Notes on the BENCHOP implementation for the adaptive radial basis function with the backward Crank Nicolson method in time

Juxi Li(jl164@le.ac.uk) and
Jeremy Levesley(jl1@le.ac.uk)

March 16, 2015

Abstract

This text describes the adaptive radial basis function with backward Crank Nicolson in time method and its implementation for the BENCHOP-project.

1 Spatial discretization and time-stepping scheme

The one dimensional non-dividend payment Black-Scholes equation is

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0, \tag{1}$$

where S is the underlying asset value which follows a random walk in continuous time, σ is a constant volatility, r is a known and constant short-term interest rate and V is the price of option. This equation can be rearranged as follows:

$$\frac{\partial u}{\partial t} + \mathcal{L}u = 0, \quad t \in [0, T]. \tag{2}$$

$\mathcal{L}$ is our partial differential operator in one or two space dimensions. We will describe our adaptive radial basis function in one dimension problem and use the method of lines to transform the partial differential equation (PDE) to a system of ordinary differential equations (ODEs).

For a given a set of collocation points $x_1, \ldots, x_{N(t)} \in \mathbb{R}$ and RBF ϕ, we construct an approximation to u

$$u_N(x,t) = \sum_{i=1}^{N(t)} \lambda(t)\phi(\|x - x_i(t)\|).$$

The N and x_i change because we are using adaptive techniques. Substituting this into the PDE above we obtain a system of ODEs which can be written in terms of the $N \times N$ matrices $A = \phi(\mid x - x_j(t) \mid)_{1 \le j \le N(t)}$, $A_x = \phi'(\mid x - x_j(t) \mid)_{1 \le j \le N(t)}$ and $A_{xx} = \phi''(\mid x - x_j(t) \mid)_{1 \le j \le N(t)}$. Because of the adaptive method we use, thus the matrices of A, A_x and A_{xx} are depending on time t. However, for notational simplicity in below, we should write $A = A^t$, $A_x = A_x^t$ and $A_{xx} = A_{xx}^t$. Our ODE system at each time step is

$$A\lambda^t = -\left[\frac{1}{2}\sigma^2 A_{xx}\lambda + \left(r - \frac{1}{2}\sigma^2\right) A_x\lambda - rA\lambda\right]. \tag{3}$$

We cannot in principle guarantee the invertibility of the linear systems arising in the method, but we have never obtained a system with zero determinant, even though small eigenvalues arise if we choose small smooth radial basis functions such as Gaussians or multiquadrics for ϕ. Premulitiplication by A gives

$$\lambda^t = -\left[\frac{1}{2}\sigma^2 A^{-1}A_{xx} + \left(r - \frac{1}{2}\sigma^2\right) A^{-1}A_x - rI\right]\lambda. \tag{4}$$

We can rewrite this as

$$\begin{aligned} \lambda^t &= P\lambda, \\ \text{where } P &= -\tfrac{1}{2}\sigma^2 A^{-1}A_{xx} - (r - \tfrac{1}{2}\sigma^2)A^{-1}A_x + rI. \end{aligned} \tag{5}$$

Now, any backward time integration schemes can be applied to find the unknown coefficient vector λ. We will use the backward Crank Nicolson scheme for our time integration:

$$\frac{\lambda^t - \lambda^{t-\Delta t}}{\Delta t} = \frac{1}{2}P(\lambda^t + \lambda^{t-\Delta t}), \quad 0 < t \le T. \tag{6}$$

If we rearrange the terms above we will have

$$(I - \frac{1}{2}\Delta t P)\lambda^t = \left(I + \frac{1}{2}\Delta t P\right)\lambda^{t-\Delta t}. \tag{7}$$

By defining the follow new matrices L and R:

$$\begin{aligned} L &= (I - \tfrac{1}{2}\Delta t P), \\ R &= (I + \tfrac{1}{2}\Delta t P), \end{aligned} \tag{8}$$

then we have a simple linear system of equations:

$$\begin{aligned} L\lambda^t &= R\lambda^{t-\Delta t}, \\ \lambda^t &= L^{-1}R\lambda^{t-\Delta t}, \end{aligned} \tag{9}$$

$0 < t \le T$ with increment of time Δt. We note that good approximation of the initial condition is really important, where the function we are approximating has a derivative singularity. We use adaptive approximation to cluster nodes near to points where the function is least smooth. As time evolves, the solution smooths and we need less points to gain the same approximation quality.

2 Adaptive algorithm

In the one dimensional problem, the approximation of U is obtained by global radial basis function, using the multiquadric basis function, which we evaluate by comparison with a local approximation using piecewise cubic spline interpolation at the 8 points nearest to x_i, $2 \leq i \leq N-1$. A good approximation is defined by an error threshold in the residual which is based on predefined thresholds err_{ref} and err_{crs}, and residual is $r_{err} = \mid u_N(x_i) - \tilde{u}_{\mathcal{N}_{x_i}}(x_i) \mid$, where $\mathcal{N}_{x_i}$ denotes the 8 nearest points, not including x_i, and $\tilde{u}_{\mathcal{N}_{x_i}}$ is the local approximation to u_N based on these points.

Let err_{ref} and err_{crs} be thresholds for refining points and coarsening points respectively. The refinement and coarsening strategy can be summarized as follows:

For a given set of interpolation points $x_1, \ldots, x_N$, we will not reconstruct two end points (first and last interpolation points)

1. - For each x_i, determine $\mathcal{N}_{x_i}$;
   - Compute $\tilde{u}_{\mathcal{N}_{x_i}}$, the local interpolant using cubic splines;
   - Compute $r_{err} = \mid u_N(x_i) - \tilde{u}_{\mathcal{N}_{x_i}}(x_i) \mid$.
2. Refine if $r_{\text{err}} > \text{err}_{ref}$.
3. Coarsen if $r_{\text{err}} < \text{err}_{crs}$.

3 Benchmark problem 1

- We truncate the computational domain, $log(S_{\min}) \leq log(S) \leq log(S_{\max})$. The initial condition and boundary condition is

$$\begin{aligned} u(x,T) &= \max\{e^x - K, 0\}, \\ u(\log(S_{\min}), t) &= \alpha(t), \quad 0 \leq t \leq T, \\ u(\log(S_{\max}), t) &= \beta(t), \quad 0 \leq t \leq T. \end{aligned} \tag{10}$$

- The time stepping method used is the backward Crank Nicolson scheme.

3.1 Problem 1

- The computation of Δ at time 0 is calculated by $\sum_{j=1}^{N} \lambda_j^0 \phi'(\mid x - x_j(0) \mid)$ where $\phi'(\mid x - x_j(0) \mid)$ is partial derivative of $\phi(\mid x - x_j(0) \mid)$ and 0 means at time 0.
- The computation of Γ at time 0 is calculated by $\sum_{j=1}^{N} \lambda_j^0 \phi''(\mid x - x_j(0) \mid)$ where $\phi''(\mid x - x_j(0) \mid)$ is partial derivative of $\phi(\mid x - x_j(0) \mid)$ and 0 means at time 0.
- The computation of ν at time 0 is calculated by $\frac{u(log(S_0), 1.001\sigma) - u(log(S_0), \sigma)}{0.001 * \sigma}$ where u is an interpolation to the computed solution.

Problem	S_{min}	S_{max}	$N_u = 140$	M = 160	ε
1a) SP	30	2K	160	80	3.5e-5
1b) SP	30	2K	160	100	1.8e-5
1c) SP	30	2K	40	100	2.6e-5
1a) CP	30	2K	50	80	9.1e-5
1b) CP	30	2K	40	80	1.9e-5
1c) CP	30	2K	40	150	9.9e-5
1a) Δ SP	30	2K	160	80	6.0e-5
1a) ΓSP	30	2K	160	110	8.6e-5
1a) ν SP	30	2K	160	80	8.6e-5
1a) Δ CP	30	2K	60	80	2.9e-5
1a) ΓCP	30	2K	80	100	6.2e-5
1a) ν CP	30	2K	160	80	1.1e-5

Table 1: Here SP and CP mean Standard Parameter and Challenging Parameter respectively, N_u is starting uniform nodes and ε is max error in all three evaluate points