On the numerical solution of the heat equation I: propagation in unbounded domains

Jing-Rebecca Li       Leslie Greengard

April 11, 2006

Abstract

We describe a fast solver for the inhomogeneous heat equation in unbounded domains. It relies on a recently developed spectral approximation of the free-space heat kernel [8] coupled with the nonuniform fast Fourier transform [2, 4, 7, 13]. Unlike finite difference and finite element techniques, diffusion into an infinite medium is satisfied analytically, avoiding the 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.

Key words. heat equation; unbounded domain; integral representation; spectral approximation.

1 Introduction

The solution of the heat equation (the diffusion equation) in unbounded domains 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 and finite element methods, these must be coupled to artificial (non-reflecting) boundary conditions imposed on the finite computational domain in order to simulate the effects of diffusion into an infinite medium. There is an extensive literature on such techniques that we will not attempt to review here.

Instead, we describe a very different approach, based on the exact solution of the governing equation using convolution with the free-space Green’s function. In the simplest setting, suppose that one wishes to solve the isotropic inhomogeneous heat equation in $\mathbb{R}^d$:

$$U_t(x, t) - \nabla^2 U(x, t) = f(x, t), \quad t > 0,$$

subject to the initial condition

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

where $f(x, t)$ and $U_0(x)$ are smooth and compactly supported in the box $B = [-R/2, R/2]^d$, centered at the origin. More concretely, we suppose that $f(x, t)$ and $U_0(x)$ are $k$-times differentiable ($f(x, t), U_0(x) \in C^k(B)$). We assume that no physical boundaries are present.

From standard potential theory [11, 15], the solution can be written as

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

where $k(x, t)$ is the free-space heat kernel.
where the fundamental solution for the heat equation in $\mathbb{R}^d$ is
\[
  k(x, t) := \frac{e^{-|x|^2/4t}}{(4\pi t)^{d/2}}, \quad t > 0.
\]

We will refer to the first integral in (3) as an \textit{initial potential} and the second integral as a \textit{volume potential}. Naive discretization of these integrals leads to an enormously expensive numerical method - 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, correctness of the far-field behavior, etc. appear to be overwhelmed by questions of cost. In recent years, however, several fast algorithms have been developed [8–10, 17] that lead to nearly optimal schemes, for which the work required is of the order $O(NM \log N)$. They involve a fairly substantial amount of numerical and analytic machinery. In the present case, 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 equation (1) and (2), significant advantage can be obtained by considering its Fourier transform. For this, we let
\[
  \hat{U}(s, t) = \int_{\mathbb{R}^d} e^{i\langle s, x \rangle} U(x, t) \, dx, \quad U(x, t) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} e^{-i\langle s, x \rangle} \hat{U}(s, t) \, ds.
\]

It is obvious from (1), (2), and well-known, that $\hat{U}(s, t)$ satisfies the ordinary differential equation
\[
  \frac{d\hat{U}}{dt}(s, t) = -\|s\|^2 \hat{U}(s, t) + \hat{f}(s, t),
\]
where
\[
  \hat{f}(s, t) = \int_{\mathbb{R}^d} e^{i\langle s, x \rangle} f(x, t) \, dx.
\]
An elementary calculation shows that
\[
  \hat{U}(s, t) = e^{-\|s\|^2 \Delta t} \hat{U}(s, t - \Delta t) + \Phi(s, t),
\]
where
\[
  \Phi(s, t, \Delta t) = \int_{t-\Delta t}^{t} e^{-\|s\|^2 (t-\tau)} \hat{f}(s, \tau) \, d\tau.
\]
Thus, in the spectral (Fourier) domain, history dependence is no longer an issue. 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 $\hat{f}(s, t)$ from $f(x, t)$, and we need to compute the inverse transform to obtain $U(x, t)$ from $\hat{U}(s, t)$. This last step turns out to be somewhat subtle.

3 Quadrature and discretization

Let us first describe a straightforward “product integration” rule for (8), based on polynomial approximation of $\hat{f}(s, t)$ in time, followed by analytic integration. Linear approximation of
\[
\hat{f}(s, t) \text{ yields second order accuracy and the rule } [9]
\]
\[
\Phi(s, t, \Delta t) = W_0(s, \Delta t) \hat{f}(s, t) + W_1(s, \Delta t) \hat{f}(s, t - \Delta t),
\]
\[
W_0(s, \Delta t) = \frac{e^{-z} - 1 + z}{z^2} \Delta t, \quad W_1(s, \Delta t) = \frac{1 - e^{-z} - z e^{-z}}{z^2} \Delta t,
\]
where \( z = \|s\|^2 \Delta t \).

For a cubic approximation of \( \hat{f}(s, \tau) \), yielding fourth order accuracy in time, we have
\[
\Phi(s, t, \Delta t) = W_0(s, \Delta t) \hat{f}(s, t) + W_1(s, \Delta t) \hat{f}(s, t - \Delta t) + W_2(s, \Delta t) \hat{f}(s, t - 2\Delta t) + W_3(s, \Delta t) \hat{f}(s, t - 3\Delta t),
\]
where
\[
W_0(s, \Delta t) = \frac{(6 + 2z^2 - 6z)e^{-z} + (6z^3 - 11z^2 - 6 + 12z)}{6z^4} \Delta t,
\]
\[
W_1(s, \Delta t) = \frac{(-z^2 + 2z^3 + 6 - 4z)e^{-z} + (-6z^2 - 6 + 10z)}{-2z^4} \Delta t,
\]
\[
W_2(s, \Delta t) = \frac{(2z + 2z^2 - 6)e^{-z} + (-8z + 3z^2 + 6)}{-2z^4} \Delta t,
\]
\[
W_3(s, \Delta t) = \frac{(z^2 - 6)e^{-z} + (6 + 2z^2 - 6z)}{6z^4} \Delta t.
\]

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 twelve digit accuracy.

### 3.1 The forward transform

Given the quadrature rule (9) or (11), we still need to compute \( \hat{f}(s, t) \) from the data \( f(x, t) \).
(The same transform is also required at \( t = 0 \) to compute \( \hat{U}(s, 0) \) from \( U(x, 0) \).) Since we have assumed that the data is supported in the box \( B = [-R/2, R/2]^d \), the Fourier transform is simply
\[
\hat{f}(s, t) = \int_{-R/2}^{R/2} \cdots \int_{-R/2}^{R/2} e^{is \cdot x} f(x, t) \, dx,
\]
where \( x = (x_1, \ldots, x_d) \), \( s = (s_1, \ldots, s_d) \).

If \( f(x, t) \) is given on a uniform mesh with \( N \) points in each dimension, the trapezoidal rule yields
\[
\hat{f}(s, t) \approx \left( \frac{R}{N} \right)^d \sum_{n_1=1}^N \cdots \sum_{n_d=1}^N e^{is \cdot x_n} f(x_n, t), \quad (12)
\]
where \( x_n = (-R/2 + n_1 \ast R/N, \ldots, -R/2 + n_d \ast R/N) \). While we have not, as yet, determined where \( \hat{f}(s, t) \) is to be computed, let us recall that the data \( U(x, 0), f(x, t) \in C^k(B) \). Thus, we are justified in truncating the domain of integration in the Fourier domain at, say, \( ||s|| = P \), with an error that decays like \( P^{-k} \). This bounds the oscillatory behavior of the integrand in (12). Together with the smoothness of \( f(x, t) \), it follows that the trapezoidal rule will yield high order accuracy [3]. More precisely, it is easy to verify that \( N \) must be of the order \( O(PR) \) to ensure that the integrand is well-resolved and that the quadrature error is of the order \( O(N^{-k}) \). We will refer to \( P \) as the *high-frequency cutoff*. It remains only to determine the actual nodes in the Fourier domain \( s_j \) where we wish to obtain \( \hat{f}(s_j, t) \) and, therefore, \( \hat{U}(s_j, t) \).
3.2 The inverse transform

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

\[ U(x, t) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} e^{-is \cdot x} \hat{U}(s, t) \, ds. \] (13)

Thus, we need to devise a quadrature for (13). As discussed above, since the data are smooth, we are justified in truncating the domain of integration in the Fourier domain at \( \|s\| = P \) with an error that decays like \( P^{-k} \). (If the data were band-limited at frequency \( P \), of course, then truncation would introduce no error.) The real difficulty lies in developing an efficient rule for the finite Fourier integral:

\[ U(x, t) \approx \frac{1}{(2\pi)^d} \int_{\|s\| < P} e^{-is \cdot x} \hat{U}(s, t) \, ds. \] (14)

The problem of discretizing (14) was addressed in [8], where it was shown (in one dimension) that a mesh which clustered toward the origin in transform \( j \) for \( \|s\| = P \) with an error that decays like \( P^{-k} \). (If the data were band-limited at frequency \( P \), of course, then truncation would introduce no error.) The real difficulty lies in developing an efficient rule for the finite Fourier integral:

\[ U(x, t) \approx \frac{1}{(2\pi)^d} \int_{\|s\| < P} e^{-is \cdot x} \hat{U}(s, t) \, ds. \] (14)

Theorem 1. (adapted from [8]) 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,

\[ \left| \int_a^b e^{isx} \hat{U}(s, t) \, ds - \sum_{k=1}^n e^{is_k x} \hat{U}(s_k, t) w_k \right| \leq \sqrt{2\pi} \frac{(b-a)}{\sqrt{n}} \left[ R \frac{(b-a)}{2n} + \sqrt{\frac{\log(1/\epsilon)}{n}} \right]^{2n} + O(\epsilon) \] (15)

for \( |x| \leq R \).

Note that, in the estimate (15), 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 exponential clustering toward the origin that allows us to correctly capture the behavior of diffusion in the infinite medium.

Corollary 1. (adapted from [8]) Let \( \epsilon > 0 \) be the desired precision, let \( L_{\text{min}} = -\log(1/\epsilon) \) and let \( L_{\text{max}} = [\log P] \), 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,

\[ \left| \int_{|s| < P} e^{-is \cdot x} \hat{U}(s, t) \, ds - \sum_{j=L_{\text{min}}}^{L_{\text{max}}} \sum_{k=1}^{n(j)} (e^{is_{j,k} x} + e^{-is_{j,k} x}) \hat{U}(s_{j,k}, t) w_{j,k} \right| = O(\epsilon) \] (16)

for \( |x| \leq R \).

We will denote by \( N_t = N_t(\epsilon, R, P) \) the total number of nodes required in one dimension. Using a tensor product of this one-dimensional rule, \( O(N_t^d) = O((\log(1/\epsilon) + RP)^d) \) nodes are required in \( d \) dimensions. We refer the reader to the original paper [8] for additional details.
3.3 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,

$$U(x, t) \approx \frac{1}{(2\pi)^d} \sum_{j=1}^{N_1} \cdots \sum_{d=1}^{N_1} e^{-i a_j x} \hat{U}(s_j, t) w_{j_1} \cdots w_{j_d},$$  \hspace{1cm} (17)

where $s_j = (s_{j_1}, \ldots, s_{j_d})$.

The summation in (17) must be carried out for each evaluation point $x$. Direct evaluation at each of the $N^d$ points $x_n$ defined in Section 3.1 would require $O(N^d \cdot N^d)$ work. Likewise, the sum (12) must be evaluated at each of the $N_1^d$ spectral nodes, also requiring $O(N_1^d \cdot N^d)$ work.

Since the $s_j$ 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 nonuniform Fast Fourier Transforms (NUFFTs). Detailed descriptions and additional references can be found in the papers [2, 4–7, 13, 16]. Like the FFT, these algorithms evaluate sums of the form (17) and (12) in $O((N_1^d + N^d) \log (N_1^d + N^d))$ work, for any fixed precision.

4 The numerical scheme

An informal description of the algorithm follows, for second order accuracy in time.

---

**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 (17) according to Corollary 1.d) Compute weights $W_0, W_1$ from (10).

**Step 2: Transform initial data**

a) Initialize $\hat{U}(s, 0)$ at all $N_1^d$ spectral nodes by computing the forward NUFFT of $U(x, 0)$.
b) Compute $\hat{f}(s, 0)$ at all $N_1^d$ spectral nodes by computing the forward NUFFT of $f(x, 0)$.

**Step 3: For $m = 1, \ldots, M$**

a) Compute $\hat{f}(s, m\Delta t)$ at all $N_1^d$ spectral nodes by computing the forward NUFFT of $f(x, m\Delta t)$.
b) Update $\hat{U}(s, t)$ according to (7).

$$\hat{U}(s, m\Delta t) = e^{-|s|^2 \Delta t} \hat{U}(s, (m-1)\Delta t) + W_0(s, \Delta t) \hat{f}(s, m\Delta t) + W_1(s, \Delta t) \hat{f}(s, (m-1)\Delta t)$$

c) Compute $U(x, t)$ from $\hat{U}(s, t)$ using (17) and the NUFFT.

The modifications necessary for fourth order accuracy in time 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 the second and fourth order accurate schemes, so that their execution times are nearly identical. The fourth order scheme does, however, require twice as much storage since it is a four-level marching scheme.
5 Numerical results

We first test the performance of the algorithm in two dimensions with the exact solution

\[ u_{\text{exact}}(x_1, x_2, t) = e^{-\frac{(x_1-0.05)^2 + (x_2-0.15)^2}{4(t+0.01)}} + e^{-\frac{(x_1-0.1)^2 + (x_2-0.2)^2}{4x0.01}} \sin(3t). \]

It is straightforward to compute \( U_0(x_1, x_2) \) (to which only the first term contributes) and the inhomogeneous source term \( f(x_1, x_2, t) \) (to which only the second term contributes). We choose the computational domain to be \([-2, 2] \times [-2, 2]\); both \( U_0(x_1, x_2) \) and \( f(x_1, x_2, t) \) are negligible outside of it.

In Figure 1 we show the convergence of the algorithm in the \( L_2 \) norm using the linear and cubic product integration rules for (8). As expected, we see that the algorithm is second and fourth order accurate, respectively. We should note that the algorithm converges exponentially fast in space, since the data is infinitely differentiable.

For a tolerance of 10\(^{-10}\), a high-frequency cut-off of \( P = 46 \) was required, with \( N_1 = 466 \) quadrature nodes in the Fourier domain and \( N = 71 \) points in the physical space mesh. For a tolerance of 10\(^{-6}\), the required values were \( P = 25 \), \( N_1 = 222 \), and \( N = 41 \). For a tolerance of 10\(^{-3}\), \( P = 10 \) was sufficient, with \( N_1 = 104 \) and \( N = 21 \). Recall that in two dimensions, the tensor product rule requires \( N_1^2 \) total nodes. Finally, for the three tolerances above, 0.12, 0.03 and 0.01 seconds were required per time step on a laptop computer with a 2 GHz Pentium M processor.

![Convergence of 2nd and 4th order schemes](image1.png)

(a) Error

![Convergence of 2nd and 4th order schemes](image2.png)

(b) Relative Error

Figure 1: Convergence rate of algorithm.

5.1 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 [14]. 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

\[ u_t - \nabla^2 u = -\frac{30\phi^2(1-\phi)^2}{S} \phi, \]

\[ \vec{e}^2 \phi_t - \vec{e}^2 \nabla^2 \phi = \phi(1-\phi)(\phi - \frac{1}{2}) + 30\phi^2(1-\phi)^2 \vec{e} \alpha S u, \]
with unknowns \(\phi(x,t)\) and \(u(x,t)\). The quantity \(\phi(x,t)\) 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 \(T = T_M + \Delta T u\), where \(T_M\) is the equilibrium melting temperature and \(\Delta T\) 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 \(\phi\) and \(u\), we are really talking about the enlargement of the support of \(\phi - 1\) and \(u + 1\), respectively. The parameter \(S = \frac{\tilde{c}}{c}\) (\(c\) 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), \(\tilde{c}\) controls the interface thickness, \(\alpha\) and \(m\) are related to specific material properties and the length scale.

Typically, the coupled equations (18-19) are solved by finite difference methods, with artificially boundary conditions imposed at the computational boundary. (In [14], homogeneous Neumann conditions were used for both \(u\) and \(\phi\).) However, a potential difficulty with this approach lies in the difference in the diffusion characteristics of \(u\) and \(\phi\) when the under-cooling is moderate. For example, there are materials for which the typical under-cooling parameter is in the range \(S = 0.002 - 0.1\) [12]. 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 the incorrect boundary condition for \(u\) will pollute the result for both \(u\) and \(\phi\) as the simulation continues. In [14] 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 our Fourier representation-based algorithm solves the free space heat equation without the need for artificial boundary conditions, requiring only that the supports of the initial data and the source are contained within the computational boundary, we use it to solve (18) for \(u\) (or more precisely, \(u + 1\)). Note that the source for (18) 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 \(u\) is contained in the computational domain at all times is satisfied. The computational domain, of course, also needs to contain the support of \(u + 1\) at the start of the simulation.

With these requirements satisfied, our method for computing \(u\) in the computational domain (hence the accuracy of the computed source term in (19) for \(\phi\)) is unaffected by the diffusion of \(u + 1\) out of the computational boundary during the simulation. The equation (19) is solved with a simple implicit finite difference method (the backward Euler method in time and the standard five-point stencil for the discrete approximation of the Laplacian). The corresponding linear system is solved using FISHPAK [1].

Numerical results are shown in Figure 2 with a time step of \(\Delta t = 0.0005\). The relevant parameters are \(S = 0.05\), \(\alpha = 400\), \(m = 0.035\), \(\tilde{c} = 0.025\). In the first column of Figure 2, we show the contour plots of \(u\) and \(\phi\) at various times computed with the finite difference method for both equations (18) and (19), using the computational domain \([0,5] \times [0,5]\). (The nonlinear source terms are treated explicitly - that is, their value is taken at time \(t\) when marching from time \(t\) to \(t + \Delta t\).) The colors correspond to the values of \(u\), the black lines are four contours curves for \(\phi\), corresponding to the values 0.01, 0.1, 0.5, 0.99 (thickest), respectively. Again, \(\phi \equiv 0\) represents bulk solid and \(\phi \equiv 1\) represents bulk liquid. In the second column of Figure 2 we show the results from using the Fourier-based algorithm for (18), again on the computation domain \([0,5] \times [0,5]\). Finally, in the third column of Figure 2, we reverted to using the finite difference method for both (18) and (19) but enlarged the computational domain to \([-2.5, 7.5] \times [-2.5, 7.5]\). Clearly, the results from the hybrid approach agree with those from the finite difference method on the enlarged domain. The finite difference method on the original computational domain is already quite inaccurate at time \(t = 4\).

In the present case, the advantages of the Fourier/integral representation are modest. The longer the simulation time, of course, the greater the power of the spectral representation.
6 Conclusions and generalizations

We have described a simple Fourier-based method for the solution of the heat equation in free space with smooth data. It allows for accurate long-time simulations without the need for artificial boundary conditions on a finite computational domain. Modest improvements in the performance of the scheme can be obtained by careful tuning of the quadrature rule used for the inverse Fourier transform.

More important, however, is 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 [8, 10]. 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

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

where

$$u_H(x, t) = \int_{t - \delta}^{t} \int_{\mathbb{R}^d} k(x - y, t - \tau) f(y, \tau) \, dy \, d\tau,$$

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

The spectral representation is used only for $u_H$ and a quadrature approximation is used for $u_L$. The papers [8, 10] present the relevant framework (although for layer potentials rather than volume potentials). Work in this direction is in progress and will be reported at a later date.

References


Figure 2: Simulation of phase field model. FD=finite difference. IR=using integral representation. Results from the integral representation approach on the original computational domain \((0, 5) \times (0, 5)\) agree with those from the finite difference approach on an enlarged domain \((-2.5, 7.5) \times [-2.5, 7.5]\).