UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

Fast Wavelet Based Methods for Certain Time Dependent Problems (Ph.D. Thesis)

An Jiang

July 1996

CAM Report 96-20

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90024-1555

UNIVERSITY OF CALIFORNIA

Los Angeles

Fast wavelet based methods for certain time dependent problems

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Mathematics

by

An Jiang

1996

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT OF THE DISSERTATION

Fast wavelet based methods for certain time dependent problems

by

An Jiang

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 1996

Professor Stanley Osher, Chair

Observing that under a lot of circumstances only the solution at some particular points for a PDE are needed, we develop fast algorithms to solve parabolic equations at one point without calculating the solution in the whole space-time domain. This is achieved by taking advantage of the sparse wavelet representation of finite difference methods or the operator itself and using only parts of the representation to compute the local solution. The complexity for solving at one point is only $\mathcal{O}(\log^4(N))$ when the equation has time independent coefficients. When the coefficients do depend on time, the complexity is $\mathcal{O}(N\log^3(N))$.

CHAPTER 1

Introduction

During the last few years a number of fast computational algorithms have been developed for elliptic problems. These are techniques for which the number of arithmetic operations needed are close to linear as a function of the number of unknowns. Examples of algorithms of such complexity are multigrid methods and the so-called fast Poisson solvers. The wavelet based methods for elliptic problems formulated as integral equations also belong to this category ([**BCoR**]).

There has not been the same progress for hyperbolic and parabolic methods. In general, classical numerical techniques for these problems are already optimal. However we mention that in [**GrS1**] the multidimensional heat operator, with u_0 and f(see (1.1) below) both zero, but with inhomogeneous boundary data given at M points, was treated. There the closed form of the solution evaluated at M points at time level N was obtained in $\mathcal{O}(MN)$ rather than $\mathcal{O}(N^2M^2)$ operations. Also, in [**GrS2**] the same authors obtained an algorithm for evaluating the sum of N Gaussians at M arbitrary distributed points in $\mathcal{O}(N+M)$ operations. But their interesting method appears to need an explicit analytic representation of the heat kernel, effectively ruling out variable coefficient problems.

In the work of B.Engquist, S.Osher and S.Zhong [**EnOZh**], they introduced a

fast algorithm for solving hyperbolic and parabolic equations. Consider a evolution equation

$$\partial_t u + L(x, \partial_x)u = f(x), \quad x \in \Omega \in R^d, \quad t > 0, \tag{1.1}$$

$$u(x, 0) = u_0(x),$$

with boundary conditions, where L is a differential operator.

An explicit algorithm typically has the form

$$u^{n+1} = Au^n + F,$$

$$u^0 = u_0,$$

with u_0 and $F \in \mathbf{R}^{N^d}$ given vectors. This formula admits a closed form solution given by

$$u^n = A^n u^0 + \sum_{j=0}^{n-1} A^j F.$$

This form can be used to compute the solution $A^n u_0$, for $F = 0$, in $\log(n)$ steps by repeated squaring of A: $A, A^2, A^4, \ldots$. The latter squarings would involve almost dense matrices even if we started with very sparse A and the algorithm, as stated, is useless. However, for an appropriate representation of A in a wavelet basis all of the powers A^v may stay sparse and the algorithm using repeated squaring should be advantageous. This leads to an algorithm with $\mathcal{O}(N^d \log(N))$ complexity for many hyperbolic and parabolic equations.

In this paper we will investigate more wavelet based fast methods for parabolic equations.

Under a lot of circumstances only the solution at isolated points is needed. An interesting example occurs when calculating stock option prices using the Black-Scholes Equation. It is impossible in traditional finite difference methods to obtain local solutions without calculating the solution in the whole space-time domain. We develop a fast wavelet algorithm to solve evolution equations at one point without obtaining the whole solution. When the coefficients in the equation are not time dependent, we can use the above idea of repeated squaring. The resulting complexity for solving at one point is only $\mathcal{O}(\log^4(N))$, where N is number of grid points in the domain.

We also investigate certain important time-dependent problems, which are not discussed in **[EnOZh]**. When the coefficients, and thus the A matrices, do depend on time, we can calculate each of them and their wavelet transforms in advance. After that it is efficient to calculate the solutions for different initial conditions. In the following chapters we will describe those methods in detail.

The adaption to higher dimension is quite straight forward. In fact this method is preferable for multidimensional problems, because of the simple fact that $\log(N^d) = d\log(N)$.

Numerical results are presented in Chapter 5.

CHAPTER 2

Background

Fourier transforms have a long history of being used to solve differential equations. Despite the advantages of Fourier transforms, they are not well adapted to the local analysis of a function. A local perturbation may significantly affect all coefficients, therefore no terms of the expansions can be safely ignored. One would prefer to have an effective local analysis in many circumstances, for example in studying singularities or shock formation in PDE. The wavelet transform is a tool that cuts up data or functions or operators into different frequency components, and then studies each component with a resolution matched to its scale. It is a useful tool to achieve both space and frequency localization.

2.1 Compactly Supported Orthonormal Wavelets

The wavelets we will be using are Daubechies' compactly supported orthonormal wavelets.

A wavelet basis of this kind is built from an averaging function φ and a differencing function ψ, which satisfy the following relations:

$$\varphi(x) = \sqrt{2} \sum_{k=0}^{2M-1} h_{k+1} \varphi(2x-k) \tag{2.1}$$

$$\psi(x) = \sqrt{2} \sum_{k=0}^{2M-1} g_{k+1} \varphi(2x-k) \tag{2.2}$$

where

$$g_k = (-1)^{k-1} h_{2M-k+1}, \qquad k = 1, \cdots, 2M \tag{2.3}$$

and

$$\int \varphi(x) dx = 1. \tag{2.4}$$

The coefficients $\{h_k\}_{k=1}^{2M}$ are chosen so that the functions

$$\psi_k^j(x) = \psi_{I_{j,k}} = 2^{-j/2} \psi(2^{-j}x - k + 1), \tag{2.5}$$

where j and k are integers and $I_{j,k}$ is the interval $[2^j(k-1), 2^j k]$, form an orthonormal basis and, in addition, the function ψ has M vanishing moments

$$\int \psi(x) x^m dx = 0 \qquad m = 0, \cdots, M-1. \tag{2.6}$$

We also need the dilations and translations of the scaling function φ,

$$\varphi_k^j(x) = \varphi_{I_{j,k}} = 2^{-j/2} \varphi(2^{-j}x - k + 1). \tag{2.7}$$

Suppose $\{s_1^0, s_2^0, ..., s_{2^v}^0\}$ are the samples of a function f on the finest scale. The

wavelet coefficients is computed via the pyramid scheme

$$\begin{array}{ccccccc} s_k^0 & \to & s_k^1 & \to & s_k^2 & \to & s_k^3 \cdots \\ & \searrow & & \searrow & & \searrow & \\ & d_k^1 & & d_k^2 & & d_k^3 \cdots & \end{array}$$

This is implemented in $\mathcal{O}(N)$ operations using:

$$s_k^j = \sum_{p=1}^{p=2M} h_p s_{p+2k-1}^{j-1},$$
$$d_k^j = \sum_{p=1}^{p=2M} g_p s_{p+2k-1}^{j-1},$$

and the s_k^j, d_k^j are viewed as periodic sequences with period 2^{v-j} (See **[Remark]** below).

The coordinates in the orthonormal basis consist of

$$[d_1^1, \cdots, d_{N/2}^1, d_1^2, \cdots, d_{N/4}^2, \cdots, d_1^v, s_1^v].$$

This procedure can be viewed as a linear transformation in R^N, which is the Euclidean space of all periodic sequences with the period N. We can write it as $\hat{f} = Wf$ where W is the matrix representation of the linear transform.

The inverse mapping can also be done in $\mathcal{O}(N)$ operations.

[Remark] For wavelets with more than 1 vanishing moments, the supports of φ_k^j and ψ_k^j are greater than the interval $I_{j,k}$. The functions φ_I, φ_J can have overlapping supports for $I \neq J$. As a result, the pyramid structure "spills out" of the interval $[1, N]$ on which the structure is originally defined. Therefore, it is

technically convenient to replace the original structure with a periodic one with period N. This is equivalent to replacing the wavelet with its periodized version, or replacing the function f with its periodized version. Unless f is already periodic, we introduced a discontinuity at the boundaries, which will result in large fine scale wavelet coefficients near the boundaries. This problem can be solved by using the wavelet basis for intervals described in **[CDJP]**. In their construction, the wavelets in the interior of the interval is kept as are, while the basis that "spill out" from the boundaries are replaced by adapted functions that has support inside the interval.

In two dimensions, there are two natural ways to construct the wavelet systems. The first basis is defined by three basis functions: $\psi_I(x)\psi_{I'}(y)$, $\psi_I(x)\varphi_{I'}(y)$, and $\varphi_I(x)\psi_{I'}(y)$ to each dyadic square $I \times I'$ with $|I| = |I'|$. The representation of a matrix or an operator in this basis is called the *non-standard form*, because if the original matrix represents an operator, its non-standard form is not the representation of the operator in wavelet bases. The second basis is simply the tensor product of the one dimensional wavelets, $\psi_{I\times J} = \psi_I \otimes \psi_J$. The representation of a matrix or an operator in this basis is called the *standard form.* We will use the standard form in our wavelet algorithms. We use the dyadic intervals to label the elements. For example if $K(x,y)$ is an operator then its standard form $\hat{K}$ consists of

$$\hat{K}_{I,J} =< K\psi_I, \psi_J >= \int\int K(x,y)\psi_I(x)\psi_J(y)dxdy. \tag{2.8}$$

The standard form of a matrix can be obtained by doing the one dimensional

wavelet transform first for every row of the matrix and then for every column of the matrix. We can write this as $\hat{K} = WKW^{-1}$ where K is the original matrix and $\hat{K}$ is its standard form.

2.2 Sparse Representation of Operators

The role of the orthonormal wavelet bases in solving integral equations has been studied in **[BCoR]**, where it was observed that wide classes of operators have sparse representations in the wavelet bases, thus permitting a number of fast algorithms for applying these operators to functions. The operators which can be efficiently treated using representations in the wavelet basis include Calderon-Zygmund and pseudo-differential operators. (see **[BCoR]** for definitions.)

Consider an integral operator (Calderon-Zygmund or pseudo-differential operator),

$$T(f)(x) = \int K(x,y)f(y)dy, \tag{2.9}$$

If we construct its wavelet standard form, then we find that the entries decay very fast as the distance to the singularity increases, much much faster than in the original kernel. For example, let the kernel satisfy the conditions

$$|K(x,y)| \leq C|x-y|^{-1} \quad \text{if } x \neq y \tag{2.10}$$

$$|\partial_x^M K(x,y) + \partial_y^M K(x,y)| \leq C|x-y|^{-(M+1)} \quad \text{if } x \neq y \tag{2.11}$$

for some $M \geq 1$, and the weak-cancelation property:

$$|\int_{I*I} K(x,y)dxdy| \leq C|I|. \tag{2.12}$$

Then, by choosing the wavelet basis to have M vanishing moments, the elements $\hat{K}_{IJ}$ of the standard form of the kernel satisfy the estimate

$$|\hat{K}_{IJ}| = |< K\psi_I, \psi_{I'} >| \leq C_M(\frac{|I|}{|J|})^{1/2}(\frac{|I|}{d(I,J)})^{M+1}, \tag{2.13}$$

where $d(I,J)$ denotes the distance between I and J, and it is assumed that $|I| \leq |J|$. Thus for a given accuracy the representation of such an operator is sparse, in the famous finger shape (see Figure 2.1), and consists of only $N\log(N)$ elements, where N is the size of matrix.

That means if $\hat{K}$ is the matrix representation of the kernel in the wavelet basis, and $\hat{K}^b$ is a truncation of $\hat{K}$ obtained by removing the elements that are outside of the bands of width $b \geq 2M$ around all the shifted diagonals, then $\hat{K}^b$ is an approximation of $\hat{K}$ for b large enough, with the following error estimate:

$$||\hat{K} - \hat{K}^b|| \leq \frac{C}{b^M}\log(N). \tag{2.14}$$

We will also need the estimate for the nonstandard form of $\hat{K}$ which is also given in **[BCoR]**:

$$|< K\psi_I, \varphi_{I'} >| \leq \frac{C|I|^{M+1}}{|I|^{M+1} + \text{dist}(I,I')^{M+1}} \quad \text{for } |I| = |I'| \tag{2.15}$$

This estimate is also true for $|< K\varphi_I, \psi_{I'} >|$ and $|< K\psi_I, \psi_{I'} >|$.

As we know, it takes $\mathcal{O}(N^2)$ operations to apply a dense $N \times N$ matrix to a vector. If a matrix is smooth away from singularities, its wavelet form, either

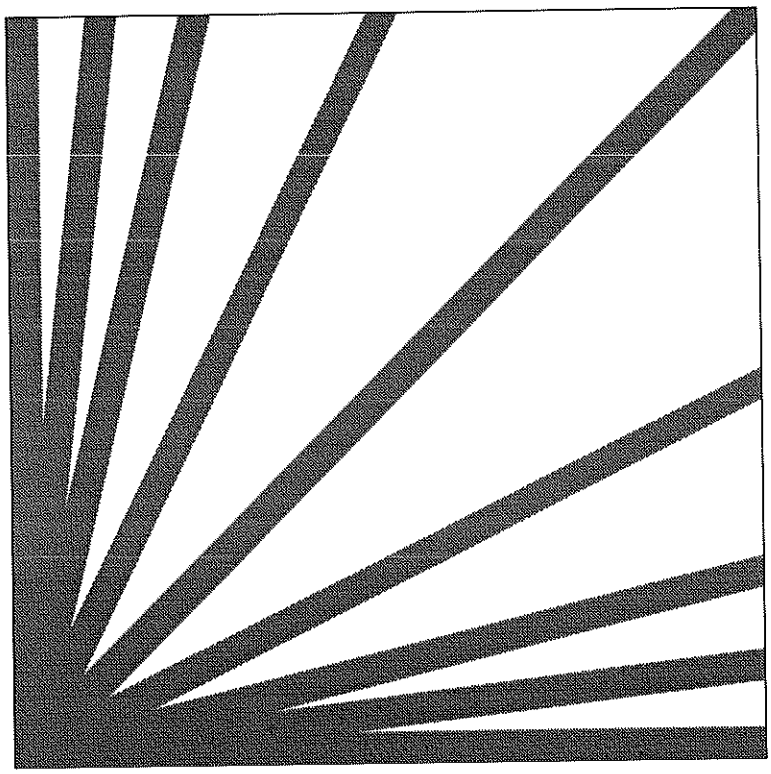

Figure 2.1: The Finger Form

standard or non-standard, is sparse. So the cost of applying its wavelet form to a vector is only $\mathcal{O}(N)$ if the non-standard form is used, and $\mathcal{O}(N\log(N))$ if the standard form is used.

2.3 Evolution Equations

In the work of B.Engquist, S.Osher and S.Zhong **[EnOZh]**, they introduced a fast algorithm for solving hyperbolic and parabolic equations. The algorithm is based on the fact that the Green's functions of hyperbolic and parabolic equations have sparse representations in wavelet bases. Their algorithm uses standard ex-

plicit finite difference schemes, and the wavelet representation is used to carry out the algebraic part.

Consider a evolution equation

$$\partial_t u + L(x, \partial_x)u = f(x), \quad x \in \Omega \in R^d, \quad t > 0, \tag{2.16}$$

$$u(x, 0) = u_0(x),$$

with boundary conditions, where L is a differential operator.

An explicit algorithm typically has the form

$$u^{n+1} = Au^n + F,$$

$$u^0 = u_0,$$

with u_0 and $F \in \mathbf{R}^{N^d}$ given vectors. This formula admits a closed form solution given by

$$u^n = A^n u^0 + \sum_{j=0}^{n-1} A^j F.$$

This form can be used to compute the solution $A^n u_0$, for $F = 0$, in $\log(n)$ steps by repeated squaring of A: $A, A^2, A^4, \ldots$.

The latter squarings would involve almost dense matrices even if we started with very sparse A and the algorithm, as stated, is useless. However, for an appropriate representation of A in a wavelet basis all of the powers A^v may stay sparse and the algorithm using repeated squaring should be advantageous.

The Green's function satisfies estimates of the types 2.10 and 2.11 with the constants uniform in time. A^v is the approximation of the Green's function. Thus

there exists a bandwidth b which is uniform in v, such that the banded version of A^v with bandwidth b is an approximation of A^v with a given accuracy.

This leads to the following algorithm with $\mathcal{O}(N^d \log(N))$ complexity for equation (2.16) with $f = 0$.

$$\hat{A} = WAW^{-1}$$

For $i = 1 : \log(n)$

$$\hat{A} = \text{TRUNC}(\hat{A} * \hat{A}, \epsilon)$$

End

$$\hat{u_0} = Wu_0$$

$$\hat{u^n} = \hat{A}\hat{u_0}$$

$$u^n = W^{-1}\hat{u^n}$$

The matrix W corresponds to a fast wavelet transform and the truncation operator sets elements in a matrix to zero if their absolute value is below a given threshold. Of course the matrices W and W^{-1} are never formed and the wavelet transforms and the inverse transforms are done using the pyramid scheme described in section 2.1.

The algorithms that we will describe in latter chapters are based on this algorithm.

CHAPTER 3

Local Solution

Observing that under many circumstances we only need to solve an equation at some isolated points, it is desirable to have a fast method to obtain solutions around certain locations.

3.1 Description of the Method

Consider the parabolic equation

$$L(x,t,\partial_x,\partial_t)u \equiv (\partial_t - a_{ij}(x)\partial_{ij} + a_i(x)\partial_i + a_0(x))u = 0, \tag{3.1}$$

where the matrix a_{ij} is positive definite. (Our method also works for all $2m^{th}$ order parabolic equations.)

The solution of this equation is

$$u(x,t) = \int G(x,y,t)u_0(y)dy. \tag{3.2}$$

The Green's function $G(x,y,t)$ has a singularity at $x = y$, thus the solution at a point x^* mainly depends on the initial condition in a neighborhood of x^*; i.e. if

we define a cut-off function

$$\Phi_{x^*,B}(x) = \begin{cases} 1, & |x^* - x| \leq B \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

then

$$\begin{aligned} \Phi_{x^*,\epsilon}(x)u(x,t) &= \int \Phi_{x^*,\epsilon}(x) * G(x,y,t)u_0(y)dy \\ &\approx \int \Phi_{x^*,\epsilon}(x) * G(x,y,t) * \Phi_{x^*,B}(y)u_0(y)dy \end{aligned}$$

provided B is large enough.

An explicit discretization typically takes the form,

$$u_j^n = u(x_j, t_n), \quad t_n = n\Delta t,$$

$$x_j = (j_1\Delta x1, ..., j_d\Delta x_d),$$

$$u^{n+1} = Au^n,$$

$$u^0 = u_0,$$

$$\Delta t = \text{const} \cdot |\Delta x|^2.$$

In the one dimensional case the matrix A is N by N ($N = 2^m$) with the number of nonzero elements in each row and column bounded by a constant. So the overall complexity for computing the solution at time equals to $\mathcal{O}(1)$ is $\mathcal{O}(N^3)$. Using the closed form $u^n = A^n u^0$, the solution can be computed in $\log(N)$ steps by repeated squaring of A : $A, A^2, A^4, \ldots, A^{2^m}$.

Now we are interested in the solution at a single point, say u_i^n. We only need $\Phi_i u^n$ where Φ_i is a projection,

$$\Phi_i(k,j) = \begin{cases} 1, & \text{if } k = j = i, \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

$$\Phi_i u = (0, 0, \ldots, u_i, 0, \ldots, 0) \tag{3.5}$$

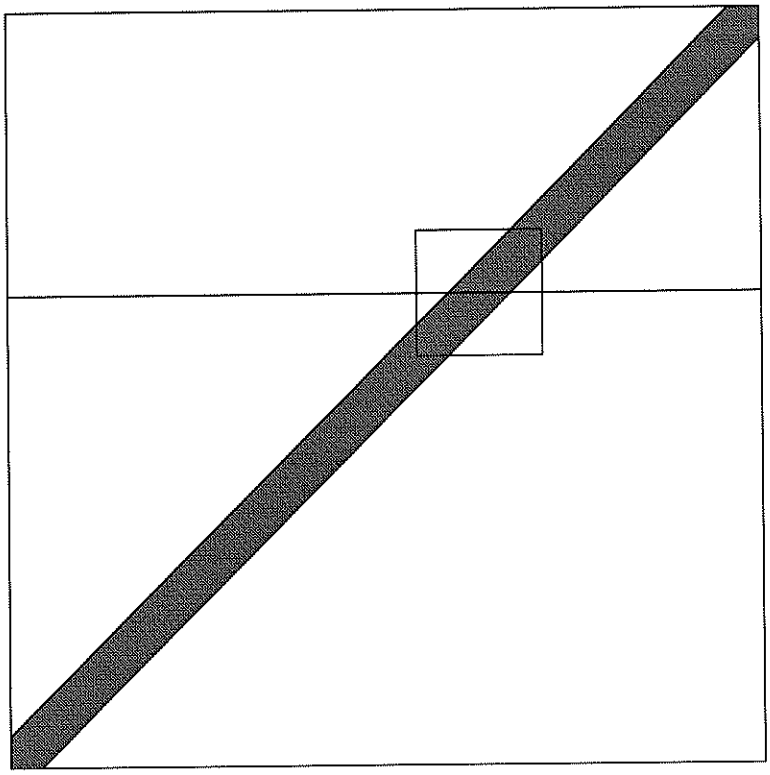

Figure 3.1: Points needed in A for calculating 1-point solution

If the matrices A^{2^i} stay banded with a fixed bandwidth B for all $1 \leq i \leq m$,

then there are only $2B$ nonzero numbers on the ith line of A^{2^i}. Hence

$$\Phi_i A^{2^k} = \Phi_i A^{2^k} \Phi_{i,B} \text{ for } k = 1, 2, ..., m, \tag{3.6}$$

where $\Phi_{i,B}$ is a wider projection,

$$\Phi_{i,B}(k,j) = \begin{cases} 1, & \text{if } k = j \text{ and } |j-i| < B, \\ 0, & \text{otherwise.} \end{cases}$$

$$\Phi_i A^n = \Phi_i \Phi_{i,B} A^n$$

So we have

$$\begin{aligned} \Phi_i A^n &= \Phi_i \Phi_{i,B} A^n \\ &= \Phi_i \Phi_{i,B} A^n \Phi_{i,B} \\ &= \Phi_i \Phi_{i,B} A^{n/2} A^{n/2} \Phi_{i,B} \\ &= \Phi_i \Phi_{i,B} A^{n/2} \Phi_{i,B} A^{n/2} \Phi_{i,B} \\ &= \Phi_i (\Phi_{i,B} A^{n/2} \Phi_{i,B})^2 \\ &= ... \\ &= \Phi_i (\Phi_{i,B} A \Phi_{i,B})^n \end{aligned} \tag{3.7}$$

Therefore we can get $\Phi_i A^n$ by computing $\Phi_{i,B} * A * \Phi_{i,B}$, $(\Phi_{i,B} A \Phi_{i,B})^2$,... , $(\Phi_{i,B} A \Phi_{i,B})^{2^m}$, i.e. by squaring only a piece of A, and the complexity for computing the solution at a point is $\mathcal{O}(\log(n))$.

All this is based on the assumption that the A^{2^i} stay banded with a fixed bandwidth B for all $1 \leq i \leq m$, which is not true. A^n is an approximation of

$G(x, y, n\Delta t)$, and the latter "spreads out" when n becomes larger. However, it is already proved in **[EnOZh]** that the wavelet transform of A^n does stay banded in the "finger form" as n increases, and the bandwidth is independent of n. This suggests that we use the idea in the wavelet transform of A.

Let us denote the wavelet transform of u as $\hat{u} = Wu$, and the wavelet transform of A as $\hat{A} = WAW^{-1}$.

Because the wavelets each have compact support, we only need $\mathcal{O}(\log(N))$ elements in $\hat{u}^n$ to get u_i^n. We call this projection P_i.

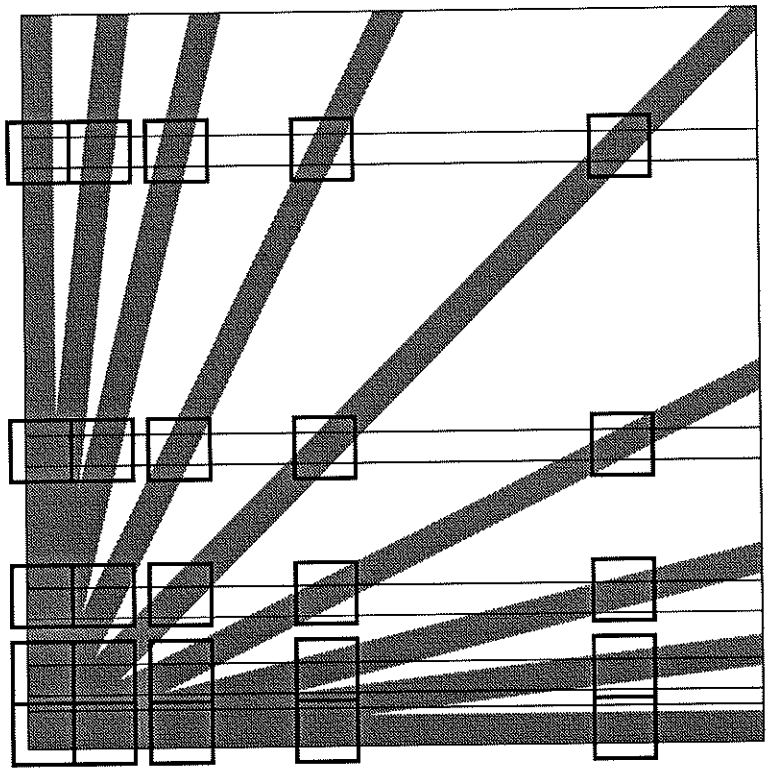

Figure 3.2: Points needed in $\hat{A}$ for calculating 1-point solution

$$P_i(j_1, j_2) = \begin{cases} 1, & \text{if } j_1 = j_2 \text{ and } |j_1 - i/2^p| < \text{constant}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.8}$$

$$\hat{u}^n = \hat{A}^n \hat{u}^0 \tag{3.9}$$

$$\Phi_i u^n = \Phi_i W^{-1} \hat{A}^n \hat{u}^0 \tag{3.10}$$

$$= \Phi_i W^{-1} P_i \hat{A}^n \hat{u}^0 \tag{3.11}$$

If each row of $\hat{A}^{2^k}(j_1, j_2)$ only has nonzero elements around $|j_2 - i/2^p| <$ constant, we can find a wider projection $P_{i,B}$,

$$P_i \hat{A}^{2^k} = P_i \hat{A}^{2^k} P_{i,B}. \tag{3.12}$$

It follows that

$$P_i \hat{A}^n = P_i (P_{i,B} \hat{A} P_{i,B})^n. \tag{3.13}$$

Figure 3.2 shows an example of the projection P_i. The above formula corresponds to taking out only those elements in the small squares and computing the squarings of only those parts.

3.2 Estimate for the standard form

The Algorithm is based on the assumption that $\hat{A}^{2^k}(j_1, j_2)$ only has nonzero elements around $|j_2 - i/2^p| <$ constant. Those positions correspond to the small

squares in Figure. 3.2.

However, as we can see in Figure 2.1 the finger form covers the full length of the bottom several rows of $\hat{A}$. Thus, it might be a problem to construct our $P_{i,B}$, which corresponds to a cut in the column direction. These bottom rows correspond to those $\int\int G(x,y)\psi_I(y)\psi_J(x)dxdy$ where $|J|$ is quite large. In the following section we are going to prove that $\int\int G(x,y)\psi_I(y)\psi_J(x)dxdy$ is small when $|I|/|J|$ is small.

In this section φ and ψ refer to the averaging and differencing functions in the wavelet analysis.

[Theorem] Suppose K is a function satisfying the following conditions:

1. size and smoothness estimates:

$$|K(x,y)| \leq C|x-y|^{-1} \quad \text{if } x \neq y \tag{3.14}$$

$$|\partial_x^M K(x,y) + \partial_y^M K(x,y)| \leq C|x-y|^{-(M+1)} \quad \text{if } x \neq y$$

2. weak-boundedness property:

$$| < K\varphi_I, \varphi_{I'} > | \leq C \quad \text{for } |I| = |I'| \tag{3.15}$$

3.

$$\int < K(x,y), \psi_I(y) > dx = 0 \tag{3.16}$$

Then we have

$$| < K\psi_I, \psi_J > | \leq C(\frac{|I|}{|J|})^{1/2+M} \frac{|I|^{1+M}}{|I|^{1+M} + \text{dist}(I,J)^{1+M}} \tag{3.17}$$

The proof of this theorem needs the following observation:

[Lemma] Let $S(x)$ be such that

$$1)\ |< S, \varphi_k >| \leq C/(1+|k|^{M+1}), \text{where } \varphi_k(x) = \varphi(x-k),$$

$$2)\ \int_{-\infty}^{\infty} S(x)dx = 0.$$

Then if $J = [k2^j, (k+1)2^j]$ and $j > 0$, we have

$$|< S, \psi_J >| \leq c2^{-j(M+\frac{1}{2})}/(1+|k|^{M+1}). \tag{3.18}$$

Proof:

ψ_J can be written as a linear combination of φ_n; i.e.,

$$\begin{aligned}\psi_J &= \sum_n \varphi_n < \psi_J, \varphi_n > \\ &= \sum_{n\in[k2^j,(k+M)2^j]} \varphi_n < \psi_J, \varphi_n > .\end{aligned}$$

Thus

$$\begin{aligned}|< S, \psi_J >| &= |\sum_n < S, \varphi_n >< \psi_J, \varphi_n >| \\ &\leq \sum_{n\in[k2^j,(k+M)2^j]} \frac{C}{1+|n|^{M+1}} < \psi_J, \varphi_n > .\end{aligned}$$

If the interval $[k2^j, (k+M)2^j]$ is away from the origin, we have

$$\begin{aligned}|< S, \psi_J >| &\leq \frac{C}{1+(\bar{k}\cdot 2^j)^{M+1}} \sum_n < \psi_J, \varphi_n >, \quad \bar{k} = \min(|k|, |k+M|) \\ &= \frac{C\cdot 2^{j/2}}{1+(\bar{k}\cdot 2^j)^{M+1}} \\ &\leq \frac{C\cdot 2^{-j(M+1/2)}}{1+|k|^{M+1}}\end{aligned}$$

The condition $\int_{-\infty}^{\infty} S(x)dx = 0$ implies the estimates is true when $[k2^j, (k+M)2^j]$ intersects with the origin, because as j becomes larger, $< S, \psi_J >$ acts as a wider average of S.

Now we are ready to prove the theorem:

After rescaling we apply the above observation to $S = K\psi_I$. Suppose the wavelets have M vanishing moments. From the weak boundedness property and the size and smoothness estimates of K we know that the estimates for $| < K\psi_I, \varphi'_I > |$ in Equation (2.15) is true:

$$| < K\psi_I, \varphi_{I'} > | \leq \frac{C|I|^{M+1}}{|I|^{M+1} + \text{dist}(I, I')^{M+1}} \quad \text{for } |I| = |I'| \tag{3.19}$$

Substitute this into the Lemma we have

$$| < K\psi_I, \psi_J > | \leq C(\frac{|I|}{|J|})^{1/2+M} \frac{|I|^{1+M}}{|I|^{1+M} + \text{dist}(I, J)^{1+M}}. \tag{3.20}$$

A similar estimate can be found in **[Meyer]**.

So if K satisfies the conditions of the theorem, then we know that $\hat{K}_{IJ}$ is small whenever $|I|/|J|$ is small, $|I|$ is small or $\text{dist}(I, J)$ is large. The number of significant elements of $\hat{K}$ is asymptotic to N.

Now let us go back to the parabolic equation (3.1)

$$L(x, t, \partial_x, \partial_t)u \equiv (\partial_t - a_{ij}(x)\partial_{ij} + a_i(x)\partial_i + a_0(x))u = 0. \tag{3.21}$$

Does the Green's function for this equation satisfies the conditions in the theorem? It is easy to check that the Green's function satisfies the size and smoothness estimate (3.14) and the weak-boundedness property (3.15). However, it doesn't

satisfy condition (3.16) in general. If the equation can be written in conservation form $u_t = \frac{\partial}{\partial x}F(x,t,\frac{\partial}{\partial x})u$ and with appropriate boundary conditions, then the Green's function satisfies $\int G(x,y,t)dx = constant$, independent of y (also t), and consequently $\int G(x,y,t)\psi_I(y)dydx = 0$. Many equations can be written in conservation form or can be transformed into conservation form by making an appropriate change of variables.

A^{2^k} is an approximation of the Green's function. Therefore the number of significant elements of $\hat{A}^{2^k}$ is asymptotic to N. They lie on diagonals $i/j = 2^k$ for k's with small absolute value. It is already proved in **[EnOZh]** that the constant in (3.19) is independent of the power of A; thus so is the constant in (3.17) .

Thus we can find a B such that $P_i\hat{A}^{2^k} \approx P_i\hat{A}^{2^k}P_{i,B}$ and by adjusting B the error can be controlled to be less then any threshold.

We obtain $P_i\hat{A}^n$ by squaring $P_{i,B} * \hat{A} * P_{i,B}$. There are $\mathcal{O}(\log^2(N))$ elements in $P_{i,B} * \hat{A}^{2^k} * P_{i,B}$; Thus each step requires $\mathcal{O}(\log^3(N))$ operations. Finally we have established that the complexity for computing the solution at a point is $\mathcal{O}(\log^4(N))$.

Here is a more detailed description of the algorithm:

Algorithm 1 (The 1-point Algorithm) An explicit finite-difference method typically looks like this:

$$v^n = Av^{n-1}$$

[Step 1] Construct the matrix A and compute $\hat{A}$, the wavelet transform of A.

[Step 2] We need to calculate the solution at a particular point, Determine the projection P according to the location of the point. Construct $B = P\hat{A}P$ from $\hat{A}$, and compute B^n using the repeated squaring idea: $B, B^2, B^4, ..., B^n$.

[Step 3] Now we are ready to calculate the solution at this point:

1. Calculate the wavelet transform of the initial data v^0, which we call $\hat{v^0}$.
2. $\hat{v^n} = B\hat{v^0}$
3. Do the inverse transform to obtain v^n.

CHAPTER 4

Time Dependent Coefficients

In the previous chapter we described a fast method for solving parabolic equations with time-independent coefficients. It is clear that in the case of time dependent coefficients the method of repeated squaring is no longer valid. However, we can still achieve big savings over the standard methods if the solution at certain points is needed for many different initial values. This kind of situation does happen, for example when calculating the stock option prices using the Black-Scholes Equation, where the volatility coefficient is allowed to depend on time as well as stock price. Of course the standard Black-Scholes model can be transformed to constant coefficients, but other important models are being developed,

4.1 Black-Scholes equation

There are two basic types of options. A *call option* gives the holder the right to buy the underlying asset by a certain date for a certain price. A *put option* gives the holder the right to sell the underlying asset by a certain date for a certain price. The date is called *expiration date* and the price is called *strike price*. Usually several options with different expiration dates or different strike prices are offered

for one stock.

Suppose an investor buys 100 call options of a stock with a strike price of \$50, the expiration date of the option is 2 months, and the price of the option is \$4 per share. Suppose in 2 months the stock price rises to \$60, he can exercise the option, i.e., buy the stock at \$50 per share. If he sells the stocks immediately, he can make a profit of \$600, taking into account the price of the options.

The Black-Scholes equation governs the price change of stock options. A European put option satisfies the following equation

$$P_t + rSP_S + \frac{1}{2}\sigma^2 S^2 P_{SS} = rP \tag{4.1}$$

with initial condition

$$P = \max(X - S, 0) \qquad \text{at } t = T. \tag{4.2}$$

Here is a description of the variables:

P: Value of European put option to sell one share

S: Stock price

t: Time

X: Strike price of option

T: Time of expiration of option

r: Risk free interest rate

σ: Volatility of stock price

The solution at $S =$ *current stock price* and $t = 0$ is needed for calculating the option prices. Calculating option values for different strike prices corresponds to

using different initial values.

In the standard Black-Scholes model σ and r are both constants. In some more sophisticated models r might depend on time and σ might depend on both time and stock price.

Suppose we use N grid points in our finite difference method. If $T = \mathcal{O}(1)$, the complexity of calculating the solution at $t = 0$ using an explicit method is $\mathcal{O}(N^3)$, because of the parabolic term $\frac{1}{2}\sigma^2 S^2 P_{SS}$. We can reduce it to $\mathcal{O}(N^2)$ by using implicit methods, such as the Crank-Nicolson method. But if we want to calculate the solution again and again using different initial values, each solution will cost the same $\mathcal{O}(N^3)$ or $\mathcal{O}(N^2)$ time.

The new algorithms we shall describe can compute solutions for different initial conditions with a very small cost after a quite expensive preparation time. The preparation cost can be reduced in some important special cases.

4.2 The basic algorithm

The basic algorithm is the following:

Algorithm 2 (The Basic Algorithm) Use an implicit method such as Crank-Nicolson. It looks like this:

$$A(n)P^n = B(n)P^{n-1} \tag{4.3}$$

[Step 1] Compute $K^n = A(n)^{-1}B(n)$ and then compute the wavelet transform of K^n for every time level. Store $\hat{K}^n$, the wavelet form of K^n , in files. Note that the final solution P^N would be simply $K^N K^{N-1}...K^1 P^0$.

[Step 2] This is used when we need to calculate the solution at a particular point. As in the time-independent coefficient case, only $P\hat{K}^n P$, a banded version of $\hat{K}^n$ is needed. The banded version has $\mathcal{O}(\log^2 N)$ significant entries. We load them from the file we saved in step 1 and calculate $P\hat{K}P = (P\hat{K^N}P)(P\hat{K^{N-1}}P)...(P\hat{K^1}P)$.

[Step 3] Now we are ready to calculate the solution at this point using all kinds of initial data.

1. Calculate the wavelet transform of the initial data P^0, which we call $\hat{P^0}$.
2. $\hat{P^N} = \hat{K}\hat{P^0}$
3. Do the inverse transform for $\hat{P^N}$ and obtain P^N.

[Complexity Count]

[Step 1] The preparation stage.

1. The complexity of LU factorization for each time step is $\mathcal{O}(N)$, because $L(n)$ is banded.
2. The complexity of obtaining $K(n) = L(n)^{-1}L'(n)$ is $\mathcal{O}(N^2)$.
3. Calculating $\hat{K^n}$ costs $\mathcal{O}(N\log(N))$.

Thus the cost for each time step is $\mathcal{O}(N^2 + N\log(N))$, and the total cost is $\mathcal{O}(N^3 + N^2\log(N))$.

[Step 2] The cost is $\mathcal{O}(N\log^3(N)$ because the cost of each time step is the cost of calculating a $\mathcal{O}(\log(N))$ by $\mathcal{O}(\log(N))$ matrix multiplication, which is $\mathcal{O}(\log^3(N))$.

[Step 3] This step consists of the wavelet transform and inverse transform of a N-vector and a multiplication of a $\mathcal{O}(\log(N))$ by $\mathcal{O}(\log(N))$ matrix and a $\mathcal{O}(\log(N))$ vector. The cost is $\mathcal{O}(N + \log^2(N))$ and is actually very low as will be shown with numerical experiments below.

[Remark 1] Step 1, the main preparation stage is the most expensive step of this algorithm, especially the wavelet transform part. This step could be very expensive compare to standard methods. However, if we need the solution of lots of different initial values at very few points this algorithm is more efficient than the usual methods.

[Remark 2] If we change step 2 to computing the full version of $\hat{K}$ instead of the banded version, we can obtain the matrix for computing the whole solution for any initial values. The extra cost is $\mathcal{O}(N^2\log(N))$, because each step costs $\mathcal{O}(N\log(N))$, as explained in **[EnOZh]**. This could be a good algorithm when the whole solution of a equation is needed for lots of different initial conditions. But this extra cost is actually very expensive (similar to the cost of [step 1] and it is unnecessary if all we need is the solution at a few points.

4.3 Variations

The algorithm we described in the previous section has an attractive operation count but the preparation stage, especially the wavelet transform part, is very time consuming. This inspires us to find a way to cut the number of wavelet transforms. This is indeed possible if the equation has some special properties. We can significantly reduce the preparation time in those cases.

Let us first consider the case that the parabolic term (second order term) can be written as a product of a function of time and the space dependent part. In this case we can change the scaling of time and make the second order term time-independent. Suppose the equation is

$$u_t = \alpha(t)\beta(x)u_{xx} + \text{lower order terms.}$$

We can make a change of variable

$$\frac{d\tau}{dt} = \alpha(t).$$

The resulting equation for τ would have a time-independent second order term:

$$u_\tau = \frac{dt}{d\tau}u_t = \beta(x)u_{xx} + \text{lower order terms.}$$

Now if we use a semi-implicit method, i.e. one, which only is implicit for the second order term, we only need to compute the inverse of the implicit part once (because it is time-independent), thus significantly decrease the preparation time.

Suppose after the time scaling the equation is:

$$u_t = a(x)u_{xx} + b(x,t)u_x + c(x,t)u. \tag{4.4}$$

A semi-implicit scheme

$$(I - \frac{1}{2}a(x)D_+D_-)u^{n+1} = (I + \frac{1}{2}a(x)D_+D_- + b(x,t)D_0 + c(x,t))u^n$$

would have a matrix form of

$$L_1 u^{n+1} = (L_2 + B^n D_0 + C^n)u$$

where

L_1: matrix form of $I - \frac{1}{2}a(x)D_+D_-$,

L_2: matrix form of $I + \frac{1}{2}a(x)D_+D_-$,

D_0: matrix form of difference operator D_0,

B^n, C^n: diagonal matrices, matrix form of $b(x,t)$ and $c(x,t)$.

Let K^n be the matrix that computes u^{n+1} from u^n:

$$u^{n+1} = K^n u^n.$$

Then we have

$$K^n = L_1^{-1}L_2 + L_1^{-1}B^n D_0 + L_1^{-1}C^n.$$

The wavelet form of K^n is:

$$\hat{K^n} = WKW' = WL_1^{-1}L_2W' + WL_1^{-1}B^n D_0 W' + WL_1^{-1}C^n W'$$

We can save the preparation time in each time step by calculating the following time-independent components in advance: $WL_1^{-1}L_2W'$, WL_1^{-1}, D_0W' and W'.

At each time step, we need to calculate

$$\begin{aligned} P\hat{K(n)}P &= PWKW^{-1}P \\ &= PWL_1^{-1}L_2'P + PWL_1^{-1}B(n)D_0W'P + PWL_1^{-1}C(n)W'P \end{aligned}$$

which cost $\mathcal{O}(N\log^2(N))$.

Here is a more detailed description:

Algorithm 3 (Variation 1)

[Step 1] Compute $WL_1^{-1}L_2W'$, WL_1^{-1}, D_0W' and W'.

[Step 2] Calculate

$$\begin{aligned} P\hat{K}^nP &= PWK^nW^{-1}P \\ &= PWL_1^{-1}L_2W'P + PWL_1^{-1}B^nD_0W'P + PWL_1^{-1}C^nW'P \end{aligned}$$

and then compute

$$P\hat{K}P = (P\hat{K}^NP)(PK^{\hat{N-1}}P)...(P\hat{K}^1P).$$

[Step 3] (Same as in the Basic Algorithm) Now we are ready to calculate the solution at the point using all kinds of initial data.

1. Calculate the wavelet transform of the initial data P^0, which we call $\hat{P}^0$.

2. $\hat{P^N} = \hat{K}\hat{P^0}$.

3. Do the inverse transform for $\hat{P^N}$ and obtain P^N.

If the $b(x,t)$ and $c(x,t)$ in equation 4.4 can be separated into a function of t and a time-independent part, i.e.,

$$u_t = a(x)u_{xx} + b_1(t)b_2(x)u_x + c_1(t)c_2(x)u$$

Then we can further save the preparation time because each $\hat{K^n}$ is just a linear combination of several time-independent terms:

$$\hat{K^n} = L_1^{-1}L_2 + b_1(t)L_1^{-1}B_2 + c_1(t)L_1^{-1}C_2$$

Algorithm 4 (Variation 2)

[Step 1] Compute the wavelet form of $L_1^{-1}L_2$, $L_1^{-1}B_2$, and $L_1^{-1}C_2$. We denote them as W_1, W_2 and W_3.

[Step 2] Calculate

$$P\hat{K^n}P = PW_1P + b_1(t)PW_2P + c_1(t)PW_3P$$

and

$$P\hat{K}P = (P\hat{K^N}P)(PK^{\hat{N}-1}P)...(P\hat{K^1}P).$$

[Step 3] (Same as in Basic Algorithm) Now we are ready to calculate the solution at this point using all kinds of initial data.

1. Calculate the wavelet transform of the initial data P^0, which we call $\hat{P^0}$.

2. $\hat{P^N} = \hat{K}\hat{P^0}$.

3. Do the inverse transform for $\hat{P^N}$ and obtain P^N.

Consider equation

$$u_t = L(x, t, \frac{\partial}{\partial x})u. \tag{4.5}$$

If the linear operator $L(x, t, \frac{\partial}{\partial x})$ commutes with its indefinite integral,

$$\int_0^t L(x, t, \frac{\partial}{\partial x})dt, \tag{4.6}$$

we can replace the equation with

$$u_t = \bar{L}(x, \frac{\partial}{\partial x})u, \tag{4.7}$$

where $\bar{L}(x, \frac{\partial}{\partial x})$ is the average of $L(x, t, \frac{\partial}{\partial x})$ over t. For example, the solution for the equation

$$u_t = a(t)u_{xx} + b(t)u_x.$$

is the same as the solution of the equation

$$u_t = \bar{a}u_{xx} + \bar{b}u_x$$

where $\bar{a} = \frac{1}{T}\int_0^T a(t)dt$ and $\bar{b} = \frac{1}{T}\int_0^T b(t)dt$. So we can solve it as a time-independent problem using the method of repeated squaring.

Algorithm 5 (Commuting Case) In equation $u_t = L(x, t, \frac{\partial}{\partial x})u$, if the linear operator $L(x, t, \frac{\partial}{\partial x})$ commutes with its indefinite integral, we can replace it with a time-independent equation and solve it using the algorithm for time-independent equations.

CHAPTER 5

Numerical Results

A C++ program has been written implementing the algorithms of the preceding section, and numerical experiments have been performed on Sparc 10 stations. We calculated each example in 2 ways: using a standard finite difference scheme and a wavelet scheme. The error and CPU time of each scheme is presented in the tables.

Example 1. The Heat Equation

Consider the following parabolic problem:

$$u_t = \partial_x(a(x)\partial_x u) \tag{5.1}$$

$$u(x,0) = u_0(x)$$

with periodic boundary conditions and the following choices:

$$a(x) = 0.5 + 0.25\sin(2\pi x)$$

$$u_0(x) = \sin(4\pi x)$$

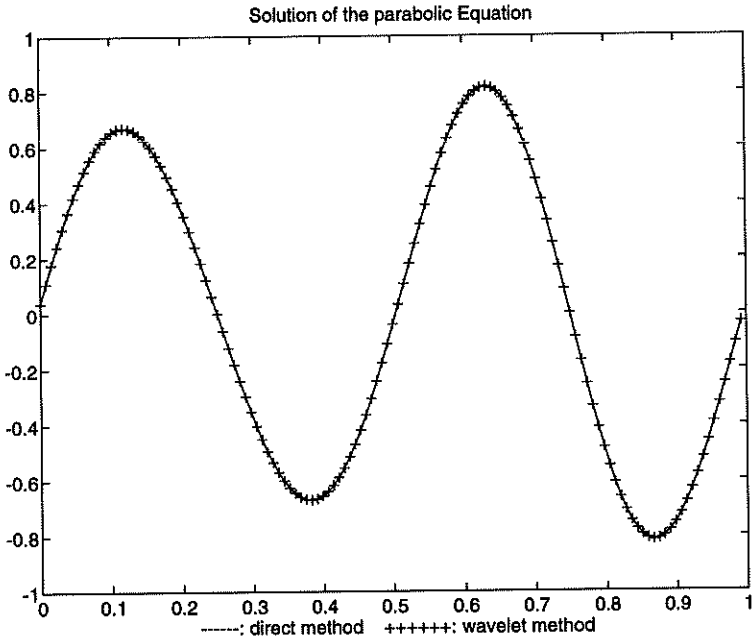

Figure 5.1: Parabolic Equation: Solutions by Direct Method and by Wavelet Method

A central difference scheme is used with $\Delta t = 0.25h^2$. Numerical results and the number of elements in $\hat{A}$ that are used to obtain the results are shown in table 5.1 and 5.2. It is clear that the results obtained by tracing only $\mathcal{O}(\log^2(N))$ elements are comparable with those obtained by direct method. The solution is shown in Figure 5.1. The wavelet transform of the matrix A^n at $t = 0$ and $t = 1/32$ is shown in Figure 5.2 and Figure 5.3 respectively.

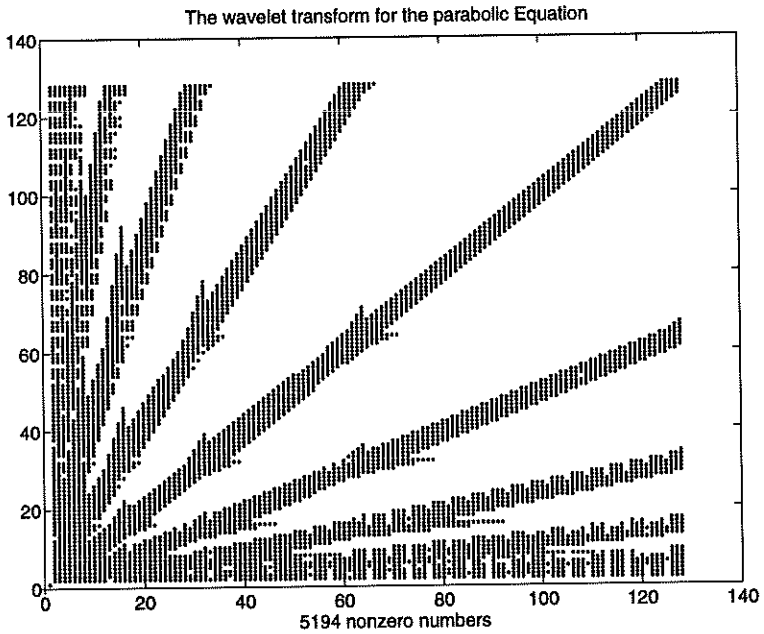

Figure 5.2: Parabolic Equation: Elements That are Greater Than 10^{-6} in $\hat{A}$

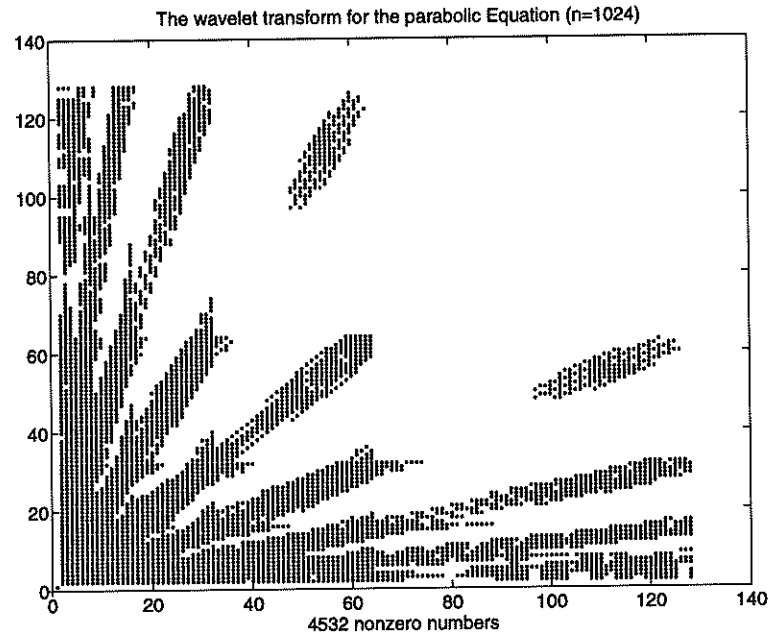

Figure 5.3: Parabolic Equation: Elements That are Greater Than 10^{-6} in $\hat{A}^n$

grids	$\|\|u_d - u_e\|\|_{\text{inf}}$	elements used	$\|\|u_w - u_e\|\|_{\text{inf}}$
128	$1.49 \cdot 10^{-4}$	50^2	$1.4854 \cdot 10^{-4}$
256	$3.7134 \cdot 10^{-5}$	59^2	$3.6378 \cdot 10^{-5}$
512	$9.2353 \cdot 10^{-6}$	68^2	$1.6767 \cdot 10^{-5}$

Table 5.1: Parabolic Equation: Errors of different methods, t=1/32

grids	$\|\|u_d - u_e\|\|_{\text{inf}}$	elements used	$\|\|u_w - u_e\|\|_{\text{inf}}$
128	$4.3279 \cdot 10^{-5}$	50^2	$5.5503 \cdot 10^{-5}$
256	$1.1030 \cdot 10^{-5}$	59^2	$5.617 \cdot 10^{-5}$
512	$2.9535 \cdot 10^{-6}$	68^2	$6.2006 \cdot 10^{-5}$

Table 5.2: Parabolic Equation: Errors of different methods, t=1/512

Example 2. Black-Scholes equation

Here we solve the Black-Scholes Equation

$$P_t + rSP_S + \frac{1}{2}\sigma^2 S^2 P_{SS} = rP \tag{5.2}$$

$$\sigma = 0.5$$

$$r = 0.05$$

$$0 \leq S \leq \infty$$

with initial condition

$$P = \max(X - S, 0) \qquad \text{at } t = T \tag{5.3}$$

Solution needed at $t = 0$.

The exact solution is:

$$d = \frac{\log(X) - \log(S) - T(r - \frac{1}{2}\sigma^2)}{\sigma\sqrt{2T}} \tag{5.4}$$

$$P = \frac{1}{2}Xe^{-rT}\,\mathrm{Erf}(d) - \frac{1}{2}S\,\mathrm{Erf}(d - \sigma\sqrt{T/2})$$

where

$$\mathrm{Erf}(z) = \frac{2}{\sqrt{\pi}}\int_{-\infty}^{z} e^{-t^2}dt$$

We compute the solution in the interval (0,5) at $T = 0.25$, using 256 grid points. Table 5.3 shows the error of direct method and wavelet method, and the number

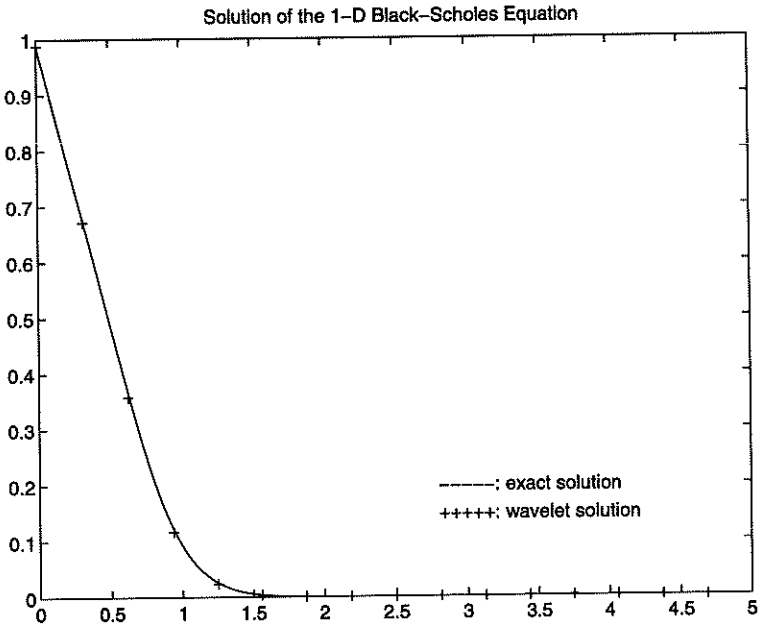

Figure 5.4: Black-Scholes Equation: The solution by Wavelet Method and the exact solution

of elements traced in A^n. The solution is shown in Figure 5.4. The wavelet transform of the matrix A^n at $t = 0$ and $t = 0.25$ is shown in Figure 5.5 and Figure 5.6 respectively.

grids	$\|\|u_d - u_e\|\|_{\text{inf}}$	elements used	$\|\|u_w - u_e\|\|_{\text{inf}}$
128	$1.5853 \cdot 10^{-4}$	54^2	$1.55518 \cdot 10^{-4}$
256	$9.98835 \cdot 10^{-5}$	65^2	$9.90298 \cdot 10^{-5}$

Table 5.3: Black-Scholes Equation: Errors of different methods, t=0.25

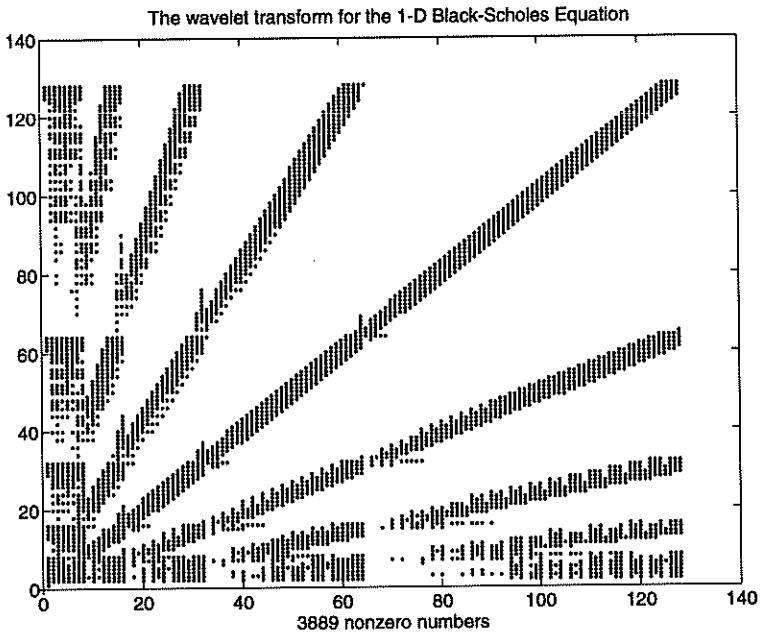

Figure 5.5: Black-Scholes Equation: Elements That are Greater Than 10^{-6} in $\hat{A}$

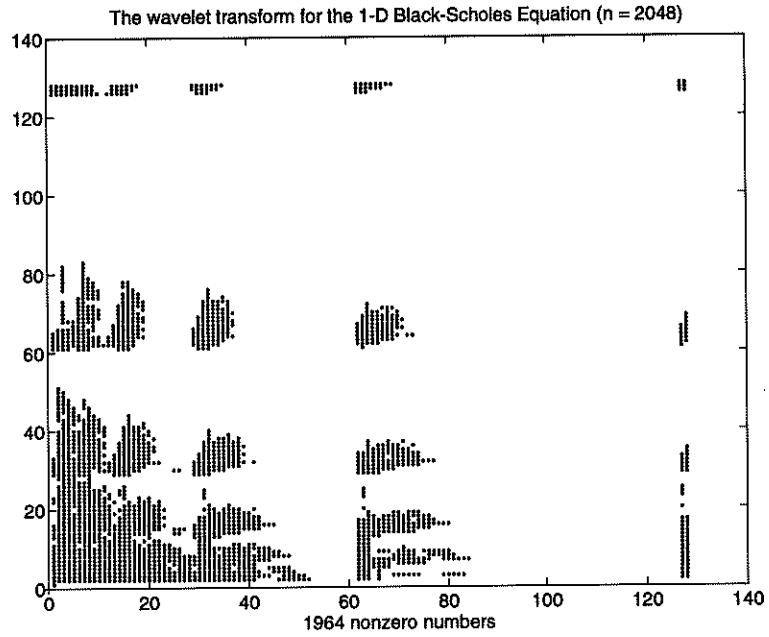

Figure 5.6: Black-Scholes Equation: Elements That are Greater Than 10^{-6} in $\hat{A}^n$

Example 3. 2D parabolic equation

Consider the following parabolic problem:

$$u_t = a_{11}u_{xx} + 2a_{12}u_{xy} + a_{22}u_{yy} \tag{5.5}$$

$$u(x, y, 0) = u_0(x, y)$$

with periodic boundary conditions and the following choices:

$$a11(x, y) = 0.5 + 0.25\sin(2\pi y)$$

$$a12(x, y) = 0.115\sin(2\pi y)\cos(2\pi x)$$

$$a22(x, y) = 0.5 + 0.25\cos(2\pi x)$$

$$u_0(x, y) = \sin(4\pi y) + \cos(8\pi y)$$

A central difference scheme is used on a 32 by 32 grid. The maximum error of the wavelet method is 0.002377. The wavelet transform of the matrix A^n at $t = 0$ and $t = 0.01$ is shown in Figure 5.7 and Figure 5.8 respectively.

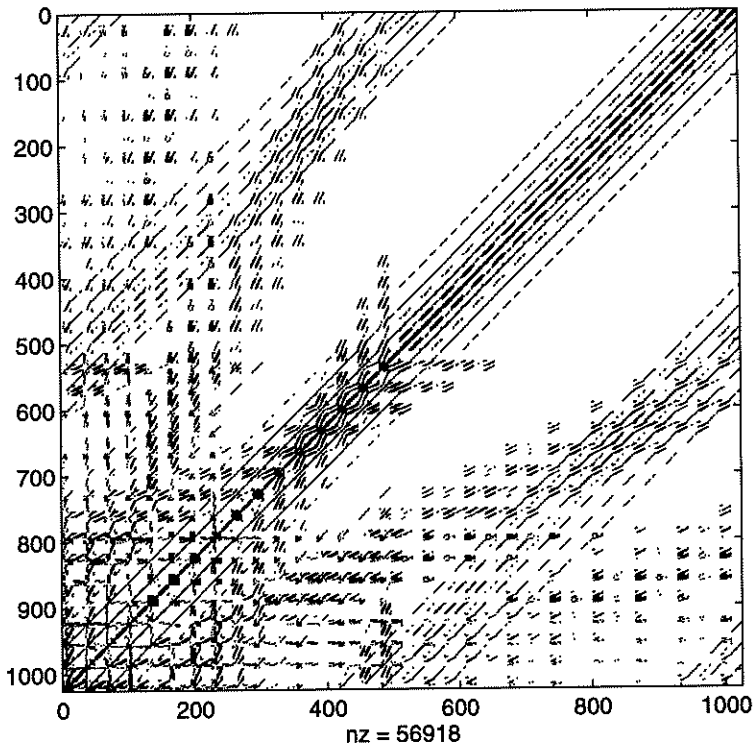

Figure 5.7: 2D Parabolic Equation: Elements That are Greater Than 10^{-4} in $\hat{A}$

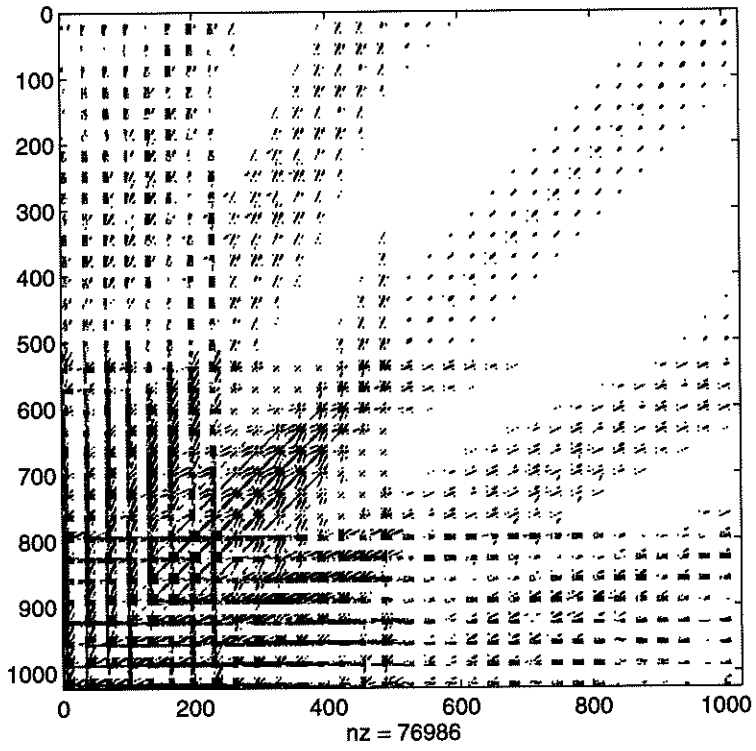

Figure 5.8: 2D Parabolic Equation: Elements That are Greater Than 10^{-6} in $\hat{A}^n$

Example 4. Black-Scholes Equation, Time-dependent Coefficients

In this example, we solve the Black-Scholes Equation

$$P_t + rSP_S + \frac{1}{2}\sigma^2 S^2 P_{SS} = rP \tag{5.6}$$

with two choices of time dependent volatility function

$$\sigma_1(S,t) = F(S)G(t) \quad \text{where}$$

$$G(t) = 1 - 0.2t \quad \text{for } 0 \le t \le 1$$

$$F(S) = \begin{cases} 0.3 & \text{if } 0 \le S \le 50, \\ 0.3 - 0.2\frac{S-50}{100} & \text{if } 50 \le S \le 150, \\ 0.1 & \text{if } S \ge 150. \end{cases}$$

$$\sigma_2(t) = 0.2G(t)$$

and the following parameters:

stock price $= S_0 = 100$,

strike price $= K = 100$,

interest rate $= r = 0.10$,

dividend yield $= \delta = 0$,

time to expiration $= T = 1$.

Standard Crank-Nicolson Scheme is used:

$$L = -rSD_0 - \frac{1}{2}\sigma^2 S^2 D_+ D_- + r \tag{5.7}$$

$$(I - \frac{1}{2}L)P^{n+1} = (I + \frac{1}{2}L)P^n \tag{5.8}$$

We compute solution in domain $[0, 200]$ using 128, 256 and 512 grid points. The basic algorithm is used. The error and time for σ_2 is shown in table 5.4 and table 5.5. The error for σ_1 is shown in table 5.6.

grids	error of normal method	error of wavelet method
128	0.0172265	0.0146943
256	0.00683063	0.00479132
512	0.002970085	0.00130449

Table 5.4: Time-dependent Black-Scholes Equation: Error of Basic Algorithm

grids	normal method	wavelet method	preparation
128	0.316634	0.03332	42.3316
256	1.41645	0.04999	245.24
512	5.8152	0.07984	1393.92

Table 5.5: Time-dependent Black-Scholes Equation: Computation Time of Basic Algorithm

grids	error(compare with direct method)
128	6.68611e-5
256	1.1706e-4
512	1.8336e-4

Table 5.6: Time-dependent Black-Scholes Equation: Error of Basic Algorithm

Example 5. Parabolic Equation, Time-dependent Coefficients

Consider the parabolic problem

$$u_t = a(x)u_{xx} + b(t)c(x)u_x \tag{5.9}$$

$$u(x,0) = u_0(x)$$

with zero boundary conditions and the following choices:

$$a(x) = 0.1(1 + 0.5\sin(2\pi x))$$

$$b(t) = 1 + 0.5t$$

$$c(x) = 0.1\pi\cos(2\pi x)$$

$$u_0(x) = \sin(4\pi x)$$

Variation 2 is used with $\Delta t = \Delta x$. The error of direct method and the wavelet method is shown in Table 5.7. The time used for each method is shown in Table 5.8. The results obtained by Variation 2 are comparable with those obtained by direct method. We can see Variation 2 saves preparation time as compared to the basic algorithm (Please refer to Table 5.5).

grids	error(compare with direct method)	# of elements traced
128	1.52719e-5	58^2
256	2.50445e-5	72^2
512	3.57722e-5	86^2

Table 5.7: Time-dependent Parabolic Equation, Variation 2, Error

grids	normal method	wavelet method	preparation
128	0.41665	0.016666	9.59962
256	1.69993	0.033332	35.7819
512	7.13305	0.049998	122.278

Table 5.8: Time-dependent Parabolic Equation, Variation 2, Time

Example 6. Parabolic Equation, Commuting Case

Consider the following parabolic problem:

$$u_t = b(t)\partial_x(a(x)\partial_x)u \tag{5.10}$$

$$u(x,0) = u_0(x)$$

with zero boundary conditions and the following choices:

$$a(x) = 0.1(1 + 0.5\sin(2\pi x))$$

$$b(t) = 1 + 0.5t$$

$$u_0(x) = \sin(4\pi x)$$

Variation 3 is used with $\Delta t = \Delta x$. The error of the direct method and the wavelet method is shown in table 5.9. The time used for each method is shown in table 5.10. The results obtained by Variation 3 are comparable with those obtained by direct method. We can see that Variation 3 is very fast.

grids	error(compare with direct method)	# of elements traced
128	3.49771e-5	58^2
256	2.93186e-5	72^2
512	3.17191e-5	86^2

Table 5.9: Time-dependent Parabolic Equation, Commuting Case, Error

grids	normal method	wavelet method	wavelet transform
128	0.42364	0.566644	0.383318
256	1.67888	1.06662	1.51661
512	7.21105	1.83326	5.28312

Table 5.10: Time-dependent Parabolic Equation, Commuting Case, Time

CHAPTER 6

Conclusion

Compare to Fourier transforms, wavelet transforms have the advantage of space localization. This inspires us to investigate algorithms of obtaining local solutions.

In **[EnOZh]** the authors designed a fast method using repeated squaring of $\hat{A}$, where A is the matrix representation of a finite difference method and $\hat{A}$ is its wavelet form. Observing that for a local solution only a small number of entries in $\hat{A}$ needs to be calculated, we derived the algorithm for computing local solutions using the repeated squaring idea.

If the equation has time dependent coefficients, the method of repeated squaring is no longer valid. The matrix A is time dependent , so is its wavelet form. Explicit finite difference methods for parabolic equations has a very strict restriction on time steps; i.e., $dt/(dx)^2 < c$. We need to use $\mathcal{O}(n^2)$ time steps for solving the equation to $\mathcal{O}(1)$ time, where n is the resolution of space variable. This is a serious disadvantage for using wavelet methods to solve equations with time dependent coefficients, because we need to compute $\hat{A}$ in every time step, which is very expensive.

We can take much larger steps when using implicit methods. Only $\mathcal{O}(n)$ time steps are needed for computing the equation to $\mathcal{O}(1)$ time. Standard implicit

methods have the form of $Av^{n+1} = Bv^n$. We never compute A^{-1} explicitly, because it is a dense matrix. Instead, we solve a linear system using LU factorization at each time step. However, A^{-1} is sparse in wavelet representation. Thus we can construct its wavelet form explicitly and turn the method into an explicit one. We have

$$\hat{v}^{n+1} = C\hat{v}^n, \tag{6.1}$$

where

$$C = (WA^{-1}W')(WBW'), \quad \hat{v} = Wv. \tag{6.2}$$

The matrix C is sparse in the finger form. This is an explicit method without the strict time step restriction of standard explicit finite difference method. When calculating local solutions, we only need to compute a small number of entries in $\hat{v}^n$. We can compute only these entries thus save the computation time. This is the basic idea of our fast algorithms. This algorithm is also useful for computing the whole solution in the case that the vector $\hat{u}^n$ is sparse in the wavelet form.

A limitation of the wavelet methods is that they require larger storing space as compared to normal finite difference methods. This is because we need to construct explicitly the matrix representation of finite difference methods in wavelet form. This requires a lot more storage space as compared to the normal finite difference methods, even when we take advantage of the sparse structure of the matrix's wavelet form.

For the repeated squaring algorithm, the extension to higher dimension is quite straight forward. In fact this method is preferable for multidimensional problems, because of the simple fact that $\log(N^d) = d\log(N)$. An example is presented in chapter 4. We can also generalize the algorithms for the time-dependent coefficient case to higher dimension by using ADI methods.

Reference

[B] G. Beylkin, " On wavelet-based algorithms for solving differential equations" in "Wavelets: Mathematics and Applications", J. Benedetto and M. Frazier, eds., CRC Press, 1994, pp.449-466.

[BCoR] G. Beylkin, R. Coifman and V. Rokhlin, "Fast wavelet transforms and numerical algorithms I", *Comm. Pure Appl. Math.*,, **64**(1991), pp.141-184.

[CDJV] A. Cohen, I. Daubechies, B. Jawerth, and P. Vial, "Multiresolution analysis, wavelets and fast algorithms on an interval", C. R. Acad. Sci., ser. 1(1992).

[Dau] I. Daubechies, "Orthonormal basis of compactly supported wavelets", *Comm. Pure,Appl. Math.* **41**(1988), pp.909-996.

[Dau2] I. Daubechies, "Ten Lectures on wavelets", CBMS **61**(1992).

[EnOZh] B. Engquist, S. Osher, and S. Zhong, "Fast wavelet algorithms for linear evolution equations", *ICASE Report*, **92-14**(1992).

[GrS1] L. Greengard and J. Strain, "A fast algorithm for the evaluation of heat potentials", *Comm. Pure,Appl. Math.* **43**(1990), pp.949-964.

[GrS2] L. Greengard and J. Strain, "The fast Gauss transform", *SIAM I. Sci. Statist. Comput.,*, **12**(1991), pp. 79-94.

[Meyer] Y. Meyer, "Wavelets and operators" in "Analysis at Urbana", Vol. 1, E.Berkson, N.T.Peck, and J.Uhl, eds., London Math. Soc., Lecture Notes Series 137,1989, pp 256-365.