Projeto Computational 1

Métodos Computacionais em Finanças

2024/2025

Afonso da Conceição Ribeiro (ist1102763)

Mestrado em Engenharia e Ciência de Dados

Instituto Superior Técnico – Universidade de Lisboa

Índice

1.

(a)

Let $V(S,t)$ be the price of an option with underlying price S at calendar time $t \in [0,T]$. For all vanilla contracts (European/American, call/put), V satisfies the Black-Scholes partial differential equation

$$\frac{\partial V}{\partial t} + \frac{\sigma^2}{2} S^2 \frac{\partial^2 V}{\partial S^2} + r\, S\, \frac{\partial V}{\partial S} - r\, V = 0, \qquad 0 < S < S^*, \quad 0 \le t \le T, \tag{1}$$

subject to terminal and boundary data that depend on the contract type. To turn the terminal-value problem (1) into an initial-value problem we introduce the forward time

$$t \;=\; T - t, \qquad U(S,t) \;:=\; V(S, T-t).$$

(also with t to simplify the notation).
Then U obeys

$$\begin{cases} \frac{\partial U}{\partial t} = \frac{\sigma^2}{2} S^2 \frac{\partial^2 U}{\partial S^2} + r\, S\, \frac{\partial U}{\partial S} - r\, U, & 0 < S < S^*,\ 0 < t < T \\ U(S,0) = u_0(S), & 0 \le S \le S^*, \\ U(0,t) = u_a(t), & 0 \le t \le T, \\ U(S^*,t) = u_b(t), & 0 \le t \le T. \end{cases} \tag{2}$$

For each contract, the functions u_0, u_a, u_b are assumed known.
We discretise the spatial axis $S \in [0, S^*]$ with a uniform grid

$$S_i = i\, h_S, \qquad h_S = \frac{S^*}{N_S}, \qquad i = 0, \dots, N_S,$$

and approximate the equation's derivatives in space at a certain point S_i for a fixed time, yielding, for the interior nodes $i = 1, \dots, N_S - 1$,

$$\frac{\partial U}{\partial t}(S_i,t) = U(S_{i-1},t)\,\underbrace{(\frac{\sigma^2}{2}\, i^2 - \frac{r\, i}{2})}_{:=\alpha_i} + U(S_i,t)\,\underbrace{(-\sigma^2\, i^2 - r)}_{:=\beta_i} + U(S_{i+1},t)\,\underbrace{(\frac{\sigma^2}{2}\, i^2 + \frac{r\, i}{2})}_{:=\gamma_i}.$$

Collecting the interior unknowns into the vector

$$W(t) \;:=\; \Big[U_1(t), \dots, U_{N_S-1}(t)\Big]^\top,$$

we obtain the ODE system

$$W'(t) = A\, W(t) + b(t), \tag{3}$$

where $A \in \mathbb{R}^{(N_S-1)\times(N_S-1)}$ is tridiagonal with band entries $\alpha_i, \beta_i, \gamma_i$ and $b(t)$ encodes the (possibly time-dependent) boundary values $u_a(t)$ and $u_b(t)$.

Finally, we use the fourth order Runge-Kutta method:

$$\begin{cases} f_1 = h\, F(t, W), \\ f_2 = h\, F(t + \frac{h}{2}, W + \frac{f_1}{2}), \\ f_3 = h\, F(t + \frac{h}{2}, W + \frac{f_2}{2}), \\ f_4 = h\, F(t + h, W + f_3), \\ W(t+h) = W(t) + \frac{1}{6}\Big(f_1 + 2f_2 + 2f_3 + f_4\Big), \end{cases}$$

with $F(t,W) = AW + b(t)$.

Input and output of `MOL_RK4`

Input: r risk-free rate [year^{-1}];

σ volatility [year$^{-1/2}$];

T maturity (calendar time horizon);

S^* truncation of the S-domain;

N_S, N_t spatial and temporal resolutions;

$u_0(S)$ payoff function $U(S,0)$;

$u_a(t)$ left boundary $U(0,t)$;

$u_b(t)$ right boundary $U(S^*,t)$.

Output: A Matlab *struct* out $= \{$ `.S`, `.t`, `.U` $\}$, where

`.S` the (N_S+1)-vector of spatial nodes S_i;

`.t` the (N_t+1)-vector of forward times t;

`.U` the $(N_S+1)\times(N_t+1)$ array with entries $U_{i,j} \approx U(S_i,t_j)$.

```
function out = MOL_RK4(r, sigma, T, S_star, NS, Nt, u_0, u_a, u_b)
    %% Spatial grid
    hS   = S_star / NS;
    h    = T / Nt;
    S    = 0 : hS : S_star;
    i    = (1:NS-1)';


    %% Finite-difference coefficients α_i, β_i, γ_i
    alpha_i = 0.5*sigma^2 .* i.^2  - 0.5*r .* i;
    beta_i  =     -sigma^2 .* i.^2  -     r;
    gamma_i = 0.5*sigma^2 .* i.^2  + 0.5*r .* i;

    A = spdiags([alpha_i beta_i gamma_i], [-1 0 1], NS-1, NS-1); % A_ML


    %% Initial condition W(0) (pay-off)
    W = u_0(S(2:NS)).';


    %% Pre-allocate storage for all slices
    U = zeros(NS+1, Nt+1);
    U(:,1) = u_0(S).'; % (t=0 forward)
    b = zeros(NS-1,1); % b_ML(t)


    %% RK-4 loop (forward time)
    for n = 0:Nt-1
        t = n * h;

        % Boundary values V(S=0,t) and V(S=S*,t)
        V_left  = u_a(t); % U(0,t)
        V_right = u_b(t); % U(S*,t)
        b(1) = alpha_i(1) * V_left;

        % Runge-Kutta stages
        f1 = h * (A*W + b);

```

```
        t_half    = t + 0.5*h;
        V_left_h = u_a(t_half);
        b(1)      = alpha_i(1) * V_left_h;
        f2 = h * (A*(W + 0.5*f1) + b);
        f3 = h * (A*(W + 0.5*f2) + b);

        t_next    = t + h;
        V_left_n = u_a(t_next);
        b(1)      = alpha_i(1) * V_left_n;
        f4 = h * (A*(W + f3) + b);

        % RK-4 update
        W = W + (f1 + 2*f2 + 2*f3 + f4)/6;

        % Assemble full vector V(S_i, t_next) and store
        U(:,n+2) = [V_left_n ; W ; u_b(t_next)];
    end


    %% Output
    out.S = S;
    out.t = 0:h:T;
    out.U = U;
end
```

(b)

Specification for a European put

For a European put with strike K, the data functions are

$$u_0(S) = \max(K - S, 0), \qquad u_a(t) = K\, e^{-rt}, \qquad u_b(t) = 0.$$

The routine sets the parameters r, σ, T, K, S^* and the grids (N_S, N_t), constructs function handles for u_0, u_a, u_b, calls `MOL_RK4`, and finally produces

- a 2D slice $V(S, 0) = U(S, T)$ at calendar time $t = 0$, and
- a 3D surface $V(S, t) = U(S, T - t)$ for $0 \le t_{\text{cal}} \le T$.

```
%% Parameters
r      = 0.06;
sigma  = 0.3;
T      = 1;
K      = 10;
S_star = 15;

NS = 400;
Nt = 13000;

%% Functions for European put option
u_0 = @(S) max(K-S,0);
u_a = @(t) K*exp(-r*t);
u_b = @(t) 0*t;

%% Run
sol = MOL_RK4(r,sigma,T,S_star,NS,Nt,u_0,u_a,u_b);

%% 2D price at calendar time t=0 (last column)
V_today = sol.U(:,end);
figure;
plot(sol.S, V_today), grid on
title('European put V(S,0)')

%% 3D surface V(S,t)
[Tgrid, Sgrid] = meshgrid(T - sol.t, sol.S); % calendar time axis
figure;
mesh(Tgrid, Sgrid, sol.U)
xlabel('t'),
ylabel('S'),
zlabel('V(S,t)')
title('European put option value, V(S,t)')
view(135,30);
```

Discussion of the numerical results

With the default grid $(N_S, N_t) = (400, 13000)$m the computation achieves $|V_{\text{num}} - V_{\text{BS}}| < 10^{-3}$ for all $S \in [0, 15]$, when compared with the closed-form Black-Scholes price.

Figure 1 shows the option price at today's date ($t = 0$). The curve starts at $V(0, 0)$, is decreasing and convex, and approaches 0 for $S \gg K$, exactly as theory predicts.

Figure 2 depicts the full surface $V(S, t)$. At $t = T$, the surface equals the kinked pay-off $\max(K - S, 0)$; as t decreases, the surface becomes smooth in S.

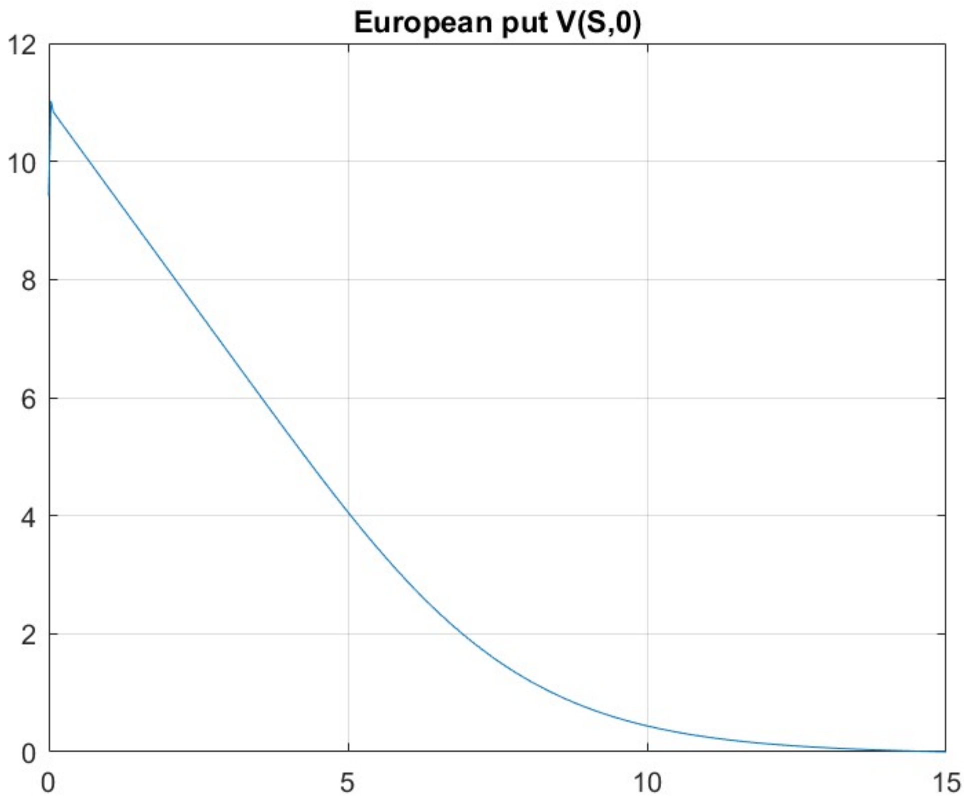

Figura 1: European put value $V(S, 0)$

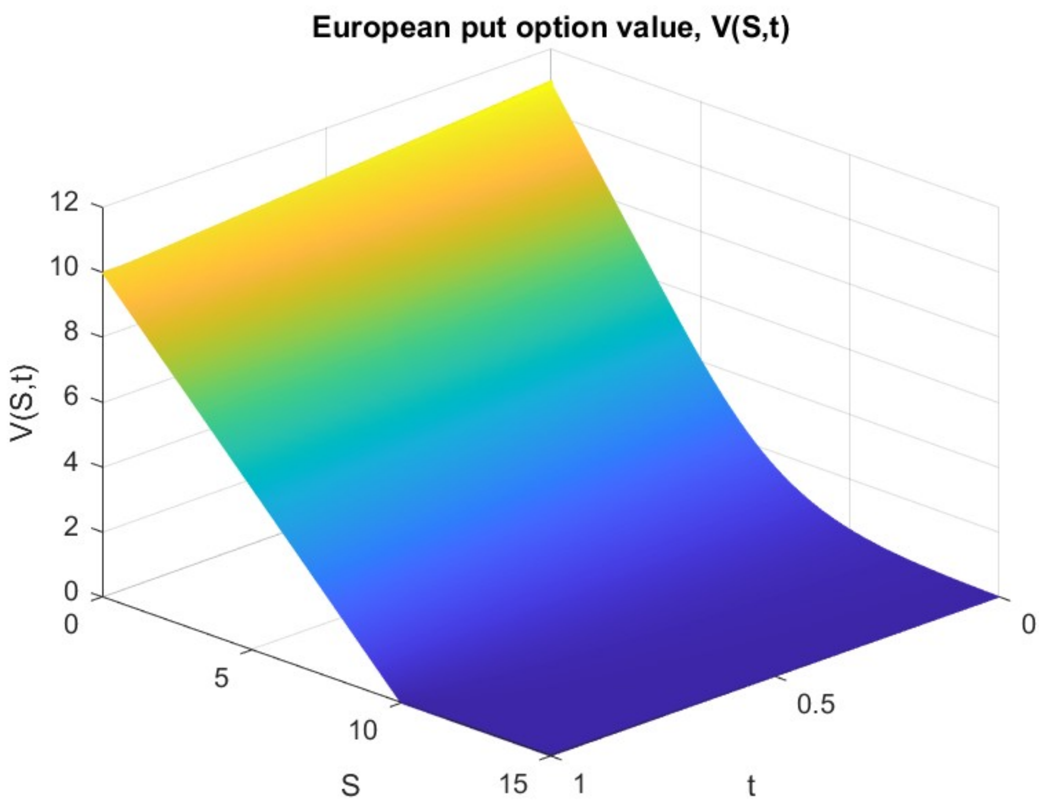

Figura 2: Surface $V(S, t)$ for $0 \leq t \leq T$ (calendar time axis)

2.

Let $V(S,t)$ be the price of an American put option with underlying price S at time $t \in [0,T]$. It satisfies the Black-Scholes inequality, which takes the form of a variational inequality:

$$\min\left\{V(S,t)-(K-S)^+,\ \frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}+rS\frac{\partial V}{\partial S}-rV\right\}=0, \tag{4}$$

subject to the boundary conditions

$$V(0,t)=K, \qquad V(S^*,t)=0, \qquad V(S,T)=\max(K-S,0),$$

for all $t \in [0,T]$ and some large $S^* > K$.
To solve (4) numerically, we discretise the time and asset axes:

$$S_i = i\,h_S, \quad h_S=\frac{S^*}{N_S}, \qquad t_n = n\,h_t, \quad h_t=\frac{T}{N_t},$$

for $i = 0,\dots,N_S$, $n = 0,\dots,N_t$. Let V_i^n denote the approximation to $V(S_i,t_n)$.
We apply the Crank-Nicolson scheme to the PDE part in (4), with central differences in space:

$$\begin{aligned}
\alpha_i &= \frac{1}{4}h_t\left(\sigma^2 i^2 - ri\right),\\
\beta_i &= -\frac{1}{2}h_t(\sigma^2 i^2 + r),\\
\gamma_i &= \frac{1}{4}h_t\left(\sigma^2 i^2 + ri\right).
\end{aligned}$$

These define a tridiagonal matrix A acting on the interior points $V_1^n,\dots,V_{N_S-1}^n$. The Crank-Nicolson system at each time step reads

$$BV^n = d^{n+1}, \qquad B = I-\tfrac{1}{2}h_t A, \quad d^{n+1}=\left(I+\tfrac{1}{2}h_t A\right)V^{n+1}+\mathrm{BC}.$$

However, due to the early exercise constraint $V_i^n \geq \max(K-S_i,0)$, this becomes a linear complementary problem. We solve it with the PSOR method:

$$V_i^{(k+1)} = \max\left\{\max(K-S_i,0),\ V_i^{(k)}+\omega\frac{r_i^{(k)}}{B_{ii}}\right\},$$

where $r_i^{(k)}$ is the residual at iteration k.

Input and output of `CN_PSOR`

Input: r risk-free rate;

σ volatility;

T maturity;

K strike price;

S^* truncation of the S-domain;

N_S, N_t grid resolutions.

Output: A Matlab *struct* out $= \{$ `.S`, `.t`, `.V` $\}$, where

`.S` the $(N_S + 1)$-vector of asset prices S_i;

`.t` the $(N_t + 1)$-vector of times t_n;

`.V` the $(N_S + 1) \times (N_t + 1)$ matrix of values V_i^n.

```
function [S_grid, t_grid, V] = CN_PSOR(r, sigma, T, K, S_star, NS, Nt)
    %% Grids
    dS     = S_star / NS;
    dt     = T / Nt;
    S_grid = linspace(0, S_star, NS+1)';
    t_grid = linspace(0, T, Nt+1);

    %% Grid indices
    I = 2:NS;
    S = S_grid(I);
    M = NS - 1;

    %% Coefficients for tridiagonal matrices
    i = (1:M)';  % i indexes interior points
    alpha = 0.25 * dt * (sigma^2 * i.^2 - r * i);
    beta  = -0.5 * dt * (sigma^2 * i.^2 + r);
    gamma = 0.25 * dt * (sigma^2 * i.^2 + r * i);

    main_diag = 1 - beta;
    lower_diag = -alpha(2:end);
    upper_diag = -gamma(1:end-1);

    %% Matrix B (I - dt/2*A)
    B = spdiags([[lower_diag; 0], main_diag, [0; upper_diag]], -1:1, M, M);

    %% Matrix A+ (I + dt/2*A)
    main_plus = 1 + beta;
    low_plus = alpha(2:end);
    up_plus  = gamma(1:end-1);
    A_plus = spdiags([[low_plus; 0], main_plus, [0; up_plus]], -1:1, M, M);

    %% Payoff and boundary conditions
    V = zeros(NS+1, Nt+1);
    V(:, end) = max(K - S_grid, 0);
    V(1, :) = K; V(end, :) = 0;
    payoff = max(K - S, 0);

    %% Relaxation
    omega = 1.3;
    tol = 1e-7;
    max_iter = 500;
```

```

    %% Time-stepping loop
    for n = Nt:-1:1
        rhs = A_plus * V(I, n+1);

        % Boundary conditions
        rhs(1)   = rhs(1)   + alpha(1)   * (V(1, n+1) + V(1, n));
        rhs(end) = rhs(end) + gamma(end) * (V(end, n+1) + V(end, n));

        % Initial guess = last solution
        x = V(I, n+1);

        % PSOR iterations
        for it = 1:max_iter
            x_old = x;
            for i = 1:M
                if i == 1
                    residual = rhs(i) - B(i,i)*x(i) - B(i,i+1)*x(i+1);
                elseif i == M
                    residual = rhs(i) - B(i,i-1)*x(i-1) - B(i,i)*x(i);
                else
                    residual = rhs(i) - B(i,i-1)*x(i-1) - B(i,i)*x(i) - B(i,
 i+1)*x(i+1);
                end
                x(i) = max(payoff(i), x(i) + omega * residual / B(i,i));
            end
            if norm(x - x_old, inf) < tol
                break;
            end
        end

        V(I, n) = x;
    end
end
```

Numerical results for the American put

We set the parameters

$$r = 0.06, \quad \sigma = 0.3, \quad T = 1, \quad K = 10, \quad S^* = 15,$$

and use the grid $(N_S, N_t) = (400, 1000)$.

Figure 3 shows the value of the option at three calendar times $t = 0, 0.5, 1$. At $t = T$ the price matches the payoff $\max(K - S, 0)$, while for $t < T$ the value lies strictly above it in the continuation region.

Figure 4 shows the continuation region: the values $V(S, t)$ that are strictly above the payoff. This surface is smooth in both S and t and bounded below by the free boundary $S_f(t)$.

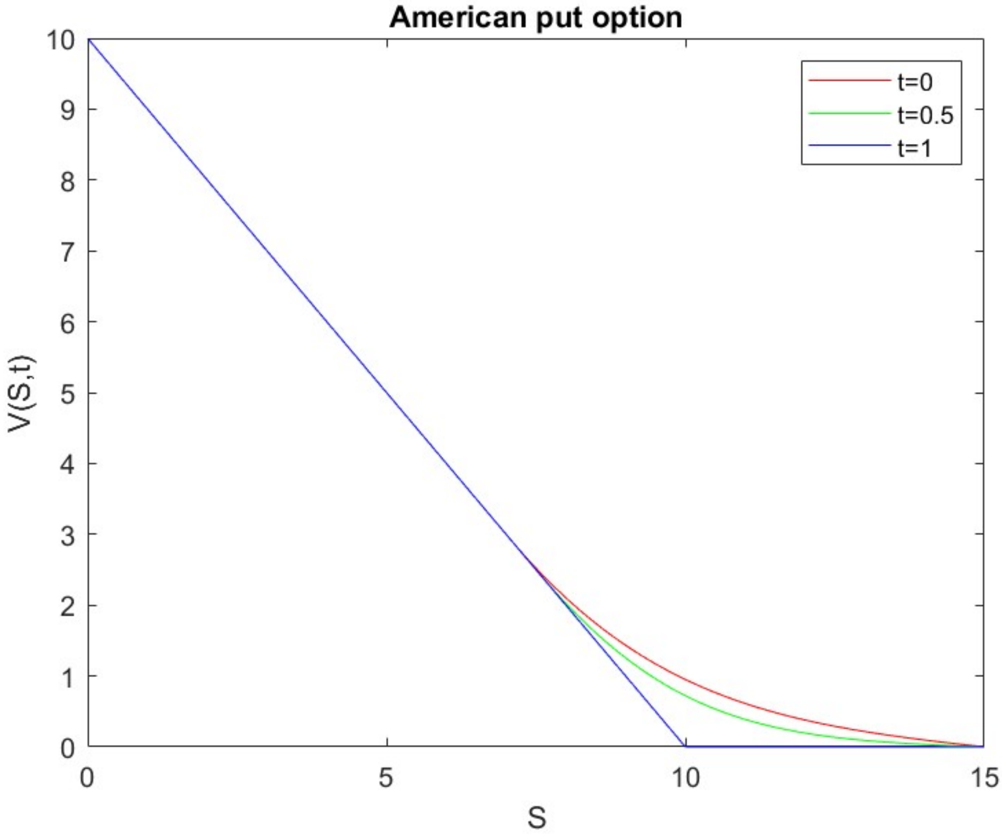

Figura 3: American put value $V(S,t)$ for $t = 0, 0.5, 1$

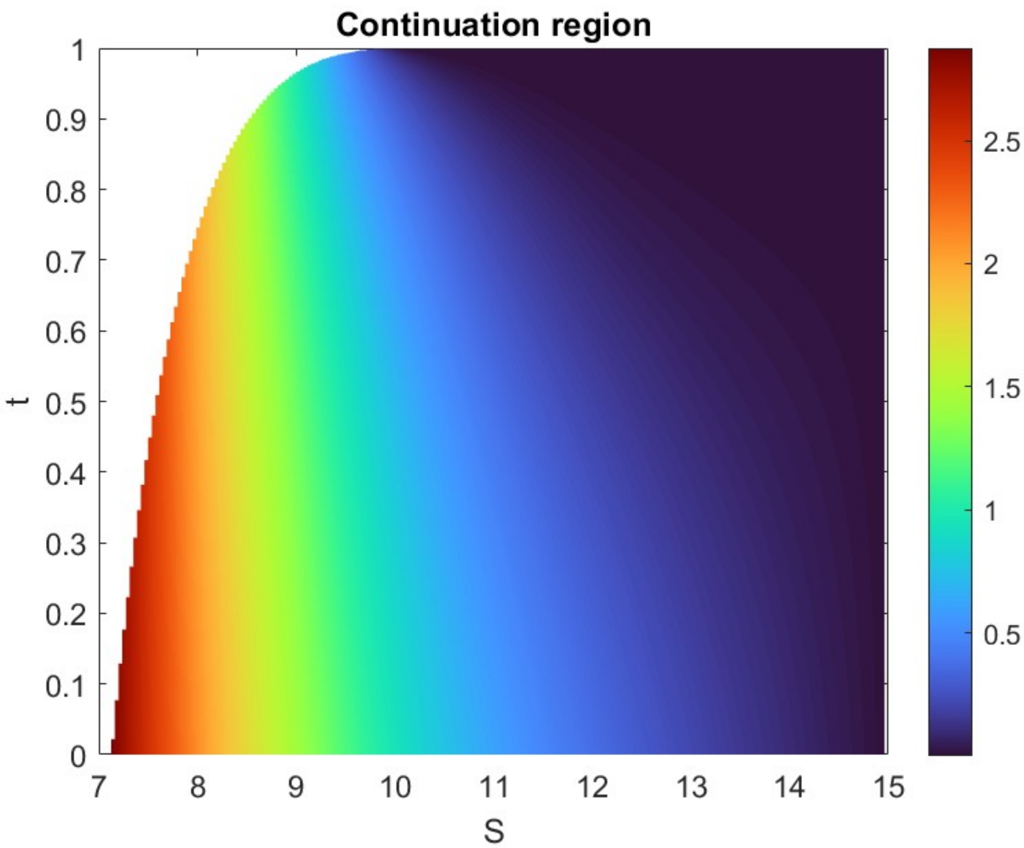

Figura 4: Continuation region $V(S,t) > (K-S)^+$