

Evaluation of Financial Options Using Radial Basis Functions in Mathematica

Michael Kelly

In the academic literature there are two common approaches for the evaluation of financial options. These are stochastic calculus and partial differential equations. The former is the method of choice for statisticians and theoreticians, while the latter is the principal tool of physicists and computer scientists because it lends itself to practical implementation schemes. Occasionally small modifications such as linear regression and binomial trees are used, but these are usually treated within either of the two previously mentioned fields. Rarely do the practitioners of these fields compare and contrast methodologies, let alone admit completely different approaches. While Radial Basis Function (RBF) methodology has previously been applied to solving some differential equations, there are very few papers considering its applicability to financial mathematics. The purpose of this article is to show not only that RBF can solve many of the evaluation problems for financial options, but that with *Mathematica* it can do so with accuracy and speed.

■ Introduction

A radial basis function (RBF) is a function $\phi(x, x_i)$ that depends only on the distance r between $x \in \mathcal{R}^d$ and a fixed point $x_i \in \mathcal{R}^d$

$$\phi(x, x_i) = \phi(\|x - x_i\|). \quad (1)$$

Each function $\phi(x, x_i)$ is radially symmetric about the center x_i . Since their discovery in the early 1970s by Hardy [1, 2], RBFs have become a primary tool for the interpolation of multidimensional scattered data. In the 1990s, Kansa [3, 4] showed that RBF methods are applicable to solving elliptic parabolic and some hyperbolic partial differential equations (PDEs). While this approach has some similarities with finite difference (FD) formulas, there are significant differences in that FD stencils typically extend over a subset of the data points at which derivative approximations are sought, and FD formulas are obtained by differentiating polynomial interpolants rather than RBF interpolants. But the fact that RBF methods allow the solution of parabolic PDEs means that they are applicable to the Black-Scholes PDE and hence to the evaluation of financial options.

The purpose of this article is to apply global RBFs within *Mathematica* as a spatial collocation scheme for solving European and American option pricing models by extending and implementing the work in this area that has recently been

done by Hon and Mao [5] and Fasshauer, Khaliq, and Voss [6]. While some results have been presented, none of the papers mentioned actually exhibit any programs or discuss the programming difficulties that are inherent in RBF models of parabolic PDEs. A number of Runge-Kutta time integration schemes are adopted for the time derivatives of the option model. We show that these schemes result in highly accurate approximations when compared with existing numerical techniques and are inherently more stable than the more commonly used finite element methods.

■ Radial Basis Function Methodology for the Black-Scholes Partial Differential Equation

It has been shown by Black and Scholes [7] that, assuming the underlying asset price is risk-neutral, all European options satisfy a lognormal parabolic PDE, now called the Black-Scholes PDE. If we let $V(S, t)$ be the value of the option with underlying asset S and elapsed time t , risk-free interest rate r , dividend yield q , and volatility or annualized standard deviation of the asset price σ , then the equation governing all European options is:

$$V^{(0,1)}(S, t) + (r - q) S V^{(1,0)}(S, t) + \frac{1}{2} S^2 \sigma^2 V^{(2,0)}(S, t) - r V(S, t) = 0. \quad (2)$$

Like all other PDEs, the specification of the boundary conditions determines the type of option studied. The initial condition for the backward PDE has the following maturity payoff function, with K being the strike and T the time of maturity:

$$V(S, T) = \begin{cases} \max(K - S, 0) & \text{for put,} \\ \max(S - K, 0) & \text{for call.} \end{cases} \quad (3)$$

It is shown in [7] for European options that with $\mathcal{N}(z)$ as the cumulative distribution function of the standard normal distribution, the explicit solution is given by

$$V(S, t) = \begin{cases} S e^{-q(T-t)} \mathcal{N}(d_1) - K e^{-r(T-t)} \mathcal{N}(d_2) & \text{for calls,} \\ K e^{-r(T-t)} \mathcal{N}(-d_2) - S e^{-q(T-t)} \mathcal{N}(-d_1) & \text{for puts.} \end{cases} \quad (4)$$

$$d_1 = \frac{\log(S/K) + (r - q + \sigma^2/2)(T - t)}{\sigma \sqrt{T - t}}$$

$$d_2 = d_1 - \sigma \sqrt{T - t}.$$

By taking the mathematical derivatives $V^{(0,1)}(S, t)$ of the European options in equation (4), we readily obtain the hedging greek deltas

$$\Delta(S, t) = V^{(1,0)}(S, t) = \begin{cases} e^{-q(T-t)} \mathcal{N}(d_1) & \text{for calls,} \\ -e^{-q(T-t)} \mathcal{N}(-d_1) & \text{for puts.} \end{cases} \quad (5)$$

Let us make a transformation into the log space so that $S = e^y$, where $V(e^y, t) = U(y, t)$. Then observing that $V^{(1,0)}(S, t) = U^{(1,0)}(y, t) \partial_S(y) =$

$\frac{1}{S} U^{(1,0)}(y, t)$ and furthermore that $V^{(2,0)}(S, t) = \frac{1}{S^2} (U^{(2,0)}(y, t) - U^{(1,0)}(y, t))$, equations (2) and (3) now become

$$U^{(0,1)}(y, t) + \left(r - q - \frac{\sigma^2}{2} \right) U^{(1,0)}(y, t) + \frac{1}{2} \sigma^2 U^{(2,0)}(y, t) - r U(y, t) = 0 \tag{6}$$

with initial condition

$$U(y, T) = \begin{cases} \max(K - e^y, 0) & \text{for put,} \\ \max(e^y - K, 0) & \text{for call.} \end{cases} \tag{7}$$

The RBF methodology is to represent the option function $U(y, t)$ as a linear combination of RBFs $\phi_i(y)$, $i = 1, \dots, N$ at the collocation points y_i

$$U(y, t) = \sum_{j=1}^N \alpha_j(t) \phi_j(y). \tag{8}$$

There are a number of different choices for the RBFs; the most common include Hardy's multiquadratic (MQ) $\phi(y, y_j) = \sqrt{\kappa^2 + (y - y_j)^2}$ and the Gaussian $\phi(y, y_j) = e^{-\kappa^2 (y - y_j)^2}$, where κ is the shape parameter. Extensive research in [3, 4] and Goldberg et al. [8] has demonstrated that the MQ interpolation is superior when solving inhomogeneous PDEs, as is the case with the Black-Scholes equation. The specification of κ is the basis of ongoing research, but here we adopt the recommendation in [1] that $\kappa = \min(\|x - x_i\|)$ for collocation points x_i . Substituting equation (8) into (6) we now have a system of N linear equations, $i = 1, \dots, N$:

$$U^{(0,1)}(y_i, t) + \left(r - q - \frac{\sigma^2}{2} \right) U^{(1,0)}(y_i, t) + \frac{1}{2} \sigma^2 U^{(2,0)}(y_i, t) - r U(y_i, t) = 0, \tag{9}$$

which can be further reduced by observing that

$$\begin{aligned} U^{(0,1)}(y_i, t) &= \sum_{j=1}^N \alpha'_j(t) \phi(y_i, y_j) \\ U^{(1,0)}(y_i, t) &= \sum_{j=1}^N \alpha_j(t) \phi^{(1,0)}(y_i, y_j) \\ U^{(2,0)}(y_i, t) &= \sum_{j=1}^N \alpha_j(t) \phi^{(2,0)}(y_i, y_j), \end{aligned} \tag{10}$$

where the partial derivatives of the MQ RBFs, initially described in equation (1) and in the preceding paragraph with the shape parameter κ , can be easily determined in *Mathematica*.

$$\text{In[103]:= } \phi[\mathbf{y1_}, \mathbf{y2_}] := \sqrt{(\mathbf{y1} - \mathbf{y2})^2 + \kappa^2};$$

$$\{\mathcal{D}[\phi[\mathbf{y}_i, \mathbf{y}_j], \mathbf{y}_i], \mathcal{D}[\phi[\mathbf{y}_i, \mathbf{y}_j], \{\mathbf{y}_i, \mathbf{2}\}]\}$$

$$\text{Out[104]= } \left\{ \frac{\mathbf{y}_i - \mathbf{y}_j}{\sqrt{\kappa^2 + (\mathbf{y}_i - \mathbf{y}_j)^2}}, \frac{1}{\sqrt{\kappa^2 + (\mathbf{y}_i - \mathbf{y}_j)^2}} - \frac{(\mathbf{y}_i - \mathbf{y}_j)^2}{(\kappa^2 + (\mathbf{y}_i - \mathbf{y}_j)^2)^{3/2}} \right\}$$

Now define the following elements of the $N \times N$ matrices Φ , Φ_y , and $\Phi_{y,y}$ with the N -vector $\underline{\alpha}(t)$

$$\Phi_{i,j} = \phi(y_i, y_j), (\Phi_y)_{i,j} = \phi^{(1,0)}(y_i, y_j), (\Phi_{y,y})_{i,j} = \phi^{(2,0)}(y_i, y_j), \alpha_j = (\underline{\alpha})_j. \quad (11)$$

Substituting equations (10), (11), and the previous differentiation output into equation (9), while observing that the elements of equation (10) actually define matrix products, results in the following matrix equation:

$$\Phi \underline{\alpha}' + \frac{1}{2} \sigma^2 \Phi_{y,y} \underline{\alpha} + \left(r - q - \frac{1}{2} \sigma^2 \right) \Phi_y \underline{\alpha} - r \Phi \underline{\alpha} = 0. \quad (12)$$

Solving equation (12) for $\underline{\alpha}$ allows the estimation of $U(y, t) = \Phi \cdot \underline{\alpha}(t)$ and hence of the original option price $V(S, t)$. It has been shown by Powell [9] that the matrix Φ is invertible. This allows us to rewrite equation (12) as a matrix differential equation for $\underline{\alpha}$

$$\underline{\alpha}' = \Phi^{-1} \cdot \left(\frac{1}{2} \sigma^2 \Phi_{y,y} \underline{\alpha} + \left(r - q - \frac{1}{2} \sigma^2 \right) \Phi_y \underline{\alpha} - r \Phi \underline{\alpha} \right) = B \cdot \underline{\alpha}. \quad (13)$$

B is the following $N \times N$ matrix and since all of its components, such as Φ and the identity matrix I , are known it is readily computed:

$$B = rI - \left(r - q - \frac{1}{2} \sigma^2 \right) \Phi^{-1} \cdot \Phi_y - \frac{1}{2} \sigma^2 \Phi^{-1} \cdot \Phi_{y,y}. \quad (14)$$

There are two approaches to solving equation (13): analytical and a backward time integration scheme. Using the work in [9], it follows that, if Φ is invertible and the eigenvectors of B are independent, then equation (12) has a solution for $\underline{\alpha}$ in terms of its eigenvectors \underline{w}_i and eigenvalues λ_i . Using *Mathematica's* `DSolve` operator we can explicitly analyze possible solutions to equation (13).

```
In[105]:= Block[{vecα, matB, n = 2},
  vecα = Array[α_# [t] &, n]; matB = Array[B_#1, #2 &, {n, n}];
  DSolve[Thread[D[vecα, t] == matB.vecα], vecα, {t}]
```

A very large output was generated. Here is a sample of it:

$$\left\{ \left\{ \alpha_1 [t] \rightarrow -\frac{\left(\frac{1}{e^{2t}} \langle \langle 1 \rangle \rangle - \frac{1}{e^{2t}} \langle \langle 1 \rangle \rangle \right) C[2] \{ \langle \langle 1 \rangle \rangle \}_{1,2}}{\sqrt{\{ \langle \langle 1 \rangle \rangle \}_{1,1}^2 + 4 \{ \langle \langle 1 \rangle \rangle \}_{1,2} \{ \langle \langle 1 \rangle \rangle \}_{2,1} - \langle \langle 1 \rangle \rangle + \{ \langle \langle 1 \rangle \rangle \}_{2,2}^2}} + \frac{C[1] \langle \langle 1 \rangle \rangle}{2 \sqrt{\langle \langle 1 \rangle \rangle}}, \right. \right.$$

$$\left. \alpha_2 [t] \rightarrow \langle \langle 1 \rangle \rangle \right\}$$

Show Less Show More Show Full Output Set Size Limit...

Choosing larger values for n , the dimension of the matrix B suggests the following general solution:

$$\underline{\alpha}(t) = \sum_{i=1}^N k_i e^{\lambda_i t} \underline{w}_i. \tag{15}$$

Then taking the time derivative of both sides and recalling that $\lambda_i \underline{w}_i = B \cdot \underline{w}_i$ because of the definition of $\{ \lambda_i, \underline{w}_i \}$ as the eigensystem of B , we get:

$$\underline{\alpha}'(t) = \sum_{i=1}^N k_i e^{\lambda_i t} \lambda_i \underline{w}_i = \sum_{i=1}^N k_i e^{\lambda_i t} B \cdot \underline{w}_i = B \cdot \sum_{i=1}^N k_i e^{\lambda_i t} \underline{w}_i = B \cdot \underline{\alpha}(t). \tag{16}$$

This result satisfies equation (13) and shows that (15) yields an analytical solution for $\underline{\alpha}(t)$. Further, define the vectors \underline{k} , $\underline{\lambda}$, and $\underline{U}(y, t)$ that consist of the components k_i , λ_i , and $U(y_i, t)$ as well as the matrix W that has as columns the eigenvectors \underline{w}_i , $i = 1, \dots, N$. Now observe that collocating equation (8) about the central points y_j converts it into the explicit matrix equation

$$\underline{U}(y, t) = \Phi \cdot \underline{\alpha}(t) = \Phi \cdot \sum_{i=1}^N \underline{w}_i k_i e^{\lambda_i t} = \Phi \cdot W \cdot \underline{k} \cdot e^{\underline{\lambda} t}. \tag{17}$$

It can be seen from equation (17) that $\underline{U}(y, t)$, and hence $V(S, t)$, can be calculated once \underline{k} is determined. Since the result holds for all values of time t , then it also holds for the initial condition expressed by equation (7) at maturity time T . Putting $t = T$ in the above result yields

$$\underline{k} = W^{-1} \cdot \Phi^{-1} \cdot \frac{\underline{U}(y, T)}{e^{\underline{\lambda} T}}. \tag{18}$$

While this result is sufficient for an explicit solution that will cover European options, it is not capable of calculating path-dependent options that have to be updated in a time-dependent manner. In fact, if we choose to again use the DSolve operator for larger values of the dimension of the matrix B , then we will get very complicated polynomial expressions that have to be solved by the RootSum function. Here is an example.

```
In[106]:= Block[{vecα, matB, n = 3},
  vecα = Array[α_# [t] &, n]; matB = Array[B_#1, #2 &, {n, n}];
  DSolve[Thread[D[vecα, t] == matB.vecα], vecα, {t}]
```

A very large output was generated. Here is a sample of it:

Out[106]=

$$\left\{ \left\{ \alpha_1[t] \rightarrow \right. \right.$$

$$C[3] \text{RootSum}\left[\langle\langle 22 \rangle\rangle + \#1 \langle\langle 1 \rangle\rangle_2 \langle\langle 1 \rangle\rangle \langle\langle 1 \rangle\rangle \left\{ \langle\langle 1 \rangle\rangle \right\}_{3,3} - \left\{ \langle\langle 1 \rangle\rangle \right\}_{1,1} \right.$$

$$\left. \left\{ \{-31.3273, \langle\langle 119 \rangle\rangle, -\langle\langle 20 \rangle\rangle\}, \langle\langle 119 \rangle\rangle, \left\{ \langle\langle 1 \rangle\rangle \right\}_{2,2} \right.$$

$$\left. \left\{ \{-31.3273, -19.4459, \langle\langle 118 \rangle\rangle, -0.1308\}, \right.$$

$$\left. \langle\langle 119 \rangle\rangle, \left\{ \langle\langle 1 \rangle\rangle \right\}_{3,3} \&, \frac{\langle\langle 1 \rangle\rangle - \langle\langle 1 \rangle\rangle + \langle\langle 1 \rangle\rangle}{\langle\langle 1 \rangle\rangle} \& \right] +$$

$$\left. \left. \langle\langle 1 \rangle\rangle + C[1] \text{RootSum}\left[\langle\langle 1 \rangle\rangle \right], \langle\langle 1 \rangle\rangle, \langle\langle 1 \rangle\rangle \right\} \right\}$$

Show Less Show More Show Full Output Set Size Limit...

For the purpose of calculating American-style options, we will consider times $t_n = T - n \Delta t$ and define $\underline{U}(y, t_n) = \underline{U}^n$, $\underline{q}(t_n) = \underline{q}^n$. Now we use backward-difference time integration schemes that can be any of the explicit first-order (BD1):

$$\underline{q}^n = \underline{q}^{n-1} - \Delta t B \cdot \underline{q}^{n-1} = (I_N - \Delta t B) \underline{q}^{n-1}, \quad (19)$$

the explicit second-order backward-difference time integration scheme (BD2):

$$\begin{aligned} R_1 &= -\Delta t B \cdot \underline{q}^{n-1} \\ R_2 &= -\Delta t B \cdot \left(\underline{q}^{n-1} + \frac{R_1}{2} \right) \\ \underline{q}^n &= \underline{q}^{n-1} + \frac{R_1 + R_2}{2}, \end{aligned} \quad (20)$$

the explicit fourth-order backward-difference time integration scheme (BD4):

$$\begin{aligned}
 R_1 &= -\Delta t B \cdot \underline{\alpha}^{n-1} \\
 R_2 &= -\Delta t B \cdot \left(\underline{\alpha}^{n-1} + \frac{R_1}{2} \right) \\
 R_3 &= -\Delta t B \cdot \left(\underline{\alpha}^{n-1} + \frac{R_2}{2} \right) \\
 R_4 &= -\Delta t B \cdot (\underline{\alpha}^{n-1} + R_3) \\
 \underline{\alpha}^n &= \underline{\alpha}^{n-1} + \frac{R_1 + 2 R_2 + 2 R_3 + R_4}{6},
 \end{aligned}
 \tag{21}$$

or the implicit backward-difference time integration scheme (ID θ):

$$\underline{\alpha}^n = \underline{\alpha}^{n-1} - \Delta t B \cdot (\theta \underline{\alpha}^{n-1} + (1 - \theta) \underline{\alpha}^n),
 \tag{22}$$

■ European Options Numerical Specification

□ Analytical Results

In order to evaluate European options, we need to specify their boundary conditions. Remember that all options will satisfy the Black-Scholes PDE of equation (2) with the initial condition of equation (3), but that for further differentiation between options we also need to specify the boundary conditions as $S \rightarrow 0$ and $S \rightarrow \infty$:

$$\begin{aligned}
 V(0, t) &= \begin{cases} K e^{-r(T-t)} & \text{for put,} \\ 0 & \text{for call,} \end{cases} \\
 V(S, t) &\rightarrow \begin{cases} 0 \text{ as } S \rightarrow \infty & \text{for put,} \\ S \text{ as } S \rightarrow \infty & \text{for call.} \end{cases}
 \end{aligned}
 \tag{23}$$

By using these equations we can now implement the numerical solution of European options. Even though we have given equations for both call- and put-style options, for the sake of brevity and to avoid what would essentially be very similar programs, we restrict ourselves to put options only. The time domain is subdivided into M subintervals, $\Delta t = \frac{T}{M}$, so that for any prior time $t_n = T - n \Delta t$, n lies in the range $0 \leq n \leq M$, with $n = 0$ corresponding to the maturity date T . The spatial discretization will be for N collocation points y_i , $1 \leq i \leq N$. The *Mathematica* program for the Black-Scholes European options and their deltas according to equations (4) and (5) can be readily implemented.

```

In[107]:= ClearAll[NormPDF, NormCDF, d1, d2, BSCall, BSPut,
putDelta, callDelta, Δy, y, t, Δt, φ, φ̂, φy, φyy, φinv, B,
λ, w, W, UT, kvec, α, U, Vanalytical, DeltaAnalytical];
d1[σ_, r_, q_, x_, y_, t_] := (Log[x/y] + (r - q + σ²/2) t) / (σ √t);
d2[σ_, r_, q_, x_, y_, t_] := (Log[x/y] + (r - q - σ²/2) t) / (σ √t);
    
```

```

NormCDF[z_] := (1 + Erf[z/√2])/2;
NormPDF[z_] := Exp[-z^2/2]/√(2π);
BSCall[σ_, r_, q_, S_, K_, T_] :=
  S e^{-qT} NormCDF[d1[σ, r, q, S, K, T]] -
  K e^{-rT} NormCDF[d2[σ, r, q, S, K, T]];
BSPut[σ_, r_, q_, S_, K_, T_] := K e^{-rT} NormCDF[-d2[σ, r, q, S, K, T]] -
  S e^{-qT} NormCDF[-d1[σ, r, q, S, K, T]];
putDelta[σ_, r_, q_, S_, K_, T_] :=
  -e^{-qT} NormCDF[-d1[σ, r, q, S, K, T]];
callDelta[σ_, r_, q_, S_, K_, T_] :=
  e^{-qT} NormCDF[d1[σ, r, q, S, K, T]];
putGamma[σ_, r_, q_, S_, K_, T_] = D[putDelta[σ, r, q, S, K, T], S];
callGamma[σ_, r_, q_, S_, K_, T_] = D[callDelta[σ, r, q, S, K, T], S];

```

For the sake of reproducing actual prices, we specify the following market parameters. The risk-free interest rate is 1%, the dividend yield is 0, the volatility is 30%, the strike is \$100, S_{\min} of 1 gives a y_{\min} of 0, and $\log(S_{\max}) = y_{\max} = 6$. Let there be $N = 121$ collocation points (N_{pts}) and subdivide the time to maturity $T = 1$ year into $M = 100$ subintervals. We chose rather large values for M and N that can be reused later for the American options, but are more than sufficient for evaluating European options. We later consider the implicit time recursion model with $\theta = 0.5$, which corresponds to the Crank-Nicolson FD scheme. An upper limit on recursion is set at 1000. Further, observe that we calculate not just one value for the analytical and approximation results but all of the possible values for the option over its spatial and time domains.

```

In[118]:= (* Option parameters *)
r = 0.1; σ = 0.3; q = 0; K = 100; T = 1; M = 100; θ = 0.5; Smax = e^6 // N;
Npts = 121; Smin = 1; $RecursionLimit = 1000; (* 0 ≤ t ≤ T,
t = (T - n Δt) is time elapsed, 0 ≤ n ≤ M, 1 ≤ i, j ≤ Npts,
τ is time remaining, S = e^y, y = Log[S], ymin = 0, Smax = Exp[ymax] *)

```

Our purpose is to compare the exact results for European options from the given Black-Scholes formulas with the analytical results of equation (17), explicit formulas of equations (19) to (21), and the implicit time integration scheme of equation (22). Note that $\phi(y_i, y_j)$ is the RBF as described earlier in equations (1), (10), and (11). With S_{\min} and S_{\max} the smallest and largest values of the stock price, then the spatial increment in the y domain with N (or N_{pts}) collocation points is $\Delta y = \frac{\log(S_{\max}) - \log(S_{\min})}{N-1}$ and choosing $\kappa = 4\Delta y$ in $\phi(y_i, y_j)$, we can now define Φ , Φ_y , and $\Phi_{y,y}$ according to (11) as:

```

In[119]:= Δy = (Log[Smax] - Log[Smin]) / (Npts - 1); Δt = T / M;
φ[y1_, y2_] := √((y1 - y2)^2 + 16 Δy^2);
φ̂ = Table[φ[y1, y2], {y1, Log[Smin], Log[Smax], Δy},
  {y2, Log[Smin], Log[Smax], Δy}];

```

```

ϕy = Table[Evaluate[D[ϕ[y1, y2], y1]],
  {y1, Log[Smin], Log[Smax], Δy}, {y2, Log[Smin], Log[Smax], Δy}];
ϕyy = Table[Evaluate[D[ϕ[y1, y2], y1, y1]],
  {y1, Log[Smin], Log[Smax], Δy}, {y2, Log[Smin], Log[Smax], Δy}];

```

The definition of the matrix B is then determined from (14) and used to calculate the eigensystem of B as $\{\lambda_i, w_i\}$. The initial condition of (7) is used to determine $U(y, T)$ (or UT), which is substituted into (18) to calculate \underline{k} (or kvec), and this is further substituted into (15) to yield $\underline{\alpha}(t)$.

```

In[123]:= ϕinv = Inverse[ϕ];
B = r IdentityMatrix[Npts] - σ² ϕinv.ϕyy/2 - (r - q - σ²/2) ϕinv.ϕy;
{λ, w} = Eigensystem[B];
W = Transpose[w];
UT = Table[Max[K - Exp[y], 0], {y, Log[Smin], Log[Smax], Δy}];
kvec = Inverse[W].ϕinv.UT/Exp[λ T];
α[t_] := Re[W.(kvec * Exp[λ t])];

```

Initial and boundary conditions of (7) and (23) are used to obtain $U(0, t)$ and $U(y, T)$. The general formulas for $U(y, t)$, $0 \leq t \leq T$, $0 \leq y \leq \log(S_{\max})$ can now be specified by (17). Finally, recalling that $y = \log(S)$, t is the elapsed time, and τ is the time remaining, we can get the analytical formula for $V(S, t)$ and its hedging delta in terms of $U(\log(S), t)$ by simply taking the derivatives of the RBFs $\phi(y, y_j)$.

```

In[130]:= U[0, t_] := K Exp[-r (T - t)];
U[y_, T] := Max[K - Exp[y], 0];
U[y_, t_] /; y ≥ Log[Smax] := 0;
U[y_, t_] /; y < Log[Smin] :=
  Table[ϕ[y, yj], {yj, Log[Smin], Log[Smax], Δy}].α[t];
Vanalytical[S_, τ_] := U[Log[S], T - τ];
Vanalytical[S_] := Table[ϕ[Log[S], yj],
  {yj, Log[Smin], Log[Smax], Δy}].(Re[W.kvec]);
DeltaAnalytical[S_] := Table[Evaluate[D[ϕ[y, yj], y]] /. {y → Log[S]},
  {yj, Log[Smin], Log[Smax], Δy}].(Re[W.kvec])/S;
GammaAnalytical[S_] :=
  Table[Evaluate[D[ϕ[y, yj], {y, 2}]] /. {y → Log[S]},
  {yj, Log[Smin], Log[Smax], Δy}].(Re[W.kvec])/S²;

```

□ Explicit and Implicit Results

The explicit and implicit backward-difference time integration schemes given by equations (19) to (22) can now be implemented using straightforward Do loops. We start at time $t = T$ with $U(y, T)$ set to UT as determined by (7) and step backward through time, considering earlier times $t_n = T - n \Delta t$, while defining $\underline{U}(y, t_n) = \underline{U}^n$, $\underline{\alpha}(t_n) = \underline{\alpha}^n$. There is one further modification that needs to be un-

dertaken in each step of the Do loop; namely, that the boundary condition (23) must be satisfied. Note that $T - t = T - t_n = n \Delta t$ in the exponent. This modification occurs at $S = S_{\min}$ and hence when $y = y_1$. This necessitates the rewriting of $U(y_1, t) = U^1(t)$ (or $U[[1]]$).

```
In[138]:= ClearAll[U0, α0, n, α1, α2, α4, αi, U1, U2, U4, Ui, Un, Vlexplicit,
  V2explicit, V4explicit, Vimplicit, Delta1Explicit,
  Delta2Explicit, Delta4Explicit, DeltaImplicit];
U0 = UT; α0 = ϕinv.U0;
Module[{Un}, α1[0] = α0;
  Do[α1[n] = (IdentityMatrix[Npts] - Δt B).α1[n - 1]; Un = ϕ.α1[n];
  Un[[1]] = K Exp[-r n Δt]; α1[n] = ϕinv.Un, {n, 1, M}]];
Module[{Un, R1, R2}, α2[0] = α0;
  Do[R1 = -Δt B.α2[n - 1]; R2 = (IdentityMatrix[Npts] - Δt B/2).R1;
  α2[n] = α2[n - 1] + (R1 + R2)/2; Un = ϕ.α2[n];
  Un[[1]] = K Exp[-r n Δt]; α2[n] = ϕinv.Un, {n, 1, M}]];
Module[{Un, R1, R2, R3, R4}, α4[0] = α0;
  Do[R1 = -Δt B.α4[n - 1]; R2 = (IdentityMatrix[Npts] - Δt B/2).R1;
  R3 = (R1 - Δt B.R2/2); R4 = (R1 - Δt B.R3);
  α4[n] = α4[n - 1] + (R1 + 2 R2 + 2 R3 + R4)/6; Un = ϕ.α4[n];
  Un[[1]] = K Exp[-r n Δt]; α4[n] = ϕinv.Un, {n, 1, M}]];
Module[{Un, B2 = (IdentityMatrix[Npts] - θ Δt B), B1}, B1 = B2 + Δt B;
  αi[0] = α0; Do[αi[n] = Inverse[B1].B2.αi[n - 1]; Un = ϕ.αi[n];
  Un[[1]] = K Exp[-r n Δt]; αi[n] = ϕinv.Un, {n, 1, M}]]];
```

Now the initial and boundary conditions according to equations (7) and (23) are specified.

```
In[144]:= U1[y_, 0] := Max[K - Exp[y], 0]; U2[y_, 0] := Max[K - Exp[y], 0];
U4[y_, 0] := Max[K - Exp[y], 0]; Ui[y_, 0] := Max[K - Exp[y], 0];
U1[y_, n_Integer] /; y ≥ Log[Smax] := 0;
U2[y_, n_Integer] /; y ≥ Log[Smax] := 0;
U4[y_, n_Integer] /; y ≥ Log[Smax] := 0;
Ui[y_, n_Integer] /; y ≥ Log[Smax] := 0;
```

Again the general formulas for $U(y, t)$, $0 \leq t \leq T$; $0 \leq y \leq \log(S_{\max})$ can now be specified according to the left-hand side of equation (17). We can also get the explicit and implicit approximation formulas for $V(S, t)$ by using the updated values for $\alpha(t_n) = \alpha^n$. The hedging delta formulas are given in terms of $U(\log(S), t)$ by simply taking the derivatives of the RBFs $\phi(y, y_j)$.

```
In[148]:= U1[y_, n_Integer] /; y < Log[Smax] :=
  Table[ϕ[y, yj], {yj, Log[Smin], Log[Smax], Δy}].α1[n];
U2[y_, n_Integer] /; y < Log[Smax] :=
  Table[ϕ[y, yj], {yj, Log[Smin], Log[Smax], Δy}].α2[n];
U4[y_, n_Integer] /; y < Log[Smax] :=
  Table[ϕ[y, yj], {yj, Log[Smin], Log[Smax], Δy}].α4[n];
Ui[y_, n_Integer] /; y < Log[Smax] :=
```

```

Table[ $\phi$ [y, yj], {yj, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_1$ [n];
V1explicit[S_, n_Integer] := U1[Log[S], n];
V2explicit[S_, n_Integer] := U2[Log[S], n];
V4explicit[S_, n_Integer] := U4[Log[S], n];
Vimplicit[S_, n_Integer] := Ui[Log[S], n];
V1explicit[S_] :=
Table[ $\phi$ [Log[S], yj], {yj, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_1$ [M];
V2explicit[S_] := Table[ $\phi$ [Log[S], yj],
{yj, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_2$ [M];
V4explicit[S_] := Table[ $\phi$ [Log[S], yj],
{yj, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_4$ [M];
Vimplicit[S_] := Table[ $\phi$ [Log[S], yj],
{yj, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_1$ [M];
Delta1Explicit[S_] := Table[Evaluate[D[ $\phi$ [y, yj], y]] /. {y  $\rightarrow$  Log[S]},
{yj, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_1$ [M]/S;
Delta2Explicit[S_] := Table[Evaluate[D[ $\phi$ [y, yj], y]] /. {y  $\rightarrow$  Log[S]},
{yj, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_2$ [M]/S;
Delta4Explicit[S_] := Table[Evaluate[D[ $\phi$ [y, yj], y]] /. {y  $\rightarrow$  Log[S]},
{yj, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_4$ [M]/S;
DeltaImplicit[S_] := Table[Evaluate[D[ $\phi$ [y, yj], y]] /. {y  $\rightarrow$  Log[S]},
{yj, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_1$ [M]/S;
Gamma1Explicit[S_] := Table[Evaluate[D[ $\phi$ [y, yj], {y, 2}]] /.
{y  $\rightarrow$  Log[S]}, {yj, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_1$ [M]/S2;
Gamma2Explicit[S_] := Table[Evaluate[D[ $\phi$ [y, yj], {y, 2}]] /.
{y  $\rightarrow$  Log[S]}, {yj, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_2$ [M]/S2;
Gamma4Explicit[S_] := Table[Evaluate[D[ $\phi$ [y, yj], {y, 2}]] /.
{y  $\rightarrow$  Log[S]}, {yj, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_4$ [M]/S2;
GammaImplicit[S_] := Table[Evaluate[D[ $\phi$ [y, yj], {y, 2}]] /.
{y  $\rightarrow$  Log[S]}, {yj, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_1$ [M]/S2;
In[168]:= TableForm[
Table[(PaddedForm[N[#1], {7, 4}] &)/@{S, BSPut[ $\sigma$ , r, q, S, K, T],
Vanalytical[S], V1explicit[S],
V2explicit[S], V4explicit[S], Vimplicit[S]},
{S, 80, 200, 10}], TableHeadings  $\rightarrow$ 
{None, {"S", "Exact", "Analytic", "BD1", "BD2", "BD4", "ID $\theta$ "}}},
TableAlignments  $\rightarrow$  Center]

```

Out[168]//TableForm=

S	Exact	Analytic	BD1	BD2	BD4	ID θ
80.0000	16.2425	16.2271	15.9010	15.9039	15.9067	15.9069
90.0000	11.0036	10.9866	10.6530	10.6510	10.6489	10.6492
100.0000	7.2179	7.2008	6.8569	6.8523	6.8476	6.8479
110.0000	4.6135	4.5975	4.2397	4.2350	4.2303	4.2306
120.0000	2.8899	2.8753	2.5017	2.4985	2.4953	2.4956
130.0000	1.7825	1.7696	1.3801	1.3790	1.3778	1.3781
140.0000	1.0870	1.0756	0.6715	0.6722	0.6729	0.6732
150.0000	0.6575	0.6474	0.2304	0.2325	0.2345	0.2348
160.0000	0.3955	0.3867	-0.0416	-0.0386	-0.0357	-0.0353
170.0000	0.2371	0.2297	-0.2084	-0.2049	-0.2015	-0.2012
180.0000	0.1419	0.1366	-0.3105	-0.3067	-0.3031	-0.3027
190.0000	0.0849	0.0826	-0.3725	-0.3686	-0.3649	-0.3645
200.0000	0.0509	0.0532	-0.4094	-0.4055	-0.4018	-0.4014

Table 1. European put option prices determined by analytic, explicit, and implicit RBF methods.

```
In[169]:= TableForm[Table[
  (PaddedForm[N[#1], {7, 4}] &) /@ {S, putDelta[ $\sigma$ , r, q, S, K, T],
    DeltaAnalytical[S], Delta1Explicit[S],
    Delta2Explicit[S], Delta4Explicit[S], DeltaImplicit[S]},
  {S, 80, 200, 10}], TableHeadings  $\rightarrow$ 
  {None, {"S", "Exact  $\Delta$ ", "Analytic  $\Delta$ ", "BD1  $\Delta$ ",
    "BD2  $\Delta$ ", "BD4  $\Delta$ ", "ID $\theta$   $\Delta$ "}}], TableAlignments  $\rightarrow$  Center]
```

Out[169]//TableForm=

S	Exact Δ	Analytic Δ	BD1 Δ	BD2 Δ	BD4 Δ	ID θ Δ
80.0000	-0.6028	-0.6030	-0.6037	-0.6043	-0.6048	-0.6048
90.0000	-0.4474	-0.4475	-0.4484	-0.4488	-0.4492	-0.4491
100.0000	-0.3144	-0.3144	-0.3156	-0.3157	-0.3158	-0.3158
110.0000	-0.2116	-0.2114	-0.2129	-0.2128	-0.2128	-0.2128
120.0000	-0.1376	-0.1375	-0.1391	-0.1389	-0.1387	-0.1387
130.0000	-0.0873	-0.0871	-0.0886	-0.0884	-0.0882	-0.0882
140.0000	-0.0543	-0.0541	-0.0555	-0.0553	-0.0552	-0.0552
150.0000	-0.0333	-0.0331	-0.0343	-0.0342	-0.0341	-0.0341
160.0000	-0.0202	-0.0200	-0.0211	-0.0210	-0.0210	-0.0210
170.0000	-0.0122	-0.0120	-0.0129	-0.0129	-0.0129	-0.0129
180.0000	-0.0073	-0.0070	-0.0079	-0.0079	-0.0079	-0.0079
190.0000	-0.0044	-0.0040	-0.0048	-0.0048	-0.0048	-0.0047
200.0000	-0.0026	-0.0020	-0.0028	-0.0028	-0.0028	-0.0028

Table 2. European put option deltas determined by analytic, explicit, and implicit RBF methods.

```

In[170]:= TableForm[Table[
  (PaddedForm[N[#1], {7, 4}] &)/@{S, putGamma[σ, r, q, S, K, T],
  GammaAnalytical[S], Gamma1Explicit[S],
  Gamma2Explicit[S], Gamma4Explicit[S], GammaImplicit[S]},
  {S, 80, 200, 10}], TableHeadings →
  {None, {"S", "Exact Γ", "Analytic Γ", "BD1 Γ",
  "BD2 Γ", "BD4 Γ", "IDθ Γ"}}, TableAlignments → Center]

```

Out[170]//TableForm=

S	Exact Γ	Analytic Γ	BD1 Γ	BD2 Γ	BD4 Γ	IDθ Γ
80.0000	0.0161	0.0085	0.0086	0.0086	0.0085	0.0085
90.0000	0.0146	0.0097	0.0096	0.0097	0.0097	0.0097
100.0000	0.0118	0.0087	0.0086	0.0087	0.0087	0.0087
110.0000	0.0088	0.0069	0.0068	0.0068	0.0069	0.0069
120.0000	0.0061	0.0050	0.0050	0.0050	0.0050	0.0050
130.0000	0.0041	0.0034	0.0034	0.0034	0.0034	0.0034
140.0000	0.0026	0.0022	0.0022	0.0022	0.0022	0.0022
150.0000	0.0016	0.0014	0.0014	0.0014	0.0014	0.0014
160.0000	0.0010	0.0009	0.0009	0.0009	0.0009	0.0009
170.0000	0.0006	0.0006	0.0006	0.0006	0.0006	0.0006
180.0000	0.0004	0.0003	0.0003	0.0003	0.0003	0.0003
190.0000	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002
200.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

Table 3. American put option gammas determined by the Black-Scholes theory and by the analytic, explicit, and implicit RBF methods.

■ American Options Numerical Specification

□ Analytical Results

American options will satisfy the same boundary conditions of equation (23) as European options when $S \rightarrow 0$ and $S \rightarrow \infty$, but must also satisfy the early exercise condition. This means that the current American value $V_A(S_t, t)$ is the maximum of the conditional expected future discounted value $\mathbb{E}(e^{-r\Delta t} V(S_{t+\Delta t}, t + \Delta t) | \mathcal{F}_t) = V_E(S, t)$, which is the same as the European option price and the exercise price $K - S_t$. That is,

$$V_A(S_t, t) = \max(V_E(S_t, t), K - S_t) = \max(V_E(S_t, t), U(\log(S_t), T)). \tag{24}$$

While we can implement this formula directly into the RBF formalism simply by taking the European option code in the previous section and comparing it with the exercise prices, this will not yield satisfactory results. The American options are path-dependent and can be exercised at any time, so they do not have one final boundary at maturity, but an infinite number of boundaries that describe the moving free boundary. There is no explicit formula for the American put, and it is likely that there will only ever be numerical approximations.

```

In[171]:= ClearAll[VAmerAnalytical, DeltaAmerAnalytical];
VAmerAnalytical[S_, τ_] := Max[U[Log[S], T - τ], U[Log[S], T]];
VAmerAnalytical[S_] :=
  Max[Table[φ[Log[S], yj], {yj, Log[Smin], Log[Smax], Δy}].
    (Re[W.kvec]), K - S, 0];
DeltaAmerAnalytical[S_] := Max[
  Table[Evaluate[D[φ[y, yj], y]] /. {y → Log[S]},
    {yj, Log[Smin], Log[Smax], Δy}].(Re[W.kvec])/S, -1];
GammaAmerAnalytical[S_] :=
  Table[Evaluate[D[φ[y, yj], {y, 2}]] /. {y → Log[S]},
    {yj, Log[Smin], Log[Smax], Δy}].(Re[W.kvec])/S^2;

```

□ Explicit and Implicit Results

The explicit and implicit backward-difference time integration schemes given by equations (19) to (22) can again be implemented using Do loops. As before, start at time $t = T$ with $U(y, T)$ set to UT as determined by (7) and step backward through time considering earlier times $t_n = T - n \Delta t$ while defining

$\underline{U}(y, t_n) = \underline{U}^n$, $\underline{\alpha}(t_n) = \underline{\alpha}^n$. Again observe that $T - t = T - t_n = n \Delta t$ in the exponent. As with the European options, there are further modifications that need to be undertaken in each step of the Do loop; namely, that the exercise condition (23) must be satisfied for all points y_i . At each time step t_n and each collocation point y_i we determine the i^{th} European option value of \underline{U}^n and then compare it with the exercise value $K - S^i = K - e^{y_i} = K - S_{\min} e^{(i-1)\Delta y}$.

```

In[176]:= ClearAll[Un, α1a, α2a, α4a, αia, U1Amer, U2Amer,
  U4Amer, UiAmer, V1AmerExplicit, V2AmerExplicit,
  V4AmerExplicit, ViAmerImplicit, Delta1AmerExplicit,
  Delta2AmerExplicit, Delta4AmerExplicit, DeltaAmerImplicit];
U0 = UT; α0 = φinv.U0;
Module[{Un}, α1a[0] = α0;
  Do[α1a[n] = (IdentityMatrix[Npts] - Δt B).α1a[n - 1]; Un = φ.α1a[n];
    Do[Un[[i]] = Max[Un[[i]], K - Smin Exp[(i - 1) Δy]], {i, 1, Npts}];
    α1a[n] = φinv.Un, {n, 1, M}]];
Module[{Un, R1, R2}, α2a[0] = α0;
  Do[R1 = -Δt B.α2a[n - 1]; R2 = (IdentityMatrix[Npts] - Δt B/2).R1;
    α2a[n] = α2a[n - 1] + (R1 + R2) / 2; Un = φ.α2a[n];
    Do[Un[[i]] = Max[Un[[i]], K - Smin Exp[(i - 1) Δy]], {i, 1, Npts}];
    α2a[n] = φinv.Un, {n, 1, M}]];
Module[{Un, R1, R2, R3, R4}, α4a[0] = α0;
  Do[R1 = -Δt B.α4a[n - 1]; R2 = (IdentityMatrix[Npts] - Δt B/2).R1;
    R3 = (R1 - Δt B.R2/2); R4 = (R1 - Δt B.R3);
    α4a[n] = α4a[n - 1] + (R1 + 2 R2 + 2 R3 + R4) / 6; Un = φ.α4a[n];
    Do[Un[[i]] = Max[Un[[i]], K - Smin Exp[(i - 1) Δy]], {i, 1, Npts}];

```

```

 $\alpha_{4a}[n] = \phi_{\text{inv}} \cdot \text{Un}, \{n, 1, M\}];$ 
Module[{Un, B2 = (IdentityMatrix[Npts] -  $\theta \Delta t$  B), B1}, B1 = B2 +  $\Delta t$  B;
 $\alpha_{ia}[0] = \alpha_0$ ; Do[ $\alpha_{ia}[n] = \text{Inverse}[B1] \cdot B2 \cdot \alpha_{ia}[n - 1]$ ; Un =  $\phi \cdot \alpha_{ia}[n]$ ;
Do[Un[[i]] = Max[Un[[i]], K - Smin Exp[(i - 1)  $\Delta y$ ]], {i, 1, Npts}];
 $\alpha_{ia}[n] = \phi_{\text{inv}} \cdot \text{Un}, \{n, 1, M\}];$ 

```

Here the initial and boundary conditions according to equations (7) and (23) are specified.

```

In[182]:= U1Amer[y_, 0] := Max[K - Exp[y], 0];
U2Amer[y_, 0] := Max[K - Exp[y], 0];
U4Amer[y_, 0] := Max[K - Exp[y], 0];
UiAmer[y_, 0] := Max[K - Exp[y], 0];
U1Amer[y_, n_Integer] /; y  $\geq$  Log[Smax] := 0;
U2Amer[y_, n_Integer] /; y  $\geq$  Log[Smax] := 0;
U4Amer[y_, n_Integer] /; y  $\geq$  Log[Smax] := 0;
UiAmer[y_, n_Integer] /; y  $\geq$  Log[Smax] := 0;

```

Again, the general formulas for $U(y, t)$, $0 \leq t \leq T$, $0 \leq y \leq \log(S_{\text{max}})$ can now be specified according to the left-hand side of equation (17). We can also get the explicit and implicit approximation formulas for $V(S, t)$ by using the updated values for $\alpha(t_n) = \alpha^n$. The hedging delta formulas are given in terms of $U(\log(S), t)$ by simply taking the derivatives of the RBFs $\phi(y, y_j)$.

```

In[186]:= U1Amer[y_, n_Integer] /; y < Log[Smax] :=
Table[ $\phi[y, y_j]$ , {y_j, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_{1a}[n]$ ;
U2Amer[y_, n_Integer] /; y < Log[Smax] :=
Table[ $\phi[y, y_j]$ , {y_j, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_{2a}[n]$ ;
U4Amer[y_, n_Integer] /; y < Log[Smax] :=
Table[ $\phi[y, y_j]$ , {y_j, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_{4a}[n]$ ;
UiAmer[y_, n_Integer] /; y < Log[Smax] :=
Table[ $\phi[y, y_j]$ , {y_j, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_{ia}[n]$ ;
V1AmerExplicit[S_, n_Integer] :=
Max[U1Amer[Log[S], n], U1Amer[Log[S], 0]];
V2AmerExplicit[S_, n_Integer] :=
Max[U2Amer[Log[S], n], U2Amer[Log[S], 0]];
V4AmerExplicit[S_, n_Integer] :=
Max[U4Amer[Log[S], n], U4Amer[Log[S], 0]];
ViAmerImplicit[S_, n_Integer] :=
Max[UiAmer[Log[S], n], UiAmer[Log[S], 0]];
V1AmerExplicit[S_] := Max[Table[ $\phi[\text{Log}[S], y_j]$ ,
{y_j, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_{1a}[M]$ , K - S, 0];
V2AmerExplicit[S_] := Max[Table[ $\phi[\text{Log}[S], y_j]$ ,
{y_j, Log[Smin], Log[Smax],  $\Delta y$ }] .  $\alpha_{2a}[M]$ , K - S, 0];
V4AmerExplicit[S_] := Max[Table[ $\phi[\text{Log}[S], y_j]$ ,

```

```

{yj, Log[Smin], Log[Smax], Δy}] . α4a[M], K - S, 0];
ViAmerImplicit[S_] := Max[Table[φ[Log[S], yj],
{yj, Log[Smin], Log[Smax], Δy}] . αia[M], K - S, 0];
Delta1AmerExplicit[S_] := Table[Evaluate[D[φ[y, yj], y]] /.
{y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}] . αia[M]/S;
Delta2AmerExplicit[S_] := Table[Evaluate[D[φ[y, yj], y]] /.
{y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}] . α2a[M]/S;
Delta4AmerExplicit[S_] := Table[Evaluate[D[φ[y, yj], y]] /.
{y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}] . α4a[M]/S;
DeltaAmerImplicit[S_] := Table[Evaluate[D[φ[y, yj], y]] /.
{y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}] . αia[M]/S;
Gamma1AmerExplicit[S_] := Table[Evaluate[D[φ[y, yj], {y, 2}]] /.
{y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}] . α1a[M]/S2;
Gamma2AmerExplicit[S_] := Table[Evaluate[D[φ[y, yj], {y, 2}]] /.
{y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}] . α2a[M]/S2;
Gamma4AmerExplicit[S_] := Table[Evaluate[D[φ[y, yj], {y, 2}]] /.
{y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}] . α4a[M]/S2;
GammaAmerImplicit[S_] := Table[Evaluate[D[φ[y, yj], {y, 2}]] /.
{y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}] . αia[M]/S2;

```

For the sake of calculating actual prices, we use the same market parameters as before. As with the European options, observe that we calculate not just one value for the analytical and approximation results but all of the possible values for the American put option over its spatial and time domains. The time taken for these calculations is only incrementally longer than for the European case. In order to compare the RBF values for the American put with other schemes, we use the equal jumps additive binomial model described in Clewlow and Strickland [10], Chapter 2 and implemented here in *Mathematica*.

```

In[206]:= EqualJumpsAdditiveBinomialAmericanPut [
  S_?NumberQ, K_?NumberQ, r_?NumberQ, q_?NumberQ,
  sig_?NumberQ, T_?NumberQ, n_?NumberQ] :=
Module[{i, j, dt = T/n, nu = r - q - 0.5 sig^2,
  dx, pu, pd, disc, eqdt, dpu, dpd, edx, St, C},
  dx = Sqrt[sig^2 dt + (nu dt)^2];
  pu = 1/2 + (nu dt/dx)/2; pd = 1 - pu;
  disc = Exp[-r dt]; eqdt = Exp[q dt];
  dpu = disc pu; dpd = disc pd;
  edx = Exp[dx];
  St[-n] = S Exp[-n dx];
  Do[St[j] = St[j - 1] edx, {j, -n + 1, n}];
  Do[C[j] = Max[0, K - St[j]], {j, -n, n, 2}];
  Do[C[j] = dpu C[j + 1] + dpd C[j - 1]; St[j] = eqdt St[j];
  C[j] = Max[C[j], K - St[j]], {i, n - 1, 0, -1}, {j, -i, i, 2}];
  C[0] ]

```

We use this function with a large number $N = 500$ steps to calculate highly accurate American put prices at the same stock values as for the European options. By varying the stock values by the incremental amount of $\Delta S = \$0.2$, we can additionally get values for the American put deltas.

```
In[207]:= binomialTree = Table[EqualJumpsAdditiveBinomialAmericanPut[
      S, K, r, 0, σ, T, 500], {S, 80, 200, 10}]
Out[207]:= {20.2674, 13.1197, 8.33587, 5.21252, 3.20937, 1.95432, 1.18057,
      0.706511, 0.423226, 0.251961, 0.149678, 0.0890653, 0.0531858}

In[208]:= binomialTree2 =
      Table[EqualJumpsAdditiveBinomialAmericanPut[S, K, r, 0, σ, T, 500],
      {S, 80.2, 200.2, 10.0}]
Out[208]:= {20.0961, 13.0046, 8.26003, 5.16277, 3.17714, 1.93422, 1.1686,
      0.699655, 0.418883, 0.24923, 0.147999, 0.0880566, 0.0525953}

In[209]:= binDelta = (binomialTree2 - binomialTree)/0.2
Out[209]:= {-0.856572, -0.575645, -0.379222, -0.248721,
      -0.161147, -0.100498, -0.0598567, -0.034283, -0.0217178,
      -0.0136589, -0.00839964, -0.00504329, -0.00295229}
```

In Tables 4 and 5 we can see that while the RBF time integration schemes work very well to approximate the American put, the analytical result is not a sufficiently sensitive measure of the influence of the moving boundary.

```
In[210]:= TableForm[Table[(PaddedForm[N[#1], {7, 4}] &)/@
      {S, binomialTree[[S/10 - 7]]},
      VAmerAnalytical[S], ViAmerExplicit[S],
      V2AmerExplicit[S], V4AmerExplicit[S], ViAmerImplicit[S]],
      {S, 80, 200, 10}], TableHeadings -> {None,
      {"S", "Binomial", "Analytic", "BD1", "BD2", "BD4", "IDθ"}},
      TableAlignments -> Center]
```

Out[210]//TableForm=

S	Binomial	Analytic	BD1	BD2	BD4	IDθ
80.0000	20.2674	20.0000	20.1950	20.1914	20.1850	20.1848
90.0000	13.1197	10.9866	12.9464	12.9317	12.9172	12.9180
100.0000	8.3359	7.2008	8.1081	8.0926	8.0785	8.0797
110.0000	5.2125	4.5975	4.9468	4.9343	4.9234	4.9244
120.0000	3.2094	2.8753	2.9239	2.9155	2.9083	2.9092
130.0000	1.9543	1.7696	1.6526	1.6481	1.6445	1.6452
140.0000	1.1806	1.0756	0.8651	0.8638	0.8631	0.8637
150.0000	0.7065	0.6474	0.3824	0.3835	0.3847	0.3852
160.0000	0.4232	0.3867	0.0887	0.0913	0.0938	0.0943
170.0000	0.2520	0.2297	0.0000	0.0000	0.0000	0.0000
180.0000	0.1497	0.1366	0.0000	0.0000	0.0000	0.0000
190.0000	0.0891	0.0826	0.0000	0.0000	0.0000	0.0000
200.0000	0.0532	0.0532	0.0000	0.0000	0.0000	0.0000

Table 4. American put option prices determined by the binomial and analytic, explicit, and implicit RBF methods.

```

In[211]:= TableForm[
  Table[(PaddedForm[N[#1], {7, 4}] &) /@ {S, binDelta[[S/10 - 7]],
    DeltaAmerAnalytical[S],
    Delta1AmerExplicit[S], Delta2AmerExplicit[S],
    Delta4AmerExplicit[S], DeltaAmerImplicit[S]},
    {S, 80, 200, 10}], TableHeadings →
  {None, {"S", "Binomial Δ", "Analytic Δ", "BD1 Δ",
    "BD2 Δ", "BD4 Δ", "IDθ Δ"}}, TableAlignments → Center]

```

Out[211]//TableForm=

S	Binomial Δ	Analytic Δ	BD1 Δ	BD2 Δ	BD4 Δ	IDθ Δ
80.0000	-0.8566	-0.6030	-0.8756	-0.8776	-0.8805	-0.8805
90.0000	-0.5756	-0.4475	-0.5893	-0.5897	-0.5901	-0.5901
100.0000	-0.3792	-0.3144	-0.3891	-0.3890	-0.3888	-0.3888
110.0000	-0.2487	-0.2114	-0.2514	-0.2510	-0.2507	-0.2507
120.0000	-0.1611	-0.1375	-0.1593	-0.1589	-0.1585	-0.1585
130.0000	-0.1005	-0.0871	-0.0993	-0.0990	-0.0986	-0.0987
140.0000	-0.0599	-0.0541	-0.0611	-0.0609	-0.0606	-0.0606
150.0000	-0.0343	-0.0331	-0.0373	-0.0371	-0.0369	-0.0369
160.0000	-0.0217	-0.0200	-0.0226	-0.0225	-0.0224	-0.0224
170.0000	-0.0137	-0.0120	-0.0137	-0.0136	-0.0136	-0.0136
180.0000	-0.0084	-0.0070	-0.0082	-0.0082	-0.0082	-0.0082
190.0000	-0.0050	-0.0040	-0.0049	-0.0049	-0.0049	-0.0049
200.0000	-0.0030	-0.0020	-0.0028	-0.0028	-0.0028	-0.0028

Table 5. American put option deltas determined by the binomial and analytic, explicit, and implicit RBF methods.

A final check on the accuracy of the results for American options can be obtained by loading the powerful financial software *UnRisk* (www.unriskderivatives.com). It uses compiled backward-recursive path integration over a large grid to obtain highly accurate estimates of path-dependent and American option values. The values returned by the calculation are {Put value, Delta, Gamma, Theta, Vega, VolConvexity, DeltaVega}. In Tables 6, 7, and 8 we can see that while the RBF backward-integration methodology works very well to approximate the American put, the analytical result is still not sufficiently sensitive to the behavior of the moving boundary.

```

In[212]:= Needs["UnRisk`UnRiskFrontEnd`"];
MyEquities =
  Table[MakeEquity[S, EquityYield → q], {S, 80, 200, 10}];
MyAmericanPuts = Table[MakeVanillaEquityOption[
  MyEquities[[i]], K, {2007, 4, 23}, OptionType → "Put",
  ExerciseType → "American"], {i, 13}];
MyYieldCurve = MakeYieldCurve[r]; MyVolCurve =
  MakeVolatilityCurve[σ]; M = 100;

```

```
In[216]:= MyAmerValues =
  Table[Valuate[MyAmericanPuts[[i]], {2006, 4, 23}, {2006, 4, 24},
    MyVolCurve, MyYieldCurve, CalculateVega -> True], {i, 13}];
TableForm[Map[PaddedForm[N[#1], {7, 4}] &,
  Join[Transpose[{Range[80, 200, 10]}], MyAmerValues, 2],
  {2}], TableHeadings ->
  {None, {"S", "Put", "Delta", "Gamma", "Theta", "Vega",
    "VolConvexity", "DeltaVega"}}, TableAlignments -> Center]
```

Out[216]//TableForm=

S	Put	Delta	Gamma	Theta	Vega	VolConvexity	DeltaVega
80.0000	20.1072	-0.8937	0.0513	-0.0176	0.0973	0.0445	0.0426
90.0000	12.9840	-0.5784	0.0230	-0.0051	0.3113	0.0042	0.0098
100.0000	8.2379	-0.3822	0.0163	-0.0074	0.3579	0.0008	0.0003
110.0000	5.1387	-0.2465	0.0111	-0.0077	0.3342	0.0033	-0.0044
120.0000	3.1598	-0.1556	0.0073	-0.0070	0.2797	0.0069	-0.0061
130.0000	1.9209	-0.0965	0.0047	-0.0058	0.2185	0.0095	-0.0060
140.0000	1.1578	-0.0590	0.0029	-0.0045	0.1625	0.0109	-0.0051
150.0000	0.6937	-0.0357	0.0018	-0.0034	0.1167	0.0109	-0.0040
160.0000	0.4140	-0.0214	0.0011	-0.0024	0.0816	0.0097	-0.0030
170.0000	0.2465	-0.0128	0.0007	-0.0017	0.0558	0.0081	-0.0022
180.0000	0.1467	-0.0076	0.0004	-0.0012	0.0376	0.0065	-0.0015
190.0000	0.0873	-0.0045	0.0002	-0.0008	0.0251	0.0051	-0.0010
200.0000	0.0520	-0.0027	0.0001	-0.0005	0.0166	0.0038	-0.0007

```
In[217]:= TableForm[Table[(PaddedForm[N[#1], {7, 4}] &) /@
  {S, MyAmerValues[[S/10 - 7, 1]],
  VAmerAnalytical[S], V1AmerExplicit[S],
  V2AmerExplicit[S], V4AmerExplicit[S], ViAmerImplicit[S]},
  {S, 80, 200, 10}], TableHeadings -> {None,
  {"S", "Quadrature", "Analytic", "BD1", "BD2", "BD4", "ID0"}},
  TableAlignments -> Center]
```

Out[217]//TableForm=

S	Quadrature	Analytic	BD1	BD2	BD4	ID0
80.0000	20.1072	20.0000	20.2411	20.2400	20.2364	20.2347
90.0000	12.9840	10.9926	13.1319	13.1182	13.1043	13.1045
100.0000	8.2379	7.2071	8.3540	8.3375	8.3225	8.3232
110.0000	5.1387	4.6041	5.2226	5.2080	5.1953	5.1961
120.0000	3.1598	2.8822	3.2173	3.2063	3.1970	3.1978
130.0000	1.9209	1.7766	1.9584	1.9512	1.9453	1.9459
140.0000	1.1578	1.0828	1.1811	1.1770	1.1740	1.1743
150.0000	0.6937	0.6548	0.7072	0.7054	0.7043	0.7045
160.0000	0.4140	0.3943	0.4216	0.4211	0.4212	0.4214
170.0000	0.2465	0.2376	0.2513	0.2517	0.2524	0.2525
180.0000	0.1467	0.1445	0.1509	0.1517	0.1528	0.1528
190.0000	0.0873	0.0907	0.0932	0.0942	0.0954	0.0954
200.0000	0.0520	0.0615	0.0619	0.0630	0.0642	0.0642

Table 6. American put option prices determined by the quadrature and analytic, explicit, and implicit RBF methods.

```

In[218]:= TableForm[Table[
  (PaddedForm[N[#1], {7, 4}] &) /@ {S, MyAmerValues[[S/10 - 7, 2]],
    DeltaAmerAnalytical[S], Delta1AmerExplicit[S],
    Delta2AmerExplicit[S], Delta4AmerExplicit[S],
    DeltaAmerImplicit[S]},
  {S, 80, 200, 10}], TableHeadings →
  {None, {"S", "Quadrature Δ", "Analytic Δ", "BD1 Δ",
    "BD2 Δ", "BD4 Δ", "IDθ Δ"}}, TableAlignments → Center]

```

Out[218]//TableForm=

S	Quadrature Δ	Analytic Δ	BD1 Δ	BD2 Δ	BD4 Δ	IDθ Δ
80.0000	-0.8937	-0.6030	-0.8556	-0.8552	-0.8571	-0.8576
90.0000	-0.5784	-0.4475	-0.5799	-0.5801	-0.5806	-0.5807
100.0000	-0.3822	-0.3143	-0.3849	-0.3846	-0.3845	-0.3846
110.0000	-0.2465	-0.2114	-0.2491	-0.2487	-0.2483	-0.2484
120.0000	-0.1556	-0.1374	-0.1579	-0.1574	-0.1570	-0.1571
130.0000	-0.0965	-0.0871	-0.0983	-0.0979	-0.0976	-0.0976
140.0000	-0.0590	-0.0541	-0.0602	-0.0600	-0.0598	-0.0598
150.0000	-0.0357	-0.0331	-0.0364	-0.0363	-0.0361	-0.0361
160.0000	-0.0214	-0.0200	-0.0219	-0.0217	-0.0216	-0.0217
170.0000	-0.0128	-0.0120	-0.0130	-0.0129	-0.0129	-0.0129
180.0000	-0.0076	-0.0070	-0.0076	-0.0075	-0.0075	-0.0075
190.0000	-0.0045	-0.0040	-0.0043	-0.0042	-0.0042	-0.0042
200.0000	-0.0027	-0.0020	-0.0022	-0.0022	-0.0022	-0.0022

Table 7. American put option deltas determined by the quadrature and analytic, explicit, and implicit RBF methods.

```

In[219]:= TableForm[Table[
  (PaddedForm[N[#1], {7, 4}] &) /@ {S, MyAmerValues[[S/10 - 7, 3]],
    GammaAmerAnalytical[S], Gamma1AmerExplicit[S],
    Gamma2AmerExplicit[S], Gamma4AmerExplicit[S],
    GammaAmerImplicit[S]},
  {S, 80, 200, 10}], TableHeadings →
  {None, {"S", "Quadrature Γ", "Analytic Γ", "BD1 Γ",
    "BD2 Γ", "BD4 Γ", "IDθ Γ"}}, TableAlignments → Center]

```

Out[219]//TableForm=

S	Quadrature Γ	Analytic Γ	BD1 Γ	BD2 Γ	BD4 Γ	ID θ Γ
80.0000	0.0513	0.0085	0.0269	0.0262	0.0259	0.0263
90.0000	0.0230	0.0097	0.0167	0.0166	0.0165	0.0167
100.0000	0.0163	0.0087	0.0123	0.0122	0.0122	0.0123
110.0000	0.0111	0.0069	0.0089	0.0088	0.0088	0.0089
120.0000	0.0073	0.0050	0.0061	0.0061	0.0061	0.0061
130.0000	0.0047	0.0034	0.0040	0.0040	0.0040	0.0040
140.0000	0.0029	0.0022	0.0025	0.0025	0.0025	0.0025
150.0000	0.0018	0.0014	0.0016	0.0016	0.0016	0.0016
160.0000	0.0011	0.0009	0.0010	0.0010	0.0010	0.0010
170.0000	0.0007	0.0006	0.0006	0.0006	0.0006	0.0006
180.0000	0.0004	0.0003	0.0004	0.0004	0.0004	0.0004
190.0000	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002
200.0000	0.0001	0.0001	0.0002	0.0002	0.0002	0.0002

Table 8. American put option gammas determined by the quadrature and analytic, explicit, and implicit RBF methods.

□ **Optimal Exercise Boundary for the American Put**

Furthermore, these programs can now be used to efficiently determine the difficult numerical problem of the moving free boundary for the American option. The concept of the free boundary is that it is the function $FB(t)$ that describes the point at time t where for values of $S_t > FB_t$ the stock price is not so small as to warrant early exercise, and it remains better to hold onto the option, so that it behaves like the European option and satisfies the Black-Scholes PDE of equation (2). However, if $S_t \leq FB_t$, then the stock has become sufficiently small as to warrant immediate exercise. On the boundary the value of the option is the exercise price resulting in the equation

$$V(FB(t), t) = U(y(t), t) = K - FB(t) = K - e^{y(t)}. \tag{25}$$

Consider the time $t_n = T - n \Delta t$; then at that time $U(y_n, t_n) = K - e^{y_n}$ and $F(y_n) = U(y_n, t_n) - K + e^{y_n} = 0$. We now define these $F(y)$ functions.

```
In[220]:= ClearAll[F1, F2, F4, Fi];
          F1[y_, n_Integer] := U1Amer[y, n] - K + e^y;
          F2[y_, n_Integer] := U2Amer[y, n] - K + e^y;
          F4[y_, n_Integer] := U4Amer[y, n] - K + e^y;
          Fi[y_, n_Integer] := UiAmer[y, n] - K + e^y;
```

```
In[225]:= Off[FindRoot::"lstol"]
```

To check that the zero function $F(y, n)$ is working properly, we use FindRoot, starting at $y_0 = \log(K)$.

```
In[226]:= y /. FindRoot[Fi[y, M], {y, Log[K]}, AccuracyGoal -> 10,
                       WorkingPrecision -> 20, MaxIterations -> 300]
```

Out[226]= 4.3500072767206089575

At time $t_n = T$ when $n = 0$, $\text{FB}(T) = K$ and hence $y(T) = \log(K)$. For successive values of n , we move further back in time to the present and $\text{Fi}[y, n]$ determines those values y_n such that $\log(y_n) = \text{FB}(t_n)$. This procedure is implemented using `NestList`, in which successive solutions are used as the starting point for the next evaluation of the `FindRoot` function.

```
In[227]:= exerciseBdryPts = NestList[#[[1]] + 1,
      y /. FindRoot[Fi[y, #[[1]] + 1], {y, #[[2]]}, AccuracyGoal -> 10,
      WorkingPrecision -> 20, MaxIterations -> 300] &, {0, Log[K]}, M]
```

```
Out[227]:= {{0, Log[100]}, {1, 4.5561932385336261842},
  {2, 4.5436871507129802744}, {3, 4.5234387898578146500},
  {4, 4.5000071431279923528}, {5, 4.4999986047562968794},
  {6, 4.4999970891333575006}, {7, 4.4977939168859925177},
  {8, 4.4911976516584216455}, {9, 4.4814123344580258802},
  {10, 4.4675186036495674686},
  {11, 4.4499200992474832487}, {12, 4.4499941509843673905},
  {13, 4.4499996810722396164}, {14, 4.4500022628395216874},
  {15, 4.4500028801041949433}, {16, 4.4500036851563828342},
  {17, 4.4500040255741613136}, {18, 4.4500044167108636958},
  {19, 4.4497999597851553456}, {20, 4.4475529642319670852},
  {21, 4.4441290417701749413}, {22, 4.4397703062635811122},
  {23, 4.4345081974563303855}, {24, 4.4282537293048573306},
  {25, 4.4207944459214501354}, {26, 4.4117589851304339383},
  {27, 4.4008960897819213631}, {28, 4.4000309240267231047},
  {29, 4.4000098918473913176}, {30, 4.4000045843086320959},
  {31, 4.3999994779805448553}, {32, 4.3999990423301953491},
  {33, 4.3999971892173454803}, {34, 4.3999975369740232652},
  {35, 4.3999958752948114178}, {36, 4.3999959555671765290},
  {37, 4.3999948864553056755}, {38, 4.3999945828136903349},
  {39, 4.3999947408613160717}, {40, 4.3999944779944486649},
  {41, 4.3999938860176458550}, {42, 4.3999941601254691854},
  {43, 4.3999938666676987319}, {44, 4.3999934836647549014},
  {45, 4.3999929725496883359}, {46, 4.3999931351484446102},
  {47, 4.3991586818972903929}, {48, 4.3978013322568408437},
  {49, 4.3960807398960029235}, {50, 4.3940728474171419519},
  {51, 4.3918155079754839831}, {52, 4.3893266995970437892},
  {53, 4.3866107962779580188}, {54, 4.3836630710763167523},
  {55, 4.3804691705480343044}, {56, 4.3770099431876408354},
  {57, 4.3732550897390454802}, {58, 4.3691648448976707225},
  {59, 4.3646844449033349648}, {60, 4.3597333815258148126},
  {61, 4.3541604046384914749}, {62, 4.3491198902332688492},
  {63, 4.3499022166743211253}, {64, 4.3499546829139412401},
  {65, 4.3499688309030937187}, {66, 4.3499795636567580726},
  {67, 4.3499877501387976394}, {68, 4.3499899767448375606},
  {69, 4.3499928790239539119}, {70, 4.3499962008878066678},
  {71, 4.3499974597170615801}, {72, 4.3499987034351720258},
  {73, 4.3499989643146470757}, {74, 4.3500001681914532512},
  {75, 4.3499999076901291465}, {76, 4.3500017072156979079},
  {77, 4.3500027240591525012}, {78, 4.3500031967844006864},
  {79, 4.3500033453450793815}, {80, 4.3500041048372897077},
```

```
{81, 4.3500029898629113034}, {82, 4.3500044586675010166},
{83, 4.3500048971694963586}, {84, 4.3500049775243699405},
{85, 4.3500050512915995347}, {86, 4.3500057220434790997},
{87, 4.3500053675533061998}, {88, 4.3500061574554390510},
{89, 4.3500059890093150535}, {90, 4.3500064664345208233},
{91, 4.3500065229615567359}, {92, 4.3500063592788685493},
{93, 4.3500063628618517254}, {94, 4.3500064795975806952},
{95, 4.3500065188149172577}, {96, 4.3500067515741859209},
{97, 4.3500072470502405422}, {98, 4.3500067211117741922},
{99, 4.3500074910176451767}, {100, 4.3500072767202724053}]
```

```
In[228]:= ListLinePlot[Transpose[exerciseBdryPts][[2]], PlotRange -> {60, 100},
  AxesLabel -> {"Time Remaining", "Free Boundary"},
  PlotLabel -> "Free Boundary Determined by RBF"]
```

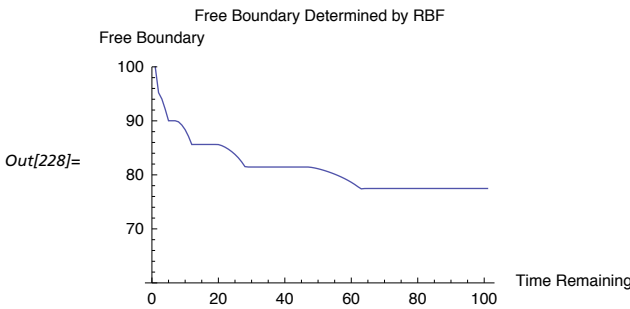


Figure 1. Moving free boundary determined by RBF.

□ 3D Graphs of American Options

The given code evaluates all American option values for every point in time t_n and at every collocation point y_j . This yields a complete picture of the American put as it evolves in time and space, as shown by the following.

```
In[228]:= ViamerOut = Table[{S, n, ViAmerImplicit[S, n]},
  {S, 80, 200, 10}, {n, M, 0, -10}];
ViamerOut = Map[#[[3]] &, ViamerOut, {2}];
```

```

In[230]:= ListPlot3D[ViamerOut, DataRange -> {{80, 200}, {0, 1}},
  AxesLabel -> {"Stock", "Time", "American Put"},
  ViewPoint -> {3.078, -1.063, 0.655},
  PlotLabel -> "American Put Determined by Implicit RBF"]

```

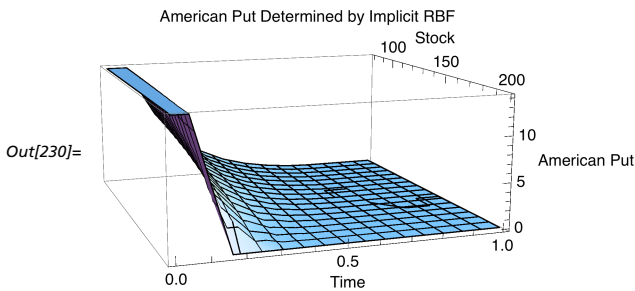


Figure 2. 3D plot of the American put determined by the implicit RBF.

Conclusion

In this article we investigated the use of radial basis functions (RBFs) as a means of solving parabolic partial differential equations (PDEs) associated with the solution of European- and American-style financial options. Unlike the usual approach, which typically involves finite difference (FD) schemes, we have used a spatial collocation approach that has resulted in analytical, explicit, and implicit evaluation schemes. We have shown that the RBF methodology can be easily represented using *Mathematica's* powerful programming idiom. Furthermore, there are a number of advantages of this approach over that of the FD method. It is quicker and yields more accurate results in the same time. It is comprehensive, resulting in a complete description of the path of the option for a complete range of values in space and time, as was displayed in Figures 1 and 2. And because the RBFs are themselves readily differentiated, the hedging parameters are also immediately calculable. Lastly, there is a range of parameters such as the shape parameter κ , the number of time steps M , and the spatial dimension N that can be adjusted so as to further improve results.

References

- [1] R. L. Hardy, "Multiquadratic Equations of Topography and Other Irregular Surfaces," *Journal of Geophysical Research*, **76**(8), 1971 pp. 1905-1915.
- [2] R. L. Hardy, "Theory and Applications of the Multiquadric-Biharmonic Method," *Computers and Mathematics with Applications*, **19**(8-9), 1990 pp. 163-208.
- [3] E. J. Kansa, "Multiquadrics—A Scattered Data Approximation Scheme with Applications to Computational Fluid Dynamics I: Surface Approximations and Partial Derivative Estimates," *Computers and Mathematics with Applications*, **19**(8-9), 1990 pp. 127-145.
- [4] E. J. Kansa, "Multiquadrics—A Scattered Data Approximation Scheme with Applications to Computational Fluid Dynamics II: Solutions to Parabolic, Hyperbolic and Elliptic Partial Differential Equations," *Computers and Mathematics with Applications*, **19**(8-9), 1990 pp. 147-161.

- [5] Y-C. Hon and X-Z. Mao, "A Radial Basis Function Method for Solving Options Pricing Models," *Financial Engineering*, **8**, 1999 pp. 31-49.
- [6] G. Fasshauer, A. Q. M. Khaliq, and D. A. Voss, "Using Meshfree Approximations for Multi-Asset American Option Problems," *Journal of Chinese Institute of Engineers (JCIE)*, **27**(4), 2004 pp. 563-571.
- [7] F. Black and M. Scholes, "The Pricing of Options and Corporate Liabilities," *Journal of Political Economy*, **81**(3), 1973 pp. 637-654.
- [8] M. A. Goldberg and C. S. Chen, "On a Method of Atkinson for Evaluating Domain Integrals in the Boundary Element Method," *Applied Mathematics and Computation*, **80**(2-3), 1994 pp. 125-138.
- [9] M. J. D. Powell, "The Theory of Radial Basis Function Approximations in 1990," *Advances in Numerical Analysis: Wavelets, Subdivision Algorithms, and Radial Basis Functions*, Vol. II (W. A. Light, ed.), Oxford: Oxford University Press, 1992 pp. 105-210.
- [10] L. Clewlow and C. Strickland, *Implementing Derivatives Models (Wiley Series in Financial Engineering)*, New York: John Wiley & Sons, 1998.

About the Author

Michael Kelly is currently the Research Director for Unetich Trading LLC, which operates at the Chicago Board of Trade. He designs automated trading programs using *Mathematica* that implement trading strategies based upon statistical rules. Previously he was a professor of mathematical and computational finance in the Stuart School of Business, Illinois Institute of Technology and a derivatives consultant. Prior to that he was senior lecturer in both mathematics and finance in what is now the School of Computing and Mathematics and the School of Economics and Finance, respectively, at the University of Western Sydney, Australia. He has also been a visiting research fellow at the University of Sydney, the Australian National University and the University of Warwick (U.K.). For at least 15 years his lectures and research have been based upon computational mathematics and maths programs such as *Mathematica* and Matlab.

Michael Kelly

Research Director

Unetich Trading LLC, Suite 2020, Chicago Board of Trade
141 W. Jackson Boulevard, Chicago, Illinois 60604-2994
mk1444@rcn.com