Wavelet Optimized Finite-Difference Approach to Solve Jump-Diffusion type Partial Differential Equation for Option Pricing

Ruppa K. Thulasiram[*]
Parimala Thulasiraman
Mohammad R. Rahman[†]

Department of Computer Science
University of Manitoba
Winnipeg, MB R3T 2N2
CANADA

Abstract

The sine and cosine functions used as the bases in Fourier analysis are very smooth (infinitely differentiable) and very broad (nonzero almost everywhere on the real line) and hence they are not effective for representing functions that change abruptly (jumps) or have highly localized support (diffusive). In response to this shortcoming, there has been intense interest in recent years in a new type of basis functions called wavelets. A given wavelet basis is generated from a single function, called a mother wavelet or scaling function, by dilation and translation. By replicating the mother wavelet at many different scales, it is possible to mimic the behavior of any function; this property of wavelets is called multiresolution. Wavelet is a powerful integral transform technique for studying many problems including financial derivatives such as options. Moreover, the approximation error is much smaller than that of the truncated Fourier expansion. Therefore, one can get better approximation of a function at jump discontinuity with the use of wavelet expansion rather than Fourier expansion.

In the current study, we employ wavelet analysis to option pricing problem manifested as partial differential equation (PDE) with jump characteristics. We have used wavelets to develop an optimum finite differencing of the differential equations manifested by financial models. In particular, we apply wavelet optimized finite-difference (WOFD) technique on the partial differential equation. We describe how Lagrangian polynomial is used to approximate the partial derivatives on an irregular grid. We then describe how to determine sparse and dense grid with wavelets. Further work on implementation is going on.

1. Introduction

Recently the subject of "wavelet analysis" has drawn much attention from both mathematicians and engineers alike. Many studies [4, 5, 7] have emerged applying

[*] Author for correspondence: tulsi@cs.umanitoba.ca
[†] Currently at Department of Computer Science, University of Calgary, Calgary, AB, Canada.

wavelets to multitude of situations, and all seem to report favorable results. Broadly defined, a wavelet is simply a wavy function carefully constructed as to have certain mathematical properties. An entire set of wavelets is constructed from a single wavelet called ”*mother wavelet*” function and this set provides useful building block functions that can be used to describe any in a large class of functions. Roughly speaking, wavelet analysis is a refinement of Fourier analysis. The Fourier transform is a method of describing an input function in terms of its frequency components. The frequency analysis using the Fourier decomposition of a function becomes “scale analysis” using wavelets. This means that it is possible to examine features of the function of any size by adjusting a scaling parameter in the analysis. With the wavelet analysis it is very easy to reveal the position of a function where the function is discontinuous. Because of the way wavelets work, the approximation error is much smaller than that of the truncated Fourier expansion and very significantly localized at the point of discontinuity. So we get better approximation of a function at discontinuity when we use wavelet expansion rather Fourier expansion. The usefulness of the wavelet transform technique is evident from the large number of fields to which it has been applied. Wavelet based techniques to solve partial differential equations (PDE) are a relatively new area of research; they have been discussed in papers by Dempster and Eswaran [12], Siddiqi and Manchanda [9], Jameson [1,2,3], Nielson [6]. This paper focuses on the methodology to the solution of Burger’s Equation in Wavelet Optimized Finite Difference Method (WOFD). We closely follow the work reported in [6] for the fundamentals and then apply he methodology to option pricing problem.

The rest of the paper is presented as follows. In section 2 we introduce the simplest wavelets, the Haar wavelets. We also introduce the basic concepts such as wavelet transforms and multiresolution analysis for the sake of completion. We then introduce Daubecheies wavelets, which have played a key role in the explosion of activity of wavelet analysis. In section 3 we discuss two of our recent studies on option pricing using fast Fourier transform and solving Black-Scholes equation using Pade approximation. In section 4 we explain how the wavelet transform framework can be used to solve PDEs especially Burger’s type equation that manifests an option pricing problem using wavelet optimized finite difference (WOFD) technique. We first describe how Lagrangian polynomial is used to approximate derivatives on an irregular grid. We then describe how to determine sparse and dense grids with wavelets. The implementation work is continuing. In section 5 we briefly describe our expectation on the implementation results for the solution of Burgers type equation. Finally, in section 6 we conclude our current study with a brief discussion on further work.

2. Wavelet Transform

In this section we shall introduce the basic concepts related to the Harr transform which is the basic or other wavelets like Daub 4 Wavelet. We will discuss the later wavelet transform in the subsequent subsection.

2.1. Harr Transform

Let us consider a function $\mathrm{f} = (f_1, f_2, f_3, ..., f_N)$ where *N* is a positive even integer which we refer to as the *length* of f**.** The values of **f** are the *N* real numbers $f_1, f_2, f_3, ..., f_N$. The Harr transform decomposes a discrete function into two sub-functions of half of its length. One sub-function is a running average or *trend;* the other sub-function is a running difference or *fluctuation.*

The *first trend* sub-function $\mathrm{a}^1 = (a_1, a_2, a_3, ..., a_{N/2})$ is computed by taking a running average as $\mathrm{a_m} = \frac{\mathrm{f_{2m-1} + f_{2m}}}{\sqrt{2}}$ .for *m= 1,2,3,…….,N/2.* The other sub-function is called *first fluctuation,* $\mathrm{d}^1 = (\mathrm{d_1, d_2, d_3, ..., d_{N/2_2}})$ is computed by taking a running difference as: $\mathrm{d_m} = \frac{\mathrm{f_{2m-1} - f_{2m}}}{\sqrt{2}}$ for *m=1,2,3,…….,N/2.* The Harr transform is performed in several stages or levels. The first level is the mapping $\mathbf{H_1}$ defined by a discrete function to its first trend $\mathbf{a^1}$ and first fluctuation $\mathbf{d^1}$**.** That is,

$$\mathrm{f} \xrightarrow{\mathrm{H1}} (\mathrm{a}^1 \mid \mathrm{d}^1)$$

The inverse of $\mathbf{H_1}$ maps the transformed function ($\mathbf{a^1}|\mathbf{d^1}$) back into the original function f via the following formula:

$$f = \left(\frac{a_1 + d_1}{\sqrt{2}}, \frac{a_1 - d_1}{\sqrt{2}}, ..., \frac{a_{N/2} + d_{N/2}}{\sqrt{2}}, \frac{a_{N/2} - d_{N/2}}{\sqrt{2}} \right).$$

2.2. Harr Wavelets

The scalar product **f.g** of the functions **f**=*(f₁,f₂.f₃,..,fₙ)* and **g** =*(g₁,g₂,g₃…gₙ)* is defined by $\mathrm{f.g} = (f_1.g_1 + f_2.g2 + ... + f_n.g_n)$. The 1-level *Haar wavelets* are defined as

$$W_1^1 = \left(\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 0,0......,0 \right)$$

$$W_2^1 = \left(0,0, \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 0,0....,0 \right)$$

.

.

.

$$W_{N/2}^1 = \left(0,0,......, \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}} \right)$$

Using 1-level Haar wavelets, we can express the values for the first fluctuation sub function $\mathbf{d^1}$ as scalar products $d_m = f.W_m^1$**.** Next, 1-level *Haar Scaling* function is defined as

$$V_1^1 = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0,0......,0 \right)$$

$$V_2^1 = \left(0,0,\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0,0....,0 \right)$$

.

.

.

$$V_{N/2}^1 = \left(0,0,......,\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}} \right)$$

Using 1-level Haar scaling function, we can express the values for the first trend sub function $\mathbf{a}^{\mathbf{1}}$ as scalar products $a_m = f.V_m^1$

2.3. Multiresolution Analysis

From the pervious section it is clear that a function is synthesized by starting with a very low resolution function and successively adding on details to create higher resolution versions and ending up with a complete synthesis of the original function at the finest resolution. This is known as *multiresolution analysis* (MRA). MRA is the heart of wavelet analysis. A function can be expressed by the sum of two functions: a first averaged signal and a first detailed signal as $f = A^1 + D^1$. Using Haar scaling function and wavelets, the averaged and detail functions can be expressed as

$$A^1 = a_1 V_1^1 + a_2 V_2^1 + ... + a_{N/2} V_{N/2}^1$$

$$D^1 = d_1 W_1^1 + d_2 W_2^1 + ... + d_{N/2} W_{N/2}^1$$

Applying the scalar product formulas for the coefficients, we can write these two formulas as follows:

$$A^1 = \left(f.V_1^1\right)V_1^1 + \left(f.V_2^1\right)V_2^1 + ... + \left(f.V_{N/2}^1\right)V_{N/2}^1$$

$$D^1 = \left(f.W_1^1\right)W_1^1 + \left(f.W_2^1\right)W_2^1 + ... + \left(f.W_{N/2}^1\right)W_{N/2}^1$$

The second level of a MRA of a function **f** involves expressing **f** as $\mathsf{f} = \mathsf{A}^2 + \mathsf{D}^2 + \mathsf{D}^1$ where, $\mathbf{A}^{\mathbf{2}}$ is the second averaged function and $\mathbf{D}^{\mathbf{2}}$ is the second detail function

$$A^2 = \left(f.V_1^2\right)V_1^2 + \left(f.V_2^2\right)V_2^2 + ... + \left(f.V_{N/4}^2\right)V_{N/4}^2$$

$$D^2 = \left(f.W_1^2\right)W_1^2 + \left(f.W_2^2\right)W_2^2 + ... + \left(f.W_{N/4}^2\right)W_{N/4}^2$$

In general, if the number *N* of function values is divisible k times into two sub functions then a k-level MRA : $f = A^k + D^k + ... + D^2 + D^1$

2.4. Duab4 Wavelets

The Daubechies wavelet transforms are defined in the same way as the Haar wavelet transform but the main difference is in the scaling function and in wavelets. Both the wavelets and scaling function has longer support that is with few more values they produce averages and differences. There are many Daubechies transforms, but all are very similar. We here just concentrate on the Daub4 wavelet as it is simplest one as well as it is used in WOFD technique for the solution of partial differential equation.

The 1-level Daub4 transform is the mapping $f \xrightarrow{D_1} \left(a^1 \mid d^1\right)$ from the function f to its first trend sub function **a**^{1} and first fluctuation sub function **d**^{1}. Each value a_m of **a**^{1}=*(a*$_1$*,…..a*$_{N/2}$*)* is equal to a scalar product: $a_m = f.V_m^1$. Similarly each value d_m is equal to a scalar product of **f** with a 1-level wavelet W_m^1. The scaling numbers [7] $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ and wavelet numbers $\beta_1, \beta_2, \beta_3, \beta_4$ are defined by

$$\alpha_1 = \frac{1+\sqrt{3}}{4\sqrt{2}}, \alpha_2 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \alpha_3 = \frac{3-\sqrt{3}}{4\sqrt{2}}, \alpha_4 = \frac{1-\sqrt{3}}{4\sqrt{2}}$$

$$\beta_1 = \frac{1-\sqrt{3}}{4\sqrt{2}}, \beta_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}, \beta_3 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \beta_4 = \frac{-1-\sqrt{3}}{4\sqrt{2}}$$

Using these numbers the scaling function and wavelets are defined by

$$V_1^1 = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0......, 0)$$

$$V_2^1 = (0, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0......, 0)$$

$$\vdots$$

$$v_{N/2}^1 = (\alpha_3, \alpha_4, 0, 0, ..., 0, \alpha_1, \alpha_2)$$

$$W_1^1 = (\beta_1, \beta_2, \beta_3, \beta_4, 0, 0......, 0)$$

$$W_2^1 = (0, 0, \beta_1, \beta_2, \beta_3, \beta_4, 0, 0......, 0)$$

$$\vdots$$

$$W_{N/2}^1 = (\beta_3, \beta_4, 0, 0, ..., 0, \beta_1, \beta_2)$$

Let us define **D**$_\mathbf{N}$ to be matrix such that the rows of **D**$_\mathbf{N}$ are the first-level Daub4 scaling function and wavelets.

$$D_N = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & 0 & 0 & . & . & 0 & 0 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 & 0 & 0 & . & . & 0 & 0 \\ 0 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & 0 & . & . & 0 \\ 0 & 0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 & 0 & . & . & 0 \\ . & . & . & .. & . & . & . & . & . & . \\ \alpha_3 & \alpha_4 & 0 & 0 & 0 & . & . & 0 & 0 & 0\alpha_1 \quad \alpha_2 \end{bmatrix}$$

The Equation shows that the rows of **D**$_\mathbf{N}$ form an orthonormal set of vectors. That is, **D**$_\mathbf{N}$ is an orthogonal matrix. Comparing the definition of the matrix DN and the definition of the 1-level Daub4 transform we see that $(a_1, d_1, a_2, d_{21}, ..., a_{N/2}, d_{N/2})^T = D_N f^T$.

3. Option pricing using FFT and Pade Approximation

In this section we briefly describe two methodologies in option pricing. Experience acquired through the use of fast Fourier transform for option pricing and the use of Pade approximation in reducing the severity of the mathematics behind the finance models for option pricing has been a driving force and motivation for the current study. In other words, these two studies are precursors to the current study using wavelets. While the FFT has limitations as mentioned before but could be easy to understand, limitations of the use of Pade scheme stems from the requirement of fundamental mathematical knowledge on the approximation theory, which may not be that easy for a finance practitioner. Hence we resort to the wavelet transform in this study. However, we present some fundamental background of these two methodologies

3.1. Fast Fourier Transform for Option Pricing

Since the sine and cosine functions used as the bases in Fourier analysis are very smooth (infinitely differentiable) and very broad (nonzero almost everywhere on the real line), they are not very effective for representing functions that change abruptly (jumps) or have highly localized support (diffusive). Moreover, the approximation error is generally large due to truncation of Fourier expansion. However, in one of the recent studies [13], this truncation error is avoided to a large extent. The computational cost of longer Fourier expansion is handled by developing a parallel algorithm for the study.

Fast Fourier Transform (FFT) has been used in many scientific and engineering applications. In the recent study [13], the FFT has been applied for a novel application in finance. Based on a previous study [14] the authors have improved a mathematical model of Fourier transform technique proposed in [15] for pricing financial derivatives so as to help design an effective and efficient parallel algorithm. They have then developed a new parallel algorithm for FFT using a swapping technique that exploits data locality [13]. They have also analyzed their algorithm theoretically and have reported the significance of the new algorithm. This algorithm was implemented on a *20* node *SunFire-6800* high performance computing system and compared the new algorithm with the traditional Cooley-Tukey [16] FFT algorithm both as stand alone comparison of the performance and in relation to our theoretical analysis and showed higher efficiency of the new algorithm. The computation of the longer Fourier expansion not only reduces the error bounds inherent with Fourier transform but use high performance algorithm reduces the time required to do the computation to a very large extent.

3.2. Pade Approximation

Option pricing is one of the prominent and challenging problems in computational finance. Various explicit and implicit schemes were developed for the option pricing problem. Explicit methods usually need less computation because it does not entail solving a set of linear equations at each time step. Implicit methods, on the other hand, have better stability and convergence properties but are computationally more demanding. Many of the schemes used in finance are first order in time. Courtedon [17] developed a second order accurate finite difference method for valuing options. For simplest problems, one can generally transform the Black-Scholes model to the simple heat equation for which well developed solution schemes are available. Mayo [18] developed a fourth order method in the log of asset prices to evaluate American options.

The readers are directed to the book [19] for further information on finite-differencing for financial derivatives.

In general, it is often difficult to reduce the mathematical severity of the finance model to a simpler form. Towards this perspective, a recent study [20] has developed a second order L_0 stable discrete parallel algorithm using *Pade* approximation for experimentation towards high performance architectures. This algorithm is suitable for more complicated option pricing problems although in [20, 21] the authors have applied to the Black-Scholes model. For simulation purposes, the authors have implemented this algorithm and evaluated the European options. Numerical results are compared with those obtained using other commonly used numerical methods and shown that the new algorithm is robust and efficient than the traditional schemes.

A popular finite-difference method for solving PDEs is the Crank-Nicholson method which is based on $(1,1)$ Pade approximation. Higher order Pade approximation gives higher accuracy in time. However, this could lead to higher computational complexity. Therefore, a balance of accuracy and available resources dictated our
selection of $(2,0)$ Pade to illustrate the higher order finite difference method in solving the option pricing problems. This is a new contribution in the use of Pade approximation to the field of computational finance in [20]. Moreover, the $(2,0)$ method is L_0-stable, and hence it has better stability than the Crank-Nicholson scheme which is only A_0-stable. Moreover, with the L_0-stability, the error remains bounded, which is not guaranteed in A_0-stable algorithms, one of which is Crank-Nicholson scheme. Employing the Pade scheme to the B-S model leads to [20]
$$BU(t-k) = U(t) - \frac{l}{2}[BC(t-k) + C(t)], t = 0, k, 2k.....$$
where
$$A = \begin{bmatrix} b_2 & c_2 & 0 & 0 & 0 & 0 & ...... & . & 0 & 0 \\ a_2 & b_2 & c_2 & \beta_4 & 0 & 0 & . & . & 0 & 0 \\ \multicolumn{10}{c}{...} \\ . & . & . & .. & . & . & . & . & . & . \; . \; . \; . \; . \\ \multicolumn{10}{r}{...a_{M-1} \; b_{M-1} \; c_{M-1}} \\ 0 & 0 & 0 & 0 & 0 & . & . & 0 & 0 & 0 b_M \quad c_M \end{bmatrix},$$

$C(t) = [a_1 u(0,t), 0, ...0]^T$ and U is a vector of asset prices. This discretized form of the Black-Scholes model can be implemented with some careful consideration on the parametric conditions, as described in [20]. The disadvantage of this scheme is the requirement of fundamental knowledge on the approximation theory.

4. Wavelet Optimized Finite-Differencing

In this section we outline an approach to solve a type of partial differential equation with the aid of wavelets. This method is due to Jameson [1,2,3] and is called wavelet optimized finite difference (WOFD) technique. Wavelets provide a perfect mechanism for grid selection where sparse grids are placed in regions of the domain where the activities are smooth (for example in a fluid mechanics problem, where flow is smooth; or in an option pricing problem the underlying asset price behave smoothly) and fine grids are placed in regions of the domain where the activities are chaotic (for example, in fluid mechanics problem, flow features are rough or perhaps highly oscillatory; or in option pricing problem with high frequency underlying asset price behavior). In other words, if a portion of the computational domain of an option pricing problem where asset price behavior is composed of large smooth features, then a high order low grid point density is optimal. If another portion of the same option pricing problem is composed of high frequency features, then optimality is obtained by increasing the density of the grid points and decreasing the order. Before describing the wavelet optimized finite-differencing scheme we describe two other differencing techniques based on interpolation and finite-differencing, which lays foundation for the focus of the current study.

4.1. Generating Difference Equation: Interpolation

For a vector **f** of *N* numbers, we can get a better approximation of the derivative **f’** at *i*th point if there are more elements around the *i*th point of **f.** Common finite difference formulas are found by fitting an algebraic polynomial of degree *q* locally around the *i*th point of a vector **f** of evenly spaced elements to obtain difference approximations of accuracy *q-1*. Interpolations with algebraic polynomials are probably the most common and popular way to generate differencing coefficients. One simply fits the polynomial to the data, followed by differentiation of the polynomial, and finally one evaluates the polynomial at the point of interest. The well known Lagrange interpolation formula [11] for algebraic interpolation is

$$P(x) = f(x_0).L_{n,0}(x) + f(x_1).L_{n,1}(x) + ... + f(x_n).L_{n,n}(x)$$

where

$$L_{n,k}(x) = \frac{(x-x_0)(x-x_1).....(x-x_{k-1})(x-x_{k+1})......(x-x_{n-1})(x-x_n)}{(x_k-x_0)(x_k-x_1)......(x_k-x_{k-1})(x_k-x_{k+1}).......(x_k-x_{n-1})(x_k-x_n)}$$

$$= \prod_{i=0,i\neq k}^{n} \frac{(x-x_i)}{(x_k-x_i)}$$

for each k=0,1,…..,n. Differentiation of the above equation *d* times yields

$$P^d(x) = f(x_0)L_{n,0}{}^d(x) + f(x_1)L_{n,1}{}^d(x) + ... + f(x_n)L_{n,n}{}^d(x)$$

We are interested here with the first and second order derivatives. It can be easily shown that the first and second order derivatives can be found by the following equation.

$$L_{n,k}{}^1(x) = \frac{d}{dx}L_{n,k}(x) = \frac{\sum_{l=0,l\neq k}^{n}\prod_{m=0,m\neq k,l}^{n}(x-x_m)}{\prod_{i=0,i\neq k}^{n}(x_k-x_i)}$$

$$L_{n,k}{}^{2}(x)=\frac{d^2}{dx^2}L_{n.,k}(x)=\frac{\sum_{l=0,l\neq k}^{n}\sum_{m=0,m\neq k,l}^{n}\prod_{n=0,n\neq k,l,m}^{n}(x-x_n)}{\prod_{i=0,i\neq k}^{n}(x_k-x_i)}$$

4.2. Generating Difference Equation: finite-differencing

We describe the discretization using finite-differencing here following the steps reported in [21]. Consider the B-S model $\frac{\partial u}{\partial t}+\frac{\sigma^2 S^2}{2}\frac{\partial^2 u}{\partial S^2}+rs\frac{\partial u}{\partial S}-ru=0$ where *u* is the option price, *t* is time, *σ* is volatility, *S* is the asset price, and *r* is the interest rate. By changing the variables, we can reduce the above B-S equation to the diffusion equation. The B-S equation is discretized as follows: A finite number of equally spaced time steps between the current date (*t = 0*) and the maturity date of the option, (*t = T*) are chosen. That is, $\Delta t=T/N$ and the total of (*N + 1*) times are considered. Similarly, a finite number of equally spaced asset prices (N_j) are also chosen. We suppose that $S_{\max}$ is the maximum price an asset can reach. We define $\Delta S=S_{\max}/2N_j$ and consider a total of $2N_j+1$ asset prices including the current asset price. By the above discretization, a grid consisting of a total of $(N+1)(2N_j+1)$ points can be constructed [21] . The grid point (i,j) corresponds to time $i\Delta t$ and price $j\Delta S$. The variable, $u(i,j)$ refers to the value of the option at the grid point (i,j). The accuracy of the original system will be dictated by the convective term $\frac{\partial u}{\partial S}$ rather than the diffusion term $\frac{\partial^2 u}{\partial S^2}$. Therefore, central-differencing for the convection term and forward or backward-differencing for the diffusion term would be sufficient. However, if central differencing is applied to the diffusion term as well, the additional cost incurred due to central-differencing could be recovered with the multithreaded implementation of the algorithm [21]. The solution scheme is marched in the time direction until it reaches a steady state. The computed values of the layer *(c-1)* are used to calculate the values of layer *c*, the number of computational layers depend on the relative error. The value $u_{i,j}^{c}$ denotes the option value at (i,j) grid point in c^{th} computational time layer. To compute the option values at the c^{th} layer we use the values from the $(c-1)^{th}$ layer, which can be expressed as $u_{i,j}^{c}=\rho u_{i+1,j-1}^{c-1}+(1-2\rho)u_{i+1,j}^{c-1}+\rho u_{i+1,j+1}^{c-1}$. Here $\rho=l/h^2$, where *l* denotes time step τ and *h* denotes space step *x*. If τ is from *0* to *T* and *x* is from $x_{\min}$ to $x_{\max}$, then $N\times l=T$ and $2N_j\times h=x_{\min}-x_{\max}$. Therefore, the terminal condition $t=T$ is reformed to the initial condition $\tau=0$. The relative *err* is calculated as $err=u_{i,j}^{c}-u_{i,j}^{c-1}$. The computation is stopped once the *err* falls below a certain preset threshold value.

We employ similar steps in wavelet assisted discretization of the transformed Black-Scholes equation.

4.3. Grid Generation with Wavelets

WOFD technique applies finite difference on a grid which is defined by the magnitude of wavelet coefficients at various scales. That is wavelet can detect oscillations in a function at any location and scale. The function *f(x)* may be decomposed into a set of wavelet coefficients that depend on two parameters, one for location and one for scale, say d_k^j, where k is the location parameter and j is the scale parameter. For example, if coefficient $d_5^2 > \varepsilon,$ where ε is a user-defined sensitivity threshold, then one can add a grid point at location x_{20}, since the wavelet coefficients at scale *j=2* represent local high frequencies in the physical space at scale $4\Delta x,$ i.e., $5 * 4\Delta x = x_{20}$. x_i represents the numerical value of the i^{th} grid point. One can add more grid points in any region around large wavelet coefficients, and it is more efficient to do so.

4.4. Burgers equation

We now focus on Burgers equation which is frequently used to study adaptive methods because it models formation of shocks. Shocks in a physical phenomenon such as fluid mechanics could be a sudden jump in the flow behavior while in a finance problem it would mean sudden changes in an important parameter such as the underlying asset price. This single equation has terms that closely duplicate the physical properties of many engineering problems such flow equations. For our purpose, Burgers' equation can be seen a manifestation of, for example, Black-Scholes model. That is, the model equation should have a convective term, a diffusive term, and a time-dependent term (or reaction term). Further discussions in this section are based on the Burgers equation, a convective, diffusive, reactive equation. Black-Scholes model can be transformed into the Burgers' equation. Burgers (1948) introduced a simple nonlinear equation that meets these requirements [8]

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \mu \frac{\partial^2 u}{\partial x^2} \text{ for } t>0 \tag{1}$$

with initial and boundary conditions given by

$$u(x,0) = h(x)$$
$$u(x,t) = u(x+1,t)$$

The equation can be discretized and we get the following system

$$\begin{aligned} &\frac{d}{dt}u(t) = Lu(t) + N(u(t)), t > 0 \\ &u(0) = h \equiv [h(x_0), h(x_1), ..., h(x_{N-1})]^T \end{aligned} \tag{2}$$

where,

$$L = \mu D^2$$
$$N(u(t)) = -diag(D^1 u(t))$$

We can use a standard finite difference technique such as Crank-Nicholson [22] for time stepping, though Crank-Nicholson is an A_0 stable algorithm as mentioned before.

However, in the current study since we are optimizing the finite-difference by wavelet transform, we presume that the Crank-Nicholson scheme would be sufficient enough in the academic setting. Moreover, we presume that applying Pade approximation to equation (2) would not be difficult.

Applying the Crank-Nicholson scheme to eq.(2) can be written as:

$$\frac{u(t+\Delta t)-u(t)}{\Delta t} = L\frac{u(t+\Delta t)+u(t)}{2} + N\left(\frac{u(t+\Delta t)+u(t)}{2}\right)\frac{u(t+\Delta t)+u(t)}{2}$$

and we obtain time-stepping procedure

$$Au_{n+1} = Bu_n + \Delta t N\left(\frac{u_{n+1}+u_n}{2}\right)\frac{u_{n+1}+u_n}{2} \quad \textit{for } n=0,1,\ldots n_1\text{-}1$$

$$\mathbf{u_0=h}$$

where, $A = I - \frac{\Delta t}{2}L, B = I + \frac{\Delta t}{2}L$ and $u_n = u(n\Delta t)$

4.5. Algorithm for WOFD Method

1. Initialize model parameters such as such as domain size (L=1), grid size (N=2^{10}), diffusion parameter ($\mu = 0.002$) time stepping (dt=0.0009), initial time (time=0).
2. Initialize N sample points (0:h:L-h), h=L/N. Initialize u=sin(2*pi*x)
3. Generate Finite Difference (FD) operators. With this finite difference operators and Crank-Nicholson matrices construct Matrices A and B. Make LU decomposition of A.
4. Reconstruct u on the finest grid with the Lagrange Interpolation.
5. Define new grid by wavelet methods
6. Restrict u to new grid (depending on the new points or deleted points selected by wavelet grid evaluator)
7. Update finite difference operators according to new grid.
8. Solve the values of u_{i+1}. If we LU decompose A then the solution of the restricted grid can be found

   $temp = B*u_i - dt(\mu * u)*D^2$

   $LUu_{i+1} = temp$

   Uu_{i+1}=L\temp

   $u_{i+1} = U\backslash L\backslash temp$

9. Advance time step time=time+dt
10. Do step 3 -9 until time > 0.5.

5. Conclusions

Implementation of the above algorithm is still going on and the preliminary result from solution of the Burgers equation is encouraging. However, at present we do not have substantial results to report.

In this paper we have attempted to apply the wavelet multi-resolution analysis to solve a partial differential equation. For this purpose a well known partial equation known as Burger Equation was chosen and applied wavelet optimized finite difference technique for the solution of this model equation. The main goal for this study is to overcome the difficulties encountered in the fast Fourier transform technique and the classical Pade approximation technique in solving the option pricing problem and to discover the possibility of application of WOFD technique. We are encouraged from our current theoretical endeavor so far and expect to get substantial results in the near future.

Acknowledgements

The first two authors acknowledge partial financial support from Natural Sciences and Engineering Research Council (NSERC) of Canada and the University of Manitoba Research Grant Program (URGP).

References

[1] L. Jameson, *On the Daubechies- based wavelet differential matrix,* SIAM J. Sci. Comput., 6 (1996), pp498-516.

[2] L. Jameson, *On the Wavelet-Optimized Finite Difference Method*, ICASE report 94-9, NASA CR-194601, 1994.

[3] L. Jameson, *A wavelet optimized, very high order adaptive grid and order numerical method*, SIAM J. Sci. Comput. Vol.19, No.6, pp.1980-2013, Nov., 1998.

[4] Charles K. Chui, *An Introduction to wavelets,* Academic press, San Diego, CA 92101, 1992.

[5] Charles K. Chui, *Wavelets: A mathematical tool for signal processing,* SIAM, Philadelphia PA19104-2688, 1997.

[6] Ole M. Nielson, *Wavelets in Scientific Computing,* Ph.D. Dissertation, Department of Mathematical Modeling, Technical University of Denmark, Denmark 1998.

[7] James S. Walker, *A premier on: wavelets and their scientific applications,* CRC Press LLC .1999

[8] G.D Smith, *Numerical Solution of Partial Differential Equation: Finite difference methods.* Oxford University Press, 1985.

[9] Abul Hassan Siddiqi , *Fast wavelet-based algorithms for option pricing,* Proc. world Multi conference on Systemic, Cybernetics and Informatics, July 2002, Orlando, FL, USA.

[10] Dale A. Anderson and John C. Tannehill, *Computational Fluid Mechanics and Heat Transfer,* Hemisphere publishing corp.QA901.A53 1984.

[11] Richard I. Burden & J. Douglas Faires, *Numerical Analysis*, PWS-KENT publishing company Boston, 1989.

[12] M.A.H Dempster & A. Eswaran, *Solution of PDEs by wavelet methods*, Working Paper 25, 2001.

[13] S. Barua, R.K. Thulasiram and P. Thulasiraman, *High Performance Computing for a Finance Application Using fast Fourier Transform,* Springer Lecture Notes on Computer Science, (Proc. of the European Parallel Computing Conference, Lisbon, Portugal, Aug-Sep, 2005) (to appear).

[14] R. K. Thulasiram and P. Thulasiraman, *Performance Evaluation of a Multithreaded Fast Fourier Transform Algorithm for Derivative Pricing*, The Journal of Supercomputing, **26**, 1, 43-58, Aug. 2003.

[15] P. Carr and D. Madan, *Option Valuation using the Fast Fourier Transform*, The Journal of Computational Finance, **2**, 4, pp.61-73, 1999.

[16] J.W. Cooley and P.A. Lewis and P.D. Welch, *The Fast Fourier Transform and its Application to Time Series Analysis*, Chapter in "In Statistical Methods for Digital Computers", pp. 377-423", Wiley, 1997, New York.

[17] G. Courtedon, *A more accurate finite difference approximation for the valuation of options,* Journal of Financial and Quantitative Analysis, **17**, pp.697-705, 1982.

[18] A. Mayo, *Fourth Order Accurate Implicit Finite Difference Method for Evaluating American Options,* Proc. (CD-RoM) of the International Conference on Computational Finance, London, England, June 2000.

[19] D. Tavalla and C. Randall, *Pricing Financial Instruments: The Finite Difference Method*, John Wiley and Sons, New York, NY, 2000.

[20] R. K. Thulasiram , C. Zhen and A. Gumel, *A second order L_0 stable algorithm for evaluating European options*, Proc. International Symposium on High Performance Computing Systems and Applications **(HPCS)**, Winnipeg, MB, Canada, pp.17-23, May 2004.

[21] R. K. Thulasiram, C. Zhen, A. Chhabra, P. Thulasiraman and A. Gumel, *A second order L_0 stable algorithm for evaluating European options*, Intl. J. High Performance Computing and Networking (accepted – for spring 2006 issue).

[22] D. A. Anderson and J. C. Tannehill and R. H. Pletcher, *Computational Fluid Dynamics and Heat Transfer*, Hemisphere Publishing Corporation, 1984.