Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Physics 226 (2007) 1891–1901

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

On the numerical solution of the heat equation I: Fast solvers in free space

Jing-Rebecca Li [*], Leslie Greengard

INRIA-Rocquencourt, Projet POEMS, Domaine de Voluceau – Rocquencour, 78153 Le Chesnay Cedex, France

Received 11 April 2006; received in revised form 19 June 2007; accepted 19 June 2007
Available online 7 July 2007

Abstract

We describe a fast solver for the inhomogeneous heat equation in free space, following the time evolution of the solution in the Fourier domain. It relies on a recently developed spectral approximation of the free-space heat kernel coupled with the non-uniform fast Fourier transform. Unlike finite difference and finite element techniques, there is no need for artificial boundary conditions on a finite computational domain. The method is explicit, unconditionally stable, and requires an amount of work of the order $O(NM \log N)$, where N is the number of discretization points in physical space and M is the number of time steps. We refer to the approach as the fast recursive marching (FRM) method.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Heat equation; Free space; Unbounded domain; Integral representation; Spectral approximation

1. Introduction

The solution of the heat equation (the diffusion equation) in free space or in unbounded regions arises as a modeling task in a variety of engineering, scientific, and financial applications. While the most commonly used approaches are based on finite difference (FD) and finite element (FE) methods, these must be coupled to artificial (non-reflecting) boundary conditions imposed on a finite computational domain in order to simulate the effect of diffusion into an infinite medium. These boundary conditions are discussed, for example, in [5,14–17]. Here, we describe a mathematically much more straightforward approach, which we will refer to as the fast recursive marching (FRM) method. It is based on evaluating the exact solution of the governing equation, using convolution in space and time with the free-space Green's function. One advantage of this approach is that essentially no convergence theory is required. The error in the solution is simply the quadrature error in evaluating the solution. In the present paper, we restrict our attention to the simplest setting, namely the isotropic inhomogeneous heat equation in $\mathbb{R}^d$:

$$U_t(\mathbf{x},t) - \nabla^2 U(\mathbf{x},t) = f(\mathbf{x},t), \quad t > 0, \tag{1}$$

[*] Corresponding author. Tel.: +33 1 39 63 53 55; fax: +33 1 39 63 58 84.
E-mail address: jingrebecca.li@inria.fr (J.-R. Li).

0021-9991/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2007.06.021

in the absence of physical boundaries, subject to the initial condition

$$U(\mathbf{x},0) = U_0(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d. \tag{2}$$

The functions $f(\mathbf{x},t)$ and $U_0(\mathbf{x})$ are assumed to be compactly supported in the box $B = [-R/2, R/2]^d$, centered at the origin. We also assume that $f(\mathbf{x},t)$ and $U_0(\mathbf{x})$ are k-times differentiable: $f(\mathbf{x},t) \in C^k(B \times [0,T])$ and $U_0(\mathbf{x}) \in C^k(B)$. In subsequent works we will consider complex geometry and discontinuous data.

From standard potential theory [13,23], the solution can be written as

$$U(\mathbf{x},t) = \int_{\mathbb{R}^d} k(\mathbf{x}-\mathbf{y},t)U_0(\mathbf{y})\,\mathrm{d}\mathbf{y} + \int_0^t \int_{\mathbb{R}^d} k(\mathbf{x}-\mathbf{y},t-\tau)f(\mathbf{y},\tau)\,\mathrm{d}\mathbf{y}\,\mathrm{d}\tau, \tag{3}$$

where the fundamental solution for the heat equation in $\mathbb{R}^d$ is

$$k(\mathbf{x},t) := \frac{\mathrm{e}^{-\|\mathbf{x}\|^2/4t}}{(4\pi t)^{\frac{d}{2}}}, \quad t > 0. \tag{4}$$

We will refer to the first integral in (3) as an *initial potential* and the second integral as a *volume potential*. There is a substantial literature on Green's function methods for problems of diffusion (see for example [3]). However, straightforward discretization of the above integrals leads to an enormously expensive numerical scheme – the solution is dependent on the full space–time history of the diffusion process. With N points in the discretization of the domain and M time steps, it is easy to see that $O(N^2M + N^2M^2)$ work is required. Thus, the obvious advantages of the approach (stability, robustness, and the correctness of the far-field behavior) appear to be overwhelmed by the problems of cost.

In recent years, however, several fast algorithms have been developed [10–12,26] that lead to nearly optimal schemes, for which the work required is $O(MN\log N)$. Related methods can be found in [18,20,27]. They involve a fairly substantial amount of numerical and analytic machinery. In the present context, where we need to evaluate volume potentials with smooth data, a simpler method can be developed based entirely on the continuous Fourier transform. After outlining the algorithm itself, we illustrate its performance with some numerical examples from materials science, simulating dendritic solidification.

2. Fourier representation of the solution

While (3) describes the solution to the heat equations (1) and (2), significant advantage can be obtained by considering its Fourier transform. For this, we let

$$\hat{U}(\mathbf{s},t) = \int_{\mathbb{R}^d} \mathrm{e}^{\mathrm{i}\mathbf{s}\cdot\mathbf{x}} U(\mathbf{x},t)\,\mathrm{d}\mathbf{x}, \quad U(\mathbf{x},t) = \frac{1}{(2\pi)^d}\int_{\mathbb{R}^d} \mathrm{e}^{-\mathrm{i}\mathbf{s}\cdot\mathbf{x}}\hat{U}(\mathbf{s},t)\,\mathrm{d}\mathbf{s}. \tag{5}$$

It is obvious from (1), (2), and well known that $\hat{U}(\mathbf{s},t)$ satisfies the ordinary differential equation

$$\frac{\mathrm{d}\hat{U}}{\mathrm{d}t}(\mathbf{s},t) = -\|\mathbf{s}\|^2\hat{U}(\mathbf{s},t) + \hat{f}(\mathbf{s},t), \tag{6}$$

where

$$\hat{f}(\mathbf{s},t) = \int_{\mathbb{R}^d} \mathrm{e}^{\mathrm{i}\mathbf{s}\cdot\mathbf{x}} f(\mathbf{x},t)\,\mathrm{d}\mathbf{x}.$$

An elementary calculation shows that

$$\hat{U}(\mathbf{s},t) = \mathrm{e}^{-\|\mathbf{s}\|^2\Delta t}\hat{U}(\mathbf{s},t-\Delta t) + \Phi(\mathbf{s},t,\Delta t), \tag{7}$$

where

$$\Phi(\mathbf{s},t,\Delta t) = \int_{t-\Delta t}^{t} \mathrm{e}^{-\|\mathbf{s}\|^2(t-\tau)}\hat{f}(\mathbf{s},\tau)\,\mathrm{d}\tau. \tag{8}$$

Thus, in the Fourier domain, history dependence is no longer an issue; $\hat{U}(\mathbf{s},t)$ is simply *damped* and *updated* at each time step, as indicated in (7). While this is not a new observation, it is worth introducing some notation; we will refer to $\Phi(\mathbf{s},t,\Delta t)$ in (8) as the *update integral*.

Why, then, is this not the standard procedure for computing heat flow? The answer is that we need a quadrature rule for (8), we need to compute $\hat{f}(\mathbf{s},t)$ from $f(\mathbf{x},t)$, and we need to compute the inverse transform to obtain $U(\mathbf{x},t)$ from $\hat{U}(\mathbf{s},t)$. The first task is easy, the second is a matter of finding a suitable "fast algorithm", and the third is somewhat subtle. We will address these issues in order.

2.1. Quadratures for the update integral

The update integral (8) can be computed to high order accuracy using a standard product integration approach [4], that is to say, polynomial approximation of $\hat{f}(\mathbf{s},t)$ as a function of time, followed by analytic integration. Linear approximation of $\hat{f}(\mathbf{s},t)$ yields second order accuracy and the rule [11]

$$\Phi(\mathbf{s},t,\Delta t) = W_0(\mathbf{s},\Delta t)\hat{f}(\mathbf{s},t) + W_1(\mathbf{s},\Delta t)\hat{f}(\mathbf{s},t-\Delta t), \tag{9}$$

$$W_0(\mathbf{s},\Delta t) = \frac{\mathrm{e}^{-z}-1+z}{z^2}\Delta t, \quad W_1(\mathbf{s},\Delta t) = \frac{1-\mathrm{e}^{-z}-z\mathrm{e}^{-z}}{z^2}\Delta t, \tag{10}$$

where $z = \|\mathbf{s}\|^2\Delta t$.

For a cubic approximation of $\hat{f}(\mathbf{s},\tau)$, yielding fourth order accuracy in time, we have

$$\Phi(\mathbf{s},t,\Delta t) = W_0(\mathbf{s},\Delta t)\hat{f}(\mathbf{s},t) + W_1(\mathbf{s},\Delta t)\hat{f}(\mathbf{s},t-\Delta t) + W_2(\mathbf{s},\Delta t)\hat{f}(\mathbf{s},t-2\Delta t) + W_3(\mathbf{s},\Delta t)\hat{f}(\mathbf{s},t-3\Delta t), \tag{11}$$

where

$$\begin{aligned}
W_0(\mathbf{s},\Delta t) &= \frac{(6+2z^2-6z)\mathrm{e}^{-z} + (6z^3-11z^2-6+12z)}{6z^4}\Delta t,\\
W_1(\mathbf{s},\Delta t) &= \frac{(-z^2+2z^3+6-4z)\mathrm{e}^{-z} + (-6z^2-6+10z)}{-2z^4}\Delta t,\\
W_2(\mathbf{s},\Delta t) &= \frac{(2z+2z^2-6)\mathrm{e}^{-z} + (-8z+3z^2+6)}{-2z^4}\Delta t,\\
W_3(\mathbf{s},\Delta t) &= \frac{(z^2-6)\mathrm{e}^{-z} + (6+2z^2-6z)}{6z^4}\Delta t.
\end{aligned}$$

A word of caution concerning the computation of the weights: if z is small, the formulae above are subject to significant cancellation error. In that case, the weights can be computed by Taylor expansion of the exponentials, carried out to a sufficient number of terms. A reasonable compromise is to switch from the analytic expression to the Taylor series if $z < 10^{-3}$. Four terms in the Taylor series are then sufficient for 12 digit accuracy.

2.2. The forward transform

Given the quadrature rule (9) or (11), we still need to compute $\hat{f}(\mathbf{s},t)$ from the data $f(\mathbf{x},t)$. The same transform, of course, is also required at $t=0$ to compute $\hat{U}(\mathbf{s},0)$ from $U(\mathbf{x},0)$. Since we have assumed that the data are supported in the box $B = [-R/2,R/2]^d$, the Fourier transform is simply

$$\hat{f}(\mathbf{s},t) = \int_{-R/2}^{R/2}\cdots\int_{-R/2}^{R/2} \mathrm{e}^{\mathrm{i}\mathbf{s}\cdot\mathbf{x}} f(\mathbf{x},t)\,\mathrm{d}\mathbf{x}, \tag{12}$$

where $\mathbf{x} = (x_1,\ldots,x_d)$, $\mathbf{s} = (s_1,\ldots,s_d)$.

While we have not, as yet, determined where $\hat{f}(\mathbf{s},t)$ is to be computed, let us recall that the source $f(\mathbf{x},t) \in C^k(B)$. Thus, $\hat{f}(\mathbf{s},t) = \mathrm{O}(\|\mathbf{s}\|^{-k})$ for large $\mathbf{s}$. A straightforward calculation shows that if ϵ is the error tolerance, then evaluation of (12) needs to be done only for $\|\mathbf{s}\| \leqslant P$, where $P = \mathrm{O}(1/\epsilon)^{\frac{1}{k}}$. We will refer to P as the *high-frequency cutoff*. This bounds the oscillatory behavior of the term $\mathrm{e}^{\mathrm{i}\mathbf{s}\cdot\mathbf{x}}$ in the integrand of (12). Together with the fact that $f(\mathbf{x},t)$ is smooth, it follows that the trapezoidal rule applied to (12) with N points will yield $\mathrm{O}(N^{-k})$ accuracy [4]. The error will begin decaying rapidly once N is of the order $\mathrm{O}(PR)$, meaning that the integrand is well resolved. The same reasoning holds for the initial values $\hat{U}(\mathbf{s},0)$. If $f(\mathbf{x},t)$ is given on a uniform mesh with N points in each dimension, the trapezoidal rule yields

$$\hat{f}(\mathbf{s},t) \approx \left(\frac{R}{N}\right)^d \sum_{n_1=1}^{N} \cdots \sum_{n_d=1}^{N} \mathrm{e}^{\mathrm{i}\mathbf{s}\cdot\mathbf{x}_n} f(\mathbf{x}_n,t), \tag{13}$$

where $\mathbf{x}_n = (-R/2 + n_1(R/N), \ldots, -R/2 + n_d(R/N))$.

It remains only to determine the actual nodes in the Fourier domain $\mathbf{s}_j$ where we wish to obtain $\hat{f}(\mathbf{s}_j,t)$ and, therefore, $\hat{U}(\mathbf{s}_j,t)$.

2.3. The inverse transform

In order to compute the solution in physical space, we need to evaluate the inverse Fourier transform

$$U(\mathbf{x},t) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \mathrm{e}^{-\mathrm{i}\mathbf{s}\cdot\mathbf{x}} \hat{U}(\mathbf{s},t)\,\mathrm{d}\mathbf{s}. \tag{14}$$

Thus, we need to devise a quadrature for (14).

As discussed above, since the data are smooth, we are justified in truncating the domain of integration in the Fourier domain at $\|\mathbf{s}\| = P = \mathrm{O}(1/\epsilon)^{\frac{1}{k}}$, with an error ϵ. (If the data were band-limited at frequency P, of course, then this truncation would introduce no error.) The real difficulty lies in developing an efficient rule for the finite Fourier integral:

$$U(\mathbf{x},t) \approx \frac{1}{(2\pi)^d} \int_{\|\mathbf{s}\|<P} \mathrm{e}^{-\mathrm{i}\mathbf{s}\cdot\mathbf{x}} \hat{U}(\mathbf{s},t)\,\mathrm{d}\mathbf{s}. \tag{15}$$

The problem of discretizing (15) was addressed in [10], where it was shown (in one dimension) that a mesh which clustered toward the origin in s-space on dyadic intervals was capable of resolving the integrand for all time to any desired precision ϵ. Intuitively, the reason for this exponential clustering at the origin can be understood from considering the Fourier transform of the free space Green's function $\frac{\mathrm{e}^{-\|\mathbf{x}\|^2/4t}}{(4\pi t)^{\frac{d}{2}}}$ itself, namely the function $\mathrm{e}^{-\|\mathbf{s}\|^2 t}$. For large t, this function is sharply peaked near $\mathbf{s} = 0$ and the accurate resolution of this function is what ensures that the lowest frequency terms diffuse into the infinite region correctly.

A slight modification of Theorem 2.1 in [10] yields

Theorem 1 (adapted from [10]). *Let $[a,b]$ be a dyadic interval of the form $[2^j, 2^{j+1}]$, let $\epsilon > 0$ be the desired precision and let $\{s_1,\ldots,s_n\}$ and $\{w_1,\ldots,w_n\}$ be the nodes and weights for the n-point Gauss–Legendre quadrature scaled to $[a,b]$. Then,*

$$\left| \int_a^b \mathrm{e}^{\mathrm{i}sx} \hat{U}(s,t)\,\mathrm{d}s - \sum_{k=1}^{n} \mathrm{e}^{\mathrm{i}s_k x} \hat{U}(s_k,t) w_k \right| \leqslant \sqrt{2\pi} \frac{(b-a)}{\sqrt{n}} \left[\frac{R(b-a)}{2n} + \sqrt{\frac{\log(1/\epsilon)}{n}} \right]^{2n} + \mathrm{O}(\epsilon) \tag{16}$$

for $|x| \leqslant R$.

Note that, in the estimate (16), both terms in square brackets must be small for the quadrature to be accurate. For the first term to be small, the number of nodes must scale like the length of the interval in s-space. The second term is more interesting. It requires that there be at least a constant number of nodes on each interval, *no matter how small*. It is this requirement that forces the exponential clustering of nodes toward the origin.

Corollary 1 (adapted from [10]). *Let $\epsilon > 0$ be the desired precision, let $L_{\min} = -\log(1/\epsilon)$ and let $L_{\max} = \lceil \log P \rceil$, where P is the high-frequency cutoff. Further, let $\{s_{j,1},\ldots,s_{j,n(j)}\}$ and $\{w_{j,1},\ldots,w_{j,n(j)}\}$ be the nodes and weights for the n(j)-point Gauss–Legendre quadrature on the interval $[2^j, 2^{j+1}]$, where $n(j) = \max(R2^{j+3/2}$, $8\log(1/\epsilon))$. Then, in one space dimension,*

$$\left| \int_{|s|<P} \mathrm{e}^{-\mathrm{i}s\cdot x} \hat{U}(s,t)\,\mathrm{d}s - \sum_{j=L_{\min}}^{L_{\max}} \sum_{k=1}^{n(j)} (\mathrm{e}^{\mathrm{i}s_{j,k}x} + \mathrm{e}^{-\mathrm{i}s_{j,k}x}) \hat{U}(s_{j,k},t) w_{j,k} \right| = \mathrm{O}(\epsilon) \tag{17}$$

for $|x| \leqslant R$.

We will denote by $N_1 = N_1(\epsilon,R,P)$ the total number of nodes required in one dimension. Using a tensor product of this one-dimensional rule, $\mathrm{O}(N_1^d) = \mathrm{O}((\log(1/\epsilon) + RP)^d)$ nodes are required in d dimensions. We leave it to the reader to verify that if the trapezoidal rule were employed for (15), then $O(\frac{RP}{\epsilon})^d$ nodes would be required in order to be accurate for all time $t > 0$.

2.4. The nonuniform FFT

Let us denote by $s_1, \ldots, s_{N_1}$ and $w_1, \ldots, w_{N_1}$ the full set of one-dimensional quadrature nodes and weights described in the preceding section. Then,

$$U(\mathbf{x}, t) \approx \frac{1}{(2\pi)^d} \sum_{j_1=1}^{N_1} \cdots \sum_{j_d=1}^{N_1} e^{-i\mathbf{s}_j \cdot \mathbf{x}} \hat{U}(\mathbf{s}_j, t) w_{j_1} \cdots w_{j_d}, \tag{18}$$

where $\mathbf{s}_j = (s_{j_1}, \ldots, s_{j_d})$.

The summation in (18) must be carried out for each evaluation point $\mathbf{x}$. Direct evaluation at each of the N^d points $\mathbf{x}_n$ defined in Section 2.2 would require $O(N_1^d \cdot N^d)$ work. Likewise, the sum (13) must be evaluated at each of the N_1^d spectral nodes, also requiring $O(N_1^d \cdot N^d)$ work.

Since the $\{\mathbf{s}_j\}$ are not uniformly spaced, the classical fast Fourier transform (FFT) cannot be applied to accelerate this computation. In the last decade, however, suitable fast algorithms have been developed that we refer to as non-uniform fast Fourier transforms (NUFFTs). Detailed descriptions and additional references can be found in [2,6–9,21,24]. Like the FFT, these algorithms evaluate sums of the form (18) and (13) in $O((N_1^d + N^d)\log(N_1^d + N^d))$ work, for any fixed precision.

3. The numerical scheme

With the full complement of tools in place, we can now provide an informal description of the fast recursive marching (FRM) method. For kth order accuracy in time, we refer to the method as FRM(k).

The FRM(2) method

```
Step 1: Initialization
    (a) Select time step $\Delta t$ and number of time steps $M$.
    (b) Select precision $\epsilon$ for quadrature in Fourier domain.
    (c) Obtain $N_1$ quadrature nodes and weights in Fourier space for (18) according to Corollary 1.
    (d) Compute weights $W_0$, $W_1$ from (10).
Step 2: Transform initial data
    (a) Initialize $\hat{U}(\mathbf{s},0)$ at all $N_1^d$ spectral nodes by computing the forward NUFFT of $U(\mathbf{x},0)$.
    (b) Compute $\hat{f}(\mathbf{s},0)$ at all $N_1^d$ spectral nodes by computing the forward NUFFT of $f(\mathbf{x},0)$.
Step 3: For $m = 1,\ldots,M$
    (a) Compute $\hat{f}(\mathbf{s},m\Delta t)$ at all $N_1^d$ spectral nodes by computing the forward NUFFT of $f(\mathbf{x},m\Delta t)$.
    (b) Update $\hat{U}(\mathbf{s},t)$ according to (7).
        $\hat{U}(\mathbf{s},m\Delta t) = e^{-\|\mathbf{s}\|^2\Delta t}\hat{U}(\mathbf{s},(m-1)\Delta t) + W_0(\mathbf{s},\Delta t)\hat{f}(\mathbf{s},m\Delta t) + W_1(\mathbf{s},\Delta t)\hat{f}(\mathbf{s},(m-1)\Delta t)$
    (c) Compute $U(\mathbf{x},t)$ from $\hat{U}(\mathbf{s},t)$ using (18) and the NUFFT.
```

The modifications necessary for the fourth order accurate FRM(4) method are straightforward.

Note that the total computational cost is of the order $O(M \cdot (N_1^d + N^d)\log(N_1^d + N^d))$, since two NUFFT calculations are required per time step. This is true for both FRM(2) and FRM(4), so that their execution times are nearly identical. FRM(4) does, however, require twice as much storage since it is a four-level marching scheme.

4. Numerical results

We first test the performance of the algorithm in two dimensions with a known exact solution $U_{\text{exa}}(x_1, x_2, t)$, constructed so that $U_{\text{exa}}(x_1, x_2, t) = V(x_1, x_2, t) + W(x_1, x_2, t)$, where V corresponds to an initial potential and W corresponds to a volume potential. For this, we let

$$V(x_1,x_2,t) = \frac{e^{-\frac{(x_1-0.05)^2+(x_2-0.15)^2}{4(t+0.01)}}}{4\pi(t+0.01)}.$$

Clearly V satisfies the homogeneous heat equation with initial data

$$V(x_1,x_2,0) = \frac{e^{-\frac{(x_1-0.05)^2+(x_2-0.15)^2}{4(0.01)}}}{4\pi(0.01)}.$$

We let W satisfy the inhomogeneous heat equation

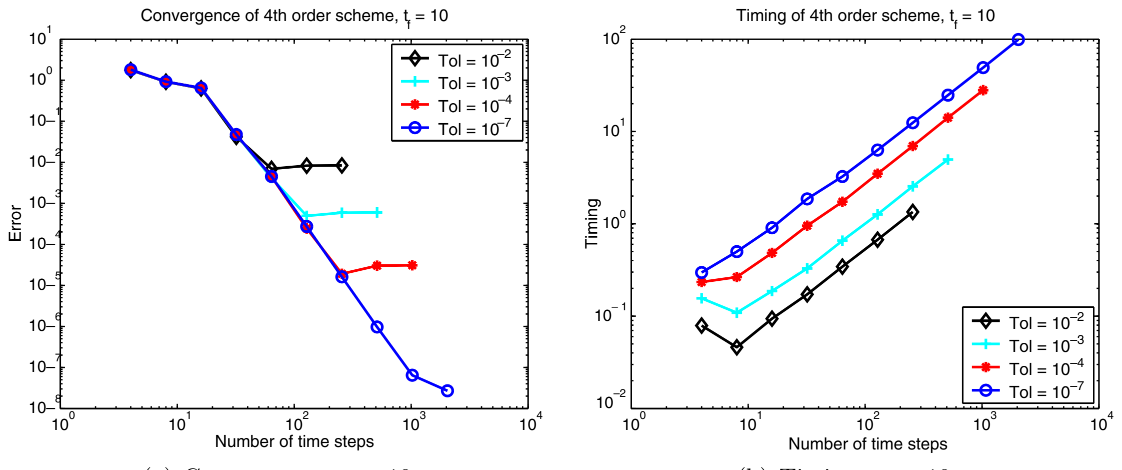
$$\begin{aligned} &W_t = \nabla^2 W + f, \\ &W(x_1,x_2,0) = 0, \\ &f(x_1,x_2,t) = e^{-\frac{(x_1-0.1)^2+(x_2-0.2)^2}{0.04}}(100\sin(5.1t)+200\cos(4.1t)). \end{aligned}$$

To obtain the reference solution, W is computed via the volume potential formula (3), using analytic integration in space and high order Gauss–Legendre quadrature in time. We choose the computational domain to be $[-2,2]\times[-2,2]$; both $U_{\text{exa}}(x_1,x_2,0)$ and $f(x_1,x_2,t)$ are negligible outside of it. The constants in W are chosen so that the L_2 norm of the solution is approximately 1, meaning that the errors shown in the following table and figures have the same order of magnitude as the relative errors.

In Fig. 1 we show results for the fourth order fast recursive marching scheme FRM(4), based on the cubic product integration rule (8), for four different levels of error tolerance: $tol = 10^{-2}$, 10^{-3}, 10^{-4}, 10^{-7}. The parameters required for these tolerances are given in Table 1. We set the NUFFT tolerance to *tol* and run the simulation until $t_f = 10$.

Fig. 1a shows that the convergence rate is consistent with a fourth order accurate scheme until the tolerance level has been achieved, at which point no further reduction in error is obtained. The computation time (Fig. 1a) is clearly linear in the number of time steps for each value of *tol*.

In Table 2 we compare the timings of FRM(4) and a fourth order in time, implicit finite difference method (IFD4). For IFD4, we use the five point Laplacian in space and either a fourth order backward differentiation formula (BDF) in time or a singly implicit Runge–Kutta method. The results are approximately the same for these two marching schemes, so we only present results for the BDF approach. We chose to enforce homogeneous Neumann boundary conditions on the boundary of the computational domain, since this is often used in dendritic solidification simulations [22,25], the application discussed below. Homogeneous Neumann conditions are compatible with the use of FISHPACK [1] to solve the "modified" Helmholtz problem that arises at each time step.



(a) Convergence, $t_f = 10$ (b) Timing, $t_f = 10$

Fig. 1. Convergence of FRM(4) – the fast recursive marching scheme with fourth order accurate quadrature in time.

Table 1
Parameter selection in FRM scheme

tol	P	N	N_1
10^{-2}	12	21	72
10^{-3}	18	31	80
10^{-4}	25	41	152
10^{-7}	35	51	224

The first column indicates the desired precision *tol*. The second, third and fourth columns indicate (for each dimension) the high-frequency cutoff, the number of nodes in physical space and the number of nodes in Fourier space, corresponding to the selected value of *tol*. The actual number of nodes used in this two-dimensional calculation are N^2 and N_1^2. In this particular case, N and N_1 grow only logarithmically with *tol*, since the data are infinitely differentiable.

Table 2
Comparison of fourth order fast recursive marching scheme FRM(4) and a standard implicit finite difference method (IFD4)

	$tol = 10^{-2}$		$tol = 10^{-3}$		$tol = 10^{-4}$		$tol = 10^{-7}$	
	FRM(4)	IFD4	FRM(4)	IFD4	FRM(4)	IFD4	FRM(4)	IFD4
$t = 1$								
$x_{\max}$	2	3	2	4	2	6	2	n/a
N^2	$(21)^2$	$(61)^2$	$(31)^2$	$(321)^2$	$(41)^2$	$(1201)^2$	$(51)^2$	n/a
M	8	16	16	32	32	128	128	n/a
Total time	0.06	0.1	0.2	5	0.9	246	6	n/a
Time/step	0.01	0.01	0.01	0.15	0.03	2	0.05	n/a
$t = 10$								
$x_{\max}$	2	6	2	8	2	10	2	n/a
N^2	$(21)^2$	$(121)^2$	$(31)^2$	$(641)^2$	$(41)^2$	$(2001)^2$	$(51)^2$	n/a
M	64	128	128	256	256	512	1024	n/a
Total time	0.3	2	1	111	7	2374	50	n/a
Time/step	0.01	0.02	0.01	0.43	0.03	4.6	0.05	n/a
$t = 100$								
$x_{\max}$	2	10	2	15	2	n/a	2	n/a
N^2	$(21)^2$	$(201)^2$	$(31)^2$	$(1201)^2$	$(41)^2$	n/a	$(51)^2$	n/a
M	512	1024	1024	2048	2048	n/a	16,384	n/a
Total time	3	38	10	2889	55	n/a	828	n/a
Time/step	0.01	0.04	0.01	1.4	0.03	n/a	0.05	n/a

For IFD4, entries marked by "n/a" means that we were not able to compute a solution due to memory constraints (see text for explanation).

We present results for the same four levels of error tolerance as before, at three different final times, $t_f = 1,\ 10,\ 100$. The number of discretization points in physical space is N^2 and the number of time steps is M. The L_2 errors are computed at $t = t_f$ on the domain $[-2,2] \times [-2,2]$. For each method, the computational domain is $[-x_{\max}, x_{\max}] \times [-x_{\max}, x_{\max}]$. For FRM(4), since the initial data and the source are both localized to $[-2,2] \times [-2,2]$, the computational domain can remain $[-2,2] \times [-2,2]$ for all four levels of tolerance for all time. But with IFD4, the computational domain grows larger with larger t_f and smaller *tol*, since Neumann conditions are *not* the correct artificial boundary conditions on the square. Formally, it is easy to verify that the domain must grow like $O(\sqrt{t_f})$ in each linear dimension, so that the number of grid points must grow like $O(t_f)$ in a two-dimensional simulation.

For $t_f = 10$ and error tolerances of 10^{-2}, 10^{-3}, 10^{-4}, and 10^{-7}, FRM(4) took 0.3, 1, 7, and 50 s, respectively, on a laptop computer with a 2 GHz Pentium M processor. The finite difference method IFD4 took 2, 111, 2374 s for the first three error levels, respectively, and we were not able to compute a solution for $tol = 10^{-7}$ due to memory constraints.

For $t_f = 100$ and the same four error tolerances, FRM(4) took 3, 10, 55, and 828 s, respectively. IFD4 took 38 and 2889 s for the first two error levels, and we were not able to compute a solution at the two higher levels of accuracy. For the error level 10^{-3}, the number of unknowns in physical space is $(31)^2$ for FRM(4) and it is

$(1201)^2$ for IFD4, because IFD4 must expand the computational domain from $[-2,2]\times[-2,2]$ to the domain $[-15,15]\times[-15,15]$.

4.1. Phase field model

We next illustrate the usefulness of the method in modeling the process of dendritic solidification in a pure material in two dimensions. This example, and the following explanation of the model, were drawn from [22]. We have take the simple isotropic phase field model equations (rather than the more complicated anisotropic model the authors derived in the original paper) which simulate dendritic growth into an under-cooled liquid. The coupled system of heat equations governing the process are

$$u_t - \nabla^2 u = -\frac{30\phi^2(1-\phi)^2}{S}\phi_t, \tag{19}$$

$$\frac{\tilde{\epsilon}^2}{m}\phi_t - \tilde{\epsilon}^2\nabla^2\phi = \phi(1-\phi)\left(\phi-\frac{1}{2}\right) + 30\phi^2(1-\phi)^2\tilde{\epsilon}\alpha S u, \tag{20}$$

with unknowns $\phi(\mathbf{x},t)$ and $u(\mathbf{x},t)$. The quantity $\phi(\mathbf{x},t)$ is known as the phase field variable, with $\phi\equiv 0$ and $\phi\equiv 1$ corresponding to the bulk solid and liquid phases, respectively. The temperature is $T = T_\mathrm{M} + \Delta T u$, where T_M is the equilibrium melting temperature and ΔT is the under-cooling (the difference between the melting temperature and the initial temperature at the boundary of the domain). At the start of the simulation, in large areas of the computational domain (except where the initial seed configuration is placed) $\phi\equiv 1$ and $u\equiv -1$. Thus, when we speak of the diffusion of ϕ and u, we are really talking about the enlargement of the support of $\phi-1$ and $u+1$, respectively. The parameter $S=\frac{c\Delta T}{L}$ (c is the specific heat per unit volume, L is the latent heat per unit volume) is the Stefan number or dimensionless under-cooling (the quantity L/c is the unit of under-cooling), $\tilde{\epsilon}$ controls the interface thickness, α and m are related to specific material properties and the length scale.

Typically, the coupled equations (19) and (20) are solved by a finite difference method, most often using a backward differentiation formula in time [25]. For the nonlinear equation in (20), Newton's method is used as an inner iteration. This problem is posed in an infinite domain, so that artificially boundary conditions need to be imposed at the computational boundary. Homogeneous Neumann conditions are frequently used for both u and ϕ [22,25].

A potential difficulty with using a finite difference method for this application lies in the difference in the diffusion characteristics of u and ϕ when the under-cooling is small. For example, there are materials for which the typical under-cooling parameter is in the range $S = 0.002$–0.1 [19]. When this is the case, the support of $u+1$ grows to be much larger than the support of $\phi-1$ as time increases. Thus, unless the computational domain is made quite large, the support of $u+1$ reaches the computational boundary early in the simulation, and an incorrect boundary condition for u will pollute the result for both u and ϕ as the simulation continues. In [22] the authors were constrained to choose a large under-cooling, $S = 0.5$, for their numerical simulations, so that the supports of $u+1$ and $\phi-1$ stay on the same scale.

Since the fast recursive marching scheme solves the free space heat equation directly, we require only that the supports of the initial data and the source are contained within the computational boundary. We can then use it to solve (19) for u (or more precisely, $u+1$). Note that the source for (19) is localized on the support of $\phi-1$ (it is 0 when $\phi=1$). Thus, as long as the computational domain contains the support of $\phi-1$ at all times, the requirement that the *source* for $u+1$ be contained in the computational domain at all times is satisfied, even if $u+1$ itself diffuses out. The computational domain, of course, also needs to contain the support of $u+1$ at the start of the simulation.

In this example, Eq. (20) is solved using the standard five-point stencil for the Laplacian and the implicit Euler method in time, in order to illustrate as simply as possible that FRM is able to handle diffusion into an unbounded medium in this more complex context. The coupling of a higher order BDF method with an efficient Newton solver for Eq. (20) and FRM for (19) will be the topic of a separate paper.

With the notation of the previous section, we will use FRM(2) (with a linear approximation we automatically get second order) for Eq. (19) and IFD1 for Eq. (20), where the nonlinear source terms are treated

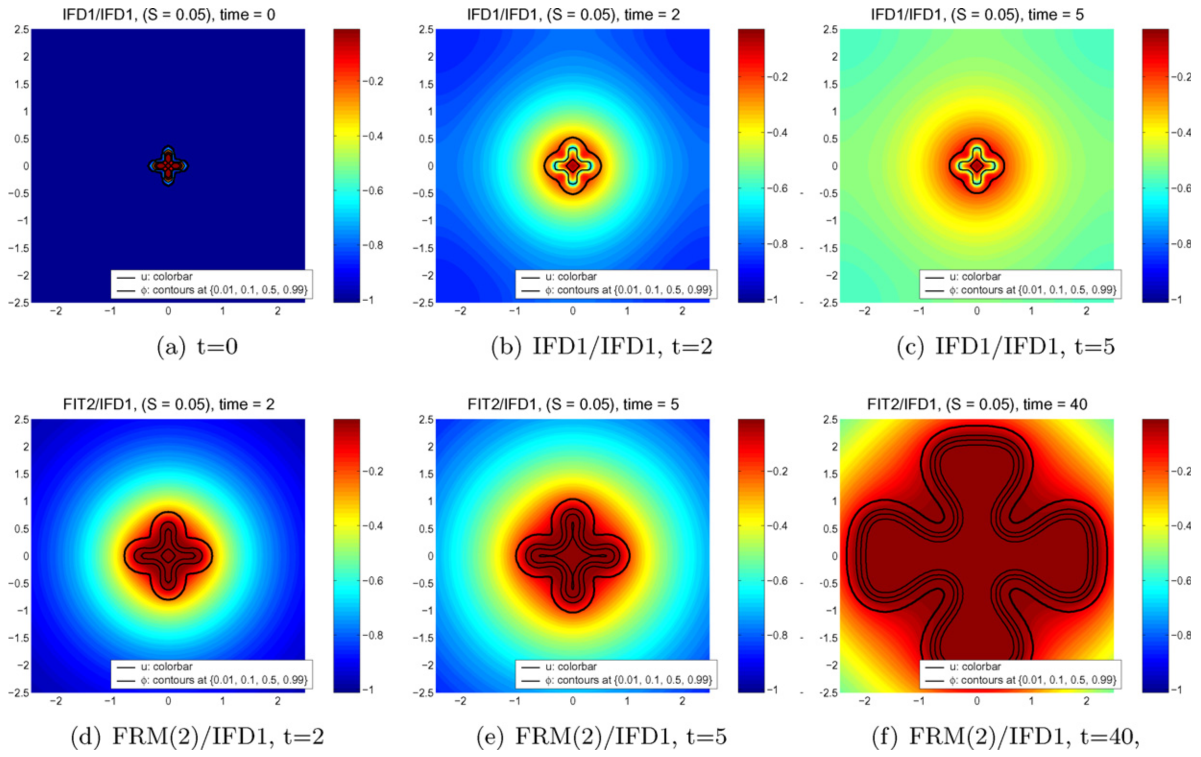


Fig. 2. Comparison of solving the thermal field equation (19) using FRM(2) versus solving it using IFD1 and homogeneous Neumann boundary conditions. The phase equation (20) is solved using IFD1 in both cases. Heat diffuses out correctly with FRM(2)/IFD1 but is trapped with IFD1/IFD1.

explicitly – that is, their value is taken at time t when marching from time t to $t + \Delta t$. The resulting Helmholtz equation from IDF1 is solved using FISHPACK [1], which uses cyclic reduction for separable PDEs.

Numerical results are shown in Fig. 2 with the time step $\Delta t = 0.0025$. The relevant parameters are $S = 0.05$, which is considered low, $\alpha = 400$, $m = 0.035$, $\tilde{\epsilon} = 0.025$. In the first row of Fig. 2, we show the contour plots of u and ϕ at various times, where IFD1 is used to solve *both* Eqs. (19) and (20), using the computational domain $[-2.5, 2.5] \times [-2.5, 2.5]$. The background color (or grayscale) plot corresponds to field values of u on the computational domain. The black lines are contour lines for ϕ, corresponding to the values 0.01, 0.1, 0.5, 0.99 (outermost and thickest), respectively. Again, $\phi \equiv 0$ represents bulk solid and $\phi \equiv 1$ represents bulk liquid. In the second row of Fig. 2 we show the (converged) results from using FRM(2) for Eq. (19) and IDF1 for Eq. (20), on the computational domain $[-2.5, 2.5] \times [-2.5, 2.5]$.

It is easy to see that, when the thermal field is computed with IFD1 using simple Neumann boundary conditions, the heat is trapped at the boundaries and the simulation is no longer accurate even at $t = 2$ (Fig. 2b). If it is computed with FRM(2), heat diffuses out of the computational domain correctly and simulation can continue until the solidification front reaches the computational boundary (Fig. 2f).

In this example, both FRM(2) and IFD1 use the same number of discretization points in physical space, $N^2 = 100^2$. The computational time of FRM(2) is between 4 and 8 times that of IFD1. Both have the CPU time estimate of $O(N^2 \log N)$ per time step, but the present FRM implementation has a larger constant prefactor associated with the $O(\cdot)$ notation. This is a small increase when compared to the additional work that would be required if the computational domain were enlarged to ensure accuracy in the IFD1 calculation.

5. Conclusions and generalizations

We have described the fast recursive marching method, a simple Fourier-based method for the solution of the heat equation in free space with smooth initial data and a smooth source term. It allows for efficient and accurate long-time simulations without the need for artificial boundary conditions on a finite computational domain. The convergence theory can be stated trivially – the error in the solution is the quadrature error in computing the space–time integral (3). The CPU time of the method is nearly optimal, requiring $O(N^2 M \log N)$ work for M time steps, with N^2 points in the spatial discretization. There are a number of ways in which the present implementation can be optimized, and we mention a few. Inspection of the data in the preceding section shows that the number of points in the spectral representation, N_1, is larger than N by a small constant. More careful analysis of the relation between P, N and N_1 can reduce this constant, as can improvements in the clustering quadrature method discussed in Section 2. Finally, the CPU time per step is dominated by the NUFFT and improvements in that algorithm translate into improvements in any FRM scheme.

In subsequent works, we will describe the extension of the present scheme to the case of discontinuous source data and/or complex geometry. This will involve the full machinery described in [10,12]. More precisely, heat potentials, such as the volume potential in (3), are decomposed into a "history" part u_H and a "local part" u_L, according to

$$u(\mathbf{x},t) = \int_0^t \int_{\mathbb{R}^d} k(\mathbf{x}-\mathbf{y},t-\tau) f(\mathbf{y},\tau)\,\mathrm{d}\mathbf{y}\,\mathrm{d}\tau = u_H(\mathbf{x},t) + u_L(\mathbf{x},t),$$

where

$$u_H(\mathbf{x},t) = \int_0^{t-\delta} \int_{\mathbb{R}^d} k(\mathbf{x}-\mathbf{y},t-\tau) f(\mathbf{y},\tau)\,\mathrm{d}\mathbf{y}\,\mathrm{d}\tau,$$
$$u_L(\mathbf{x},t) = \int_{t-\delta}^{t} \int_{\mathbb{R}^d} k(\mathbf{x}-\mathbf{y},t-\tau) f(\mathbf{y},\tau)\,\mathrm{d}\mathbf{y}\,\mathrm{d}\tau.$$

The spectral representation is used only for u_H and a quadrature approximation is used for u_L. Layer potentials are decomposed in the same manner.

The full set of tools will allow for the construction of fast, explicit, and unconditionally stable solvers for the heat equation in interior or exterior domains with complex boundaries.

Acknowledgment

This work was supported by the Applied Mathematical Sciences Program of the U.S. Department of Energy under Contract DEFG0288ER25053.

References

[1] John Adams, Paul Swarztrauber, Roland Sweet, FISHPACK: a package of FORTRAN subprograms for the solution of separable elliptic partial differential equations, The National Center for Atmospheric Research, Boulder, Colorado, USA.
[2] Gregory Beylkin, On the fast Fourier transform of functions with singularities, Appl. Comput. Harmonic Anal. 2 (1995) 363–383.
[3] C.A. Brebbia, J.C.F. Telles, L.C. Wrobel, Boundary Element Techniques, Springer, Berlin, 1984.
[4] P.J. Davis, P. Rabinowitz, Methods of Numerical Integration, Academic Press, San Diego, 1984.
[5] Eric Dubach, Artificial boundary conditions for diffusion equations: numerical study, J. Comput. Appl. Math. 70 (1) (1996) 127–144.
[6] A. Dutt, V. Rokhlin, Fast fourier transforms for nonequispaced data, SIAM J. Sci. Comput. 14 (1993) 1368–1383.
[7] J.A. Fessler, B.P. Sutton, Nonuniform fast Fourier transforms using min–max interpolation, IEEE Trans. Signal Proc. 51 (2003) 560–574.
[8] K. Fourmont, Non-equispaced fast fourier transforms with applications to tomography, IEEE Trans. Signal Proc. 9 (2003) 431–450.
[9] L. Greengard, J.-Y. Lee, Accelerating the nonuniform fast Fourier transform, SIAM Rev. 46 (2004) 443–454.
[10] Leslie Greengard, Patrick Lin, Spectral approximation of the free-space heat kernel, Appl. Comput. Harmonic Anal. 9 (1) (2000) 83–97.
[11] Leslie Greengard, John Strain, A fast algorithm for the evaluation of heat potentials, Comm. Pure Appl. Math. 43 (8) (1990) 949–963.
[12] Leslie Greengard, John Strain, The fast Gauss transform, SIAM J. Sci. Statist. Comput. 12 (1) (1991) 79–94.
[13] Ronald B. Guenther, John W. Lee, Partial Differential Equations of Mathematical Physics and Integral Equations, Prentice Hall, Englewood Cliffs, NJ, 1988.
[14] Thomas Hagstrom, Radiation boundary conditions for the numerical simulation of waves, Acta Numerica, 1999, vol. 8. of Acta Numerica, Cambridge University Press, Cambridge, 1999, pp. 47–106.
[15] L. Halpern, J. Rauch, Absorbing boundary conditions for diffusion equations, Numer. Math. 71 (2) (1995) 185–224.

[16] Houde Han, Zhongyi Huang, A class of artificial boundary conditions for heat equation in unbounded domains, Comput. Math. Appl. 43 (6-7) (2002) 889–900.
[17] Houde Han, Zhongyi Huang, Exact and approximating boundary conditions for the parabolic problems on unbounded domains, Comput. Math. Appl. 44 (5-6) (2002) 655–666.
[18] Jingfang Huang, Ming-Chih Lai, Yang Xiang, An integral equation method for epitaxial step-flow growth simulations, J. Comput. Phys. 216 (2) (2006) 724–743.
[19] S.-C. Huang, M.E. Glickman, Fundamentals of dendritic solidification: I. Steady state tip growth, Acta Metall. (1981) 701–734.
[20] M.T. Ibañez, H. Power, An efficient scheme for BEM for heat transfer problems using Fourier series, Boundary Elements, XXI (Oxford, 1999), Int. Ser. Adv. Bound. Elem., vol. 6, WIT Press, Southampton, 1999, pp. 155–167.
[21] June-Yub Lee, Leslie Greengard, The type 3 nonuniform FFT and its applications, J. Comput. Phys. 206 (1) (2005) 1–5.
[22] B.T. Murray, A.A. Wheeler, M.E. Glickman, Simulations of experimentally observed dendritic growth behavior using a phase-field model, J. Cryst. Growth (1995) 386–400.
[23] Witold Pogorzelski, Integral Equations and their Applications, Pergamon Press, Oxford, 1966.
[24] D. Potts, G. Steidl, M. Tasche, Fast Fourier transforms for nonequispaced data: A tutorial, in: J.J. Benedetto, P. Ferreira (Eds.), Modern Sampling Theory Mathematics and Applications, Birkhäuser, Boston, 2001, pp. 249–274.
[25] R.J. Braun, B.T. Murray, Adaptive phase-fieldnext term computations of dendritic crystal growth, J. Cryst. Growth 174 (1–4) (1997) 41–53.
[26] John Strain, Fast adaptive methods for the free-space heat equation, SIAM J. Sci. Statist. Comput. 15 (1992) 185–206.
[27] Johannes Tausch, A fast method for solving the heat equation by layer potentials, J. Comput. Phys. 224 (2) (2007) 956–969.