Technical report, IDE0922 , September 24, 2009

A new method of pricing multi-options using Mellin transforms and Integral equations

Master's Thesis in Financial Mathematics

Olesya Vasilieva

School of Information Science, Computer and Electrical Engineering
Halmstad University

A new method of pricing multi-options using Mellin transforms and Integral equations

Olesya Vasilieva

Halmstad University
Project Report IDE0922

Master's thesis in Financial Mathematics, 15 ECTS credits

Supervisor: PD Dr. Matthias Ehrhardt
Examiner: Prof. Ljudmila A. Bordag
External referee: Prof. Vladimir Roubtsov

September 24, 2009

Department of Mathematics, Physics and Electrical Engineering
School of Information Science, Computer and Electrical Engineering
Halmstad University

First of all I would like to thank Prof. Ljudmila A. Bordag for giving me the possibility to study at the Halmstad University, for her kind help, support and useful comments during all the study time.

I would also like to thank:

my parents for their love and help.

my classmates for their help and engagement during the whole study, especially Anna Mikaelyan and Ljudmila Petrova.

Best Thanks to the Halmstad University as well.

Contents

Abstract

In this thesis a new method for the option pricing will be introduced with the help of the Mellin transforms. Firstly, the Mellin transform techniques for options on a single underlying stock is presented. After that basket options will be considered. Finally, an improvement of existing numerical results applied to Mellin transforms for 1-basket and 2-basket American Put Option will be discussed concisely. Our approach does not require either variable transformations or solving diffusion equations.

Chapter 1

Introduction

Nowadays derivative markets become extremely popular, this popularity even exceeds that of the stock exchange [16, 19, 20]. Option price estimation as the most interesting of the derivatives has many approaches and is multi-faceted [3, 6]. In many cases more money is invested in options than in the underlying assets.
An option on an underlying asset is an asymmetric contract that is negotiated today with the following conditions in the future. The holder has either the right, but not the obligation to buy, as it is the case with the European Call option, or the possibility to sell, as in the case of the European Put Option, an asset for a certain price at a prescribed date in the future [9, 10, **?**]. The American type of option can be exercised at any time up to and including the date of expiry. There are two good reasons why the American option style generally receives more attention. First of all most exchange-traded options are of this type. And secondly, the pricing of the American option results in a free boundary value problem in the Black-Scholes framework which makes it an interesting mathematical research topic [21].
In general there exists no closed-form solution for the American option problem and hence there is a need for several numerical approaches.
The problem of pricing the American option becomes even more complicated if the option is used as a hedging tool and contains multiple underlying assets, so called multi-asset option or basket option. No extensive research has been done in that direction.
In my thesis one of the techniques called Mellin transformation [2, 15, 18] applied to the Black-Scholes equation [1, 4, 12] and suggested by Panini [13, 14] for basket options will be considered and new integral representations for the price and the free boundary of the American option in one dimension will be presented.
We will improve the numerical part of Panini's computations by using the

Newton's method for the free boundary condition at first for one-basket option and try to extend it afterwards to the two dimensional case. The quadrature schemes will be improved as well.
The thesis is organized as follows. In Chapter 2 the method and notation are given. This provides an introduction to the terminology and basic principles of the option pricing using the new method with the help of the Mellin transforms. In Section 2.1 we consider the Mellin transform techniques for options on a single underlying stock in European case. Section 2.2 describes American put options for one stock and presents analytical solutions for that case. The numerical solution using Newton's method for American case and it's motivation are presented in the Section 2.3. In Chapter 3 we apply the Mellin transform for the basket of two underlying assets. The expression for European put basket options will be derive in Section 3.1. Section 3.2 is devoted to the American case of basket options on two stocks. After that we discuss the improvement and numerical results applied to Mellin transforms for 1-basket and 2-basket American put options. Finally, we compare the numerical results of our improvements with the results obtained by Panini [13] using other methods. All results are presented, discussed and interpreted in Section 3.4. A collection of program codes can be found in the Appendix.

Chapter 2

Mellin transforms

In this chapter we will consider the Mellin transform used in context of the Black-Scholes formulae by Panini, Srivastav [13, 14] and Jodar [5]in 2005. The authors use Mellin transforms to derive at first an equation for the price of a European put on a single underlying stock [13, 14]. This case will be extended to American put options later. It is assumed, there are no dividends.

2.1 European Put Options

The Black-Scholes equation [1] for the price of a European put $p(S,t)$ reads

$$\frac{\partial p}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 p}{\partial S^2} + rS\frac{\partial p}{\partial S} - rp = 0, \tag{2.1}$$

where $0 < t < T$ and $0 < S < \infty$ and σ is a volatility of the market prices, r is a risk-free interest rate. We let K denotes the exercise price and S(T) denotes the asset price at the expiry date.
Note that $p(S,t) \to 0$ as $S \to \infty$ and satisfies the terminal condition

$$p(S,T) = \theta(S) = (K-S)^+ = max(K - S(T), 0)$$

and the boundary condition at $S = 0$

$$p(0,t) = Ke^{-r(T-t)}.$$

Let $\hat{p}(v,t)$ denote the Mellin transform of $p(S,t)$ which is defined by the relation

$$\hat{p}(v,t) = \int_0^\infty p(S,t) \cdot S^{v-1} dS,$$

where is v is a complex variable.
Contrary the inverse Mellin transform is defined by

$$p(S,t) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} \hat{p}(v,t) \cdot S^{-v} dv, \tag{2.2}$$

where c is a positive constant.
Hence, the price of the European put is given by

$$p(S,t) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} \hat{\theta}(v) \cdot e^{\frac{1}{2}\sigma^2 h(v)(T-t)} \cdot S^{-v} dv. \tag{2.3}$$

By transforming variables using several algebraic transformations, the above expression can be simplified to [12]

$$p(S,t) = K \cdot e^{-r(T-t)} \mathcal{N}(-d_2) - S \cdot \mathcal{N}(-d_1), \tag{2.4}$$

where $\mathcal{N}(\cdot)$ denotes the distribution function for a standard normal random variable

$$\mathcal{N}(x) = \frac{1}{2\pi} \int_{-\infty}^{x} e^{-\frac{s^2}{2}} ds.$$

2.2 American Put Options

If we consider now the American put option problem, the early exercise feature of this option gives rise to a free boundary problem. As far as we know, there exists no closed-form analytical expression for the value of American put or its associated free boundary.
The Black-Scholes equation for the price of an American put $P(S,t)$ satisfies the nonhomogeneous equation

$$\frac{\partial P}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 P}{\partial S^2} + rS\frac{\partial P}{\partial S} - rP = f(S,t), \tag{2.5}$$

where $0 < t < T$ and $0 < S < \infty$.
The inhomogenity in (2.5) is given by

$$f = f(S,t) = \begin{cases} -r \cdot K, & \text{if} \quad 0 < S \leq S^*(t) \\ 0 & \text{if} \quad S > S^*(t). \end{cases}$$

The final time condition is

$$P(S,T) = (K-S)^+$$

and the free boundary $S^* = S^*(t)$ is determined by smooth pasting conditions

$$P(S^*,t) = K - S^*, \qquad \frac{\partial P}{\partial S} \mid_{S=S^*} = -1.$$

As before, the authors apply the Mellin transform and get for the price $P(S,t)$

$$P(S,t) = p(S,t) + \frac{rK}{2\pi i} \int_t^T \int_{c-i\infty}^{c+i\infty} S^{-v} \frac{(S^*(x))^v}{v} e^{\frac{1}{2}\sigma^2 h(v)(x-t)}\, dv\, dx. \tag{2.6}$$

Substituting $S = S^*(t)$ from (2.6) and using the first smooth pasting condition, they obtain the following integral equation for the free boundary

$$K - S^*(t) = p(S^*(t),t) + \frac{rK}{2\pi i} \int_t^T \int_{c-i\infty}^{c+i\infty} \frac{1}{v} (\frac{S^*(t)}{S^*(x)})^{-v} e^{\frac{1}{2}\sigma^2 h(v)(x-t)}\, dv\, dx. \tag{2.7}$$

Note, by using Mellin transform the smooth solution will be provided automatically. Hence the second smooth pasting condition is not required.

2.3 Motivation of Newton's method for an American Put Option

Now we want to provide a motivation of using Newton's method [17] for the calculation of the free boundary $S^*(\tau)$.
The equation for the price $P(S,\tau)$ of the 1-basket American put option is given by

$$\begin{aligned} P(S,\tau) = p(S,\tau) + rK \int_0^\tau e^{-r\xi} \mathcal{N}\left(-\frac{1}{\sigma\sqrt{\xi}} \left(log\left(\frac{S(\tau)}{S^*(\tau-\xi)} \right) \right.\right. \\ \left.\left. + \xi(r - \frac{1}{2}\sigma^2) \right) \right) d\xi. \end{aligned} \tag{2.8}$$

The boundary condition on a free boundary gives an integral equation for $P(S,\tau)$

$$\begin{aligned} S^*(\tau) = K - p(S^*(\tau),\tau) - rK \int_0^\tau e^{-r\xi} \mathcal{N}\left(-\frac{1}{\sigma\sqrt{\xi}} \left(log\left(\frac{S^*(\tau)}{S^*(\tau-\xi)} \right) \right.\right. \\ \left.\left. + \xi(r - \frac{1}{2}\sigma^2) \right) \right) d\xi. \end{aligned} \tag{2.9}$$

The "European part" looks as follows

$$p(S) = Ke^{-r\tau}\mathcal{N}(-d_2) - S\mathcal{N}(-d_1), \tag{2.10}$$

where

$$d_2 = d_1 - \sigma\sqrt{\tau}, \quad d_1 = \frac{1}{\sqrt{\tau}}\left(\ln\frac{S}{K} + (r + \frac{\sigma^2}{2})\tau\right). \tag{2.11}$$

Panini has used the recursive method of the simple iteration with initial $S_0^*(\tau) = K$

$$S_n^*(\tau) = K - p(S_{n-1}^*(\tau), \tau) - rK\int_0^\tau e^{-r\xi}\mathcal{N}\left(-\frac{1}{\sigma\sqrt{\xi}}\left(\ln\left(\frac{S^*(\tau)}{S^*(\tau-\xi)}\right)\right.\right.$$
$$\left.\left. + \xi(r - \frac{1}{2}\sigma^2)\right)\right) d\xi. \tag{2.12}$$

It is not optimal to use it due to the low order of convergence. So, we can consider the possibility to improve, i.e. to accelerate this computation using a faster method, for example, Newton's method. To do so, we rewrite the equation in the form $F(x) = 0$, $(x(\tau) \equiv S^*(\tau))$

$$x^{n+1} = x^n - \frac{x^n - K + p(x^n, \tau_k)}{1 + \frac{\partial p}{\partial x}(x^n, \tau_k)}, \qquad n = 0, 1, \ldots. \tag{2.13}$$

Newton's method has the quadratic rate of convergence, but it requires a higher effort as demands calculation of matrix Jacobian. Also, the Newton's method of converges only locally from a close initial approximation x^0. For more detailed finding-out of character of convergence we shall calculate the first and the second derivative of the function $F(x) \equiv x - K + p(x, \tau_k)$. We differentiate $F(x)$ as a complex function.

$$F_x(x) = 1 + \frac{\partial p(x,\tau)}{\partial x} = 1 + \frac{\partial}{\partial x}(Ke^{-r\tau}\mathcal{N}(-\rho + \sigma\sqrt{\tau} - x\mathcal{N}(-\rho)), \tag{2.14}$$

where $\rho = \frac{1}{\sigma\sqrt{\tau}}[\ln(\frac{x}{K}) + (\rho + \sigma^2)\tau]$.

$$F_x(x) = 1 - Ke^{-r\tau}\mathcal{N}'(-\rho + \sigma\sqrt{\tau})\frac{\partial\rho}{\partial x} + x\mathcal{N}'(-\rho)\frac{\partial\rho}{\partial x} - \mathcal{N}(-\rho). \tag{2.15}$$

We shall consider the obvious property for the function $\mathcal{N}(x)$

$$N'(a) = \frac{1}{\sqrt{2\pi}}e^{\frac{-a^2}{2}}, \tag{2.16}$$

from which follows

$$N^{'}(a+h) = \frac{1}{\sqrt{2}\pi} e^{\frac{-(a+h)^2}{2}} = \frac{1}{\sqrt{2}\pi} e^{\frac{-a^2}{2}} e^{-ah} e^{\frac{-h^2}{2}} = \mathcal{N}^{'}(a) e^{-ah} e^{\frac{-h^2}{2}}. \quad (2.17)$$

In our case it turns out

$$\begin{aligned} N^{'}(-\rho + \sigma\sqrt{\tau}) &= N^{'}(-\rho) e^{\rho\sigma\sqrt{\tau}} e^{-\frac{\sigma^2\tau}{2}} \\ &= N^{'}(-\rho) e^{\frac{\ln(x)}{K} + r + \frac{\sigma^2}{2}\tau} e^{-\frac{\sigma^2\tau}{2}} = N^{'}(-\rho)\frac{x}{K} e^{\rho\tau}. \end{aligned} \quad (2.18)$$

After substitution of $N^{'}(-\rho)$ into $F_x(x)$ we have

$$F_x(x) = 1 - Ke^{-r\tau}\mathcal{N}^{'}(-\rho)\frac{x}{K} e^{\rho\tau}\frac{\partial\rho}{\partial x} + xN^{'}(-\rho)\frac{\partial\rho}{\partial x} - \mathcal{N}(-\rho) = 1 - N(-\rho). \quad (2.19)$$

As function $0 < \mathcal{N}(x) < 1$ we have received, that the first derivative is always positive.

$$F_x(x) = 1 - \mathcal{N}^{'}(-\rho) > 0, \quad if \quad \tau > 0. \quad (2.20)$$

Let us find the second derivative

$$F_{xx}(x) = \frac{\partial}{\partial x}(F_x(x)) = 0 - \mathcal{N}^{'}(-\rho)\frac{\partial - \rho}{\partial x} = \mathcal{N}^{'}(-\rho)\frac{\partial\rho}{\partial x} = \frac{\mathcal{N}^{'}(-\rho)}{x\sigma\sqrt{\tau}} > 0, \quad (2.21)$$

since $\frac{\partial\rho}{\partial x} = \frac{1}{x\sigma\sqrt{\tau}}$.

We have obtained that the second derivative is always positive

$$F_{xx}(x) = \frac{N^{'}(-\rho)}{x\sigma\sqrt{\tau}} > 0$$

. It means the Newton's method provides monotone convergence on the right (from the right side). Hence, it is quite reliable under the condition that the initial approximation of x^0 more than a required solution that we are looking for. Therefore Newton's method is quite good approach for the solution of the given system (without considering the integral or other words G). The method will be stable as well if any constant will be added to this function. It does not influence the first and the second derivatives of the function $F(x)$. Subject to (2.20) the Newtons method in formulae (2.13) was applied for $n > 0$, i.e. by $n = 0$ the simple recursion was applied.

2.4 An interpretation of the results

Now the results for 1-basket option will be discussed with the help of graphs and tables.
The results of our program and its correctness by using Newton's method can be checked in comparison with the results which were obtained by Panini. The left figure shows the results of Panini's work using the recursive function for American put option. The right figure shows the results by applying Newton's method for the free boundary $S^*(\tau)$. The similarity of both graphs is obvious.

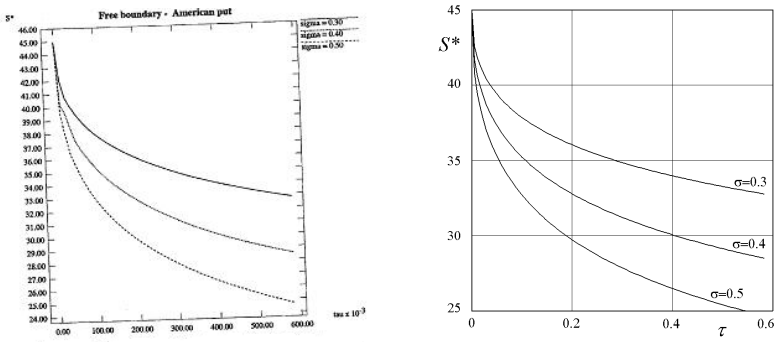

Figure 2.1: Free boundary for an American put with $K = 45$, $T = 0,5833$ and different choices for σ.

Now we want to present the difference of computations for the free boundary by American put option on a single underlying asset, i.e. an integral error from function M. The Table 2.1 shows the convergence rate of the approx-

The size of the grid, M	*Trapezium*	*Trapezium Plus*
50	0.003728	0.002714
100	0.001868	0.000610
200	0.000921	0.000099
400	0.000452	0.000037
800	0.000227	0.000018

Table 2.1: *"The average error by the computation for the free boundary" (integral error from function M).*

imated solution for the free boundary $S^*(\tau)$ and ($\tau \in [0, T]$) by increasing the grid size M on the interval $[0, T]$. The computations were done with the initial values $K = 45$, $T = 0,5833$, $\sigma = 0,5$, $r = 0.0488$. In order to check

the given program for convergence by using Newton's method, we need to compare the difference between graphs with the different number of nodes M along time τ. In the left column the grid size is given with M=50, 100, 200, 400, 800 nodes in form of doubling the previous value. The norm is calculated like the average of 25 nodes on the interval, which is common for all given grids

$$\|s^*(\tau)\| = \frac{1}{25}\sum_{i=1}^{25}\left|s^*(i\frac{T}{25})\right|$$

. The norm of the difference is calculated in the same way. In the central column the trapezoidal rule for the integral computation is used for the convergence. The trapezoidal rule has the second order of accuracy, but the error for the free boundary is reduced just linearly in this case. It means if the grid size is double increased, the error is in approximately reduced by factor 2. It depends on the behavior of the integrand function in the integral

$$\int_0^\tau f(\epsilon)\,d\epsilon,$$

which has the behavior $f(\epsilon) \sim A + B\sqrt{\epsilon}$ for very small ϵ. The trapezoidal rule for such function provides the considerable error in the neighbourhood $\epsilon = 0$, which influences all of the values $s^*(\tau)$.
For the elimination of this error the trapezoidal rule should be modified on the left end of the interval of the grid. The idea of the modification of the trapezoidal rule is the substitution on the end of the interval of the grid instead

$$\frac{f(0)+f(\Delta\tau)}{2}$$

another formulae is used

$$\frac{2f(0)+25f(9/25\Delta\tau)+9f(\Delta\tau)}{36}. \tag{2.22}$$

This formulae provides the exact value of the integral on the interval $[0, \Delta\tau]$ for the function like $f(\epsilon) \sim A+B\sqrt{\epsilon}$ and increases the accuracy of the integral computation on the whole interval $[0, \tau]$. The results of this modification called *"Trapezium Plus"* are given in the right column of the Table 2.2. By the $M = 50 \to 200$ the convergence rate is close to the quadratic, further the rate decreases up to linear. In a whole the convergence of the modified method is really faster compared to the usual trapezoidal rule. Note that the formula (2.22) would increase the accuracy also for another integrands with the infinite derivative at the boundary of interval, for example for the

function with a logarithmic singularity.
As was already mentioned the convergence velocity of the trapezoidal method is close to linear according to the first table. It allows us to use a simple extrapolation of the solution with high accuracy with the formulae

$$\tilde{S}(\tau) = 2S_M(\tau) - S_{\frac{M}{2}}(\tau).$$

The solution has been calculated for $M = 3200$ nodes. By means of this solution it is possible to find errors (i.e. the deviation from the exact solution) of the approximate solutions and their dependence from τ. In the Table 2.2 the average relative errors for the various grids are presented. In distinction with the Table 2.1 the comparisons for all M are taken with $\tilde{S}(\tau)$ here. The

The size of the grid, M	*Trapezium*	*Trapezium Plus*
50	0.003722	0.000302
100	0.001942	0.000180
200	0.001000	0.000100
400	0.000511	0.000054
800	0.000260	0.000029

Table 2.2: *"Relative average error of computation for the free boundary" (integral error from function M).*

relative errors are computed by the following formulae

$$\delta S_M = \|(S_M(\tau) - \widetilde{S}_M(\tau))/\widetilde{S}_M(\tau)\| = \frac{1}{M}\sum_{i=1}^{M} \left|S_M(\tau) - \widetilde{S}_M(\tau)\right|/\widetilde{S}_M(\tau).$$

From the table we can see that by applying the modification of the trapezoidal rule the average error is less approximately in 10 times in compare with the usual trapezoidal rule. So, on the Figure 2.2 we have the dependence $\delta S_M(\tau)$on $\tau \in [0.1, T]$ of the relative errors for the different choices of the grid size M. On the left graph the usual trapezoidal method is used, on the right one the trapezoidal rule is presented. On both graphs the errors are increased, i.e. zoomed by 1000 times. We shall especially note, that the vertical scale on the right figure is increased by 10 times. Because of the increased scale on the right graph are more distinctly visible the errors oscillation.
By comparing both graphs it is obvious, that using the modification error decreases by 10 times on the interval $\tau \in [0.1, T]$. The uniform reduction of an error took place here though the correction (2.22) was applied only on

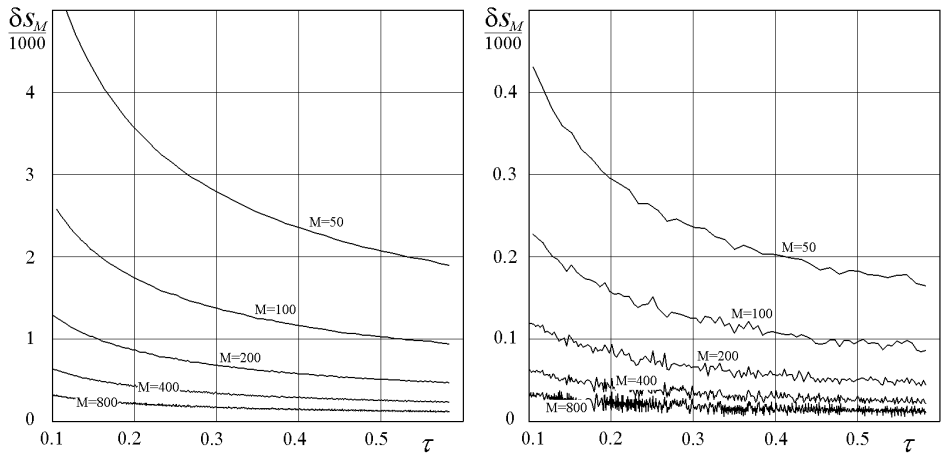

Figure 2.2: "The error of the computation of the free boundary S^* with different choices of the grid size"

an edge of an integration interval. Thus, the suggested modification of the trapezoidal method allows to raise the grid accuracy of calculations in 10 times without any change. It is necessary to mention as well that for both methods the reduction of an error takes place with the growth of τ.
The efficiency of the modification can be estimated in computational effort. It is visible from the table and graphs, for example, that the error by the modification on grid $M = 100$ is less than without it on a grid with $M = 800$ nodes. It means that the simple modification by the same accuracy allows to use more coarse grid with less nodes and in $8 * 8 = 64$ time to reduce an operating time of the program.
The Figure 2.3 shows the isolines of the function of the American put option. As we can see the smooth conjunction of the family of the strict vertical isolines with the curve linear isolines. The boundary where the vertical isolines begin to curve is the free boundary $S^*(\tau)$.

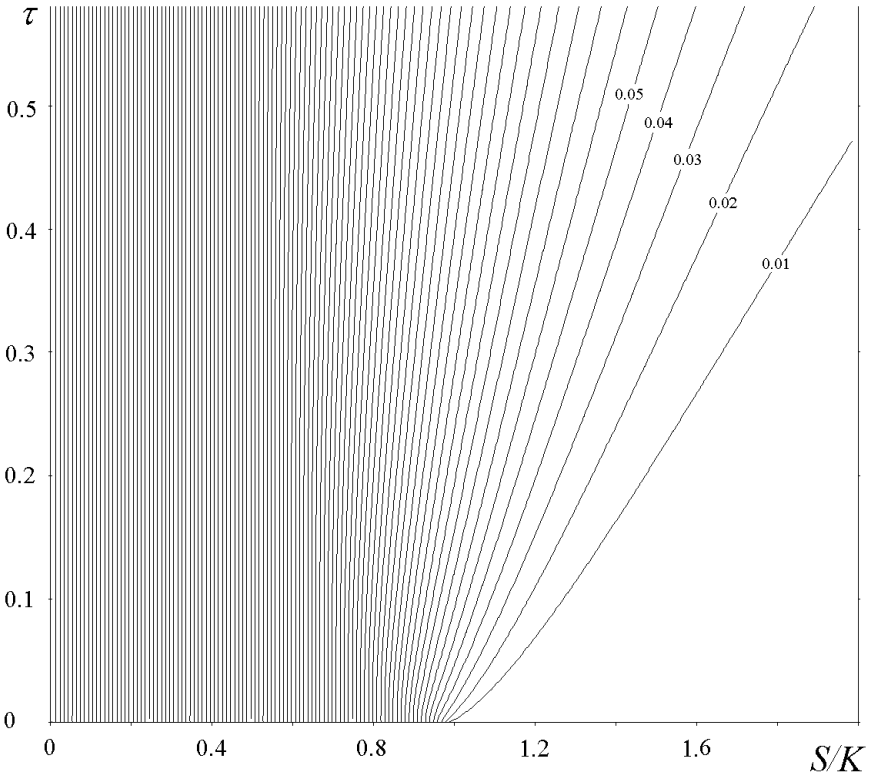

Figure 2.3: "The isolines of the space for the option function $p(s, \tau)$ with $K = 45, \sigma = 0.5, T = 0,5833$"

Chapter 3

Mellin basket transforms

Mellin transforms in two dimensions [13, 14] will be applied in this chapter for the derivation of an integral equation for put options on a basket of two stocks S_1 and S_2. Both underlying stocks pay no dividends and follow a geometric Brownian motion as described earlier according to the assumptions of the Black-Scholes model [1, 7]. This transform was introduced by Panini and Srivastav. Numerical results to support this approach will be presented in the next subchapter.

3.1 European Basket Options

First the representation of the European put option on a basket of two stocks will be determined. It is assumed, that

- K is the exercise price of our basket at time $t = T$;
- The volatility of the market prices are σ_1, σ_2;
- The coefficient of correlation ρ does not depend on time;
- The factor of the baskets growth r, the interest rate, is also assumed to be constant.

We are seeking for the price of the option in the moment of time of the option buying, i.e. $\tilde{p}(S_1, S_2, t)$ at time $t = 0$. Further dimensionless values will be used s_1, s_2, p with reversed time $t = T - \tau$. It means $S_1 = Ks_1$, $S_2 = Ks_2$, $\tilde{p} = Kp$ and the moment of time of selling $t = T$ now is $\tau = 0$. So, we are looking for the option price $p(S_1, S_2, \tau)$ at the final time $\tau = T$. The function of the European put option $p(S_1, S_2, \tau)$ on a basket of two stocks

with $0 < s_1 < \infty$, $0 < s_2 < \infty$ and $\tau > 0$ is satisfied by the Black-Scholes equation

$$\frac{\partial p}{\partial \tau} - \frac{1}{2}\sigma_1^2 s_1^2 \frac{\partial^2 p}{\partial s_1^2} - \rho\sigma_1\sigma_2 s_1 s_2 \frac{\partial^2 p}{\partial s_1 \partial s_2} - \frac{1}{2}\sigma_2^2 s_2^2 \frac{\partial^2 p}{\partial s_2^2} - r s_1 \frac{\partial p}{\partial s_1} + r s_2 \frac{\partial p}{\partial s_2} + rp = 0, \tag{3.1}$$

with the initial and boundary conditions

$$\begin{array}{lll} p(s_1, s_2, 0) & = & \theta(s_1, s_2) = (1 - s_1 - s_2)^+, \\ p(0, 0, t) & = & e^{-r\tau}, \\ p(s_1, s_2, \tau) \to 0 & as & s_1 + s_2 \to \infty. \end{array} \tag{3.2}$$

For the remaining boundaries ($s_1 = 0$, $s_2 > 0$ and $\tau > 0$ and $s_1 > 0$, $s_2 = 0$ and $\tau > 0$) there is no need to pose any conditions, because the unknown function can be defined also from the solution of the equation (3.1). But this condition is related to the condition (7.2) anyway, so it should not be written separately.
We denote the double Mellin transform [13, 14, 15] of $p(s_1, s_2, \tau)$ by $\hat{p}(v_1, v_2, t)$. The function $\hat{p}(v_1, v_2, \tau)$ is a complex function of the complex variables (v_1, v_2), which is defined by Re $v_1 > 0$ and Re $v_2 > 0$

$$\hat{p}(v_1, v_2, \tau) = \int_0^\infty \int_0^\infty p(s_1, s_2, \tau) s_1^{v_1 - 1} s_2^{v_2 - 1} ds_1 ds_2. \tag{3.3}$$

The inverse transformation looks as follows

$$p(s_1, s_2, \tau) = \frac{1}{(2\pi i)^2} \int_{c_1 - i\infty}^{c_1 + i\infty} \int_{c_2 - i\infty}^{c_2 + i\infty} \hat{p}(v_1, v_2, \tau) S_1^{-v_1} S_2^{-v_2} dv_1 dv_2. \tag{3.4}$$

Hence, the outcome of the Mellin transform as in (3.1) can be simplified to

$$\frac{d\hat{p}(v_1, v_2, \tau)}{d\tau} + Q(v_1, v_2)\hat{p}(v_1, v_2, \tau) = 0, \tag{3.5}$$

where

$$Q = Q(v_1, v_2) = \frac{\sigma_1^2}{2} v_1^2 + \rho\sigma_1\sigma_2 v_1, v_2 + \frac{\sigma_2^2}{2} v_2^2 - (r - \frac{\sigma_1^2}{2}) v_1 - (r - \frac{\sigma_2^2}{2}) v_2 - r. \tag{3.6}$$

We obtain the general solution of (3.5)

$$\hat{p}(v_1, v_2, \tau) = A(v_1, v_2)e^{-Q(v_1,v_2)\tau}, \tag{3.7}$$

where $A(v_1, v_2)$ is a constant of integration. Taking the initial condition (3.2) to account we get $A(v_1, v_2) = \hat{\theta}(v_1, v_2)$, where $\hat{\theta}$ is the Mellin transform of the payoff condition (3.2):

$$\begin{aligned} A(v_1, v_2) &= \int_0^\infty \int_0^\infty \hat{\theta}(s_1, s_2) s_1^{v_1-1} s_2^{v_1-1} ds_1 ds_2\} \\ &= \int_0^\infty \int_0^\infty (1 - s_1 - s_2)^+ s_1^{v_1-1} s_2^{v_1-1}\, ds_1\, ds_2 \\ &= \iint\limits_{s_1+s_2<1, s_1>0, s_2>0} (1 - s_1 - s_2) s_1^{v_1-1} s_2^{v_1-1} ds_1 ds_2. \end{aligned} \tag{3.8}$$

For the computation of $A(v_1, v_2)$ we make the following substitution of the variables $(s_1, s_2) \to (\alpha, \beta)$

$$s_1 = \alpha\beta, \qquad s_1 = \alpha(1 - \beta). \tag{3.9}$$

Then $ds_1 ds_2 \to \alpha d\alpha d\beta$, the domain

$$(s_1 + s_2 < 1, s_1 > 0, s_2 > 0) \to (0 < \alpha < 1, 0 < \beta < 1)$$

and the integral (3.8) will be transformed to

$$\begin{aligned} A(v_1, v_2) &= \iint\limits_{0<\alpha<1, 0<\beta<1} (1 - \alpha)(\alpha\beta)^{v_1-1}(\alpha(1 - \beta))^{v_2-1} \alpha d\alpha d\beta \\ &= \int_0^1 \int_0^1 (1 - \alpha)(\alpha\beta)^{v_1-1}(\alpha(1 - \beta))^{v_2-1} \alpha d\alpha d\beta. \end{aligned}$$

We separate the variables and integrate along α

$$\begin{aligned} A(v_1, v_2) &= \int_0^1 (1 - \alpha)\alpha^{v_1+v_2-1}\, d\alpha \cdot \int_0^1 \beta^{v_1-1}(1 - \beta)^{v_2-1}\, d\beta \\ &= \left(\frac{1}{v_1 + v_2} - \frac{1}{v_1 + v_2 - 1}\right) \cdot \int_0^1 \beta^{v_1-1}(1 - \beta)^{v_2-1}\, d\beta. \end{aligned} \tag{3.10}$$

Taking into account that the integral along β is the beta-function, we get

$$\begin{aligned} A(v_1, v_2) &= \frac{1}{(v_1+v_2)(v_1+v_2+1)} \cdot \int_0^1 \beta^{v_1-1}(1-\beta)^{v_2-1}\, d\beta \\ &= \frac{B(v_1, v_2)}{(v_1+v_2)(v_1+v_2+1)}. \end{aligned} \tag{3.11}$$

The combination of (3.4), (3.7) and (3.11) provides the expression for the price of the European put option on a basket of two stocks in the integral form

$$p(s_1, s_2, \tau) = \frac{1}{(2\pi i)^2} \int_{c_1-i\infty}^{c_1+i\infty} \int_{c_2-i\infty}^{c_2+i\infty} \frac{B(v_1, v_2)}{(v_1+v_2)(v_1+v_2+1)} e^{\tau Q(v_1,v_2)} S_1^{-v_1} S_2^{-v_2} dv_1 dv_2. \tag{3.12}$$

Alternatively, the representation of the beta-functions through gamma-function can be used and yields

$$p(s_1, s_2, \tau) = \frac{1}{(2\pi i)^2} \int_{c_1-i\infty}^{c_1+i\infty} \int_{c_2-i\infty}^{c_2+i\infty} \frac{\Gamma(v_1)\Gamma(v_2)}{\Gamma(v_1+v_2+2)} e^{\tau Q(v_1,v_2)} S_1^{-v_1} S_2^{-v_2} dv_1 dv_2. \tag{3.13}$$

3.2 American Basket Options

Now we will consider the double Mellin transforms in order to derive the expression for the price of an American put on a basket of two stocks. It differs to a European put option due to the right to sell the basket of stocks at any time from 0 to T and not only at the final time T. Let us denote the price of an American put option $\widetilde{P}(S_1, S_2, t)$.

As was already shown in the one-dimensional case, a free boundary problem emerges as a result of the early exercise aspect of the American option. The free boundary is now a curve in the plane (S_1, S_2), in each time t, for an option on a 2-basket. Over this curve is the area of assets that are more expensive than K, the put is held in this region (the continuation region, is refered to by C_t) the price is defined as the solution of the Black-Scholes equation for a European put option in the two-dimensional case.

Below the curve is the so called exercise region ε_t. The put is exercised there

and the payoff function is denoted by $\widetilde{P}(S_1, S_2, t)$. The option price is determined there by the condition of the obligated (unconditional or commited) selling on the settled (agreed) price K. In this case the option price is defined by the difference between the agreed price and the market price of both assets:

$$\widetilde{P}(S_1, S_2, t) = K - (S_1 + S_2). \tag{3.14}$$

So, that means in the exercise region ε_t the domain for the Black-Scholes equation may be broadened. The free boundary between these two regions is an additional unknown value in the general case. The behavior of the option price in both regions can be described by an inhomogeneous Black-Scholes equation, which right part depends on the unknown free boundary.
As for the European case we will use here the dimensionless values s_1, s_2 and P and the reversed time $\tau = T - t$ as well.

$$S_1 = Ks_1, \quad S_2 = Ks_2, \quad \hat{P} = KP$$

It means that the latest time to sell the basket $t = T$ is now $\tau = 0$. The unknown function of the option price reads now $P(s_1, s_2, \tau)$ and the solution of (3.14) is given by

$$P(s_1, s_2, \tau) = 1 - s_1 - s_2. \tag{3.15}$$

Hence, the price of the put $P = P(s_1, s_2, \tau)$ can be defined by the non-homogeneous forward-in-time equation

$$\frac{\partial P}{\partial t} + \frac{1}{2}\sigma_1^2 s_1^2 \frac{\partial^2 P}{\partial s_1^2} + \rho\sigma_1\sigma_2 s_1 s_2 \frac{\partial^2 P}{\partial s_1 \partial s_2} + \frac{1}{2}\sigma_2^2 s_2^2 \frac{\partial^2 P}{\partial s_2^2} + rs_1 \frac{\partial P}{\partial s_1} + rs_2 \frac{\partial p}{\partial s_2} - rP = f, \tag{3.16}$$

$$0 < s_1, s_2 < \infty, \tau > 0,$$

where

$$f = f(s_1, s_2, \tau) = \begin{cases} r & if \quad (s_1, s_2) \in \varepsilon_\tau \\ 0 & if \quad (s_1, s_2) \notin \varepsilon_\tau, \end{cases} \tag{3.17}$$

with the initial condition

$$P(s_1, s_2, 0) = \theta(s_1, s_2) = 1 - s_1 - s_2, \tag{3.18}$$

and the boundary condition

$$P(s_1, s_2, \tau) \to 0 \quad s_1 + s_2 \to \infty. \tag{3.19}$$

As for the free boundary conditions the right part $f(s_1, s_2, \tau) = r$ in (3.17) inside the region ε_t is given in order to the solution of (3.15) satisfies the equation (3.16). For the "smooth-pasting" of (3.16) and (3.15) we need to have the following conditions on the boundary $\partial\varepsilon_t$ of ε_t, whereby the unknown free boundary is given by

$$P(s_1, s_2, \tau)\Big|_{\partial\varepsilon_\tau} = 1 - s_1 - s_2, \tag{3.20}$$

$$\frac{\partial P}{\partial s_1}\bigg|_{\partial\varepsilon_\tau} = \frac{\partial P}{\partial s_2}\bigg|_{\partial\varepsilon_\tau} = -1, \tag{3.21}$$

where (s_1, s_2) are points on the free boundary.

Remark 1

The behaviour of the price of European and American put option by $s_1, s_2 \to 0$ are different, because this point falls into the region ε_t. Hence, the price for European put option is $p(0, 0, \tau) = e^{-r\tau}$ (see (3.2)) and for American put option is $P(0, 0, \tau) = 1$.

Remark 2

Comparing the solution of (3.15) and the initial condition (3.18) we can see that the boundary of the region ε_τ is the line $s_1 + s_2 = 1$ in the initial time $\tau = 0$ and the region looks as follows

$$\varepsilon_\tau\Big|_{\tau=0} = \{s_1 + s_2 \le 1, \quad s_1 \ge 0, \quad s_2 \ge 0\}.$$

The same procedure of Mellin transform as for the European put option should be done here. Let us denote the double Mellin transform from the function $P(s_1, s_2, \tau)$ as $\hat{P}(v_1, v_2, \tau)$. The function $\hat{P}(v_1, v_2, \tau)$ is the complex function of complex variables (v_1, v_2), which is defined by Re $v_1 > 0$, Re $v_2 > 0$ and can be expressed as

$$\hat{P}(v_1, v_2, \tau) = \int_0^\infty \int_0^\infty P(s_1, s_2, \tau) s_1^{v_1 - 1} s_2^{v_2 - 1} ds_1 ds_2. \tag{3.22}$$

The inverse transform is defined as integral from the complex variables v_1, v_2

$$P(s_1, s_2, \tau) = \frac{1}{(2\pi i)^2} \int_{c_1 - i\infty}^{c_1 + i\infty} \int_{c_2 - i\infty}^{c_2 + i\infty} \hat{P}(v_1, v_2, \tau) s_1^{-v_1} s_2^{-v_2} dv_1 dv_2. \tag{3.23}$$

Applying the Mellin transform to (3.16) we simplify as before and arrive at the nonhomogeneous ordinary differential equation.

$$\frac{d}{d\tau}\hat{P}(v_1, v_2, \tau) - Q(v_1, v_2) \cdot \hat{P}(v_1, v_2, \tau) = \hat{f}(v_1, v_2, \tau), \tag{3.24}$$

where the $Q(v_1, v_2)$ is the same coefficient as in the formula (3.6) for the European case of the put option. The function $\hat{f}(v_1, v_2, \tau)$ is the Mellin transform of $f(s_1, s_2, \tau)$ (3.17) which includes the free boundary as well. The general solution of this equation, i.e. the general integral, is the sum of the general solution of the homogeneous equation and the particular integral

$$\hat{P}(v_1, v_2, \tau) = A(v_1, v_2)e^{Q(v_1,v_2)\tau} + \int_0^\tau \hat{f}(v_1, v_2, \xi)e^{Q(v_1,v_2)(\tau-\xi)}d\xi. \tag{3.25}$$

The expression for $A(v_1, v_2)$ is determined by the initial conditions with $\tau = 0$ (3.18) and coincide with the expression from the European case as well:

$$A(v_1, v_2) = \frac{B(v_1, v_2)}{(v_1 + v_2)(v_1 + v_2 + 1)}. \tag{3.26}$$

Let us find the expression for $\hat{f}(v_1, v_2, \tau)$

$$\hat{f}(v_1, v_2, \tau) = \int_0^\infty \int_0^\infty f(s_1, s_2, \tau)s_1^{v_1-1}s_2^{v_2-1}ds_1ds_2 = \iint_{\epsilon_\tau} rs_1^{v_1-1}s_2^{v_2-1}ds_1ds_2. \tag{3.27}$$

Further we make the substitutions of the variables $(s_1, s_2) \to (\alpha, \beta)$ like in the European case

$$s_1 = \alpha\beta, \qquad s_2 = \alpha(1 - \beta). \tag{3.28}$$

These variables are suitable because the region ε_τ in time $\tau = 0$ depicts a rectangle. Recalling the Remark 2

$$\varepsilon_\tau\Big|_{\tau=0} = \{0 \le \alpha \le 1, \quad 0 \le \beta \le 1\}, \tag{3.29}$$

we see that with the growth of τ just only one boundary $\alpha = 1$ is changing. Denote the function for the free boundary in an arbitrary moment τ as $\tilde{\alpha}(\tau, \beta)$ we get

$$\varepsilon_\tau = \{0 \le \alpha \le \tilde{\alpha}(\tau, \beta), \quad 0 \le \beta \le 1\} \tag{3.30}$$

By applying this substitution of the variables to (3.27) with $ds_1 ds_2 \to \alpha d\alpha d\beta$ the integration region reads

$$\hat{f}(v_1, v_2, \tau) = r \iint\limits_{\substack{0<\alpha<\tilde{\alpha}, \\ 0<\beta<1}} s_1^{v_1-1} s_2^{v_2-1} ds_1 ds_2 = r \int\limits_0^1 \int\limits_0^{\tilde{\alpha}} (\alpha\beta)^{v_1-1} (\alpha - \alpha\beta)^{v_2-1} \alpha d\alpha d\beta. \tag{3.31}$$

We separate the variables and compute the integral along α

$$\hat{f}(v_1, v_2, \tau) = r \int\limits_0^1 \Big(\int\limits_0^{\tilde{\alpha}} \alpha^{v_1+v_2-1} d\alpha \Big) \beta^{v_1-1} (1-\beta)^{v_2-1} d\beta =$$

$$= \frac{r}{v_1 + v_2} \int\limits_0^1 \tilde{\alpha}^{v_1+v_2} \beta^{v_1-1} (1-\beta)^{v_2-1} d\beta. \tag{3.32}$$

This expression can be simplified by the assumption that the function $\tilde{\alpha}(\tau\beta)$ does not depend on β. Then we can take the $\tilde{\alpha}$ out of the integral. It turns out to the Beta function

$$\hat{f}(v_1, v_2, \tau) = \frac{r\tilde{\alpha}(\tau)^{v_1+v_2}}{v_1 + v_2} B(v_1, v_2), \qquad \text{if} \quad \tilde{\alpha}(\tau, \beta) \quad \text{independent of} \quad \beta. \tag{3.33}$$

As a result of the combination of (3.23), (3.25), (3.26) and (3.31) the expression for the price of the American put option looks as follows

$$P(s_1, s_2, \tau) = \frac{1}{(2\pi i)^2} \int\limits_{c_2^-}^{c_2^+} \int\limits_{c_1^-}^{c_1^+} \frac{B(v_1, v_2) e^{\tau Q(v_1, v_2)}}{(v_1 + v_2)(v_1 + v_2 + 1)} s_1^{-v_1} s_2^{-v_2} dv_1 dv_2 +$$

$$+ \frac{r}{(2\pi i)^2} \int\limits_0^\tau \int\limits_{c_2^-}^{c_2^+} \int\limits_{c_1^-}^{c_1^+} \frac{\Big(\int_0^1 \tilde{\alpha}(\xi, \beta)^{v_1+v_2} \beta^{v_1-1} (1-\beta)^{v_2-1} d\beta \Big) e^{(\tau-\xi) Q(v_1, v_2)}}{(v_1 + v_2) s_1^{v_1} s_2^{v_2}} dv_1 dv_2 d\xi. \tag{3.34}$$

Identifications of limits of integrating $c_1^- = c_1 - i\infty$, $c_1^+ = c_1 + i\infty$, $c_2^- = c_2 - i\infty$ and $c_2^+ = c_2 + i\infty$ here are introduced for brevity. The first term

in (3.34) is the price of the European put option, the second term contains an unknown function of the free boundary $\tilde{\alpha}(\xi,\beta)$. An equation for the free boundary we shall receive from the condition (3.20), which should be evaluated on the free boundary. We change the variables (s_1,s_2) to the variables (α,β). Let us remind, that $s_1=\alpha\beta$, $s_2=\alpha(1-\beta)$.

$$P(s_1,s_2,\tau)\Big|_{\partial\epsilon_\tau}=1-s_1-s_2 \qquad \Rightarrow \qquad \alpha+P(\alpha,\beta,\tau)\Big|_{\alpha=\tilde{\alpha}(\tau,\beta)}=1. \quad (3.35)$$

First we express the option price. It means we take the (3.35) by (α,β) and use another notation ϕ for the inner integral

$$P(\alpha,\beta,\tau)=\frac{1}{(2\pi i)^2}\int\limits_{c_2^-}^{c_2^+}\int\limits_{c_1^-}^{c_1^+}\frac{B(v_1,v_2)e^{\tau Q(v_1,v_2)}}{(v_1+v_2)(v_1+v_2+1)\alpha^{(v_1+v_2)}\beta^{v_1}(1-\beta)^{v_2}}dv_1dv_2+$$

$$+\frac{r}{(2\pi i)^2}\int\limits_0^{\tau}\int\limits_{c_2^-}^{c_2^+}\int\limits_{c_1^-}^{c_1^+}\frac{\left(\int_0^1\tilde{\alpha}(\xi,\phi)^{v_1+v_2}\phi^{v_1-1}(1-\phi)^{v_2-1}d\phi\right)e^{(\tau-\xi)Q(v_1,v_2)}}{(v_1+v_2)\alpha^{(v_1+v_2)}\beta^{v_1}(1-\beta)^{v_2}}dv_1dv_2d\xi. \quad (3.36)$$

Hence, the equation of the free boundary $\tilde{\alpha}(\xi,\beta)$ is given by

$$\tilde{\alpha}(\tau,\beta)+\frac{1}{(2\pi i)^2}\int\limits_{c_2^-}^{c_2^+}\int\limits_{c_1^-}^{c_1^+}\frac{B(v_1,v_2)e^{\tau Q(v_1,v_2)}}{(v_1+v_2)(v_1+v_2+1)\tilde{\alpha}(\tau,\beta)^{(v_1+v_2)}\beta^{v_1}(1-\beta)^{v_2}}dv_1dv_2+$$

$$+\frac{r}{(2\pi i)^2}\int\limits_0^{\tau}\int\limits_{c_2^-}^{c_2^+}\int\limits_{c_1^-}^{c_1^+}\frac{\left(\int_0^1\tilde{\alpha}(\xi,\phi)^{v_1+v_2}\phi^{v_1-1}(1-\phi)^{v_2-1}d\phi\right)e^{(\tau-\xi)Q(v_1,v_2)}}{(v_1+v_2)\tilde{\alpha}(\tau,\beta)^{(v_1+v_2)}\beta^{v_1}(1-\beta)^{v_2}}dv_1dv_2d\xi=1. \quad (3.37)$$

The function, we are seeking for, is located in the inner integral, it makes serious complications by the numerical solution of this integral equation. It makes sense to consider the simplified equation by the assumption that the function $\tilde{\alpha}(\tau,\beta)$ does not depend on the second argument. We can take for instance $\beta=0.5$ (i.e. the middle point of the free boundary).
We denote

$$\bar{\alpha}(\tau)=\tilde{\alpha}(\tau,\beta)\Big|_{\beta=0.5}$$

and use this value in the inner integral. As a result we obtain the approximated equation for a middle point of the free boundary

$$\bar{\alpha}(\tau) + \frac{1}{(2\pi i)^2} \int\limits_{c_2^-}^{c_2^+} \int\limits_{c_1^-}^{c_1^+} \frac{B(v_1,v_2) e^{\tau Q(v_1,v_2)} 2^{(v_1+v_2)}}{(v_1+v_2)(v_1+v_2+1)\bar{\alpha}(\tau)^{(v_1+v_2)}} dv_1 dv_2 +$$

$$+ \frac{r}{(2\pi i)^2} \int\limits_0^\tau \int\limits_{c_2^-}^{c_2^+} \int\limits_{c_1^-}^{c_1^+} \frac{B(v_1,v_2) e^{(\tau-\xi)Q(v_1,v_2)} 2^{v_1+v_2}}{(v_1+v_2)} \frac{\bar{\alpha}(\xi)^{v_1+v_2}}{\bar{\alpha}(\tau)^{v_1+v_2}} dv_1 dv_2 d\xi = 1. \quad (3.38)$$

Accordingly, the formulae (3.36) is converted to

$$P(\alpha,\beta,\tau) = \frac{1}{(2\pi i)^2} \int\limits_{c_2^-}^{c_2^+} \int\limits_{c_1^-}^{c_1^+} \frac{B(v_1,v_2) e^{\tau Q(v_1,v_2)} \beta^{-v_1} (1-\beta)^{-v_2}}{(v_1+v_2)(v_1+v_2+1)\alpha^{(v_1+v_2)}} dv_1 dv_2 +$$

$$+ \frac{r}{(2\pi i)^2} \int\limits_0^\tau \int\limits_{c_2^-}^{c_2^+} \int\limits_{c_1^-}^{c_1^+} \frac{B(v_1,v_2) \bar{\alpha}(\xi)^{(v_1+v_2)} e^{(\tau-\xi)Q(v_1,v_2)}}{(v_1+v_2)\alpha^{(v_1+v_2)} \beta^{v_1} (1-\beta)^{v_2}} dv_1 dv_2 d\xi. \quad (3.39)$$

For the transition from (α,β) to the old variables (s_1,s_2) the formulas $\alpha = s_1+s_2$, $\beta = s_1/(s_1+s_2)$ are used

$$\bar{P}(s_1,s_2,\tau) = \frac{1}{(2\pi i)^2} \int\limits_{c_2^-}^{c_2^+} \int\limits_{c_1^-}^{c_1^+} \frac{B(v_1,v_2) e^{\tau Q(v_1,v_2)}}{(v_1+v_2)(v_1+v_2+1) s_1^{v_1} s_2^{v_2}} dv_1 dv_2 +$$

$$+ \frac{r}{(2\pi i)^2} \int\limits_0^\tau \int\limits_{c_2^-}^{c_2^+} \int\limits_{c_1^-}^{c_1^+} \frac{B(v_1,v_2) e^{(\tau-\xi)Q(v_1,v_2)}}{(v_1+v_2) s_1^{v_1} s_2^{v_2}} \bar{\alpha}(\xi)^{v_1+v_2} dv_1 dv_2 d\xi. \quad (3.40)$$

The integral equation (3.38) and formulae (3.40) give the solution for the price of the American put option in case of two assets in the simplified statement.

3.3 Motivation of Newton's method for an American Put Option

The numerical solution of the integral equations was conducted for two underlying assets in the simplified statement above. For finding the free boundary

function $\bar{\alpha}(\tau)$ the next integral equation was solved

$$\bar{\alpha}(\tau) - 1 + \frac{1}{(2\pi i)^2} \int\limits_{c_1-i\infty}^{c_1+i\infty} \int\limits_{c_2-i\infty}^{c_2+i\infty} \frac{B(v_1,v_2) e^{\tau Q(v_1,v_2)} 2^{(v_1+v_2)}}{(v_1+v_2)(v_1+v_2+1)\bar{\alpha}(\tau)(v_1+v_2)} dv_1 dv_2 +$$

$$+ \frac{r}{(2\pi i)^2} \int\limits_0^\tau \int\limits_{c_1-i\infty}^{c_1+i\infty} \int\limits_{c_2-i\infty}^{c_2+i\infty} \frac{B(v_1,v_2) e^{\xi Q(v_1,v_2)} 2^{(v_1+v_2)}}{(v_1+v_2)\bar{\alpha}(\tau)^{(v_1+v_2)}} \bar{\alpha}(\tau-\xi)^{(v_1+v_2)} dv_1 dv_2 d\xi = 0. \tag{3.41}$$

After finding $\bar{\alpha}(\tau)$ in order to determine the price of the option for the some values (s_1, s_2, τ) we use the following formulae

$$\bar{P}(s_1,s_2,\tau) = \frac{1}{(2\pi i)^2} \int\limits_{c_1-i\infty}^{c_1+i\infty} \int\limits_{c_2-i\infty}^{c_2+i\infty} \frac{B(v_1,v_2) e^{\tau Q(v_1,v_2)}}{(v_1+v_2)(v_1+v_2+1) s_1^{v_1} s_2^{v_2}} dv_1 dv_2 +$$

$$+ \frac{r}{(2\pi i)^2} \int\limits_0^\tau \int\limits_{c_1-i\infty}^{c_1+i\infty} \int\limits_{c_2-i\infty}^{c_2+i\infty} \frac{B(v_1,v_2) e^{\xi Q(v_1,v_2)}}{(v_1+v_2) s_1^{v_1} s_2^{v_2}} \bar{\alpha}(\tau-\xi)^{v_1+v_2} dv_1 dv_2 d\xi. \tag{3.42}$$

For the numerical solution of the integral equation (3.41) we introduce the uniform grid $\{\tau_0, \tau_1, \tau_2, ..., \tau_M\}$ on the interval $\tau \in [0, T]$. We designate as α_j the unknown values $\bar{\alpha}(\tau_j)$. It is known that $\alpha_0 = 1$. With the help of (3.41) we construct the equations for others values with $j = 1, 2, ..., M$. Integrals in (3.41) are represented as the sums using the trapezoidal quadrature rule on the same grid $\{\tau_0, \tau_1, \tau_2, ..., \tau_M\}$. In outcome we build a system of non-linear algebraic equations, which one in a brief view can be presented as (3.43). This kind is similar to one that was in case of one underlying asset.

$$\begin{aligned}
&\alpha_1 - 1 + E(\alpha_1, \tau_1) + G(\alpha_0, \alpha_1, \tau_1) = 0 \\
&\alpha_2 - 1 + E(\alpha_2, \tau_2) + G(\alpha_0, \alpha_1, \alpha_2, \tau_2) = 0 \\
&\alpha_3 - 1 + E(\alpha_3, \tau_3) + G(\alpha_0, \alpha_1, \alpha_2, \alpha_3, \tau_2) = 0 \\
&\cdots \\
&\alpha_j - 1 + E(\alpha_j, \tau_j) + G(\alpha_0, \alpha_1, \alpha_2, ..., \alpha_j, \tau_j) = 0 \\
&\cdots \\
&\alpha_M - 1 + E(\alpha_M, \tau_M) + G(\alpha_0, \alpha_1, \alpha_2, ..., \alpha_M, \tau_M) = 0
\end{aligned} \tag{3.43}$$

The system is solved sequentially "from the top downward". The first equation is used to find α_1. At known α_1 the second equation is transformed to a scalar equation with one unknown. This equation is used to find α_2. At known α_1 and α_2 from the third equation we find α_3. And so on. This feature in the system needs very labored calculations for the functions E and G, which contain repeated integrals on complex variables (v_1, v_2) in the infinite area and a special beta-function that is expressed as an integral as well.
By the numerical solution was used the following computational features to increase of effectiveness.

- The iterative Newton's method with initial approximation $\alpha_j^0 = \alpha_{j-1}$ was used to solve every j-th equation step by step in top-down direction (n is number of the iteration)

$$\alpha_j^{n+1} = \alpha_j^n + \Delta, \quad \text{where} \quad \Delta = -\frac{\alpha_j^n - 1 + E(\alpha_j^n, \tau_j) + G(\alpha_0, \alpha_1, ..., \alpha_j^n, \tau_j)}{1 + \frac{\partial E(\alpha_j^n, \tau_j)}{\partial \alpha_j^n} + \frac{\partial G(\alpha_0, \alpha_1, ..., \alpha_j^n, \tau_j)}{\partial \alpha_j^n}}. \tag{3.44}$$

- The derivatives in (3.44) were computed as finite differences with small h, for example

$$\frac{\partial E(\alpha_j^n, \tau_j)}{\partial \alpha_j^n} \approx \frac{E(\alpha_j^n + h, \tau_j) - E(\alpha_j^n - h, \tau_j)}{2h}. \tag{3.45}$$

- The iterations terminated when the stopping condition is satisfied $|\Delta| < 10^{-8}\alpha_j^n$. Usually it was required from 3 up to 8 iterations.

- The repeated integrals on complex variables ($v_1 = c_1 + iy_1$, $v_2 = c_2 + iy_2$) are computed on region 240×240, i.e.

$$-120 < y_1 < 120, \qquad -120 < y_2 < 120, \tag{3.46}$$

  which one covered with a grid 720×720. For the calculus of integrals the trapezoidal rule was used as well. Taking into account that the real part of integrands (namely it is included in the formulas) is symmetrical on the arguments. Its value in a point (y_1, y_2) coincides the value in a point $(-y_1, -y_2)$ that allows to reduce the region of integrating 2-times.

- By the computation of the repeated complex integrals we take into account that the integrand goes to zero when $|y_1|$ and $|y_2|$ increase. The summation was done on a square

$$-N < y_1 < N, \qquad -N < y_2 < N, \tag{3.47}$$

where N was determined during the computation of sums from the condition that the maximum of the integrand modulus on a square perimeter less than $< 10^{-7}$. This procedure allow approximately in 100-times to diminish costs of CPU-time for calculations of repeated integrals.

- For calculus of the Beta-function the effective and rather precise analytical formula for logarithm of a function was used, through which the Beta-function is expressed (see text of the program code).

This list of features has allowed in many times to diminish computing costs in the solution of the integral equation (3.43). In average, on a grid with $M = 200$ the time of the solution on a computer with the processor 2.5 GHz was $10 - 15$ minutes.

3.4 An interpretation of the results

Now the results for 2-basket option will be discussed with the help of the graphs and tables [8], Figure 3.1. The comparison of the results for the free boundary S^* with parameters $K = 45, \sigma_1 = \sigma_2 = 0.5, r = 0.4888, \rho = 1$ with small τ. In order to show the correctness of the program code we compare the special case of 2-basket with 1-basket with the following assumptions. On the Figure 3.1 the testing result for 2-basket is presented in the case when its solution is identical with the solution of 1-basket problem by $\sigma_1 = \sigma_2 = 0.5$ and $\rho = 1$. The solution for these data can be received both by the algorithm 1-basket and by the algorithm 2-basket.

The numerical solution for free boundary is presented at small value τ, that is there, where the greatest inaccuracy of a numerical solution occurs. The large inaccuracy is induced by the infinite derivative of function $S^*(\tau)$ in $\tau = 0$. The solid line shows the high accuracy solution which is computed with the 1-basket algorithm. The small circles are numerical solutions which computed with the 2-basket algorithm on grid with $\Delta\tau = 0.00583$, i.e. in 32 times less, than in the 1-basket.

1. $\circ$ is related to the computation of 2-basket with the grid size $M = 100$ by using trapezoidal rule. By these initial conditions 2-basket is reduced to 1-basket.

2. $\bullet$ are results of the applying of our modification of the trapezoidal method.

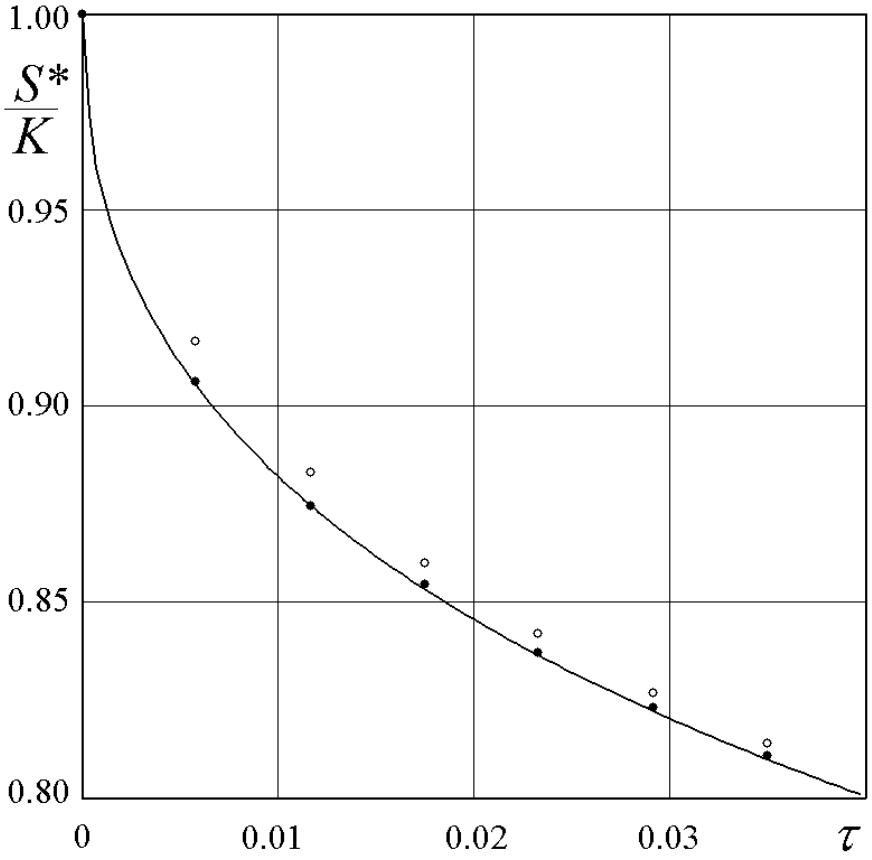

Figure 3.1: Comparison of the results for S^* with parameters $K = 45, \sigma_1 = \sigma_2 = 0.5, r = 0.4888, \rho = 1$ with small τ

As we can see on the figure it is visible, that for 2-basket, as well as in a 1-basket case, the modification improves the accuracy of calculations essentially. So, this modification, which was already introduced in Chapter 2 is more accurate. The numerical artifacts in the left corner at the bottom show

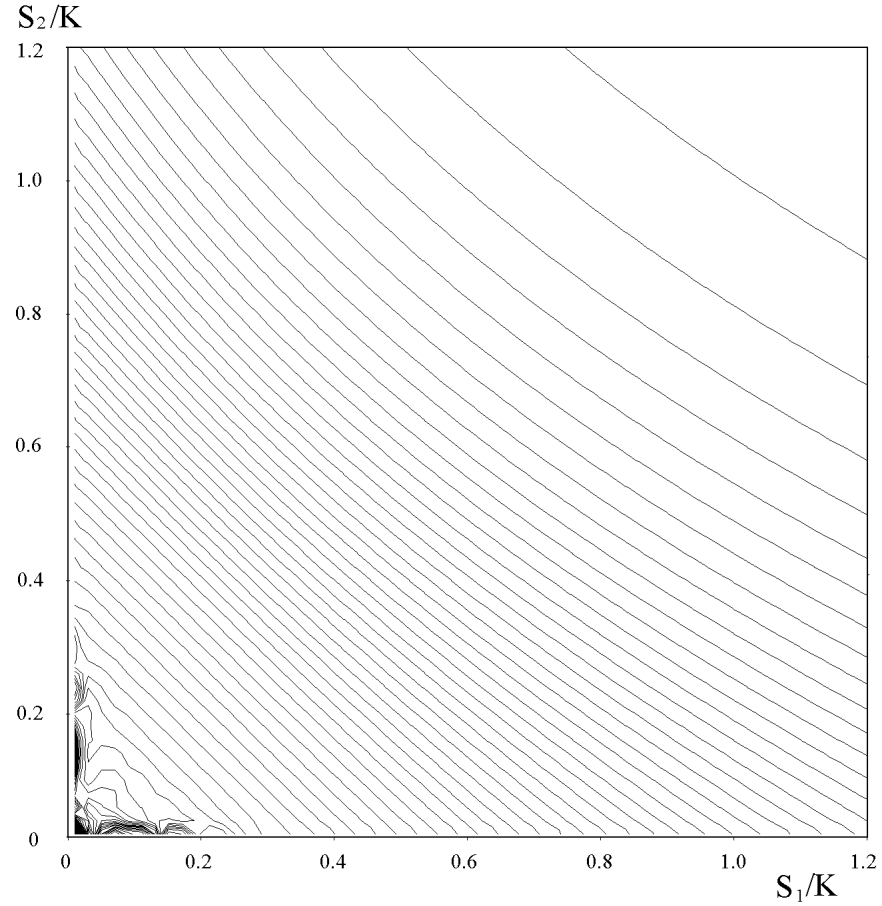

Figure 3.2: The isolines of the option prices $\bar{P}(s_1, s_2, T)$ for 2-basket with $\rho = 0$, $r = 0.0488$, $T = 0.5833$, $\sigma_1 = 0.4$, $\sigma_2 = 0.6$. The numerical artifacts in the left corner at the bottom show the region where the method does not work well.

the region where the method does not work well. The numerical artifacts in the left corner at the bottom show the region where the method does not work well. On following figures are resulted isolines of the option prices $\bar{P}(s_1, s_2, \tau)$ at the fixed value $\tau = T$ for the following initial data $r = 0.0488$, $T = 0.5833$, $\sigma_1 = 0.4$, $\sigma_2 = 0.6$ at various values $\rho = 0.$ and $\rho = 0.4$. In both figures it is visible, that at small values s_1 and s_2 the significant error of

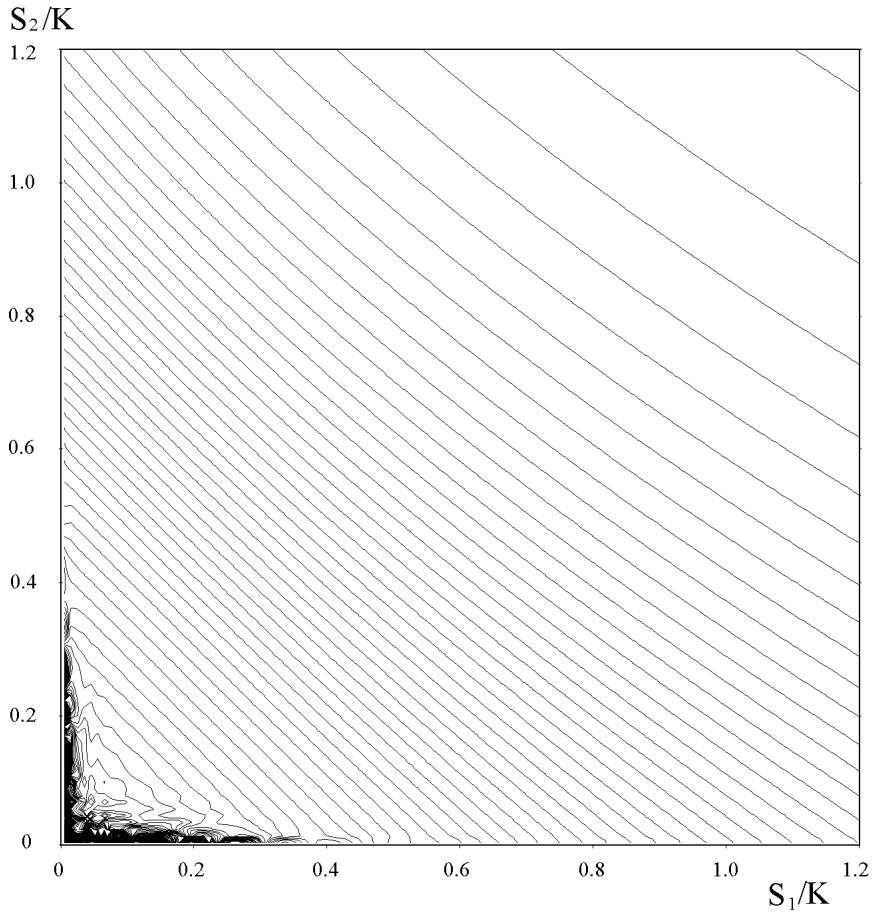

Figure 3.3: The isolines of the option prices $\bar{P}(s_1, s_2, T)$ for 2-basket with $\rho = 0.4$, $r = 0.0488$, $T = 0.5833$, $\sigma_1 = 0.4$, $\sigma_2 = 0.6$. The numerical artifacts in the left corner at the bottom show the region where the method does not work well.

the numerical solution is observed. It is connected by that at these values of arguments the integrand functions in the formulae (9.2) decrease very slowly with growth $|y_1|$ and $|y_2|$. For this reason the used region of integration with the maximal size (3.46) has appeared insufficient for the calculation of integrals with good accuracy. We point out, that in Panini [13, 14] the region with the sizes almost in 10 times less in each direction in comparison with (3.46) was used. That it is obviously not enough for the satisfied solution of the problem.

In conclusion of the paragraph we shall note, that calculation of a field isolines demands significant CPU-time as it is necessary to execute calculations under the formulae (3.42) in many points on a two-dimensional grid. On a grid 200×200 it has demanded 5 hours of CPU-time. In this respect considered integrated method obviously concedes the difference methods. However, if it is required to calculate value of the option price in one or several various points (namely such situation usually arises in practice) the opposite picture takes place. The matter is that difference methods allow to calculate the solution only as layers along time, and for calculation of the solution in one point at moment $\tau = T$ it is required to calculate all solutions in all previous layers. The integrated methods have not this defect. In this sense they have outlook of use for the multi-put options.

Chapter 4

Conclusion

In this paper we examine the integral method based on the Mellin transforms for the numerical solution of Black-Scholes equation. This equation describes the price of the option of American put on one or two underlying assets.
In the realization of the method the following new features were applied.

- Newtons method was applied for the numerical solution of the integral equation along the free boundary. It provides the higher rate of the convergence of the iterative process.

- The modification of the trapezoidal rule was suggested for the discretisation of the integrals, which is taking into account the behaviour of the integrand function near the boundary of the integration interval (the feature of the form $\sqrt{x}$). The applied modification makes it possible to reduce the computational error in approximately ten times compared to the usual trapezoidal rule.

- In the method for 2-basket problem for the computation of the nested integrals along the infinite region the dynamic procedure of shortening of the excess integration region was applied. It will be taken with the accuracy control during the computation. This procedure allows to decrease the CPU-time in a few tens. It needs just a little more additional operative memory of the computer.

The steps listed above let increase the efficiency of the investigated new method approximately in 1000 times. It makes this method more attractive for the applying for the problems of higher dimensions which is typical for basket options.

Chapter 5

Appendix

5.1 The program code for 1-basket

```
/*
 ============================================================================
 Name        : op1newton.c
 Author      : Olessia
 Version     :
 Copyright   : Your copyright notice
 Description : 1-basket in C, Newtons Method
 ============================================================================
 */

#include <math.h>
#include <stdio.h>
#include <fcntl.h>
#include <time.h>
#include <stdlib.h>
#include <string.h>
#include <sys\stat.h>
#include <io.h>

int S_MODE=S_IWRITE|S_IREAD;

// Constants of problem
double K=45., r=0.0488, T=0.5833, sigma=0.5;
//double K=45., r=0.0488, T=0.02, sigma=0.3;
```

```
// Grid parameters for [0,T] and step dtau=T/M
int M,MS,MT;   double dtau;

// Dimension for tabbing of Laplas function with step htab in [0,9]
// F(x)=frac{1}{\sqr{2\pi}}\int_0^x{exp(-x*x/2}dx
int Ntab=20000;
double Ftab[24000], htab;

// Integrand Function
double e2(double x){ return exp(-0.5*x*x); }

// Function of calculating of Ftab[] with step h=9./Ntab (Simpson method)
double TabFtab()
{ int i; double x,spi2,h,A;
  Ftab[0]=0.;  spi2=1./sqrt(2*M_PI); h=htab=9./Ntab;

  for(i=1;i<=Ntab/2;i++)
  { x=i*h; Ftab[i]=Ftab[i-1]+spi2*h*(e2(x-h)+4.*e2(x-0.5*h)+e2(x))/6.;
  }
  A=Ftab[Ntab/2]; Ftab[Ntab/2]=0.;
  for(i=Ntab/2+1;i<=Ntab;i++)
  { x=i*h; Ftab[i]=Ftab[i-1]+spi2*h*(e2(x-h)+4.*e2(x-0.5*h)+e2(x))/6.;
  }

  for(i=Ntab/2;i<=Ntab;i++)Ftab[i]+=A;

return(1);
}

// Function of  N(x) in arbitary point along Ftab[]
double N(double x)
{ int i; double s,f; if(x>9.)return(1.); if(x<-9.)return(0.);
  s=fabs(x); i=(int)(s/htab);
  f=Ftab[i]+(s-i*htab)*(Ftab[i+1]-Ftab[i])/htab; if(x<0.)f=-f;
  return(0.5+f);
}

// Function of  Sfun(tau,M,St) in arbitary point along St[M]
double Sfun(double tau,int M,double *St)
{ int i; double s,f,dt;  if(tau<0.)return(St[0]);
  dt=T/M; i=(int)(tau/dt); if(i>M)return(St[M]);
```

```
  f=St[i]+(tau-i*dt)*(St[i+1]-St[i])/dt;
  return(f);
}

// Function of Eurupean put-option
//
double p(double s,double tau)
{ double d1,d2,f,st; if(tau<0.0000001)return(0.);
  st=sigma*sqrt(tau); d1=(log(s/K)+(r+0.5*sigma*sigma)*tau)/st; d2=d1-st;
  f=K*exp(-r*tau)*N(-d2)-s*N(-d1);
  return(f);
}

// Integrand Function in American put-option (included r*K) with St[]
//
double I(double s,double tau,double ksi,double *St)
{ double d,f; if(ksi<0.0000001 || ksi>tau+0.0000001)return(0.);
  d=(log(s/Sfun(tau-ksi,M,St))+(r-0.5*sigma*sigma)*ksi)/(sigma*sqrt(ksi));
  f=1.*r*K*exp(-r*ksi)*N(-d);
  return(f);
}

double *f; int sq;

// Full integral by the trapecium method with St[]
//
double IntPol(double s,double tau,double *St)
{ int i,j; double dksi,Q,h=0.0001,a;
  dksi=dtau; i=tau/dtau+0.01; if(i<1)return(0.);
  for(j=0;j<=i;j++)f[j]=I(s,tau,j*dksi,St);
  Q=dksi*0.5*(f[0]+f[i]); if(i==1)return(1.6*Q);
  for(j=1;j<i;j++)Q=Q+dksi*f[j];

   //  modify trapecium+
if(sq==1)
{
  //minus
  Q=Q-dksi*(f[0]+f[1])/2.;
  //plus
  Q=Q+dksi*(2.*f[0]+25.*I(s,tau,9.*dksi/25.,St)+9.*f[1])/36.;
}
```

```
  return(Q);
}


//Central Function: Find Szv[] by Newton on grid M, dtau=T/M
//
int SzvOp1(int M,double *Szv)
{ int i,n,j,Niter=0;
  double tau,ksi,dksi,dS,Eur,Integ,dEur,dInt,R,c,A,B;
  double h=0.00005, eps=1e-8;

  dksi=dtau=T/M;

// Initial Value for all S^*(tau)
  for(i=0;i<=M;i++)Szv[i]=K;

tau=0.; i=0; dS=0; n=0;
//printf("%7.4f %10.8f %+11.8f %3i \n", tau,Szv[i],dS,n);

for(i=1;i<=M;i++)
{  tau=i*dtau;
// Initail for Szv[i] from i-1 layer of tau
   Szv[i]=Szv[i-1];

   if(i>=3){ c=(Szv[i-1]-Szv[i-2])/(Szv[i-2]-Szv[i-3]);
     if(c<0.1)c=0.1; if(c>0.9)c=0.9;
     Szv[i]=Szv[i-1]+c*(Szv[i-1]-Szv[i-2]);
   }
// for the others also
   for(j=i+1;j<=M;j++){ Szv[j]=Szv[i]; }

   // The Newton method iterations for Szv[i],  n<=15
   for(n=1;n<=15;n++)
   {
      // Compute  R=Szv-K+p(Szv,tau)+IntPol(tau)
      Eur=p(Szv[i],tau);
      Integ=IntPol(Szv[i],tau,Szv);
      R=Szv[i]-K+Eur+Integ;

     //derivatives as the finite differences
```

```
     dEur=(p(Szv[i]+h,tau)-p(Szv[i]-h,tau))/(h+h);
     Szv[i]+=h;   B=IntPol(Szv[i],tau,Szv);
     Szv[i]-=h+h; A=IntPol(Szv[i],tau,Szv);  Szv[i]+=h;

     dInt=(B-A)/(h+h);  if(i<=3)dInt*=i*1./3;

     // Increment for Szv[i]
     dS=-R/(1.+dEur+dInt);   Szv[i]+=dS;

     // Print
//     printf("%7.4f %10.8f %+11.8f %3i \r", tau,Szv[i],dS/Szv[i],n);

     // Exit, if Newton convergenced and goto new tau
     if(fabs(dS)<eps*Szv[i])
     {
       if(i%10==0)printf("%8.5f %10.8f %+11.8f %3i \n",tau,Szv[i],dS/Szv[i],n);
//     if(n>=8)getch();
       Niter=Niter+n; break;
     }
   }//n
 }//i
 return(Niter);
}

// Dimension points for different saving
double *S0,*S1,*Stch,*DS0,*DS1;


/*************************         ******************/
/************************* M A I N ******************/
/*************************         ******************/

int main ()
{ int i,j,n,k,L=0,Niter,Nvar,Mstart=25;
  double tau,ksi,dksi,R,s,A,sr0,sr1;
  int fn,Xtlen;  char spt[80];
  float RO[1220];

// Forming of Ftab[];
TabFtab();
```

```
// Take of Memory
n=10010;
f   =(double *)calloc(n,8);
S0  =(double *)calloc(n,8);
S1  =(double *)calloc(n,8);
DS0 =(double *)calloc(n,8);
DS1 =(double *)calloc(n,8);
Stch=(double *)calloc(n,8);

MT=1600; sq=0;
M=MT;    Niter=SzvOp1(M,S0);
M=2*MT;  Niter=SzvOp1(M,S1);

for(i=0;i<=MT;i++)Stch[i]=S1[2*i]+(S1[2*i]-S0[i]);

// File of izolines
/*
fn=open("op1.bin",O_BINARY|O_RDWR|O_CREAT|O_TRUNC);
MS=200; Xtlen=4*MS+24; M=MT;
for(i=0;i<=M;i++)
{ tau=i*dtau; RO[0]=1.*MS;  RO[MS+1]=tau;
  for(j=1;j<=MS;j++)
  { s=j*(K+K)/MS;
    if(i==0){ RO[j]=(K-s)/K; if(RO[j]<0.)RO[j]=0.; }
    else    { RO[j]=(p(s,tau)+IntPol(s,tau,Stch))/K;    }
  }
  write(fn,RO,Xtlen);
}
close(fn);
*/

Mstart=50;
for(Nvar=0;Nvar<=4;Nvar++)
{
  M=Mstart*pow(2,Nvar);

  sq=0; Niter=SzvOp1(M,S0);
  sq=1; Niter=SzvOp1(M,S1);

  for(i=0;i<=M;i++)
  { tau=i*T/M;
```

```
    DS0[i]=fabs(S0[i]-Sfun(tau,MT,Stch))/S0[i];
    DS1[i]=fabs(S1[i]-Sfun(tau,MT,Stch))/S1[i];
  }
  sr0=sr1=0.;
  for(i=0;i<=M;i++){ sr0+=fabs(DS0[i]); sr1+=fabs(DS1[i]); }
  sr0=sr0/M;  sr1=sr1/M;
  sprintf(spt,"%c %4i sr0=%11.7f sr1=%11.7f\r\n",'%',M,sr0,sr1);
  printf("\n %s",spt);

if(Nvar==0)fn=open("ap94.ada",O_BINARY|O_WRONLY|O_CREAT|O_TRUNC,S_MODE);
if(Nvar>=1)fn=open("ap94.ada",O_BINARY|O_WRONLY|O_APPEND,S_MODE);
write(fn,spt,strlen(spt));

for(i=0;i<=M;i++)
{ tau=i*T/M;
  sprintf(spt,"%8.4f %12.8f %12.8f\r\n",tau,DS0[i]*1000.,DS1[i]*1000.);
  write(fn,spt,strlen(spt));
}
close(fn);

/*
printf("\n%cSred %6.2f NewtonIter. for one S*[i] at M=%4i eps=%5.2e",
'%',Niter*1./M,M,eps);
printf("\n");
*/

if(getch()==27)exit(0);

}//Nvar

return(0);
}
```

5.2 The program code for 2-basket

```
/*
 ============================================================================
 Name        : op2newton.c
 Author      : Olessia Vassilieva
 Version     :
 Copyright   : Your copyright notice
 Description : 2-basket in C, Ansi-style
 ============================================================================
 */
#include <math.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define Hre(j,i) *(Hre+N3*(j)+(i))
#define Him(j,i) *(Him+N3*(j)+(i))
#define Hz(j,i) *(Hz+N3*(j)+(i))
#define Zz(j,i) *(Zz+N3*(j)+(i))
#define Xst(j,i) *(Xst+N3*(j)+(i))
#define Yst(j,i) *(Yst+N3*(j)+(i))
#define ReBet(j,i) *(ReBet+N3*(j)+(i))
#define ImBet(j,i) *(ImBet+N3*(j)+(i))
#define ReBe(k,j,i) *(ReBe[k]+N3*(j)+(i))
#define ImBe(k,j,i) *(ImBe[k]+N3*(j)+(i))


// Graphics
int IX0=360, IY0=700, IXD=640, IYD=640;
int N0=400,M0=400,N3,M3,Nc,Mc,JA,JB,IB;
int mxx,mxy;

double *Hre,*Him,*Hz,*Zz,*Xst,*Yst,*ReBet,*ImBet,*E[20];
double *qS,*ReBe[12],*ImBe[12];
double ReBtmp[12],ImBtmp[12];
int MakeBe=0;
int OcenSet=1;

double Xa=-160.,Xb=160.,Ya=-160.,Yb=160.;  // for Computation Iso-field

double Ha=-20.,Hb=20.,HL=0.0001; int Hmem=0;      // Discrete Isoline
double Ew1[1600][2],Ew2[1600][2];
```

```
// The constants
//double K=45., rr=0.0488, T=0.5833, sig1=0.3, sig2=0.3, ro=1.0;
double K=45., rr=0.0488, T=0.03, sig1=0.3, sig2=0.3, ro=1.0;


// The number of partitions for T and for 0<zit<1 by the fixed T;
int M=30,L=10;

// The step of the computation dtau=T/M, the step of the integration dzit=1./L;
double dtau,dzit;
// another added values
double step=0.5;
int countI;

// The array for S^*[] with the step dtau
double S[900],DD[900][120]; // Del(tau,zit)


/*********************************************************************/
/*******      FOR 2-BASKET  ****************************************/

double Cmul(double *u,double *v,double *w)
{ double re,im; re=u[0]*v[0]-u[1]*v[1]; im=u[1]*v[0]+u[0]*v[1];
  w[0]=re; w[1]=im; return(re);
}

double ReCmul(double *u,double *v){ return (u[0]*v[0]-u[1]*v[1]); }


double Cdiv(double *u,double *v,double *w)
{ double s,re,im; s=v[0]*v[0]+v[1]*v[1];
  re=(u[0]*v[0]+u[1]*v[1])/s; im=(u[1]*v[0]-u[0]*v[1])/s;
  w[0]=re; w[1]=im; return(re);
}

// Complex w=r^z
double Cpow(double r,double *z, double *w)
{ double a,b,c;
  a=log(r); b=exp(z[0]*a); c=z[1]*a; w[0]=b*cos(c); w[1]=b*sin(c);
  return(w[0]);
}
```

```
// Complex Cexp(z)
double Cexp(double *z, double *w)
{ double b,c;
  b=exp(z[0]); c=z[1]; w[0]=b*cos(c); w[1]=b*sin(c);
  return(w[0]);
}
// Complex Clog(z)
double Clog(double *z, double *w)
{ double r,b,Re,Im;  Re=z[0]; Im=z[1];
  r=sqrt(Re*Re+Im*Im); b=Im/r; w[0]=log(r);  w[1]=asin(b);
  if(Re<0.){ w[1]=M_PI-w[1]; return(w[0]);
}

//  the boundary from arg=-pi/2 till arg=3pi/2
}

double CgamLn(double *x,double *w)
{ double C[8]={
  2.5066282746310005,
  1.0000000000190015,
  76.18009172947146,
 -86.50532032941677,
  24.01409824083091,
 -1.231739572450155,
  0.1208650973866179e-2,
 -0.5395239384953e-5 };

  double y[2],u[2],v[2],z[2]; int k;

  z[0]=z[1]=0.;
  for(k=0;k<=5;k++)
  { y[0]=C[k+2]; y[1]=0.; u[0]=x[0]+k; u[1]=x[1]; Cdiv(y,u,y);
    z[0]+=y[0]; z[1]+=y[1];
  }
  z[0]+=C[1]; z[0]*=C[0]; z[1]*=C[0];  Clog(z,z);
  y[0]=x[0]-0.5; y[1]=x[1];    u[0]=x[0]+4.5; u[1]=x[1];
  Clog(u,v); Cmul(y,v,y);

  w[0]=y[0]-u[0]+z[0];
  w[1]=y[1]-u[1]+z[1];
  return(w[0]);
}
```

```
double BetV99(int N,double *u,double *v,double *w)
{ double p[2],q[2],r[2],s[2];

  CgamLn(u,p); CgamLn(v,q); s[0]=u[0]+v[0]; s[1]=u[1]+v[1]; CgamLn(s,r);
  s[0]=p[0]+q[0]-r[0];
  s[1]=p[1]+q[1]-r[1];
  Cexp(s,w);
  return(w[0]);
}


// Beta function for optimal using
// BetV45(N,w,w,G) (Combi  0.2*TrE + 0.8*Sim = (7A+8B)/15+(7Kr+6Eu)/30
// remade for the Simpsons rule  1 4 2 4 2 4 2 4 1

double BetV45(int N,double *u,double *v,double *w) // N=500-1000
{ int i,k,N1,N2; double a,b,t,dt,s,Ar,Ai,Br,Bi,Kr,Ki,Er,Ei,lo,lm,eo,cs,sn;
  double Tr[1203],Ti[1203],Sr[1203],Si[1203];
  dt=1./N;  a=u[0]-1.;  b=v[0]-1; //ur,vr in [4,5],  a,b in [3,4]
  Tr[0]=Ti[0]=Tr[N]=Ti[N]=0.;
  for(i=1;i<N;i++)
  { t=i*dt; lo=log(t); lm=log(1.-t); eo=exp(a*lo+b*lm);
    s=u[1]*lo+v[1]*lm;  Tr[i]=eo*cos(s); Ti[i]=eo*sin(s);
  }
  for(i=0;i<N;i++)
  { t=i*dt+0.5*dt; lo=log(t); lm=log(1.-t); eo=exp(a*lo+b*lm);
    s=u[1]*lo+v[1]*lm;  Sr[i]=eo*cos(s); Si[i]=eo*sin(s);
  }

  if(MakeBe)
  { for(k=0;k<L;k++)
    { N1=k*N/L; N2=(k+1)*N/L;
      Ar=0.5*(Tr[N1]+Tr[N2]); Ai=0.5*(Ti[N1]+Ti[N2]);
      for(i=N1+1;i<N2;i++){ Ar+=Tr[i]; Ai+=Ti[i]; }
      Br=Bi=0.;  for(i=N1;i<N2;i++){ Br+=Sr[i]; Bi+=Si[i]; }

      ReBtmp[k]=(Ar+2.*Br)*dt/3.;  ImBtmp[k]=(Ai+2.*Bi)*dt/3.;
    }

    w[0]=w[1]=0;
    for(k=0;k<L;k++){ w[0]+=ReBtmp[k]; w[1]+=ImBtmp[k]; }
    return(w[0]);
```

```
  }

  Ar=0.5*(Tr[0]+Tr[N]); Ai=0.5*(Ti[0]+Ti[N]);
  for(i=1;i<N;i++){ Ar+=Tr[i]; Ai+=Ti[i]; }
  Br=Bi=0.;  for(i=0;i<N;i++){ Br+=Sr[i]; Bi+=Si[i]; }
  w[0]=(Ar+2.*Br)*dt/3.;   w[1]=(Ai+2.*Bi)*dt/3.;
  return(w[0]);
}

// Function of deviation S^*(tau) from S1+S2=S^*(tau)
// q=pointer for Stau

double Del(double tau,double z)
{ int k,m; double dz,f,*q;
  dz=1./L; k=(int)(z/dz); m=(int)(tau/dtau+0.01); q=DD[m];
  f=(q[k]+(z-k*dz)*(q[k+1]-q[k])/dz); return(f);
}


double c1=3.6, c2=3.6;  // Constants in the complex integrals

//TabBet computes and save values B(w1,w2) in a grid

int TabBet()
{ int j,i,k,Ib;  double cx[2],cy[2],cu[2],cp[2];
  double dy1,dy2;
  dy1=(Xb-Xa)/M0; dy2=(Yb-Ya)/N0;  Ib=N0/2;
  for(j=0;j<=M0;j++)
  { cx[0]=c1; cx[1]=Xa+j*dy1;
    for(i=Ib;i<=N0;i++)
    { cy[0]=c2; cy[1]=Ya+i*dy2;

//      BetV45(1000,cx,cy,cp);
//      if(MakeBe)
//      { for(k=0;k<L;k++){ ReBe(k,j,i)=ReBtmp[k]; ImBe(k,j,i)=ImBtmp[k];} }
      BetV99(300,cx,cy,cp);

      ReBet(j,i)=cp[0]; ImBet(j,i)=cp[1];
//    if(Hmem){ Hre(j,i)=cp[0]-cu[0];  Xst(j,i)=cx[1]; Yst(j,i)=cy[1]; }
//    if((i-Ib)==(j-Ib)){ printf("%7.4f %12.9f %12.9f %12.9f %12.9f \n",
//       cx[1],cp[0],cp[1],cp[0]-cu[0]),cp[1]-cu[1];
//     }
```

```
    }
  }
  return(1);
}



// European for full 2-basket

double EvDel(double Del, double tau, double z)
{ int j,i,Ib;  double cx[2],cy[2],cu[2],cp[2],cz[2],cv[2];
  double dy1,dy2,PE,D,Log0,Log1,Log2,s11,s22,s12,a,ma;
  double verh[1205],bok[1205];
  int Schet=1;

  if(OcenSet){ JA=0; JB=M0; IB=N0; Schet=0;}

POVTOR:;

  Ib=N0/2; dy1=(Xb-Xa)/M0; dy2=(Yb-Ya)/N0; PE=0.;
  Log0=log(Del);  Log1=log(z); Log2=log(1-z);
  s11=sig1*sig1; s22=sig2*sig2; s12=sig1*sig2;
  /****************** auxiliary array in power***/

  for(j=0;j<=M0;j++)
  { cx[0]=c1; cx[1]=Xa+j*dy1; Cmul(cx,cx,cu);
    Ew1[j][0]=(0.5*s11*(cu[0]+cx[0])-rr*cx[0])*tau;
    Ew1[j][1]=(0.5*s11*(cu[1]+cx[1])-rr*cx[1])*tau;
    verh[j]=bok[j]=0.;
  }
  for(i=0;i<=N0;i++)
  { cy[0]=c2; cy[1]=Ya+i*dy2; Cmul(cy,cy,cu);
    Ew2[i][0]=(0.5*s22*(cu[0]+cy[0])-rr*cy[0])*tau;
    Ew2[i][1]=(0.5*s22*(cu[1]+cy[1])-rr*cy[1])*tau;
  }

  /****************** The basic double of sum for the integrals */
  for(j=JA;j<=JB;j++)
  { cx[0]=c1; cx[1]=Xa+j*dy1;
    for(i=Ib;i<=IB;i++)
    { cy[0]=c2; cy[1]=Ya+i*dy2;
      if(Schet==0 && (i<IB && j<JB))continue;
```

```
      cp[0]=ReBet(j,i); cp[1]=ImBet(j,i);
      cu[0]=cx[0]+cy[0];   cu[1]=cx[1]+cy[1]; Cdiv(cp,cu,cp);
      cu[0]=cx[0]+cy[0]+1; cu[1]=cx[1]+cy[1]; Cdiv(cp,cu,cp);
      Cmul(cx,cy,cz);
      cz[0]=ro*s12*cz[0]*tau+Ew1[j][0]+Ew2[i][0]-(cx[0]+cy[0])*Log0;
      cz[1]=ro*s12*cz[1]*tau+Ew1[j][1]+Ew2[i][1]-(cx[1]+cy[1])*Log0;

      Cexp(cz,cv); if(tau<0.5*dtau)cv[0]-=1.;

      cz[0]=-cx[0]*Log1-cy[0]*Log2;
      cz[1]=-cx[1]*Log1-cy[1]*Log2;
      Cexp(cz,cu);

      Cmul(cv,cu,cu);

      D=ReCmul(cp,cu);   if(i==IB)verh[j]=D; if(j==JB)bok[i]=D;

      D=D*dy1*dy2; if(i==Ib)D=0.5*D;
      PE=PE+D;
    }//i
  }//j

  ma=0.;
  for(j=JA;j<=JB;j++){ a=fabs(verh[j]); if(ma<a)ma=a; }
  for(i=Ib;i<=IB;i++){ a=fabs(bok[i]);  if(ma<a)ma=a; }

 if(OcenSet)
 {
 //  pprintf(20,50,14,1,"Dgr %12.10f %4i %4i %4i",ma,JA,JB,IB);
   if(ma>1.e-6 && JB<M0){ JA--; JB++; IB++; goto POVTOR;}
   if(ma<5.e-7 && JB>M0/2+10){ JA++; JB--; IB--; goto POVTOR;}
 }
 if(Schet==0){ Schet=1; goto POVTOR; }

  PE=2.*PE*exp(-rr*tau)/(4.*M_PI*M_PI);  return(PE);
}


// Integral function for 2-basket from the article of Panini
double IDel(double tau, double z, double ksi)
{ int j,i,k,Ib;  double cx[2],cy[2],cu[2],cp[2],cz[2],cpz[2];
  double dy1,dy2,PE,D,Log0,Log1,Log2,s11,s22,s12,rsk,dz,zit;
```

```
  if(ksi<0.0000001 || ksi>tau+0.0000001)return(0.);
  dy1=(Xb-Xa)/M0; dy2=(Yb-Ya)/N0; PE=0.;   Ib=N0/2;

//  Log0)=log(Del(tau,z)/Del(tau-ksi,z));
  Log0=log(Del(tau,z));  dz=1./L;

  Log1=log(z); Log2=log(1-z);
  s11=sig1*sig1; s22=sig2*sig2; s12=sig1*sig2; rsk=ro*s12*ksi;

  // printf(740,30,1,14,"%4i",countI); countI++;

  /****************** Auxiliary arrays *****/

  for(j=0;j<=M0;j++)
  { cx[0]=c1; cx[1]=Xa+j*dy1; Cmul(cx,cx,cu);
    Ew1[j][0]=(0.5*s11*(cu[0]+cx[0])-rr*cx[0])*ksi-cx[0]*(Log0+Log1);
    Ew1[j][1]=(0.5*s11*(cu[1]+cx[1])-rr*cx[1])*ksi-cx[1]*(Log0+Log1);
  }
  for(i=0;i<=N0;i++)
  { cy[0]=c2; cy[1]=Ya+i*dy2; Cmul(cy,cy,cu);
    Ew2[i][0]=(0.5*s22*(cu[0]+cy[0])-rr*cy[0])*ksi-cy[0]*(Log0+Log2);
    Ew2[i][1]=(0.5*s22*(cu[1]+cy[1])-rr*cy[1])*ksi-cy[1]*(Log0+Log2);
  }
  /****************** The basic double iteration of sum for the integrals*/
  for(j=JA;j<=JB;j++)
  { cx[0]=c1; cx[1]=Xa+j*dy1;
    for(i=Ib;i<=IB;i++)
    { cy[0]=c2; cy[1]=Ya+i*dy2;

      cu[0]=cx[0]+cy[0]; cu[1]=cx[1]+cy[1];
      zit=0.5; Cpow(Del(tau-ksi,zit),cu,cu);
      cp[0]=ReBet(j,i); cp[1]=ImBet(j,i);
      Cmul(cp,cu,cp);

      cu[0]=cx[0]+cy[0]; cu[1]=cx[1]+cy[1]; Cdiv(cp,cu,cp);

      Cmul(cx,cy,cz); cz[0]=rsk*cz[0]+Ew1[j][0]+Ew2[i][0];
      cz[1]=rsk*cz[1]+Ew1[j][1]+Ew2[i][1];
      Cexp(cz,cu);  D=ReCmul(cp,cu);

      D=D*dy1*dy2; if(i==Ib)D=0.5*D;   PE=PE+D;
    }//i
  }//j
```

```
  PE=2.*PE*rr*exp(-rr*ksi)/(4.*M_PI*M_PI);  return(PE);

}

double IntPol(double tau,double z)
{ int i,j; double dksi,Q,f[500];
  i=tau/dtau+0.01; if(i<1)return(0.); dksi=dtau;
  for(j=0;j<=i;j++)f[j]=IDel(tau,z,j*dksi);
  Q=dksi*(f[0]+f[i])/2.; if(i==1)return(Q);
  for(j=1;j<i;j++)Q=Q+dksi*f[j]; return(Q);
}


/*******************************************************/
/************************* M A I N *********************/
/*******************************************************/
/*******************************************************/
/*******************************************************/

int main ()
{ int i,j,n,k,N,l,ii,ll,Niter;
  double tau,ksi,dksi,f0,f1,f2,R,D,In,pro,norm1,norm2,s,h,Q,p,q;
  double x,ox,y,f,Eur,z,dz;
  double S1,S2,Dold[55],Dnew[55];
  double Ds,Dn,dD,IntDel,proI;

  N3=N0+3; M3=M0+3; n=(M0+3)*(N0+3); JA=0; JB=M0; IB=N0;
  for(k=1;k<=7;k++)E[k]=(double *)calloc(n,8);
  Hre=E[1]; Him=E[2]; Hz=E[3]; Xst=E[4]; Yst=E[5];
  ReBet=E[6]; ImBet=E[7];
  for(k=0;k<L;k++){ ReBe[k]=(double *)calloc(n,8);
    ImBe[k]=(double *)calloc(n,8);
  }

 Ha=Ha+0.5*HL;  Nc=N0/2; Mc=M0/2;
 dksi=dtau=T/M;  dz=dzit=1./L;

 TabBet();

// Initial value for S^*

for(i=0;i<=M;i++)
{ tau=i*dtau; S[i]=K*(1.-0.00000005*sqrt(tau/T));
```

```
  for(l=0;l<=L;l++)DD[i][l]=S[i]/K;

}
 z=0.5;  printf("\n");

 for(i=1;i<=M;i++)
 { tau=i*dtau;

// Initial values are taken from the previous layer along tau
   for(l=0;l<=L;l++){ DD[i+1][l]=DD[i][l]=DD[i-1][l]; }

   for(l=0;l<=L;l++)Dnew[l]=Dold[l]=Del(tau,l*dzit);

// Solve  R=Del-1+p(Del,tau,z)+Int(z)=0  with Newtons Method for all z

   countI=0;
   for(n=1;n<=100;n++)
   {
     l=L/2;

       if(kbhit())if(getch()==27)exit(0); // just escape with Esc

       z=l*dzit; Dn=Dnew[l]; Ds=Dold[l];

       if(n==1)OcenSet=1;
       Hmem=0;  Eur=EvDel(Dn,tau,z); Hmem=0;
       OcenSet=0;

       IntDel=IntPol(tau,z);        R=Dn-1+Eur+IntDel;

    //The derivation
       h=0.0001; pro=(EvDel(Dn+h,tau,z)-Eur)/h;
       DD[i][l]+=h; proI=1.00*(IntPol(tau,z)-IntDel)/h; DD[i][l]-=h;

       if(i<=3)proI*=i*1./3;

       dD=-R/(1.+pro+proI);   Dn=Dn+dD; Dnew[l]=Dn;

      printf("%7.4f %10.8f %10.8f %+11.8f %3i  %3i \r",
      tau,Ds*K,Dn*K,dD*K,n,JA);

     // The interpolation for the remaining values
```

```
      for(l=0;l<=L;l++)Dnew[l]=Dnew[L/2];

      // The computation of norm of delta
      dD=fabs(Dnew[L/2]-DD[i][L/2]);

      // renew delta in the basic array
      for(l=0;l<=L;l++){ DD[i][l]=Dnew[l];}

      // quit from Newtons method
      if(fabs(dD*K)<0.000001 || n==40){ printf("\n"); Niter=Niter+n; break;}

   }//n

}//i along tau

printf("\n End   Average N_iter %6.2f ", Niter*1./M);
getch();

return(0);
}
```

Notation

$B(v)$	The Beta function.
$\Gamma(v)$	The Gamma function.
K	The exercise price.
$p(S,t)$	The price of European Put option.
$P(S,t)$	The price of American Put option.
$\hat{p}(S,t)$	The Mellin transform of $p(S,t)$.
$p(S_1,S_2,\tau)$	The price of two European Put options on a basket of two stocks S_1, S_2 .
$\tilde{P}(S_1,S_2,\tau)$	The price of two American Put options on a basket of two stocks S_1, S_2 .
$\hat{P}(v_1,v_2,\tau)$	The double Mellin transform of $p(S_1,S_2,t)$.
ρ	The coefficient of correlation.
r	The risk-free interest rate.
S, S_1, S_2	Stock prices.
$\sigma, \sigma_1, \sigma_2$	The volatilities of the market prices.
$S^*(\tau)$	A free boundary.
t	The current time.
T	The expiry date.
τ	The reversed time ($\tau = T - t$).

Bibliography

[1] F. Black and M. Scholes (1973)
The Pricing of Options and Corporate Liabilties, Journal of Political Economy, **81**, pp. 637-654.

[2] Yu. A. Brychkov, H.J. Glaske, A.P. Prudnikov and V.K. Tuan (1992)
"Multidimensional Integral Transforms", Gordon and Breach, Amsterdam.

[3] L. Clewlow and C. Strickland (1998)
"Implementing Derivatives Models", John Wiley and Sons.

[4] J.C. Cortés, L. Jodar, R. Sala, P. Sevilla-Peris (2005)
Exact and numerical solution of Black-Scholes matrix equation, Applied Mathematics and Computation, **160**, pp. 607-613.

[5] L. Jodar, P. Sevilla-Peris, J. C. Cortés, R. Sala (2005)
A new direct method for solving the Black-Scholes equation, Applied Mathematics Letters, **18** (1), pp. 29-32.

[6] J.C. Cox, S.A. Ross and M. Rubinstein (1979)
Option pricing: A simplified approach, Journal of Financial Economics **7**, pp. 229-263.

[7] D.J. Higham (2004)
"An Introduction to Financial Option Valuation", Cambridge University Press, Cambridge.

[8] D.J. Higham and N. J. Higham (2000)
"MATLAB Guide", Philadelphia, PA: SIAM

[9] J.-Z. Huang, M.G. Subrahmanyam and G.G.Yu (1996)
Pricing and Hedging American options: A recursive integration method, Review of Finansial Studies, **9** (1), pp. 277-300.

[10] J. Hull (1998)
"Introduction to futures and options markets", Prentice-Hall, Inc., New Jersey.

[11] Y.K. Kwok (1998)
"Mathematical Models of Financial Derivatives", Springer Verlag, Singapore.

[12] R.C. McOwen (1996)
"Partial Differential Equations: Methods and Applications", Prentice Hall, New Jersey.

[13] R. Panini and R.P. Srivastav (2003)
Option Pricing with Mellin Transforms, Applied Mathematics Letters, **18** (4), pp. 471-474.

[14] R. Panini (2004)
Option Pricing with Mellin Transforms, Dissertation, Stony Brook University.

[15] I.S. Reed (1944)
The Mellin type of double integral, Duke Mathematical Journal, **11** (3), pp. 565-572.

[16] M.S. Ross (2003)
"An elementary Introduction to mathematical Finance", Cambridge.

[17] A.A. Samarsky and A.V.Gulin (1989)
" Numerical methods", Mir publisher, Moscow (in Russian).

[18] I.N. Sneddon (1972)
"The use of Integral Transforms", McGraw-Hill, New York.

[19] P. Wilmott (1989)
"Derivatives", Chichester: Wiley.

[20] P. Wilmott, S. Howison and J.Dewynne (1995)
"The Mathematics of Financial Derivatives", Cambridge University Press, Cambridge.

[21] A. Würfel (2007)
"Analytical und numerical solution of the Black-Scholes equation for European and American Basket Options using Mellin transformations", Diploma Thesis, TU Berlin (in German).

Glossary

The Mellin transform	is an integral transform which is defined by the relation $\hat{p}(v,t) = \int_0^\infty p(S,t) \cdot S^{v-1} dS$, where v is a complex variable.
The interest rate	is the amount of money charged as a fee for lending money, or the price of borrowing money.
Volatility	is the extent to which an economic variable, such as a price or an exchange rate, moves up and down over time.
The expiry date	is the prescribed time in the future.
The exercise price	is the prescribed purchase price.
European put option	gives its holder the right (but not the obligation) to sell to the writer a prescribed asset for a prescribed price at a prescribed time in the future.
American put option	gives its holder the right (but not the obligation) to sell to the writer a prescribed asset for a prescribed price at any time between the start date and a prescribed expiry date in the future.
A basket option	is an option on the weighted average of several underlyings.

http://economics.about.com
http://economist.com
http://glossary.itlocus.com