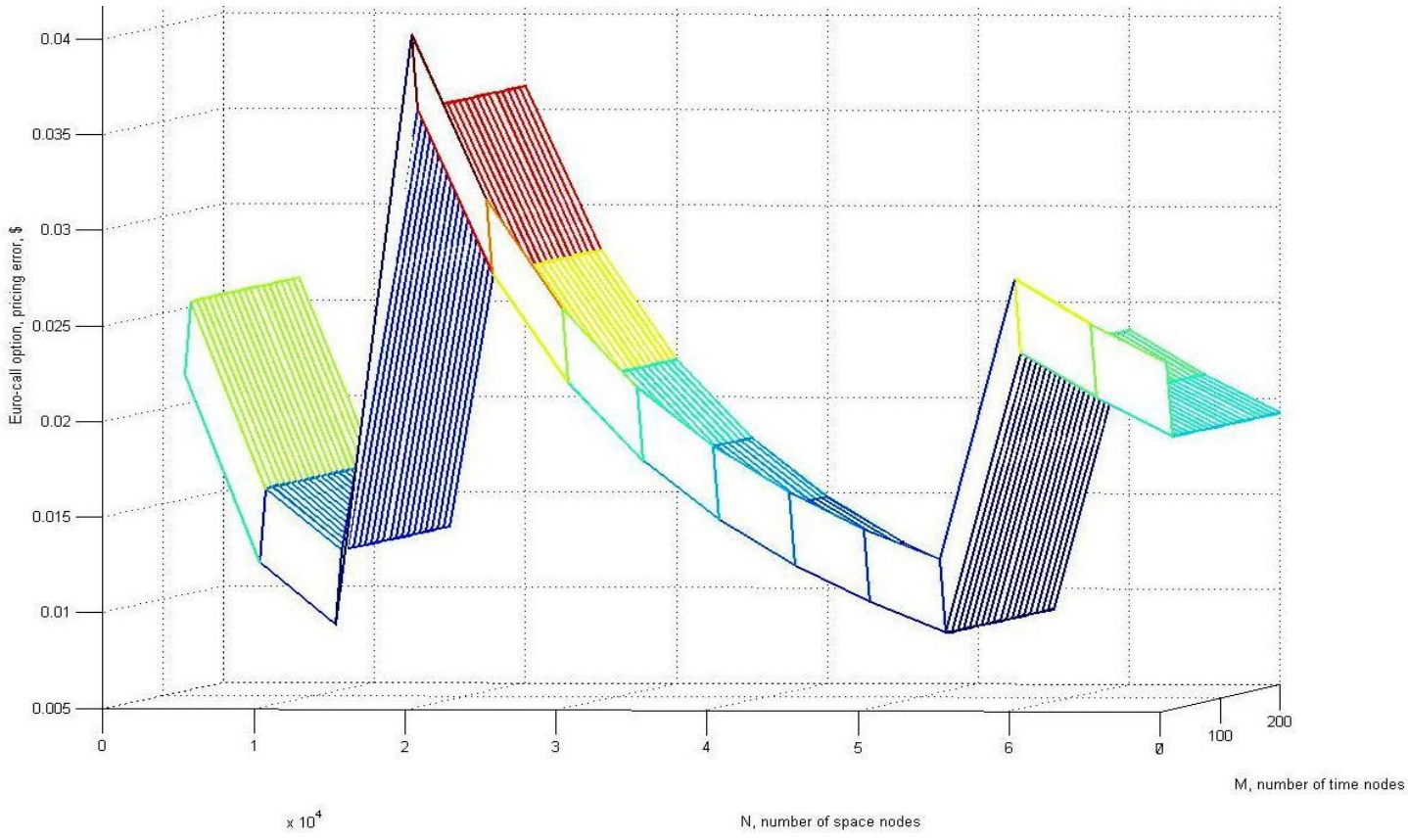
First, consider a standard European call option:

S0	**$110**
K	**$100**
r	**0.05**
sigma	**0.3**
T	**1**
B-S price:	**$21.061**

For this option, L = 5.9 and the stock price at maturity is assumed to be in [$0, $365] which appears reasonable.

Since the true price is known to be $21.061, we can run the finite difference method and get the absolute pricing error to investigate the convergence with respect to different parameters.

On the next page we can see how the error depends on the number of space and time nodes, N and M, respectively. As we see from Figure 1, there is a non-monotone convergence of the mispricing error to zero as N goes up. However, for any fixed N, the mispricing error is almost the same for all values of M greater than 60. The parameter theta ranges from 0 ("fully explicit" finite difference method) to 1 ("fully implicit"). In this example, theta is set to 0.5 which corresponds to so-called Crank-Nicholson scheme.



Fugure 1. Pricing error for vanilla European call, theta = 0.5

Let's now investigate convergence w.r.t. theta. Denote

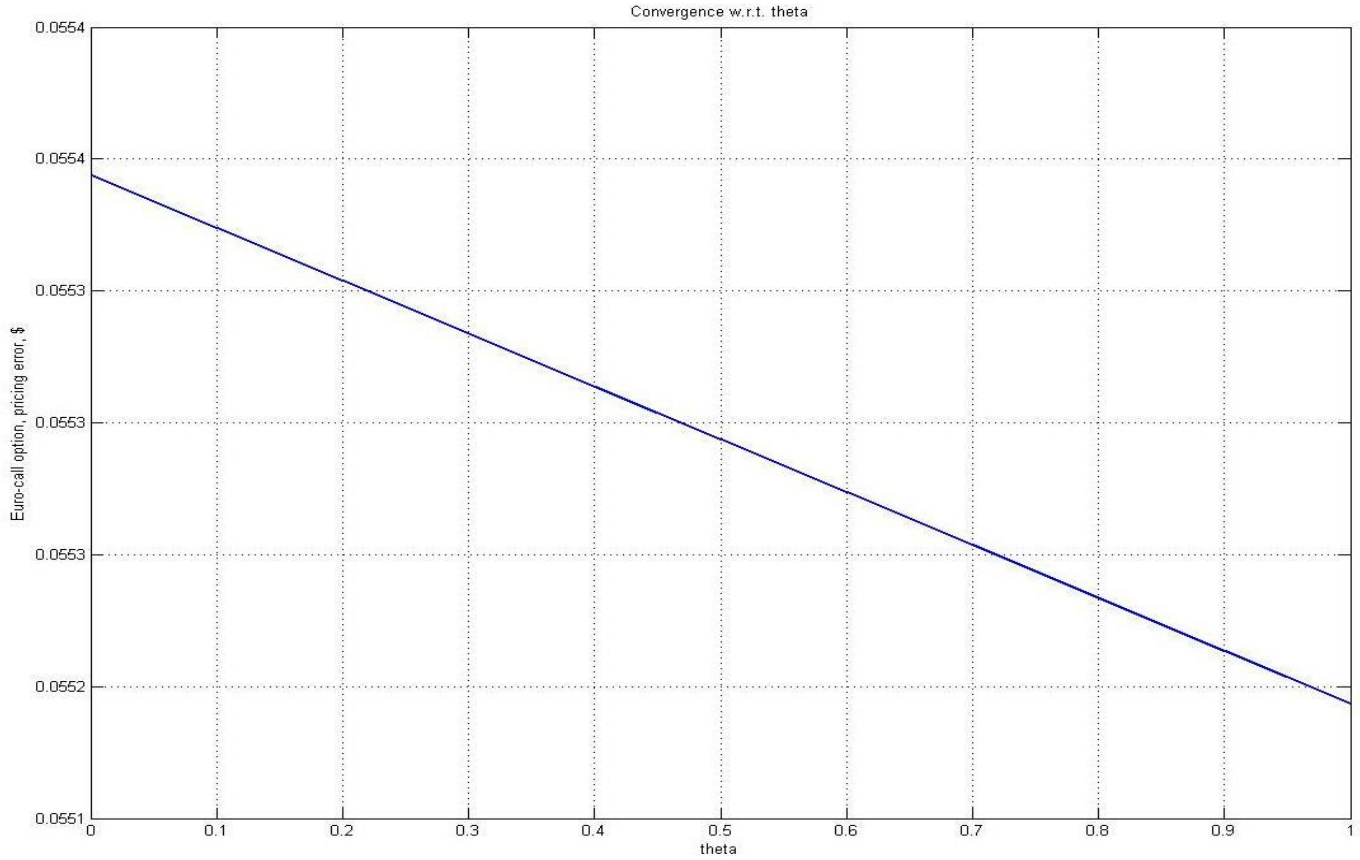
$$h = \frac{2L}{N+1},$$

$$k = \frac{T}{M},$$

That is, h and k are the sizes of state and time steps in discretization. According to the theory, for $1/2 \le theta \le 1$ it's enough to have h and k tend to zero to get convergence. For $0 \le theta < 1/2$ we have one more requirement: k/h^2 should go to zero as well. For instance, from the previous two graphs we see than N = 15000 and M = 60 should give us a small error. However, if we try to vary theta under these N and M, we'll get the following:

Theta	Mispricing error
0	1.00E+206
0.1	1.00E+47
0.2	6.32122E+26
0.3	2.68414E+14
0.4	67.1643
0.5	0.0132
0.6	0.0065
0.7	**0.0002**
0.8	0.007
0.9	0.0138
1	0.0206

Apparently, there's no convergence for theta under 0.5 because k/h^2 is not small. If we set N = 2000 and M = 20000, k/h^2 will become small enough to have convergence for all theta. Figure 2 shows that the mispricing is about 5 cents for all values of theta.



Fugure 2. Pricing error for vanilla European call vs. theta. N = 2000, M = 20000.

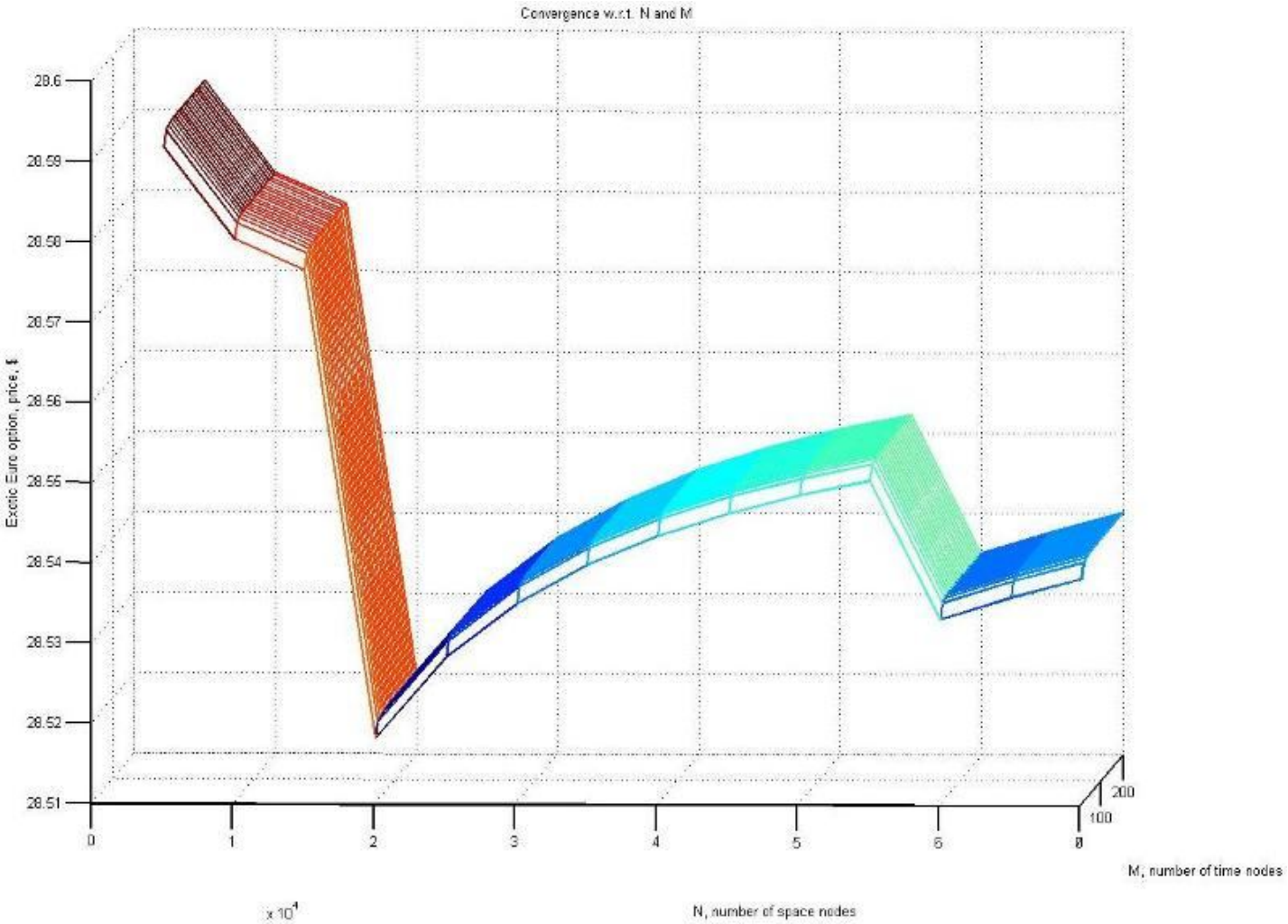
Therefore, it is advisable to use theta at least ½ to ensure unconditional convergence.

4.2 Exotic European "Root" Option

The only difference from 4.1 is that the payoff function is:

$$f(S_T) = \max\{S_T + \sqrt{S_T} - K, \ \ 0\}$$

The true price is unknown, and Figure 3 shows the estimated price vs. N and M, with theta = 0.5.



Fugure 3. Price of exotic European call, theta = 0.5

The picture is very similar to that of 4.1. There's a non-monotone convergence w.r.t. N and it looks like the true price is in $[28.54; 28.55]. It makes sense because this option offers an extra payoff of $\sqrt{S_T}$ over the standard one from 4.1 whose price was $21.06.

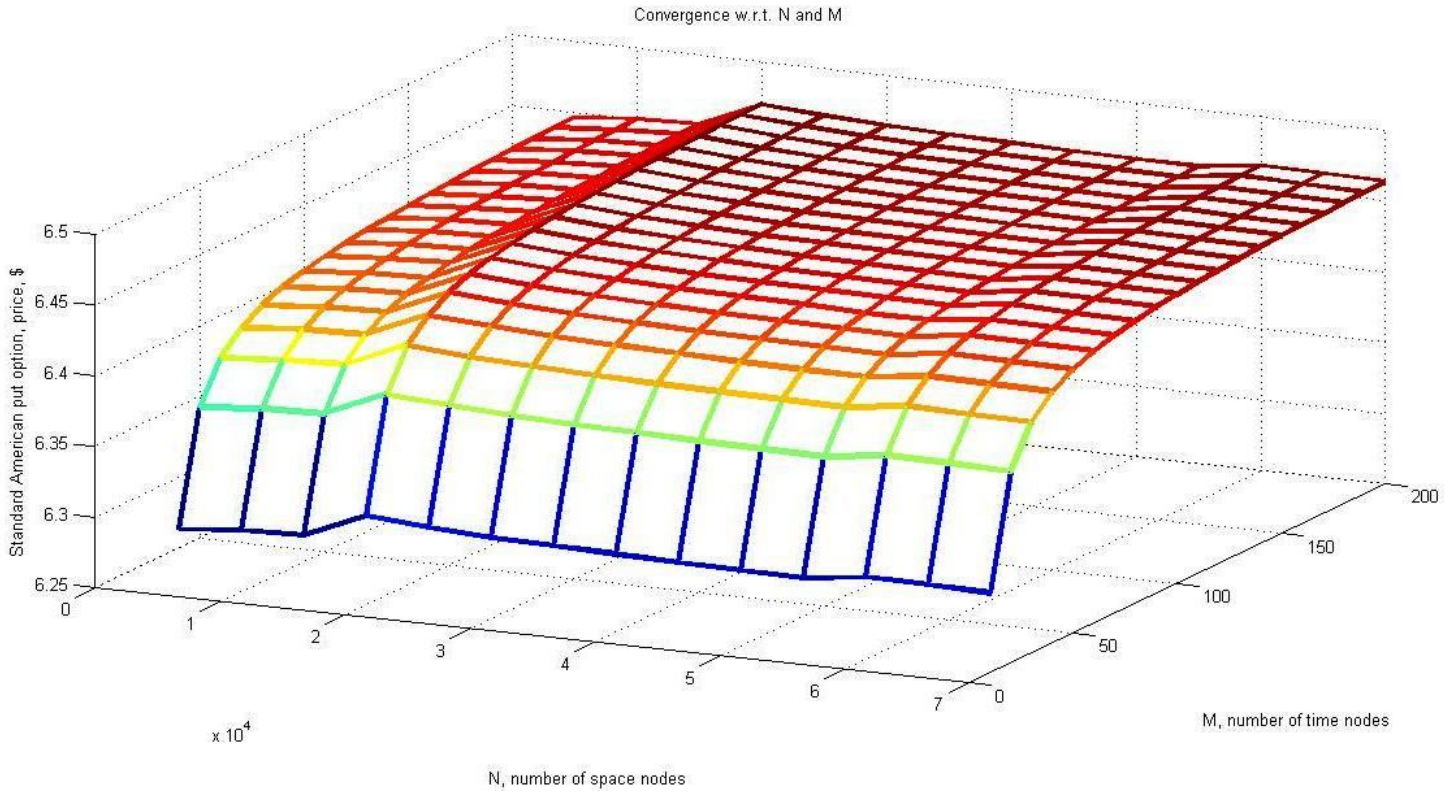
4.3 Vanilla American Put Option.

In this case, the Matlab code written for European case had to be partially modified because:

1) Discretization utilizes Neumann boundary conditions instead of Dirichlet conditions;

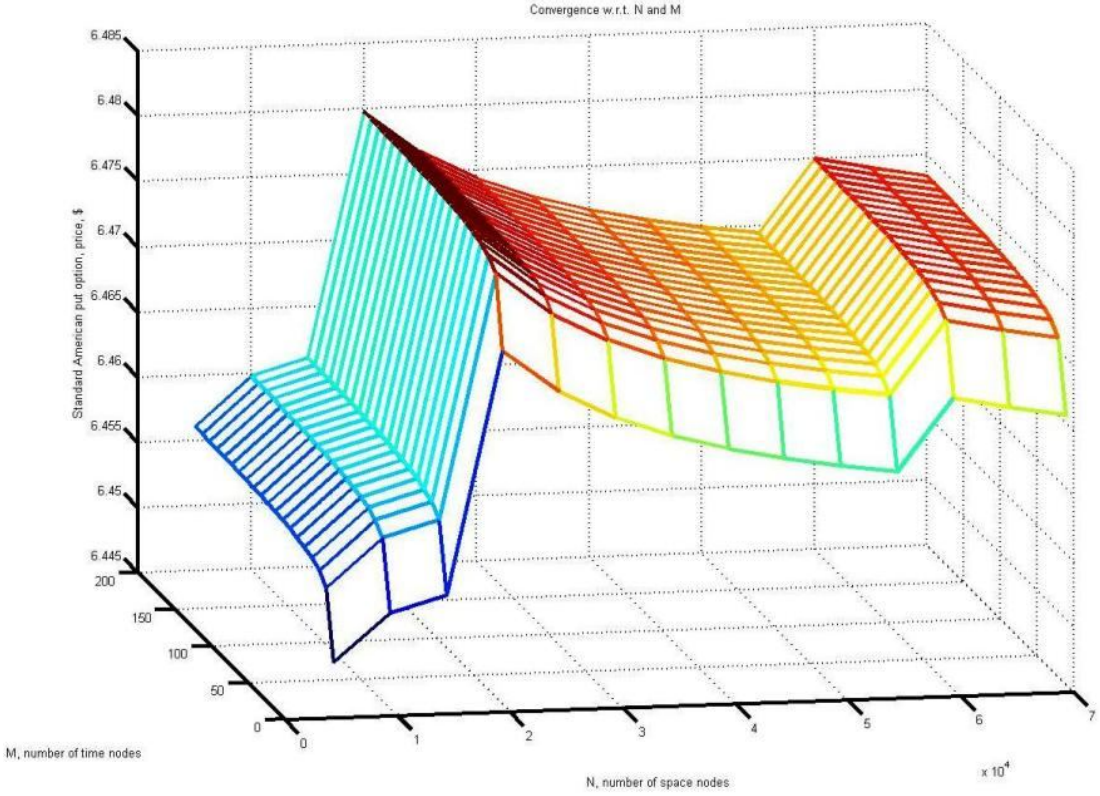
2) The current option value is the maximum of its discounted future value and immediate payoff.

Also, the convergence is unconditional only if theta = 1 (fully implicit finite difference), otherwise the smallness of k/h^2 is required in addition to the smallness of h and k. Given the same inputs as for 4.1, theta = 1, the convergence for the American put is illustrated on Figure 4.



Fugure 4. Price of Vanilla American Put, theta = 1

As opposed to the European case, here the number of time nodes makes more difference than the number of space nodes. The true option price seems to be in $[6.45; 6.50]. For theta = 0.5, we also observe convergence and the estimated price is consistent with that for theta = 1, as shown on Figure 5.



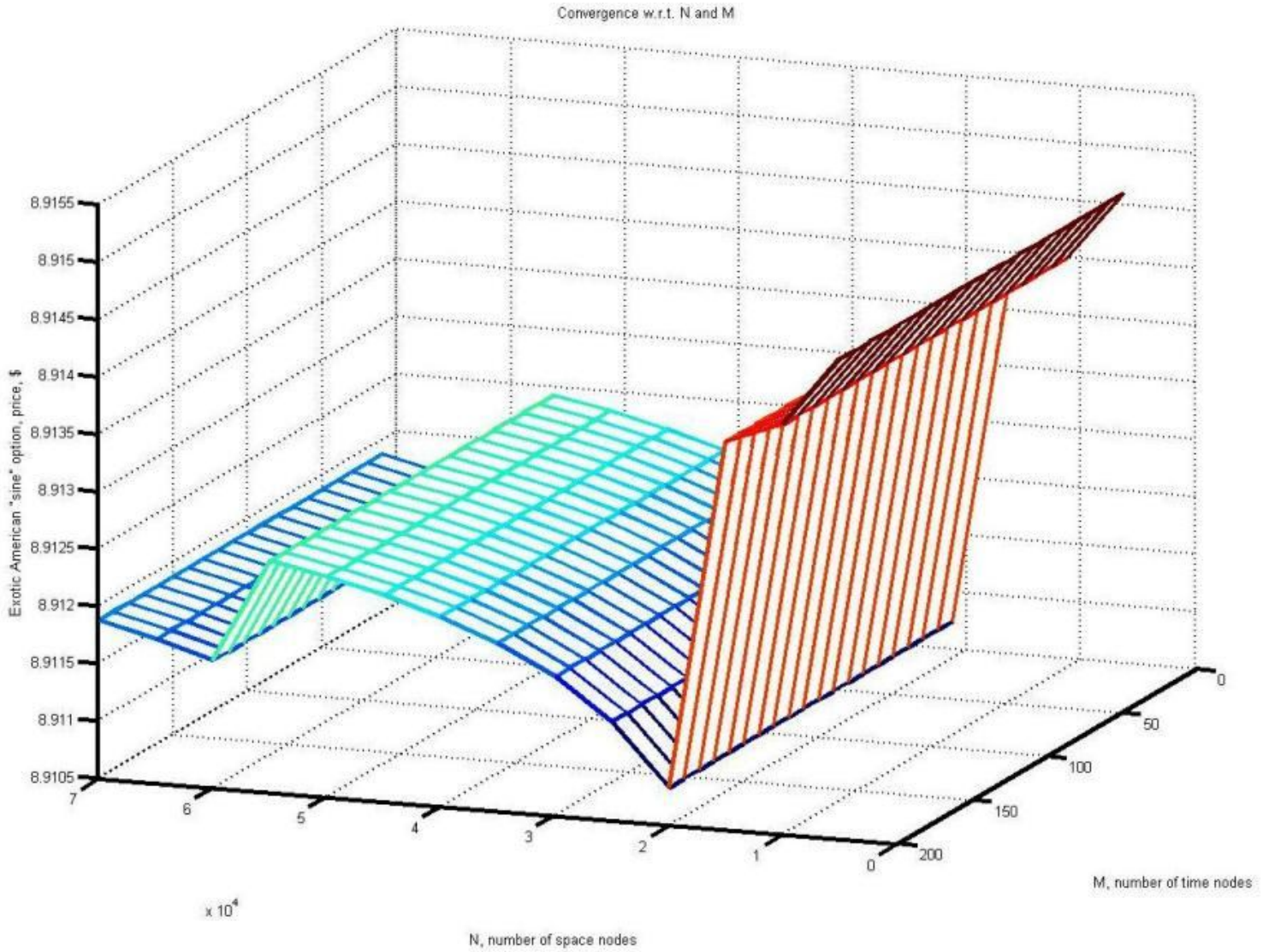
Fugure 5. Price of Vanilla American Put, theta = 0.5

4.4 Exotic American “sine” Option.

Suppose that the price process parameters are as before, but we want to price an American option that pays:

$$f(S_t) = 10 * \sin(S_t / 100)$$

Because the underlying is roughly between \$0 and \$365, I decided to divide S_t by 100 to match it to the interval [0; pi] where sin(.) is positive. The pricing results with theta = 1 are as shown on Figure 6.



Fugure 6. Price of Exotic American Option, theta = 0.5

The convergence is good and the price of this queer option is about \$8.912.

References

[1] Lamberton, Lapeyre "Introduction to Stochastic Calculus Applied to Finance", 1996