Evaluation Of Financial Options using Radial Basis Functions

Dr Michael Kelly

Stuart Graduate School of Business , Illinois Institute of Technology
mkelly@stuart.iit.edu

■ Introduction

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

$$\phi(x, x_i) = \phi(\| x - x_i \|) \tag{1}$$

Each function $\phi(x, x_i)$ is radially symmetric about the center x_i . Since their discovery in the early 1970's by Hardy[1971, 1990] RBFs have become *a* primary tool for the interpolation of multidimensional scattered data. In the 1990's Kansa[1990 *a*, 1990 *b*] showed that RBF methods were applicable to the solution of 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 mean that they are applicable to the Black Scholes PDE and hence to the evaluation of financial options.

The purpose of this paper is to apply global radial basis functions 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[1999] and Fasshauer, Khaliq and Voss[2004]. While some results have been presented, none of the papers mentioned actually exhibits any code or discusses 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. It will be shown 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 PDE

It has been shown by Black and Scholes[1973] that assuming the underlying asset price is risk−neutral, all European options satisfy a lognormal parabolic partial differential equation, 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, volatility or annualised standard deviation of the asset price σ, then the equation governing all european options is:

$$\begin{aligned} &V^{(0,1)}(S,\ t) + (r - q)\,S\,V^{(1,0)}(S,\ t) + \\ &\quad \tfrac{1}{2}\,S^2\,\sigma^2\,V^{(2,0)}(S,\ t) - r\,V(S,\ t) == 0 \end{aligned} \tag{2}$$

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

$$V(S,\ T) = \begin{cases} \text{Max}(K - S,\ 0) & \text{for Put} \\ \text{Max}(S - K,\ 0) & \text{for Call} \end{cases} \tag{3}$$

For european options Black and Scholes[1973] have shown that for $\mathcal{N}(z)$ as the cumulative distribution function of the standard normal distribution then the explicit solution is given by

$$\begin{aligned} V(S,\ t) &= \begin{cases} S\,e^{-q(T-t)}\,\mathcal{N}(d_1) - K\,e^{-r(T-t)}\,\mathcal{N}(d_2) & \text{for calls} \\ K\,e^{-r(T-t)}\,\mathcal{N}(-d_2) - S\,e^{-q(T-t)}\,\mathcal{N}(-d_1) & \text{for puts} \end{cases} \\ d_1 &= \frac{\text{Log}(S/K) + (r - q + \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}} \\ d_2 &= d_1 - \sigma\sqrt{T-t} \end{aligned} \tag{4}$$

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

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

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

$$\begin{aligned} &U^{(0,1)}(y,\ t) + (r - q - \sigma^2/2)\,U^{(1,0)}(y,\ t) + \\ &\quad \tfrac{1}{2}\,\sigma^2\,U^{(2,0)}(y,\ t) - r\,U(y,\ t) == 0 \end{aligned} \tag{6}$$

with initial condition

$$U(y,\ T) = \begin{cases} \text{Max}(K - e^y,\ 0) & \text{for Put} \\ \text{Max}(e^y - K,\ 0) & \text{for Call} \end{cases} \tag{7}$$

The RBF methodology is to represent the option function $U(y,\ t)$ as a linear combination of radial basis functions $\phi_i(y)$, $i = 1,\ ..,\ N$ 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 radial basis functions the most common being Hardy's multiquadratic (MQ) $\phi(y,\ y_j) = \sqrt{\kappa^2 + (y - y_j)}$ and the Gaussian $\phi(y,\ y_j) = e^{-\kappa^2\,(y - y_j)}$ where κ is the shape parameter. Extensive research by Kansa [1990] and Goldeberg et al [1996] have 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 Hardy's [1971] recommendation that $\kappa = \text{Min}(\| x - x_i \|)$ for collocation points x_i. Substituting equation (8) into (6) we now have a system of N linear equations, $i = 1, .., N$:

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

The above equation 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 multiquadratic RBFs initially described in equation (1) and the paragraph above with the shape parameter κ can be easily determined in Mathematica

```
ϕ[y1_, y2_] := Sqrt[(y1 - y2)^2 + κ^2];
{D[ϕ[yi, yj], yi], D[ϕ[yi, yj], {yi, 2}]}
```

$$\left\{\frac{y_i - y_j}{\sqrt{\kappa^2 + (y_i - y_j)^2}},\ \frac{1}{\sqrt{\kappa^2 + (y_i - y_j)^2}} - \frac{(y_i - y_j)^2}{\left(\kappa^2 + (y_i - y_j)^2\right)^{3/2}}\right\}$$

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

$$\begin{aligned} &\Phi_{i,j} = \phi(y_i, y_j),\ (\Phi_y)_{i,j} = \phi^{(1,0)}(y_i, y_j), \\ &(\Phi_{y,y})_{i,j} = \phi^{(2,0)}(y_i, y_j),\ \alpha_j = (\alpha)_j \end{aligned} \tag{11}$$

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

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

Solving the above equation for α allows the estimation of $U(y, t) = \Phi \cdot \alpha(t)$ and hence of the original option price $V(S, t)$. It has been shown by Powell [1992] that the matrix Φ is invertible. This allows us to rewrite equation (12) as a matrix differential equation for α

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

B is the following N x N matrix and since all its components such as Φ and the identity matrix I are known then it is readily computed:

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

There are two approaches to solving equation (13) –analytical and a backward time integration scheme. Using the work of Powell[1992] it follows that if Φ is invertible and the eigenvectors of B are independent then equation (12) has a solution for α in terms of its eigenvectors w_i and eigenvalues λ_i. Using *Mathematica*'s **DSolve** operator we can explicitly analyse possible solutions to equation (13)

```
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}]]
```

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

$$\underset{\sim}{\alpha}(t) = \sum_{i=1}^{N} k_i \, e^{\lambda_i t} \, \underset{\sim}{w_i} \tag{15}$$

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

$$\begin{aligned} \underset{\sim}{\alpha}'(t) &= \sum_{i=1}^{N} k_i \, e^{\lambda_i t} \, \lambda_i \, \underset{\sim}{w_i} = \\ &\sum_{i=1}^{N} k_i \, e^{\lambda_i t} \, B \cdot \underset{\sim}{w_i} = B \cdot \sum_{i=1}^{N} k_i \, e^{\lambda_i t} \, \underset{\sim}{w_i} = B \cdot \underset{\sim}{\alpha}(t) \end{aligned} \tag{16}$$

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

$$\underset{\sim}{U}(\underset{\sim}{y}, t) = \Phi \cdot \underset{\sim}{\alpha}(t) = \Phi \cdot \sum \underset{\sim}{w_i}_{i=1}^{N} k_i \, e^{\lambda_i t} = \Phi \cdot W \cdot \underset{\sim}{k} \cdot e^{\lambda t} \tag{17}$$

It can be seen from the above equation that $\underset{\sim}{U}(\underset{\sim}{y}, t)$ and hence $V(S, t)$ can be calculated once $\underset{\sim}{k}$ 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

$$\underset{\sim}{k} = W^{-1} \cdot \Phi^{-1} \cdot \underset{\sim}{U}(\underset{\sim}{y}, T) \Big/ e^{\lambda T} \tag{18}$$

While the above result is sufficient for an explicit solution that will cover european options it will not be capable of calculating path dependent options which 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 which have to be solved by the **RootSum** function. For example :

```
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}]]
```

For the purpose of calculating american style options we will consider times $t_n = T - n \Delta t$ and define $\underset{\sim}{U}(\underset{\sim}{y}, t_n) = \underset{\sim}{U}^n$, $\underset{\sim}{\alpha}(t_n) = \underset{\sim}{\alpha}^n$. Now we utilise backward difference time integration schemes which can be the explicit first order (BD1) :

$$\underset{\sim}{\alpha}^n = \underset{\sim}{\alpha}^{n-1} - \Delta t \, B \cdot \underset{\sim}{\alpha}^{n-1} = (I_N - \Delta t \, B) \cdot \underset{\sim}{\alpha}^{n-1} \tag{19}$$

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

$$\begin{aligned} R_1 &= -\Delta t \; B \cdot \underset{\sim}{\alpha}^{n-1} \\ R_2 &= -\Delta t \; B \cdot (\underset{\sim}{\alpha}^{n-1} + R_1 / 2) \\ \underset{\sim}{\alpha}^n &= \underset{\sim}{\alpha}^{n-1} + (R_1 + R_2) / 2 \end{aligned} \tag{20}$$

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

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

or the implicit backward difference time integration scheme (IDθ)

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

■ 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 $S \to 0$ and $S \to \infty$.

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

Using the equations above 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 code we shall restrict ourselves throughout the rest of the paper to put options only. The time domain shall be subdivided into M sub–intervals, so that $\Delta t = T/M$, so that for any prior time $t_n = T - n\,\Delta t$, n lies in the range $0 \le n \le M$, with n=0 corresponding to the maturity date T. The spatial discretisation will be for N collocation points y_i , $1 \le i \le N$. The *Mathematica* code for the Black–Scholes european options and their deltas according to equations (4) and (5) can be readily implemented:

```
Off[General::spell1];
ClearAll[NormPDF, NormCDF, d1, d2, BSCall, BSPut,
  putDelta, callDelta, Δy, y, t, Δt, ϕ, Φ, 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 + σ^2 / 2) t) / (σ Sqrt[t]);
d2[σ_, r_, q_, x_, y_, t_] :=
  (Log[x / y] + (r - q - σ^2 / 2) t) / (σ Sqrt[t]);
NormCDF[z_] := (1 + Erf[z / Sqrt[2]]) / 2;
NormPDF[z_] := Exp[-z^2 / 2] / Sqrt[2 π];
BSCall[σ_, r_, q_, S_, K_, T_] :=
  S E^-q T NormCDF[d1[σ, r, q, S, K, T]] -
   K E^-r T NormCDF[d2[σ, r, q, S, K, T]];
BSPut[σ_, r_, q_, S_, K_, T_] :=
  K E^-r T NormCDF[-d2[σ, r, q, S, K, T]] -
   S E^-q T NormCDF[-d1[σ, r, q, S, K, T]];
putDelta[σ_, r_, q_, S_, K_, T_] :=
  -E^-q T NormCDF[-d1[σ, r, q, S, K, T]];
callDelta[σ_, r_, q_, S_, K_, T_] :=
  E^-q T 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 will specify some market parameters. The risk free interest rate is 1%, the dividend yield is 0, the volatility is 30%, the strike is \$100, Smin of 1 gives a ymin of 0, Log(Smax) = ymax = 6. Let there be N = Npts = 101 collocation points and subdivide the time to maturity T = 1 year into M = 100 subintervals. We have chosen 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 will 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.

```
(* 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 will be to compare the exact results for european options from the above Black–Scholes formulae with the analytical results of equation (17), explicit formulae of (19)–(21) and implicit time integration scheme of equation (22). Note that $\phi(y_i, y_j)$ is the RBF function as described earlier in eqns (1), (10) and (11). With Smin and Smax the smallest and largest values of the stock price then the spatial increment in the y domain with Npts collocation points is Δy = (Log[Smax] − Log[Smin]) / (Npts − 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 equation (11) as:

```
Δy = (Log[Smax] - Log[Smin]) / (Npts - 1); Δt = T / M;
ϕ[y1_, y2_] := Sqrt[(y1 - y2)^2 + 16 Δy^2] ;
Φ = 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 $\{\lambda_i, w_j\}$. The initial condition of (7) is used to determine $U(y, T)$ = UT which is substituted into (18) to calculate $\underset{\sim}{k}$ = kvec and this is further substituted into (15) to yield $\underset{\sim}{\alpha}(t)$.

```
Φinv = Inverse[Φ];
B =
  r IdentityMatrix[Npts] - σ^2 Φinv.Φyy / 2 - (r - q - σ^2 / 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 formulae for $U(y, t)$, $0 \le t \le T$; $0 \le y \le$ Log[Smax] can now be specified by (17). Finally recalling that y = Log[S], t = time elapsed and τ = 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 $\phi(y, y_j)$.

```
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[Smax] :=
  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]) / S^2 ;
```

□ 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 $U(y, T)$ = UT determined by (7) and step backwards through time considering earlier times $t_n = T - n\,\Delta t$ while defining $\underset{\sim}{U}(\underset{\sim}{y}, t_n) = \underset{\sim}{U}^n$, $\underset{\sim}{\alpha}(t_n) = \underset{\sim}{\alpha}^n$. There is one further modification that needs to be undertaken 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 = Smin and hence when $y = y_1$. This necessitates the rewriting of $U(y_1, t) = U^1(t) = U$[[1]].

```
ClearAll[U0, α0, n, α1, α2, α4, αi, U1, U2, U4, Ui, Un, V1explicit,
  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}]];
```

Initial and boundary conditions according to equations (7) and (23) are specified:

```
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 formulae for $U(y, t)$, $0 \le t \le T$; $0 \le y \le$ Log[Smax] can now be specified according to the LHS of equation (17). Also we can get the explicit and implicit approximation formulae for $V(S, t)$ by using the updated values for $\alpha(t_n) = \alpha^n$. The hedging delta formulae are given in terms of U(Log[S], t) by simply taking the derivatives of the RBFs $\phi(y, y_j)$.

```
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[ϕ[y, yj], {yj, Log[Smin], Log[Smax], Δy}].αi[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[ϕ[Log[S], yj], {yj, Log[Smin], Log[Smax], Δy}].α1[M];
V2explicit[S_] :=
  Table[ϕ[Log[S], yj], {yj, Log[Smin], Log[Smax], Δy}].α2[M];
V4explicit[S_] :=
  Table[ϕ[Log[S], yj], {yj, Log[Smin], Log[Smax], Δy}].α4[M];
Vimplicit[S_] :=
  Table[ϕ[Log[S], yj], {yj, Log[Smin], Log[Smax], Δy}].αi[M];
Delta1Explicit[S_] :=
  Table[Evaluate[D[ϕ[y, yj], y]] /. {y → Log[S]},
     {yj, Log[Smin], Log[Smax], Δy}].α1[M] / S;
Delta2Explicit[S_] := Table[Evaluate[D[ϕ[y, yj], y]] /.
      {y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}].α2[M] / S;
Delta4Explicit[S_] := Table[Evaluate[D[ϕ[y, yj], y]] /.
      {y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}].α4[M] / S;
DeltaImplicit[S_] := Table[Evaluate[D[ϕ[y, yj], y]] /.
      {y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}].αi[M] / S;
Gamma1Explicit[S_] := Table[Evaluate[D[ϕ[y, yj], {y, 2}]] /.
      {y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}].α1[M] / S^2;
Gamma2Explicit[S_] := Table[Evaluate[D[ϕ[y, yj], {y, 2}]] /.
      {y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}].α2[M] / S^2;
Gamma4Explicit[S_] := Table[Evaluate[D[ϕ[y, yj], {y, 2}]] /.
      {y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}].α4[M] / S^2;
GammaImplicit[S_] := Table[Evaluate[D[ϕ[y, yj], {y, 2}]] /.
      {y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}].αi[M] / S^2;
```

```
TableForm[Table[
  (PaddedForm[N[#1], {5, 4}] &) /@ {S, BSPut[σ, r, q, S, K, T],
    Vanalytical[S], V1explicit[S],
    V2explicit[S], V4explicit[S], Vimplicit[S]},
          {S, 80, 200, 10}], TableHeadings →
  {None, {  " S", " Exact", "Analytic",
    "  BD1", "  BD2", "  BD4", "  IDθ"}}]
```

S	Exact	Analytic	BD1	BD2	BD4	IDθ
80.0000	16.2430	16.2330	16.2300	16.2320	16.2330	16.2330
90.0000	11.0040	10.9930	11.0000	10.9960	10.9930	10.9930
100.0000	7.2179	7.2071	7.2200	7.2136	7.2071	7.2071
110.0000	4.6135	4.6041	4.6173	4.6107	4.6040	4.6041
120.0000	2.8899	2.8822	2.8926	2.8874	2.8821	2.8821
130.0000	1.7825	1.7766	1.7832	1.7799	1.7766	1.7766
140.0000	1.0870	1.0828	1.0859	1.0843	1.0828	1.0828
150.0000	0.6575	0.6548	0.6553	0.6550	0.6548	0.6548
160.0000	0.3955	0.3943	0.3931	0.3937	0.3943	0.3943
170.0000	0.2371	0.2375	0.2355	0.2365	0.2375	0.2375
180.0000	0.1419	0.1445	0.1422	0.1434	0.1446	0.1446
190.0000	0.0849	0.0907	0.0884	0.0896	0.0908	0.0909
200.0000	0.0509	0.0615	0.0593	0.0605	0.0617	0.0617

Table 1. European Put Option Prices determined by RBF Analytic, Explicit and Implicit Methods

```
TableForm[Table[
  (PaddedForm[N[#1], {5, 4}] &) /@ {S, putDelta[σ, r, q, S, K, T],
    DeltaAnalytical[S], Delta1Explicit[S],
    Delta2Explicit[S], Delta4Explicit[S], DeltaImplicit[S]},
          {S, 80, 200, 10}], TableHeadings →
  {None, {  " S", "Exact Δ", "Analytic Δ",
    "  BD1 Δ", "  BD2 Δ", "  BD4 Δ", "  IDθ Δ"}}]
```

S	Exact Δ	Analytic Δ	BD1 Δ	BD2 Δ	BD4 Δ	IDθ Δ
80.0000	−0.6028	−0.6030	−0.6018	−0.6024	−0.6030	−0.6030
90.0000	−0.4474	−0.4475	−0.4467	−0.4471	−0.4475	−0.4475
100.0000	−0.3144	−0.3143	−0.3141	−0.3142	−0.3143	−0.3143
110.0000	−0.2116	−0.2114	−0.2116	−0.2115	−0.2114	−0.2114
120.0000	−0.1376	−0.1374	−0.1378	−0.1376	−0.1374	−0.1374
130.0000	−0.0873	−0.0871	−0.0875	−0.0873	−0.0871	−0.0871
140.0000	−0.0543	−0.0541	−0.0544	−0.0543	−0.0541	−0.0541
150.0000	−0.0333	−0.0331	−0.0333	−0.0332	−0.0331	−0.0331
160.0000	−0.0202	−0.0200	−0.0202	−0.0201	−0.0200	−0.0200
170.0000	−0.0122	−0.0120	−0.0120	−0.0120	−0.0120	−0.0120
180.0000	−0.0073	−0.0070	−0.0070	−0.0070	−0.0070	−0.0070
190.0000	−0.0044	−0.0040	−0.0040	−0.0040	−0.0040	−0.0040
200.0000	−0.0026	−0.0020	−0.0020	−0.0020	−0.0020	−0.0020

Table 2. European Put Option Deltas determined by RBF Analytic, Explicit and Implicit Methods

```
TableForm[Table[
  (PaddedForm[N[#1], {5, 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"}}]
```

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 BS Theory, RBF Analytic, Explicit and Implicit Methods

■ American Options Numerical Specification

□ Analytical Results

American options will satisfy the same boundary conditions of equation (23) as european options as $S \to 0$ and $S \to \infty$ 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 $\mathbb{E}(e^{-r\,\Delta t}\ V(S_{t+\Delta t},\ t+\Delta t) \mid \mathcal{F}_t) = V_E(S, t)$, which is the same as the european option price and the exercise price $K - S_t$. That is

$$V_A(S_t, t) \ = \ \text{Max}[V_E(S_t, t),\ K - S_t\,] \ = \ \text{Max}[V_E(S_t, t),\ U(\text{Log}(S_t), T)] \qquad (24)$$

While we can implement the above formula directly into the RBF formalism simply by taking the european option code in the section above and comparing it with the exercise prices, this will not yield satisfactory results because american options are path dependent and as they can be exercised at any time, have not one final boundary at maturity but an infinite number of boundaries which 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.

```
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 $U(y, T) = \text{UT}$ determined by (7) and step backwards through time considering earlier times $t_n = T - n\,\Delta t$ while defining $\underset{\sim}{U}\left(\underset{\sim}{y}, t_n\right) = \underset{\sim}{U}^n$, $\underset{\sim}{\alpha}(t_n) = \underset{\sim}{\alpha}^n$. Again observe that $T - t = T - t_n = n\,\Delta t$ in the exponent. As with the european options there are further modifications that needs 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 ith european option value $\underset{\sim}{U}^n[[i]]$ and then compare it with the exercise value $K - S^i = K - e^{y_i} = K - \text{Smin}\, e^{(i-1)\,\Delta y}$

```
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}]; α4a[n] = Φinv.Un , {n, 1, M}]];
Module[{Un, B2 = (IdentityMatrix[Npts] - Θ Δt B), B1},
  B1 = B2 + Δt B; αia[0] = α0;
  Do[αia[n] = Inverse[B1].B2.αia[n - 1]; Un = Φ.αia[n] ;
   Do[Un[[i]] = Max[Un[[i]], K - Smin Exp[(i - 1) Δy]],
    {i, 1, Npts}]; αia[n] = Φinv.Un , {n, 1, M}]];
```

Initial and boundary conditions according to equations (7) and (23) are specified:

```
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 ≥ Log[Smax] := 0;
U2Amer[y_, n_Integer] /; y ≥ Log[Smax] := 0;
U4Amer[y_, n_Integer] /; y ≥ Log[Smax] := 0;
UiAmer[y_, n_Integer] /; y ≥ Log[Smax] := 0;
```

Again the general formulae for $U(y, t)$, $0 \le t \le T$; $0 \le y \le \text{Log[Smax]}$ can now be specified according to the LHS of equation (17). Also we can get the explicit and implicit approximation formulae for $V(S, t)$ by using the updated values for $\underset{\sim}{\alpha}(t_n) = \underset{\sim}{\alpha}^n$. The

hedging delta formulae are given in terms of $U(\text{Log}[S], t)$ by simply taking the derivatives of the RBFs $\phi(y, y_j)$.

```
U1Amer[y_, n_Integer] /; y < Log[Smax] :=
  Table[ϕ[y, yj], {yj, Log[Smin], Log[Smax], Δy}].α1a[n];
U2Amer[y_, n_Integer] /; y < Log[Smax] :=
  Table[ϕ[y, yj], {yj, Log[Smin], Log[Smax], Δy}].α2a[n];
U4Amer[y_, n_Integer] /; y < Log[Smax] :=
  Table[ϕ[y, yj], {yj, Log[Smin], Log[Smax], Δy}].α4a[n];
UiAmer[y_, n_Integer] /; y < Log[Smax] :=
  Table[ϕ[y, yj], {yj, Log[Smin], Log[Smax], Δy}].αia[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[ϕ[Log[S], yj],
     {yj, Log[Smin], Log[Smax], Δy}].α1a[M], K - S, 0];
V2AmerExplicit[S_] := Max[Table[ϕ[Log[S], yj],
     {yj, Log[Smin], Log[Smax], Δy}].α2a[M], K - S, 0];
V4AmerExplicit[S_] := Max[Table[ϕ[Log[S], yj],
     {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}].α1a[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] / S^2;
Gamma2AmerExplicit[S_] := Table[Evaluate[D[ϕ[y, yj], {y, 2}]] /.
      {y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}].α2a[M] / S^2;
Gamma4AmerExplicit[S_] := Table[Evaluate[D[ϕ[y, yj], {y, 2}]] /.
      {y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}].α4a[M] / S^2;
GammaAmerImplicit[S_] := Table[Evaluate[D[ϕ[y, yj], {y, 2}]] /.
      {y → Log[S]}, {yj, Log[Smin], Log[Smax], Δy}].αia[M] / S^2;
```

For the sake of calculating actual prices, we will 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. Also the time taken is only incrementally longer than for the european case. In order to compare the RBF values for the american put with other schemes we utilise the Equal Jumps Additive Binomial model described in Clewlow and Strickland [1998], Ch 2 and implemented here in *Mathematica*.

```
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 will use the above function with a large number $N = 500$ of steps to calculate highly accurate american put prices at the same stock values as for the european options and by varying these by the incremental amount of $\Delta S = \$0.2$, we can further get values for the american put deltas.

```
binomialTree =
 Table[EqualJumpsAdditiveBinomialAmericanPut[S, K, r, 0, σ, T, 500],
  {S, 80, 200, 10}]
```

```
{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}
```

```
binomialTree2 =
 Table[EqualJumpsAdditiveBinomialAmericanPut[S, K, r, 0, σ, T, 500],
  {S, 80.2, 200.2, 10.0}]
```

```
{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}
```

```
binDelta = (binomialTree2 - binomialTree) / 0.2
```

$$\{-0.856572, -0.575645, -0.379222, -0.248721, -0.161147, -0.100498, -0.0598567, -0.034283, -0.0217178, -0.0136589, -0.00839964, -0.00504329, -0.00295229\}$$

In the next two tables 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

```
TableForm[Table[(PaddedForm[N[#1], {5, 4}] &) /@
       {S, binomialTree[[S / 10 - 7]],
    VAmerAnalytical[S], V1AmerExplicit[S],
    V2AmerExplicit[S], V4AmerExplicit[S], ViAmerImplicit[S]},
          {S, 80, 200, 10}], TableHeadings → {None, {"  S",
    "Binomial", "Analytic", "  BD1", "  BD2", "  BD4", "  IDθ"}}]
```

S	Binomial	Analytic	BD1	BD2	BD4	IDθ
80.0000	20.2670	20.0000	20.2410	20.2400	20.2360	20.2350
90.0000	13.1200	10.9930	13.1320	13.1180	13.1040	13.1050
100.0000	8.3359	7.2071	8.3540	8.3375	8.3224	8.3232
110.0000	5.2125	4.6041	5.2226	5.2080	5.1953	5.1961
120.0000	3.2094	2.8822	3.2173	3.2062	3.1970	3.1978
130.0000	1.9543	1.7766	1.9585	1.9511	1.9453	1.9459
140.0000	1.1806	1.0828	1.1811	1.1770	1.1739	1.1743
150.0000	0.7065	0.6548	0.7072	0.7054	0.7042	0.7045
160.0000	0.4232	0.3943	0.4216	0.4211	0.4211	0.4214
170.0000	0.2520	0.2375	0.2513	0.2516	0.2524	0.2525
180.0000	0.1497	0.1445	0.1509	0.1517	0.1527	0.1528
190.0000	0.0891	0.0907	0.0932	0.0942	0.0953	0.0954
200.0000	0.0532	0.0615	0.0619	0.0629	0.0641	0.0642

Table 4. American Put Option Prices determined by Binomial, RBF Analytic, Explicit and Implicit Methods

```
TableForm[Table[
  (PaddedForm[N[#1], {5, 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"}}]
```

S	Binomial Δ	Analytic Δ	BD1 Δ	BD2 Δ	BD4 Δ	IDθ Δ
80.0000	−0.8566	−0.6030	−0.8556	−0.8552	−0.8571	−0.8576
90.0000	−0.5756	−0.4475	−0.5799	−0.5801	−0.5806	−0.5807
100.0000	−0.3792	−0.3143	−0.3849	−0.3846	−0.3845	−0.3846
110.0000	−0.2487	−0.2114	−0.2491	−0.2487	−0.2483	−0.2484
120.0000	−0.1611	−0.1374	−0.1579	−0.1574	−0.1570	−0.1571
130.0000	−0.1005	−0.0871	−0.0983	−0.0979	−0.0976	−0.0976
140.0000	−0.0599	−0.0541	−0.0602	−0.0600	−0.0598	−0.0598
150.0000	−0.0343	−0.0331	−0.0364	−0.0363	−0.0361	−0.0361
160.0000	−0.0217	−0.0200	−0.0219	−0.0217	−0.0216	−0.0217
170.0000	−0.0137	−0.0120	−0.0130	−0.0129	−0.0129	−0.0129
180.0000	−0.0084	−0.0070	−0.0076	−0.0075	−0.0075	−0.0075
190.0000	−0.0050	−0.0040	−0.0043	−0.0042	−0.0042	−0.0042
200.0000	−0.0030	−0.0020	−0.0022	−0.0022	−0.0022	−0.0022

Table 5. American Put Option Deltas determined by Binomial, RBF Analytic, Explicit and Implicit Methods

A final check on the accuracy of the results for american options can be obtained by loading in the powerful financial software *UnRisk*. It uses compiled backward recursive path integration over a large grid to obtain highly accurate estimates of path dependent

and american option values. In the following two tables 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 behaviour of the moving boundary.

```
Needs["UnRisk`UnRiskFrontEnd`"];

MyEquities =
  Table[MakeEquity[S, EquityYield → q], {S, 80, 200, 10}];

MyAmericanCalls = Table[
   MakeVanillaEquityOption[MyEquities[[i]], K, {2007, 4, 23},
    OptionType → "Put", ExerciseType → "American"], {i, 13}];

MyYieldCurve = MakeYieldCurve[r];
MyVolCurve = MakeVolatilityCurve[σ];

MyAmerValues = Table[
  Valuate[MyAmericanCalls[[i]], {2006, 4, 23}, {2006, 4, 24},
   MyVolCurve, MyYieldCurve, CalculateVega → True], {i, 13}]
```

$$\begin{pmatrix}
20.1072 & -0.893656 & 0.0512762 & -0.0176433 & 0.097303 & 0.0445105 & 0.0425975 \\
12.984 & -0.578389 & 0.0229865 & -0.00509505 & 0.311269 & 0.00415878 & 0.00979675 \\
8.23795 & -0.382164 & 0.0163314 & -0.0074411 & 0.357886 & 0.000836575 & 0.000328313 \\
5.13869 & -0.246501 & 0.0110849 & -0.00771156 & 0.334157 & 0.00326843 & -0.00441884 \\
3.1598 & -0.155636 & 0.00730538 & -0.00699543 & 0.279741 & 0.00687307 & -0.00606407 \\
1.92091 & -0.0965184 & 0.00468408 & -0.00580065 & 0.218499 & 0.00953948 & -0.00598893 \\
1.15785 & -0.0590177 & 0.00293668 & -0.00451725 & 0.162526 & 0.0109103 & -0.00512659 \\
0.69369 & -0.0357036 & 0.00180909 & -0.00336124 & 0.11668 & 0.0108562 & -0.00403852 \\
0.413969 & -0.0214334 & 0.00109974 & -0.00241703 & 0.0815688 & 0.00969907 & -0.00301274 \\
0.246522 & -0.0127993 & 0.000662028 & -0.00169369 & 0.0558414 & 0.00811411 & -0.00216091 \\
0.146692 & -0.00761841 & 0.000395822 & -0.00116367 & 0.0376375 & 0.00650452 & -0.0015068 \\
0.0872869 & -0.00452574 & 0.000235544 & -0.000787377 & 0.0251196 & 0.00512845 & -0.00103112 \\
0.0520401 & -0.00268903 & 0.000139854 & -0.000526839 & 0.0165927 & 0.00384877 & -0.000693512
\end{pmatrix}$$

```
TableForm[Table[(PaddedForm[N[#1], {5, 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", "  IDθ"}}]
```

S	Quadrature	Analytic	BD1	BD2	BD4	IDθ
80.0000	20.1070	20.0000	20.2410	20.2400	20.2360	20.2350
90.0000	12.9840	10.9930	13.1320	13.1180	13.1040	13.1050
100.0000	8.2379	7.2071	8.3540	8.3375	8.3224	8.3232
110.0000	5.1387	4.6041	5.2226	5.2080	5.1953	5.1961
120.0000	3.1598	2.8822	3.2173	3.2062	3.1970	3.1978
130.0000	1.9209	1.7766	1.9585	1.9511	1.9453	1.9459
140.0000	1.1578	1.0828	1.1811	1.1770	1.1739	1.1743
150.0000	0.6937	0.6548	0.7072	0.7054	0.7042	0.7045
160.0000	0.4140	0.3943	0.4216	0.4211	0.4211	0.4214
170.0000	0.2465	0.2375	0.2513	0.2516	0.2524	0.2525
180.0000	0.1467	0.1445	0.1509	0.1517	0.1527	0.1528
190.0000	0.0873	0.0907	0.0932	0.0942	0.0953	0.0954
200.0000	0.0520	0.0615	0.0619	0.0629	0.0641	0.0642

Table 6. American Put Option Prices determined by Quadrature, RBF Analytic, Explicit and Implicit Methods

```
TableForm[Table[(PaddedForm[N[#1], {5, 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"}}]
```

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 Quadrature, RBF Analytic, Explicit and Implicit Methods

```
TableForm[Table[(PaddedForm[N[#1], {5, 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"}}]
```

S	Quadrature Γ	Analytic Γ	BD1 Γ	BD2 Γ	BD4 Γ	IDθ 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 Quadrature, RBF Analytic, Explicit and Implicit 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) which describes the point at time t where for values of $S_t > \mathrm{FB}_t$, 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 $S_t \le \mathrm{FB}_t$, 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

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

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 will define these $F(y)$ functions below :

```
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;
On[General::spell1];

     Off[FindRoot::"lstol"]
```

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

```
y /. FindRoot[Fi[y, M], {y, Log[K]}, AccuracyGoal → 10,
   WorkingPrecision → 20, MaxIterations → 300]
```

4.3500020350959964723

At time $t_n = T$ when $n = 0$, FB(T) = K and hence $y(T)$ = Log(K). For successive values of n, we move further back in time to the present and Fi[y, n] determines those values y_n such that Log(y_n) = 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.

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

ListPlot[Exp[Transpose[exerciseBdryPts][[2]]],
 PlotJoined → True, PlotRange → {60, 100},
 AxesLabel → {"Time remaining", "Free Boundary"},
 PlotLabel -> "Free Boundary determined by RBF"]
```

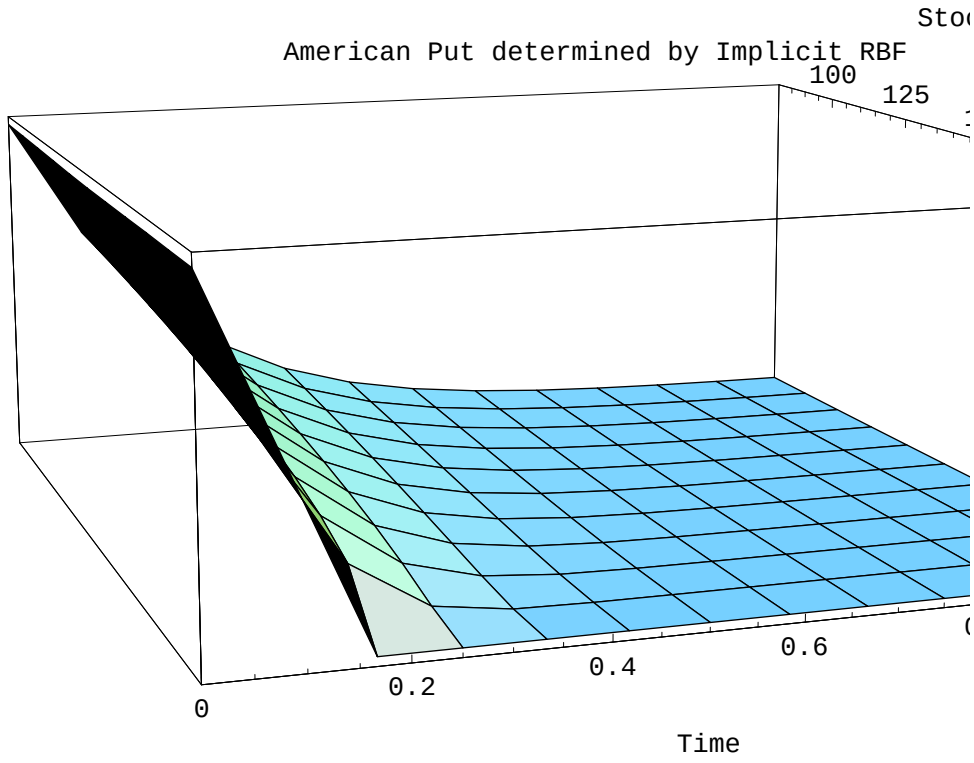
Figure 1. Moving Free Boundary determined by RBF

□ 3D Graphs of American Options

The code above 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 below:

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

```
ListPlot3D[ViamerOut, MeshRange → {{80, 200}, {0, 1}},
 AxesLabel → {"Stock", "Time", "American Put"},
 ViewPoint -> {3.078, -1.063, 0.655},
 PlotLabel -> "American Put determined by Implicit RBF"]
```




```
- SurfaceGraphics -
```

Figure 2. 3D Plot of American Put determined by Implicit RBF

■ Conclusion

In this paper we have investigated the use of Radial Basis Functions as a means of solving parabolic PDEs associated with the solution of european and american style financial options. Unlike the usual approach which typically involves finite difference schemes we have utilised a spatial collocation approach which has resulted in analytical, explicit asnd 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, yielding 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 the two figures above. And because the RBFs are themselves readily differentiated, then the hedging parameters are also immediately calculable. Lastly there are a range of parameters such as the shape parameter κ and the number of time steps M and spatial dimension N that can be adjusted so as to further improve results.

■ References

F.Black and M.Scholes. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81, pg.637, 1973.

L. Clewlow and C. Strickland. Implementing Derivatives Models. Wiley Sereies in Financial Engineering. 1998.

G. Fasshauer, A.Q.M. Khaliq, and D.A. Voss. Using Mesh free approximation for Multi Asset American options. *Journal of Chinese Institute of Engineers* (JCIE) Vol. 27(4), pgs 563−571, 2004.

R.L.Hardy. Multiquadratic equations of topographyand other irregular surfaces. *Journal of Geophysical Research*,76(8):1905−1915, 1971.

R.L.Hardy. Theory and Applications of the multiquadric−biharmonic method. *Computers and Mathematics with Applications*, 19(8/9):163−208, 1990.

Y.C.Hon and X.Z.Mao. A radial basis function method for solving options pricing models. *Financial Engineering*. Vol. 8, pgs 31−49, 1999.

M.A.Goldberg and C.S.Chen. On a method of Atkinson for evaluating domain integrals in the boundary element method. *Journal of Applioed Mathematics and Computation*, 18, pg.9, 1996.

E. 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):147−161, 1990.

E. 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):147−161, 1990.

M.J.Powell. The Theory of Radial Basis Function Approximation. *Advances in Numerical Analysis III,* Clarendon Press*,* pg.105, 1992*.*