Noname manuscript No.
(will be inserted by the editor)

Fast High-Order Integral Equation Methods for Solving Boundary Value Problems of Two Dimensional Heat Equation in Complex Geometry

Shaobo Wang · Shidong Jiang · Jing Wang

the date of receipt and acceptance should be inserted later

Abstract Efficient high-order integral equation methods have been developed for solving boundary value problems of the heat equation in complex geometry in two dimensions. First, the classical heat potential theory is applied to convert such problems to Volterra integral equations of the second kind via the heat layer potentials, where the unknowns are only on the space-time boundary. However, the heat layer potentials contain convolution integrals in both space and time whose direct evaluation requires $O(N_S^2 N_T^2)$ work and $O(N_S N_T)$ storage, where N_S is the total number of discretization points on the spatial boundary and N_T is the total number of time steps.

In order to evaluate the heat layer potentials accurately and efficiently, they are split into two parts - the local part containing the temporal integration from $t-\delta$ to t and the history part containing the temporal integration from 0 to $t-\delta$. The local part can be dealt with efficiently using conventional fast multipole type algorithms. For problems with complex *stationary* geometry, efficient separated sum-of-exponentials (SOE) approximations are constructed for the heat kernel and used for the evaluation of the history part. Here all local and history kernels are compressed only once. The resulting algorithm is very efficient with quasilinear complexity in both space and time for both interior and exterior problems. For problems with complex *moving* geometry, the spectral Fourier approximation is applied for the heat kernel and nonuniform FFT (NUFFT) is used to speed up the evaluation of the history part of heat layer potentials. The performance of both algorithms is demonstrated with several numerical examples.

S. Wang
Department of Mathematical Sciences, New Jersey Institute of Technology, Newark, New Jersey 07102. E-mail: sw228@njit.edu

S. Jiang
Department of Mathematical Sciences, New Jersey Institute of Technology, Newark, New Jersey 07102. E-mail: shidong.jiang@njit.edu

J. Wang
School of Information Science & Engineering, Yunnan University, Kunming, Yunnan, 650091, China; Department of Mathematical Sciences, New Jersey Institute of Technology, Newark, New Jersey 07102. E-mail: jing.wang.guan@ynu.edu.cn

Keywords heat equation; integral equation methods; high-order methods; heat kernels; sum-of-exponentials approximation; nonuniform FFT.

Mathematics Subject Classification (2000) 30E15, 35K05, 35K08, 45D05, 65E05, 65R10, 80A20

1 Introduction

In this paper, we consider the boundary value problem of heat equation in two dimensions:

$$\begin{cases} U_t(x,t) = \Delta U(x,t), \quad (x,t) \in \prod_{\tau=0}^{T} \Omega(\tau), \\ U(x,0) = 0, \quad x \in \Omega_0, \\ U(x,t) = f(x,t), \quad (x,t) \in \prod_{\tau=0}^{T} \Gamma(\tau), \end{cases} \tag{1}$$

where $x \in \mathbb{R}^2$, $\Omega(\tau) \in \mathbb{R}^2$ is a domain at $t = \tau$ which could be multiply connected and/or unbounded, and $\Gamma(\tau)$ is its boundary. This problem has direct applications in solidification, melting, crystal growth, and dislocation dynamics [23,25,38]. In order to solve (1) numerically, we first apply standard heat potential theory to convert it to a Volterra integral equation of the second kind using the heat double layer potential. As compared with the commonly used finite difference and finite element methods, integral formulation have a number of advantages for solving this type of problems. First, the unknown density is only on the space-time boundary $\prod_{\tau=0}^{T} \Gamma(\tau)$. This reduces the dimension of the problem by one and thus the total number of unknowns by a large extent. Second, it is easier to design high-order discretization scheme for the boundary and the unknowns on the boundary rather than the whole volume and the unknowns in the whole volume, especially in the case of complex geometry. In two dimensions, one only needs to discretize the boundary curves instead of the 2D domain. Third, it is easier to design high-order marching scheme in time using integral formulation in the case of moving geometry. Fourth, the integral formulation leads to a well-conditioned linear system which requires a constant number of iterations to solve and the associated marching scheme is unconditionally stable, while finite difference/finite element methods either require certain restriction on the Courant number for explicit schemes or need good preconditioners for solving the linear system for implicit schemes. Finally, for exterior problems, there is no need to design artificial boundary conditions to truncate the computational domain when the integral formulation is used. We would like to remark here that finite difference and finite element methods are very general and broadly applicable to a much wider class of problems, including variable coefficient problems, advection diffusion equation, and nonlinear problems, even though integral equation methods seem to be the method of choice for the particular problem we study here.

However, the heat layer potentials contain the convolution integrals in both space and time. A straightforward way to evaluate heat potentials at a sequence of time steps $t_n = n\Delta t$, for $n = 1, \ldots, N_T$, clearly requires an amount of work of the order $O(N_T^2 N_S^2)$, where N_S denotes the number of points in the discretization of the spatial boundary. When direct methods are used in the absence of

fast algorithms, it is difficult to argue that integral equation methods would be methods of choice for large-scale simulation. In the last three decades, a variety of schemes have been developed to overcome this obstacle. Many of these schemes use discrete Fourier methods to represent the history part and FFT to speed up the computation; see, for example, [3, 16, 23, 38]. However, these schemes are applicable only in regular domains and somewhat difficult to treat complex geometry. In [40], a space-time "fast-multipole-like" method is developed to overcome the cost of history-dependence for three dimensional problems. But it used a first kind integral equation formulation and a low order scheme for spatial and temporal discretization. The method also involves a hierarchical decomposition of the entire space-time domain and thus has a large prefactor in its computational complexity when high accuracy is required.

In this paper, we develop two high-order integral equation methods for solving (1). Both methods can be made arbitrarily high order in both space and time, although we use the scheme that is 16th order in space and 4th order in time for illustration. Both methods start from the splitting of the heat layer potentials into two parts - the local part that contains the temporal integration from $t-\delta$ to t and the history part that contains the temporal integration from 0 to $t-\delta$. Here δ is usually of the same order as the time step size Δt. For the local part, product integration [32] is applied on the temporal integral to convert it to a sum of several spatial convolution integrals where the so-called local kernels have logarithmic singularity. These weakly singular integrals are discretized via high-order quadratures [1] and the resulting discrete summations can then be evaluated via fast algorithms.

Two methods diverge in the treatment of the history part. Our first method is built upon an efficient sum-of-exponentials approximation to the heat kernel in [24]. By allowing general complex exponentials instead of purely oscillatory Fourier modes, the approximation can be made much more efficient. Indeed, the approximation is valid in the entire spatial domain and the number of exponentials depends only logarithmically on $\frac{T}{\delta}$, or equivalently, N_T. The approximation also separates the temporal and spatial variables so that fast algorithms in both space and time are more or less straightforward to design. However, the approximation does not separate the source point and the target point. Thus, it is only applicable for *stationary* geometry.

Our second method approximates the heat kernel via a Fourier spectral representation [13] that is valid for $t \geq \delta$ and x in a bounded domain. We then apply nonuniform FFT (see, for example, [12] and references therein) to speed up the calculation of the resulting discrete summation. The scheme has been applied to solve the heat equation in free space with smooth compactly supported data in [31]. The Fourier spectral representation of the heat kernel, however, is valid only in a bounded spatial domain and it requires excessively large number of Fourier modes when δ is very small. On the other hand, it has the advantage that the Fourier approximation of the heat kernel completely separates the temporal variable and the spatial variable, and the source point and the target point. This makes it very suitable for dealing with *moving* geometry, when the underlying spatial domain is not very large and the time step is not very small.

The rest of the paper is organized as follows. In Section 2, we review analytical apparatus that are needed for our algorithms. In Section 3, we discuss in detail two algorithms that we have developed for solving the boundary value problems

of the heat equation in two dimensions. The performance of these algorithms is demonstrated via several numerical examples in Section 4. Finally, we conclude the paper with further discussion and future research directions.

2 Preliminaries

In this section, we review analytical apparatus to be used in our numerical algorithms.

2.1 Potential Theory for the Heat Equation

The Green's function for the heat equation in two dimensions is

$$G(x,t) = \frac{1}{4\pi t} e^{-\frac{|x|^2}{4t}}, \quad x \in \mathbb{R}^2. \tag{2}$$

Suppose that σ is a function on $\prod_{\tau=0}^{T} \Gamma(\tau)$. Then the single layer potential is defined by the formula

$$S[\sigma](x,t) = \int_0^t \int_{\Gamma(\tau)} G(x-y, t-\tau)\sigma(y,\tau) ds_y d\tau, \tag{3}$$

and the double layer potential is defined by the formula

$$D[\sigma](x,t) = \int_0^t \int_{\Gamma(\tau)} \frac{\partial}{\partial n_y} G(x-y, t-\tau)\sigma(y,\tau) ds_y d\tau\,. \tag{4}$$

Here n_y is the outward unit normal vector at a point y on $\Gamma(\tau)$. The single layer potential is continuous across the boundary, but its normal derivative satisfies the jump relation

$$\lim_{z \to x^{\pm}} \frac{\partial S[\sigma]}{\partial n_x}(z,t) = \mp\frac{1}{2}\sigma(x,t) + \int_0^t \int_{\Gamma(\tau)} \frac{\partial}{\partial n_x} G(x-y, t-\tau)\sigma(y,\tau) ds_y d\tau, \tag{5}$$

where $z \to x^{\pm}$ implies that z approaches x on $\Gamma(t)$ nontangentially from the exterior $(+)$ or the interior $(-)$ side, respectively. The double layer potential satisfies the jump relation

$$\lim_{z \to x^{\pm}} D[\sigma](z,t) = \pm\frac{1}{2}\sigma(x,t) + D[\sigma](x,t). \tag{6}$$

If we represent the solution to (1) via a double layer potential, then the jump relation leads to the following second kind Volterra integral equation for the unknown density σ for the interior Dirichlet problem:

$$-\frac{1}{2}\sigma(x,t) + D[\sigma](x,t) = f(x,t). \tag{7}$$

Remark 1 For the exterior Dirichlet problem, the diagonal term will change sign. For the Neumann or Robin problem, the solution is represented via a single layer potential. The initial potential defined by the formula

$$I[U_0](x,t) = \int_{\Omega_0} G(x-y,t)U_0(y)dy \tag{8}$$

can be used to treat the nonzero initial data U_0; and the volume potential defined by the formula

$$V[h](x,t) = \int_0^t \int_{\Omega(\tau)} G(x-y,t-\tau)h(y,\tau)dyd\tau \tag{9}$$

can be used to treat the inhomogeneous term $h(x,t)$ on the right side of the heat equation. Note that we only need to evaluate the initial potential and the volume potential, i.e., the only unknown is the layer density defined on the space-time boundary. The readers are referred to [13, 19, 29, 43] for details.

Remark 2 In this paper, we assume that the boundary itself and the boundary data are both smooth so that the full potential of high-order quadrature schemes developed here can be realized. However, it is straightforward to extend our algorithm to treat, say, piecewise smooth curves in space by slighly modifying the spatial quadrature we use. Moreover, our integral equation formulation does not reduce the regularity of the solution. Indeed, the regularity of the solution with respect to the boundary data was studied in detail using the technique of layer potentials (see, for example, [5, 6, 9, 29]). In particular, we would like to point out [29, §9.2] that the double layer potential operator is compact from $C(\Gamma \times [0,T])$ into $C(\Gamma \times [0,T])$ and that the double layer potential is continuous on $\bar{\Omega} \times [0,T]$ provided the continuous density satisfies $\sigma(\cdot, 0) = 0$.

2.2 Spectral Fourier Approximation for the Heat Kernel in Free Space

Spectral Fourier approximation for the heat kernel has been studied in [13]. For the 1D heat kernel $G_1(x,t)$, the starting point is the following well-known Fourier representation:

$$G_1(x,t) = \frac{e^{-x^2/4t}}{\sqrt{4\pi t}} = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-k^2 t}e^{ikx}dk. \tag{10}$$

In [13], it is shown that in order to approximate G_1 by a discrete Fourier series for all $t \geq \delta > 0$ and $|x| \leq R$ within an absolute error ϵ, i.e.,

$$\left| G_1(x,t) - \sum_{i=1}^{N_F} w_i e^{-k_i^2 t}e^{ik_i x} \right| \leq \epsilon, \quad |x| \leq R, \quad t \geq \delta, \tag{11}$$

the number of Fourier modes needed N_F is of the order:

$$N_F = O\left(\log\left(\frac{1}{\epsilon}\right) \left(\log\left(\frac{1}{\epsilon\sqrt{\delta}}\right)\right)^{1/2} \frac{R}{\sqrt{\delta}} \right). \tag{12}$$

The spectral Fourier approximation for the heat kernel in higher dimensions is obtained via the tensor product in [13].

2.3 Efficient Sum-of-exponentials Approximation for the Heat Kernel

The Fourier approximation of the heat kernel becomes very inefficient when the spatial domain is very large and/or the time step size is very small. In [24], an efficient sum-of-exponentials approximation is developed for the heat kernel in any dimensions. The construction is based on efficient sum-of-exponentials approximations for the 1D heat kernel and the power function $1/t^\beta$ ($\beta > 0$). To be more specific, it is shown in [24] that for any $0.1 > \epsilon > 0$ and $T \geq 1000\delta > 0$, the 2D heat kernel $G(x,t)$ admits the following approximation:

$$\tilde{G}(x,t) = \sum_{j=1}^{N_2} \tilde{w}_j e^{-\lambda_j t} \sum_{k=-N_1}^{N_1} w_k e^{s_k t} e^{-\sqrt{s_k}|x|} \tag{13}$$

such that

$$|G(x,t) - \tilde{G}(x,t)| < \frac{1}{t^{d/2}} \cdot \epsilon \tag{14}$$

for any $x \in \mathbb{R}^d$, $t \in [\delta, T]$. Here N_1 is of the order

$$O\left(\log\left(\frac{T}{\delta}\right)\left(\log\left(\frac{1}{\epsilon}\right) + \log\log\left(\frac{T}{\delta}\right)\right)\right), \tag{15}$$

and N_2 is of the order

$$O\left(\log\left(\frac{1}{\epsilon}\right)\left(\log\log\left(\frac{1}{\epsilon}\right) + \log\left(\frac{T}{\delta}\right)\right)\right). \tag{16}$$

It is worth noting that the approximation (14) is valid in the entire spatial domain and the number of complex exponentials depends only logarithmically on T/δ. In [43], it is shown that $N_1 = 47$ and $N_2 = 22$ for $t \in [10^{-3}, 1]$ and $\epsilon = 10^{-9}$.

3 Numerical Algorithms

In this section, we present fast algorithms for solving the second kind Volterra integral equation (7), which in turn solves (1) for the interior Dirichlet problem. The exterior Dirichlet problem can be solved in an almost identical manner.

3.1 Split of the Layer Potentials

It is convenient both analytically and numerically to decompose the double layer potential into two pieces: a *history* part D_H and a *local* part D_L. Let δ be a small positive parameter. We have

$$D[\sigma](x,t) = D_H[\sigma](x,t) + D_L[\sigma](x,t), \tag{17}$$

where

$$D_H[\sigma](x,t) = \int_0^{t-\delta} \int_\Gamma \frac{\partial}{\partial n_y} G(x-y, t-\tau)\sigma(y,\tau) ds_y d\tau \tag{18}$$

and

$$D_L[\sigma](x,t) = \int_{t-\delta}^{t} \int_\Gamma \frac{\partial}{\partial n_y} G(x-y, t-\tau)\sigma(y,\tau) ds_y d\tau\,. \tag{19}$$

3.2 Efficient Heat Solver with Stationary Geometry Using SOE Approximation

3.2.1 Explicit Expressions of the Local Kernels

In order to evaluate the local part accurately, we switch the order of integration and carry out the integration in time semi-analytically using product integration (see, for example, [26,32]). Assuming that the density σ is smooth in time, we expand it on $[t-\delta, t]$ for each y in the form

$$\sigma(y,\tau) = \sigma_0(y) + (t-\tau)\sigma_1(y) + \cdots + \frac{(t-\tau)^{p-1}}{(p-1)!}\sigma_{p-1}(y) + O((t-\tau)^p).$$

The functions $\sigma_0(y), \ldots, \sigma_{p-1}(y)$ are obtained from the function values $\sigma(y, t-j\Delta t)$ for $j = 0, \cdots, p-1$ via standard polynomial interpolation. In other words, we have

$$\sigma(y,\tau) = [1 \quad \frac{t-\tau}{\Delta t} \quad ... \quad \frac{(t-\tau)^{p-1}}{\Delta t^{p-1}}]\, M_p \begin{bmatrix} \sigma(y,t) \\ \sigma(y,t-\Delta t) \\ ... \\ \sigma(y,t-(p-1)\Delta t) \end{bmatrix} + O((t-\tau)^p), \quad (20)$$

where M_p is the coefficient matrix for the pth order polynomial interpolation. For example, for $p = 4$ we have

$$M_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{11}{6} & 3 & -\frac{3}{2} & \frac{1}{3} \\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{6} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{6} \end{bmatrix}. \quad (21)$$

Substituting (20) into equation (19) and changing the order of integration in time and space, we obtain

$$D_L[\sigma](x,t) = \int_\Gamma [D_{L_0} \quad D_{L_1} \quad ... \quad D_{L_{p-1}}]\, M_p \begin{bmatrix} \sigma(y,t) \\ \sigma(y,t-\Delta t) \\ ... \\ \sigma(y,t-(p-1)\Delta t) \end{bmatrix} ds_y \quad (22)$$

where the local kernels D_{L_k} are given by the formula

$$\begin{aligned} D_{L_k}(x,y) &= \int_{t-\delta}^{t} \frac{\partial}{\partial n_y} G_d(x-y, t-\tau)(t-\tau)^k \, d\tau \\ &= \frac{1}{\Delta t^k} \int_{t-\delta}^{t} \frac{(x-y)\cdot n_y}{8\pi(t-\tau)^{2-k}} e^{-\frac{|x-y|^2}{4(t-\tau)}} \, d\tau. \end{aligned} \quad (23)$$

The integrals in (23) can be evaluated analytically and we obtain the following explicit expressions for the local kernels

$$D_{L_k}(x,y) = \begin{cases} \frac{(x-y)\cdot n_y}{2\pi|x-y|^2}\, e^{-\rho}\,, & k=0, \\ \frac{(x-y)\cdot n_y}{8\pi\Delta t}\, \mathrm{Ei}(\rho)\,, & k=1, \\ \frac{(x-y)\cdot n_y \delta}{8\pi\Delta t^2}\, (e^{-\rho} - \rho\mathrm{Ei}(\rho))\,, & k=2, \\ \frac{(x-y)\cdot n_y \delta^2}{16\pi\Delta t^3}\, ((1-\rho)e^{-\rho} + \rho^2\mathrm{Ei}(\rho))\,, & k=3, \end{cases} \quad (24)$$

where $\rho = \frac{|x-y|^2}{4\delta}$ and Ei is the exponential integral function [37] defined by the formula

$$\mathrm{Ei}(x) = \int_x^\infty \frac{e^{-t}}{t}\, dt.$$

3.2.2 Discretization and Compression of the Spatial Integrals in the Local Part

It is easy to see that the local kernels $D_{L_k}(x, y)$ in (24) are at most logarithmically singular. There are many high-order quadratures for discretizing such weakly singular integrals. Here we use the 16th order Alpert quadrature in [1] to discretize the spatial integrals involving these local kernels. We would also like to avoid the $O(N_S^2)$ work that would be required by direct evaluation of the matrix-vector product. A large number of fast algorithms are now available to reduce the cost of this step to $O(N_S)$ or $O(N_S \log N_S)$. These include classical fast multipole methods (FMM) with analytical expansions [8,14,15], kernel-independent fast multipole methods [10,11,35,44], HSS and H-matrix methods [7,20], recursive skeletonization factorization [21,22], and hierarchical interpolative factorization [22]. All these fast algorithms have either linear or quasilinear complexity, but they differ in the range of applicability, the ease of implementation, and the prefactors in front. We use recursive skeletonization factorization (`rskelf`) developed in [22]. In order to compute the matrix-vector product Av with A a hierarchically compressible matrix, `rskelf` first factorizes A into a product of low rank sparse matrices in the compression stage. The compression stage usually is about ten times slower than one FMM for intrinsically one dimensional problems. However, the compression needs to be done only once. And after the compression is done, the apply stage (i.e., using the resulting factorization to compute Av) is much faster than one FMM due to the much smaller prefactor in front. For *stationary* geometry, it is clear that all matrices are independent of time and thus need to be compressed only once, which makes `rskelf` much cheaper than FMMs for time marching.

`Rskelf` requires fast access of matrix entries and matrix blocks A_{ts}, where t and s are both set of indices. Although we could apply `rskelf` directly on the matrix A resulted from the discretization of the spatial integrals using the Alpert quadrature, the compression stage would be much slower since the Alpert quadrature contains nonequispaced nodes near the diagonal. In order to speed up the compression stage, we split the matrix A into three parts:

$$A = A_{eq} + A_{alp} - A_{adj}, \tag{25}$$

where A_{eq} is a matrix obtained by using the equispaced trapezoidal rule to discretize the spatial integral with diagonal entries set to 0; A_{alp} is a matrix that contains the effect of the non-equispaced points in the Alpert quadrature; and A_{adj} is a banded matrix that contains the effect of the adjacent 18 equispaced points near the diagonal from the trapezoidal rule.

Only A_{eq} is compressed by `rskelf` and the matrix entries of A_{eq} can be evaluated quickly. The matrix compression and apply costs are both $O(N_S)$ with the apply step much faster than the compression step (very often at least 100 times faster). The matrix A_{alp} is actually not sparse since we use global spectral interpolation to interpolate the function values at those nonequispaced points to equispaced points. However, since there are only 30 nonequispaced points near the diagonal and one could apply NUFFT to speed up the global interpolation. The cost of the matrix-vector product $A_{alp}\sigma$ is $O(N_S \log(N_S))$. We remark here that one could use local high-order barycentric interpolation to reduce the cost of this step to $O(N_S)$. Finally, we need to subtract the action of A_{adj} since they are not in the Alpert quadrature. Evaluating $A_{adj}\sigma$ costs $O(N_S)$.

We summarize the evaluation of the local part in Algorithm 1.

Algorithm 1 Evaluation of the Local Part

Require: Suppose σ is the density function and each local kernel D_{L_k} is split into three parts: D_{eq}, D_{alp}, and D_{adj}. Evaluate $D_{L_k}[\sigma]$.
1: Compress D_{eq} and evaluate $D_{\text{eq}}\sigma$ via `rskelf`.
2: Evaluate $D_{\text{alp}}\sigma$ via 1D NUFFT.
3: Evaluate $D_{\text{adj}}\sigma$ directly by sparse matrix-vector product.
4: Set $D_{L_k}[\sigma]$ to $D_{\text{eq}}\sigma + D_{\text{alp}}\sigma - D_{\text{adj}}\sigma$.

Remark 3 In the case of *stationary* boundary, one could also compress the matrix inverse directly using fast direct solvers [4,21,28,33,34,36]. This will lead to an optimal algorithm for long-time simulations. When the number of time step is not too large, our method is more efficient since (a) the compression cost is reduced by a large factor; (b) the apply cost is much smaller than the compression cost; and (c) it takes very few number of iterations for GMRES to converge due to the second kind Volterra structure of the integral equation.

3.2.3 Evaluation of the History Part

For the history part, approximating the kernel by its sum-of-exponentials approximation and substituting it into (18), we obtain

$$\begin{aligned} D_H[\sigma](x,t) \approx & \int_0^{t-\delta} \int_\Gamma \sum_{j=1}^{N_2} \tilde{w}_j e^{-\lambda_j(t-\tau)} \\ & \sum_{k=-N_1}^{N_1} w_k e^{s_k(t-\tau)} e^{-\sqrt{s_k}|x-y|} [(x-y)\cdot n_y]\sigma(y,\tau) ds_y d\tau \qquad (26) \\ = & \sum_{j=1}^{N_2} \tilde{w}_j \sum_{k=-N_1}^{N_1} w_k H_{j,k}(x,t), \end{aligned}$$

where each history mode $H_{j,k}$ is given by the formula

$$H_{j,k}(x,t) = \int_0^{t-\delta} e^{(-\lambda_j+s_k)(t-\tau)} V_k(x,\tau) d\tau, \qquad (27)$$

with V_k given by the formula

$$V_k(x,\tau) = \int_\Gamma e^{-\sqrt{s_k}|x-y|} [(x-y)\cdot n_y]\sigma(y,\tau) ds_y. \qquad (28)$$

Here, we have interchanged the order of summation and integration.

For each fixed τ, $V_k(x,\tau)$ can be discretized using the trapezoidal rule to achieve spectral accuracy. This is because although each integral is not smooth, the kernel of the whole history part is smooth since the heat kernel is smooth for $\tau \in [0, t-\delta]$. The resulting discrete summation can again be computed via `rskelf`. The computational cost for this step is $\mathcal{O}(N_S)$ for each k. Once the V_k have been evaluated, each history mode $H_{j,k}$ can be computed recursively, as in [13,26]:

$$H_{j,k}(x,t+\Delta t) = e^{(-\lambda_j+s_k)\Delta t} H_{j,k}(x,t) + \int_{t-\delta}^{t+\Delta t-\delta} e^{(-\lambda_j+s_k)(t+\Delta t-\tau)} V_k(x,\tau) d\tau. \qquad (29)$$

Similar to local kernels, V_k can be expressed as

$$V_k(x,\tau) = [1 \quad ... \quad \frac{(t-\delta-\tau)^{p-1}}{\Delta t^{p-1}}]\, M_p \begin{bmatrix} V_k(x,t-\delta) \\ ... \\ V_k(x,t-\delta-(p-1)\Delta t) \end{bmatrix} + O(\Delta t^p). \quad (30)$$

Then

$$\begin{aligned} &\int_{t-\delta}^{t+\Delta t-\delta} e^{(-\lambda_j+s_k)(t+\Delta t-\tau)} V_k(x,\tau) d\tau = \\ &[D_{H_0} \quad D_{H_1} \quad ... \quad D_{H_{p-1}}]\, M_p \begin{bmatrix} V_k(x,t-\delta) \\ V_k(x,t-\delta-\Delta t) \\ ... \\ V_k(x,t-\delta-(p-1)\Delta t) \end{bmatrix} + O(\Delta t^p), \end{aligned} \quad (31)$$

where $D_{H_j} (j = 0, ..., p-1)$ are given by the formulas

$$\begin{aligned} D_{H_j} &= \int_{t-\delta}^{t+\Delta t-\delta} e^{(-\lambda_j+s_k)(t+\Delta t-\tau)} \frac{(t-\delta-\tau)^j}{\Delta t^j} d\tau \\ &= (k-1)^{j+1} \Delta t \begin{cases} \frac{1-e^{-q}}{q}, & j=0, \\ \frac{1-e^{-q}-qe^{-q}}{q^2}, & j=1, \\ \frac{2-2e^{-q}-2qe^{-q}-q^2e^{-q}}{q^3}, & j=2, \\ \frac{6-6e^{-q}-6qe^{-q}-3q^2e^{-q}-q^3e^{-q}}{q^4}, & j=3, \end{cases} \end{aligned} \quad (32)$$

where

$$q = (p-1)(\lambda_j - s_k)\Delta t.$$

(32) suffers from severe cancellation error when q is small. Therefore, for $q < 10^{-3}$ we evaluate it with Taylor expansions

$$D_{H_j} = (p-1)^{j+1} \Delta t \begin{cases} 1 - \frac{1}{2}q + \frac{1}{6}q^2 - \frac{1}{24}q^3 + \frac{1}{120}q^4, & j=0, \\ \frac{1}{2} - \frac{1}{3}q + \frac{1}{8}q^2 - \frac{1}{30}q^3 + \frac{1}{144}q^4, & j=1, \\ \frac{1}{3} - \frac{1}{4}q + \frac{1}{10}q^2 - \frac{1}{36}q^3 + \frac{1}{168}q^4, & j=2, \\ \frac{1}{4} - \frac{1}{5}q + \frac{1}{12}q^2 - \frac{1}{42}q^3 + \frac{1}{192}q^4, & j=3. \end{cases} \quad (33)$$

Equivalently, each history mode $H_{j,k}$ can be seen to satisfy a simple linear ordinary differential equation. This gives another way of evaluating each history mode in $O(1)$ operations at each time step for each x without using the recurrence relation (29). Since both N_1 and N_2 are $O(\log(T/\delta)) = O(\log(N_T))$ with N_T the total number of time steps, the computational cost for the evaluation of the history part at each time step is $O(N_S \log N_T + N_S \log^2 N_T)$, with storage requirement of the order $O(N_S \log^2 N_T)$. The computational cost for the history part for the entire simulation is

$$O(N_S N_T \log N_T + N_S N_T \log^2 N_T). \quad (34)$$

Remark 4 Since we use multistep method for time marching, the parameter δ that divides the local part and the history part actually changes for every time step in order to reduce the storage cost of the history part. For example, for the 4th order scheme, when $t = 4\Delta t$, the local part contains the temporal integral from $3\Delta t$ to

$4\Delta t$ and $\delta = \Delta t$. When $t = 5\Delta t$, the local part contains the temporal integral from $3\Delta t$ to $5\Delta t$. That is, δ is now $2\Delta t$ and each history mode is simply updated via the formula

$$H_{j,k}(x, t + \Delta t) = e^{(-\lambda_j + s_k)\Delta t} H_{j,k}(x, t) \tag{35}$$

instead of (29).

The algorithm is straightforward to parallelize since the computation of each history mode is independent of one another. Furthermore, the hierarchical fast algorithms used for evaluating each $V_k(x, \tau)$ are themselves amenable to parallelization, leading to further parallelization if necessary.

We summarize the evaluation of the history part in Algorithm 2.

Algorithm 2 Evaluation of the History Part Using the SOE Approximation

Require: Suppose σ is the density function. Assume that we are at the mth step in a pth order scheme. Evaluate the history part $D_H[\sigma]$.

1: **if** $mod(m - 2, p - 1) == 0$ **then**
2: Evaluate the spatial integral (28) via `rskelf`.
3: Evaluate the recurrence relation (29) for each history mode $H_{j,k}$.
4: **else**
5: Update each history mode $H_{j,k}$ using (35).
6: **end if**
7: Evaluate $D_H[\sigma]$ via (26).

3.2.4 The Full Algorithm

We now combine the results from previous sections to obtain the full algorithm for solving the heat equation with *stationary* boundary using the sum-of-exponentials approximation.

Denote the density $\sigma(x, t_m)$ at time step m as $\sigma_m(x)$. For the 4th order scheme in time, the integral equation that we need to solve at the mth step is:

$$-\frac{1}{2}\sigma_m + D_{L_0}[\sigma_m] = b_m - D_H[\sigma_H] - D_{L_1}[\sigma_{m-1}] - D_{L_2}[\sigma_{m-2}] - D_{L_3}[\sigma_{m-3}], \tag{36}$$

where $b_m = f(x, t_m)$ and σ_H contains the density vectors σ_0, ..., $\sigma_{m-\delta/\Delta t}$ with δ specified in Remark 4.

Since there are $O(1)$ local kernels and $O(\log N_T)$ history kernels to be compressed and it takes $O(N_S)$ to compress each kernel, the cost for the `rskelf` compression stage is

$$O(N_S \log N_T). \tag{37}$$

Due to the second kind Volterra structure of the integral equation, it takes about 6 iterations for GMRES to converge when the GMRES stopping threshold is set to 10^{-12}. Thus, the cost for solving the density σ at all times is

$$O(N_T N_S \log N_S + N_S N_T \log^2 N_T). \tag{38}$$

Here the $\log N_S$ factor is from the global interpolation using NUFFT, which can be removed if one uses high-order local interpolation to compute the density at

Algorithm 3 Efficient Heat Solver for *Stationary* Geometry Using the SOE Approximation

Require: Given N_S source points src, N_{targ} target points $targ$, total number of time steps N_T, time step Δt in a pth order scheme. Solve the density σ at all time steps and evaluate the solution U to the heat equation at the final time T for all target points.

1: Separate all local kernels into three parts D_{eq}, D_{alp}, and D_{adj}. Compress D_{eq} of all local kernels and all history kernels approximated by the SOE approximation using `rskelf`.
2: **for** $m = p$ to $N_T + 1$ **do** ▷ Solving the density σ
3: Compute the right hand side of the linear system b.
4: **if** $m > p$ **then**
5: Evaluate the history part D_H using Algorithm 2.
6: **end if**
7: Evaluate the local part $D_L(\sigma)$ using Algorithm 1.
8: Use GMRES to solve the linear system (36) and obtain σ_m.
9: **end for**
10: Evaluate the solution U at the final time for all target points.

the nonequispaced points in the Alpert quadrature. And the $\log^2 N_T$ factor is from the number of complex exponentials in the SOE approximation of the heat kernel. Note here that we have ignored the dependence on ϵ and the factor $\log\log N_T$ in (15) and (16). Similarly, evaluating the solution U costs

$$O(N_T(N_S + N_{targ})\log^2 N_T + (N_S + N_{targ})\log N_S). \tag{39}$$

Remark 5 For high-order scheme, $\sigma_1, \ldots, \sigma_{p-1}$ are obtained via a low-order, say, 2nd order scheme with smaller time step size. This is standard technique for multistep methods.

3.3 Efficient Heat Solver in Moving Geometry Using NUFFT

3.3.1 Accurate Evaluation of the Local Part

We review briefly the treatment of the local part in [32] for *moving* geometry. When the boundary is function of time, i.e., $\Gamma = \Gamma(\tau)$, the kernel of the double layer potential has the following explicit expression:

$$\frac{\partial G(x(t), y(\tau); t-\tau)}{\partial n_{y(\tau)}} = \frac{e^{-\frac{|x(t)-y(\tau)|^2}{4(t-\tau)}}}{8\pi(t-\tau)^2}[(x(t)-y(\tau))\cdot n_{y(\tau)}]. \tag{40}$$

The local part of the double layer potential is given by the formula

$$D_L[\sigma](x,t) = \int_{t-\delta}^{t}\int_{\Gamma(\tau)} \frac{e^{-\frac{|x(t)-y(\tau)|^2}{4(t-\tau)}}}{8\pi(t-\tau)^2}[(x(t)-y(\tau))\cdot n_{y(\tau)}]\sigma(y(\tau),\tau)ds_{y(\tau)}d\tau, \tag{41}$$

where one has to carry out the integration in space first since the boundary is now a function of time as well. We now switch the order of integration by using the boundary at the current time $\Gamma(t)$ for spatial integration, which involves a change

of variable in the spatial variable first. We obtain

$$\begin{aligned} D_L[\sigma](x,t) = \int_{\Gamma(t)} \int_{t-\delta}^{t} & \frac{e^{-\frac{|x(t)-y(\tau)|^2}{4(t-\tau)}}}{8\pi(t-\tau)^2}[(x(t)-y(\tau))\cdot n_{y(\tau)}] \\ & \sigma(y(\tau),\tau)\frac{ds_{y(\tau)}}{ds_{y(t)}}d\tau ds_{y(t)}. \end{aligned} \tag{42}$$

In order to carry out the product integration in time as in the stationary case, we decompose the exponential function as follows:

$$e^{-\frac{|x(t)-y(\tau)|^2}{4(t-\tau)}} = e^{-\frac{|x(t)-y(t)|^2}{4(t-\tau)}} \cdot e^{-\frac{|y(t)-y(\tau)|^2}{4(t-\tau)}} \cdot e^{-\frac{(x(t)-y(t))\cdot(y(t)-y(\tau))}{2(t-\tau)}}. \tag{43}$$

The first term on the right hand side of (43) is the same as the stationary case, while the second and third terms both contain the factor $\frac{(y(t)-y(\tau))}{(t-\tau)}$, which is a smooth function of τ when the boundary undergoes a smooth motion. Let

$$\mu(\tau) = e^{-\frac{|y(t)-y(\tau)|^2}{4(t-\tau)}} \cdot e^{-\frac{(x(t)-y(t))\cdot(y(t)-y(\tau))}{2(t-\tau)}}[(x(t)-y(\tau))\cdot n_{y(\tau)}]\sigma(y(\tau),\tau)\frac{ds_{y(\tau)}}{ds_{y(t)}}. \tag{44}$$

Then (42) can be written as follows:

$$D_L[\sigma](x,t) = \int_{\Gamma(t)} \int_{t-\delta}^{t} \frac{e^{-\frac{|x(t)-y(t)|^2}{4(t-\tau)}}}{8\pi(t-\tau)^2}\mu(\tau)d\tau ds_{y(t)}. \tag{45}$$

We may now proceed as in the *stationary* case. That is, approximate $\mu(\tau)$ by a polynomial of τ, convert the integration in time to a sum of products of local kernels D_{L_m} and $\mu(t-m\Delta t)$, and then use proper quadrature to discretize the spatial integrals.

Remark 6 One will need the value of $\mu(t)$, which may be obtained by replacing $\frac{(y(t)-y(\tau))}{(t-\tau)}$ with its limiting value $y'(t)$ as $\tau \to t$.

3.3.2 Evaluation of the History Part

For the history part, we use the spectral Fourier approximation of the heat kernel. Unlike [13] in which a tensor product is applied to obtain the spectral Fourier approximation of the heat kernel in two dimensions, we first write the Fourier representation of $G(x,t)$ in polar coordinates:

$$\begin{aligned} G(x,t) &= \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-(k_1^2+k_2^2)t}e^{i(k_1x_1+k_2x_2)}dk_1dk_2 \\ &= \frac{1}{4\pi^2}\int_0^{2\pi}\int_0^{\infty} e^{-k^2t}e^{ik(x_1\cos(\theta)+x_2\sin(\theta))}kdkd\theta. \end{aligned} \tag{46}$$

We then use the generalized Gaussian quadrature to construct an optimal quadrature along the radial direction, and the trapezoidal rule discretize the integration along the azimuthal direction which achieves spectral accuracy for smooth periodic integrals. Altogether, we need 21600 Fourier modes to achieve 13-digit accuracy for $t \geq \delta = 10^3$ and $|x| \leq R = 1$. If the tensor product in [13] were used, one would need about 700 for each direction in the Fourier domain and 700^2 (i.e., about half

a million) Fourier modes to approximate G in the same region. In any case, we have the following spectral Fourier representation for G:

$$G(x,t) \approx \frac{1}{4\pi^2} \sum_{j=1}^{N_F} w_j e^{-|\xi_j|^2 t} e^{i\xi_j \cdot x}. \tag{47}$$

The Fourier representation of the double layer kernel can be obtained by replacing t, x with $t-\tau$, $x-y$, respectively, and then differentiating the resulting expression with respect to y. We have

$$\frac{\partial G(x(t), y(\tau); t-\tau)}{\partial n_{y(\tau)}} \approx -\frac{i}{4\pi^2} \sum_{j=1}^{N_F} w_j e^{-|\xi_j|^2 (t-\tau)} e^{i\xi_j \cdot (x-y)} \left(\xi_j \cdot n_{y(\tau)}\right). \tag{48}$$

Substituting the above approximation into the history part of the double layer potential, we obtain

$$\begin{aligned} D_H[\sigma](x,t) &= \int_0^{t-\delta} \int_{\Gamma(\tau)} \frac{\partial G(x(t), y(\tau); t-\tau)}{\partial n_{y(\tau)}} \sigma(y(\tau),\tau) ds_{y(\tau)} d\tau \\ &\approx -\frac{i}{4\pi^2} \sum_{j=1}^{N_F} \int_0^{t-\delta} \int_{\Gamma(\tau)} w_j e^{-|\xi_j|^2 (t-\tau)} e^{i\xi_j \cdot (x-y)} \\ &\quad \left(\xi_j \cdot n_{y(\tau)}\right) \sigma(y(\tau),\tau) ds_{y(\tau)} d\tau \\ &= -\frac{i}{4\pi^2} \sum_{j=1}^{N_F} w_j e^{i\xi_j \cdot x(t)} \int_0^{t-\delta} e^{-|\xi_j|^2 (t-\tau)} H_j(\tau) d\tau, \end{aligned} \tag{49}$$

where H_j is defined by the formula

$$H_j(\tau) = \int_{\Gamma(\tau)} e^{-i\xi_j \cdot y(\tau)} \left(\xi_j \cdot n_{y(\tau)}\right) \sigma(y(\tau),\tau) ds_{y(\tau)}, \quad j = 1, \ldots, N_F. \tag{50}$$

The integrals in (50) can be discretized via the trapezoidal rule with spectral accuracy. After that, all H_j $(j = 1, \ldots, N_F)$ can be evaluated via type-3 NUFFT with $O((N_S + N_F)\log(N_S + N_F))$ cost. The temporal integral can be evaluated via standard recurrence relation as in the stationary case with $O(N_F)$ cost for each time step. Finally, we may apply type-3 NUFFT again to evaluate the summation in (49) with $O((N_S + N_F)\log(N_S + N_F))$ cost. The algorithm is spectrally accurate and has $O(N_T(N_S + N_F)\log(N_S + N_F))$ complexity for the whole simulation.

4 Numerical Results

We have implemented the SOE algorithm with OpenMP for parallelizing the compression of local and history kernels and the evaluation of history kernels. For the NUFFT based algorithm, we use the library [30] for NUFFT and the code is entirely sequential. We remark here that one could use recently developed FINUFFT [2] to achieve better efficiency and parallelization of the NUFFT, hence speeding up the NUFFT based algorithm significantly. In this section, we illustrate the performance of the two algorithms outlined in the preceding section via

several numerical examples. All numerical experiments shown below were carried out using the scheme with the 4th order in time and 16th order in space. In the tables and figures presented below, Δt is the time step size; N_T is the total number of time steps; N_S is the total number of discretization points in space; E is the relative L^2 error of the numerical solution at the final time; r is the ratio of the relative errors of two consecutive runs. T is the total simulation time in seconds; T_f is the total factorization time for Algorithm 2 using the SOE approximation (where all kernels need to be compressed only once); and T_m is the total time for time marching in Algorithm 2. We generate boundary data by placing point heat sources outside the computational domain, and all errors are computed against analytical solutions at 20 target points inside the computational domain. Output and timings are obtained on a 60-core machine with each core a 2.50GHz Intel Xeon E7-4880 CPU with 38.4MB of cache. The SOE algorithm was run with the number of threads set to 50.

4.1 Efficient Heat Solver for *Stationary* Geometry Using the SOE Approximations

Example 1 *Exterior Dirichlet Problem with the Boundary Consisting of a 32-gram.*

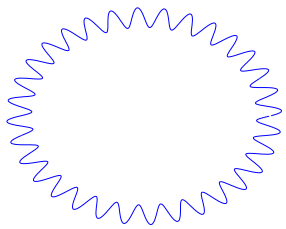

Fig. 1 32-gram boundary curve for Example 1.

We consider the exterior Dirichlet problem with the boundary consisting of a 32-gram shown in Figure 1. Here the boundary curve is roughly of size $R = 4$ and the center locates at $(0, 0)$.

We first check the order of accuracy of our algorithm. The left panel of Table 1 lists the relative L^2 error versus N_S, showing the high-order in space. The right panel lists the time step size Δt, the relative L^2 error E, and the ratio r of the relative errors of two consecutive runs. Here N_S is set to $6,250$ so that the error due to spatial discretization is negligible. We observe the convergence rate is roughly consistent with the fourth order accuracy in time (slightly better because of the smoothing effect of the heat kernel).

We now check the complexity of our algorithm. The left panel of Figure 2 shows the total CPU time for different N_S while N_T is fixed, demonstrating that the algorithm has almost linear complexity in N_S. The right panel of Figure 2 shows the total factorization time, the total marching time versus N_T when $N_S = 6,250$ is fixed. Since the geometry is stationary, the factorization needs to be carried

N_S	E
125	3.37e-2
250	1.35e-4
500	4.02e-6
1000	2.48e-9

Δt	E	r
2.00e-1	1.07e-2	
1.00e-1	6.65e-4	16.1
5.00e-2	1.76e-5	37.5
2.50e-2	2.73e-7	64.7
1.25e-2	2.47e-9	110

Table 1 Order of Accuracy for Example 1. Left: Accuracy in space - Relative L^2 error versus N_S. Here $N_T = 80$ and $\Delta t = 0.0125$ are fixed. Right: Accuracy in time - Relative L^2 error versus Δt. Here $N_S = 6,250$ is fixed.

out only once and its cost T_f is thus independent of N_T. On the other hand, we observe that the matching time T_m is almost linear with respect to N_T.

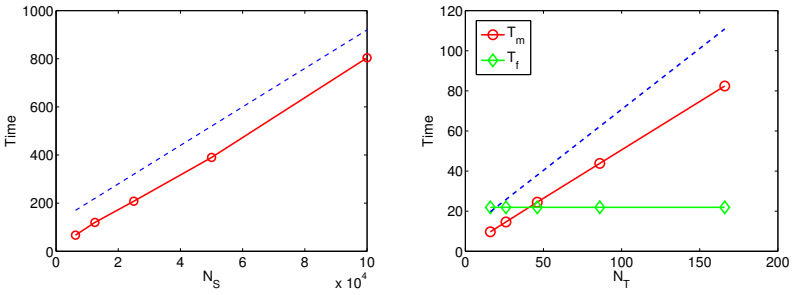

Fig. 2 Timing results for Example 1. Left: Total computational time T versus N_S. Here $N_T = 80$ and $\Delta t = 0.0125$ are fixed. Red circles represent the numerical results, while the blue dashed line represents $O(N_S)$ scaling. Right: Timing results with respect to N_T. Here $N_S = 6,250$ is fixed. Red circles represents the total matching time; green circles represent the total factorization time; and the blue dashed line represents $O(N)$ scaling. The total factorization time $T_f = 21.9s$ is the same for all different time step sizes.

Example 2 *Interior Dirichlet Problem with the Boundary Consisting of One Outer Circle and* 16 *Inner Hexagrams.*

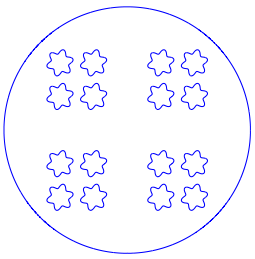

Fig. 3 Example 2: boundary consisting of one outer circle and 16 inner hexagrams.

We consider the interior Dirichlet problem with the boundary consisting of one outer circle and sixteen inner hexagrams shown in Figure 3. Here the outer circle is centered at the origin with radius 5.5. The size of each hexagram is about 0.5. The centers of the hexagrams are at (c_1, c_2) with c_1, c_2 taken from ± 1.5 and ± 3.

N_S	E
850	4.52e-3
1700	1.28e-5
2550	6.96e-7
3400	7.87e-9

Δt	E	r
2.00e-1	6.93e-3	
1.00e-1	4.69e-4	14.7
5.00e-2	4.11e-5	11.4
2.50e-2	1.49e-7	274
1.25e-2	6.52e-9	22.9

Table 2 Order of Accuracy for Example 2. Left: Accuracy in space - Relative L^2 error versus N_S. Here $N_T = 80$ and $\Delta t = 0.0125$ are fixed. Right: Accuracy in time - Relative L^2 error versus Δt. Here $N_S = 12,750$ is fixed.

Table 2 illustrates that the order of accuracy of the scheme in both space and time for Example 2. And Figure 4 shows the timing results for this example.

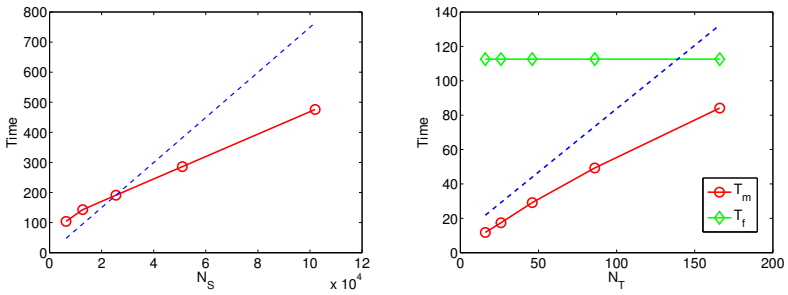

Fig. 4 Timing results for Example 2. Left: Total computational time T versus N_S. Here $N_T = 80$ and $\Delta t = 0.0125$ are fixed. Red circles represent the numerical results, while the blue dashed line represents $O(N_S)$ scaling. Right: Timing results with respect to N_T. Here $N_S = 12,750$ is fixed. Red circles represents the total matching time; green circles represent the total factorization time; and the blue dashed line represents $O(N)$ scaling. The total factorization time $T_f = 106.4s$ is the same for all different time step sizes.

Example 3 *Interior Dirichlet Problem with the Boundary Consisting of One Outer Cirle and* 64 *Inner Ellipses.*

We consider the interior Dirichlet problem with the boundary consisting of one outer circle and sixty-four inner ellipses shown in Figure 5. Here the outer circle is centered at the origin with radius 6.5. The major axis a and minor axis b of each ellipse are 0.3 and 0.2, respectively. The centers of the ellipses are at (c_1, c_2) with c_1, c_2 taken from $\pm 1, \ldots, \pm 4$.

Table 3 illustrate the order of accuracy of the algorithm in both space and time for Example 3. And Figure 6 presents the timing results for this example. The sublinear complexity with respect to N_S is due to the fact that the interaction rank between well-separated blocks is more or less independent of N_S and `rskelf` achieves higher compression rate as N_S increases.

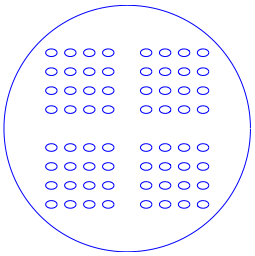

Fig. 5 Example 3: boundary consisting of one outer circle and sixty-four inner ellipses.

N_S	E
4500	2.02e-3
9100	1.87e-5
18200	6.52e-9

Δt	E	r
2.00e-1	6.58e-3	
1.00e-1	3.77e-4	17.4
5.00e-2	3.24e-5	11.6
2.50e-2	8.19e-8	396
1.25e-2	6.61e-9	12.4

Table 3 Order of Accuracy for Example 3. Left: Accuracy in space - Relative L^2 error versus N_S. Here $N_T = 80$ and $\Delta t = 0.0125$ are fixed. Right: Accuracy in time - Relative L^2 error versus Δt. Here $N_S = 26,000$ is fixed.

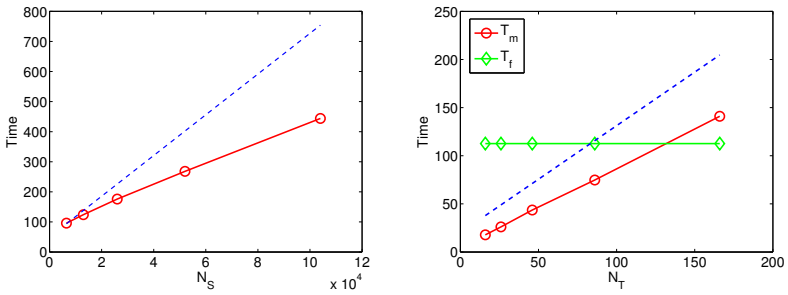

Fig. 6 Timing results for Example 3. Left: Total computational time T versus N_S. Here $N_T = 80$ and $\Delta t = 0.0125$ are fixed. Red circles represent the numerical results, while the blue dashed line represents $O(N_S)$ scaling. Right: Timing results with respect to N_T. Here $N_S = 26,000$ is fixed. Red circles represents the total matching time; green circles represent the total factorization time; and the blue dashed line represents $O(N)$ scaling. The total factorization time $T_f = 112.6s$ is the same for all different time step sizes.

4.2 Efficient Heat Solver for *Moving* Geometry Using the Spectral Fourier Approximation

Example 4 *Interior Dirichlet Problem with the Boundary Enclosed by One Outer Circle and One Inner Moving Decagram.*

We consider the interior Dirichlet problem with the boundary consisting of one outer circle and one inner moving decagram shown in Figure 7. Here, the radius of the outer circle is 0.7 and the centers of the circle and decagram are both at $(0, 0)$ at the initial time. The size of the inner decagram is roughly 0.1 and it moves

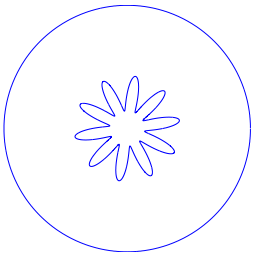

Fig. 7 Example 4: boundary consisting of one outer circle and one inner moving decagram.

towards the positive y-axis with the velocity of 0.01. At the final time, the center of the decagram is at $(0, 0.1)$.

N_S	E
200	5.09e-3
400	2.37e-5
800	9.51e-10

Δt	E	r
2.00e-1	1.93e-3	
1.00e-1	7.54e-5	25.5
5.00e-2	5.21e-7	144
2.50e-2	3.88e-8	13.4
1.25e-2	1.69e-9	22.9

Table 4 Order of Accuracy for Example 4. Left: Accuracy in space - Relative L^2 error versus N_S. Here $N_T = 80$ and $\Delta t = 0.0125$ are fixed. Right: Accuracy in time - Relative L^2 error versus Δt. Here $N_S = 2,000$ is fixed.

Table 4 illustrates that the scheme has high-order in space and slightly better than fourth order in time. Figure 8 shows the timing results for this example.

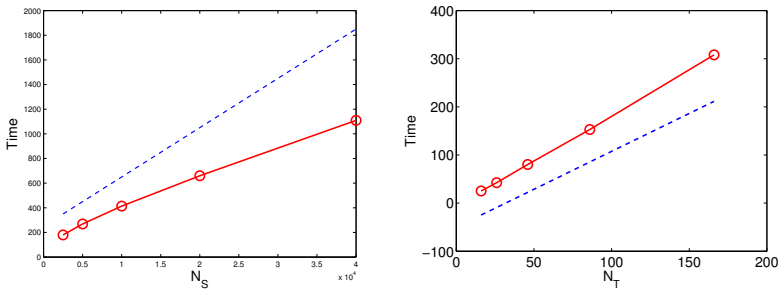

Fig. 8 Timing results for Example 4. Left: Total computational time T versus N_S. Here $N_T = 80$ and $\Delta t = 0.0125$ are fixed. Right: Total computational time T versus N_T. Here $N_S = 2,000$ is fixed. Red circles represent the numerical results, while the blue dashed line represents $O(N_S)$ scaling.

Example 5 *Interior Dirichlet Problem with the Boundary Consisting of One Outer Circle and* 64 *Inner Oscillating Circles.*

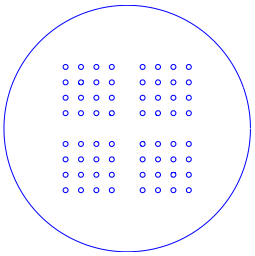

Fig. 9 Example 5: boundary consisting of one outer circle and 64 inner oscillating circles.

We consider the interior Dirichlet problem with the boundary consisting of one outer circle and 64 inner oscillating circles shown in Figure 9. Here, the outer circle is centered at the origin with radius 0.5. The radius of each inner circle is 0.01. The centers of the inner circle are initially at (c_1, c_2) with c_1, c_2 chosen from $\pm 0.25, \pm 0.185, \pm 0.125, \pm 0.0625$. The centers are then oscillating as $(c_1 + v\cos(\tau),\ c_2 + v\sin(\tau))$, where $v = 0.001$ and $\tau \in (0, 10]$ is the temporal variable.

N_S	E
780	1.37e-6
1300	4.14e-8
1820	5.39e-9
2340	2.80e-10

Δt	E	r
2.00e-1	1.58e-4	
1.00e-1	3.68e-6	42.9
5.00e-2	6.83e-8	53.9
2.50e-2	7.76e-9	8.79
1.25e-2	1.29e-9	6.01

Table 5 Order of Accuracy for Example 5. Left: Accuracy in space - Relative L^2 error versus N_S. Here $N_T = 80$ and $\Delta t = 0.0125$ are fixed. Right: Accuracy in time - Relative L^2 error versus Δt. Here $N_S = 6,500$ is fixed.

Table 5 presents the accuracy analysis for Example 5. Figure 10 shows the timing results for this example.

5 Conclusions and Further Discussions

In this paper, we have developed two fast high-order numerical algorithms for solving the boundary value problems of the heat equation in two dimensions. For problems with complex *stationary* geometry, we use the SOE approximation for the heat kernel and compress all local and history kernels only once. The resulting algorithm is very efficient with quasilinear complexity and is capable of handling both interior and exterior problems. For problems with complex *moving* geometry, we apply the spectral Fourier approximation for the heat kernel and NUFFT to speed up the evaluation of the history part of the heat potentials. The algorithm does require the compression of the local kernels at each time step due to the facts that the boundary is moving and the local kernels change correspondingly. The algorithm also has the restrictions that the physical domain shall not be very

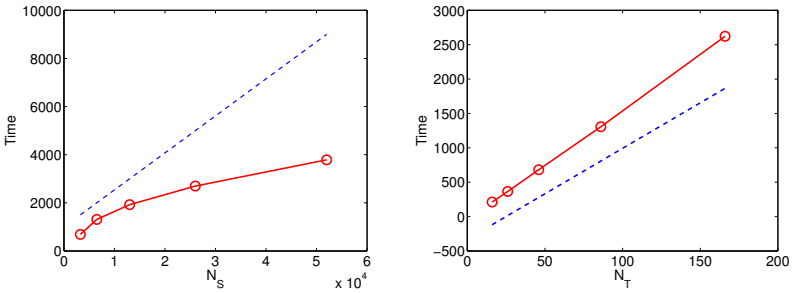

Fig. 10 Timing results for Example 5. Left: Total computational time T versus N_S. Here $N_T = 80$ and $\Delta t = 0.0125$ are fixed. Right: Total computational time T versus N_T. Here $N_S = 7,400$ is fixed. Red circles represent the numerical results, while the blue dashed line represents $O(N_S)$ scaling.

large and the time step size shall not be very small. Nevertheless, the algorithm achieves high order for complex *moving* geometry. We expect that the algorithm will have some applications in applied physics and engineering such as dislocation dynamics in materials science.

Admittedly, both algorithms have certain restrictions and there is still much room for improvement. A bootstrap method [41,42] is currently under development to remove the restrictions of the algorithms in this paper. Furthermore, even though the initial potential can be treated efficiently via the fast Gauss transform [17,18,39] and the volume potential can be treated similarly as the layer potentials, there is still work to be done in order to build an efficient and high-order solver for the general initial-boundary value problem of the heat equation. We are currently working on these projects and will report our findings on a later date. Finally, we would like to point out both algorithms can be extended to three dimensional problems and we refer the readers to [27] on a high-order algorithm for the 3D heat equation in complex geometry using the SOE approximation.

Acknowledgements S. Jiang was supported by NSF under grant DMS-1720405 and by the Flatiron Institute, a division of the Simons Foundation. Part of the work was done when J. Wang was visiting the Department of Mathematical Sciences at New Jersey Institute of Technology.

References

1. Alpert, B.K.: Hybrid Gauss-trapezoidal quadrature rules. SIAM J. Sci. Comput. **20**(5), 1551–1584 (1999)
2. Barnett, A., Magland, J.: Non-uniform fast Fourier transform library of types 1, 2, 3 in dimensions 1, 2, 3. `https://github.com/ahbarnett/finufft` (2018)
3. Brattkus, K., Meiron, D.I.: Numerical simulations of unsteady crystal growth. SIAM J. Appl. Math. **52**, 1303–1320 (1992)
4. Bremer, J.: A fast direct solver for the integral equations of scattering theory on planar curves with corners. J. Comput. Phys. **231**(4), 1879–1899 (2012)
5. Brown, M.: The method of layer potentials for the heat equation in Lipschitz cylinders. Amer. J. Math. **111**, 339–379 (1989)
6. Brown, M.: The initial-Neumann problem for the heat equation in Lipschitz cylinders. Trans. Amer. Math. Soc. **320**, 1–52 (1990)

7. Chandrasekaran, S., Dewilde, P., M. Gu, W.L., Pals, T.: A fast solver for HSS representations via sparse matrices. SIAM J. Matrix Anal. Appl. **29**, 67–81 (2006)
8. Cheng, H., Greengard, L., Rokhlin, V.: A fast adaptive multipole algorithm in three dimensions. J. Comput. Phys. **155**(2), 468–498 (1999)
9. Fabes, E.B., Riviere, N.M.: Dirichlet and Neumann problems for the heat equation in $c1$ cylinders. Proc. Sympos. Pure Math. **35**, 179–196 (1979)
10. Fong, W., Darve, E.: The black-box fast multipole method. J. Comput. Phys. **228**(23), 8712–8725 (2009)
11. Gimbutas, Z., Rokhlin, V.: A generalized fast multipole method for nonoscillatory kernels. SIAM J. Sci. Comput. **24**, 796–817 (2003)
12. Greengard, L., Lee, J.: Accelerating the nonuniform fast Fourier transform. SIAM Rev. **46**, 443–454 (2004)
13. Greengard, L., Lin, P.: Spectral approximation of the free-space heat kernel. Appl. Comput. Harmon. Anal. **9**, 83–97 (2000)
14. Greengard, L., Rokhlin, V.: A fast algorithm for particle simulations. J. Comp. Phys. **73**(2), 325–348 (1987)
15. Greengard, L., Rokhlin, V.: A new version of the fast multipole method for the Laplace equation in three dimensions. Acta. Numer. **6**, 229–270 (1997)
16. Greengard, L., Strain, J.: A fast algorithm for the evaluation of heat potentials. Comm. Pure Appl. Math. **43**, 949–963 (1990)
17. Greengard, L., Strain, J.: The fast Gauss transform. SIAM J. Sci. Statist. Comput. **12**, 79–94 (1991)
18. Greengard, L., Sun, X.: A new version of the fast Gauss transform. Documenta Mathematica **III**, 575–584 (1990)
19. Guenther, R.B., Lee, J.W.: Partial Differential Equations of Mathematical Physics and Integral Equations. Prentice-Hall, Englewood Cliffs, New Jersey (1988)
20. Hackbusch, W., Börm, S.: Data-sparse approximation by adaptive H2-matrices. Computing **69**(1), 1–35 (2002)
21. Ho, K.L., Greengard, L.: A fast direct solver for structured linear systems by recursive skeletonization. SIAM J. Sci. Comput. **34**(5), A2507–A2532 (2012)
22. Ho, K.L., Ying, L.: Hierarchical interpolative factorization for elliptic operators: integral equations. Comm. Pure Appl. Math. **69**(7), 1314–1353 (2016)
23. Ibanez, M.T., Power, H.: An efficient direct BEM numerical scheme for phase change problems using Fourier series. Comput. Methods Appl. Mech. Engrg. **191**, 2371–2402 (2002)
24. Jiang, S., Greengard, L., Wang, S.: Efficient sum-of-exponentials approximations for the heat kernel and their applications. Adv. Comput. Math. **41**(3), 529–551 (2015)
25. Jiang, S., Rachh, M., Xiang, Y.: An efficient high order method for dislocation climb in two dimensions. SIAM J. Multi. Modeling Simul. **15**(1), 235–253 (2017)
26. Jiang, S., Veerapaneni, S., Greengard, L.: Integral equation methods for unsteady Stokes flow in two dimensions. SIAM J. Sci. Comput. **34**(4), A2197–A2219 (2012)
27. Jiang, S., Wang, S.: An efficient high-order integral equation method for solving the heat equation with complex geometries in three dimensions. Proceedings of the 7th ICCM (2019). In press
28. Kong, W.Y., Bremer, J., Rokhlin, V.: An adaptive fast direct solver for boundary integral equations in two dimensions. Appl. Comput. Harmon. Anal. **31**(3), 346–369 (2011)
29. Kress, R.: Linear Integral Equations, *Applied Mathematical Sciences*, vol. 82, third edn. Springer–Verlag, Berlin (2014)
30. Lee, J.Y., Greengard, L., Gimbutas, Z.: NUFFT Version 1.3.2 Software Release. `http://www.cims.nyu.edu/cmcl/nufft/nufft.html` (2009)
31. Li, J., Greengard, L.: On the numerical solution of the heat equation. I. Fast solvers in free space. J. Comput. Phys. **226**(2), 1891–1901 (2007)
32. Li, J., Greengard, L.: High order accurate methods for the evaluation of layer heat potentials. SIAM J. Sci. Comput. **31**, 3847–3860 (2009)
33. Martinsson, P.G.: A fast direct solver for a class of elliptic partial differential equations. J. Sci. Comput. **38**(3), 316–330 (2009)
34. Martinsson, P.G., Rokhlin, V.: A fast direct solver for boundary integral equations in two dimensions. J. Comput. Phys. **205**(1), 1–23 (2005)
35. Martinsson, P.G., Rokhlin, V.: An accelerated kernel-independent fast multipole method in one dimension. SIAM J. Sci. Comput. **29**(3), 1160–1178 (2007)

36. Martinsson, P.G., Rokhlin, V.: A fast direct solver for scattering problems involving elongated structures. J. Comput. Phys. **221**(1), 288–302 (2007)
37. Olver, F.W.J., Lozier, D.W., Boisvert, R.F., Clark, C.W. (eds.): NIST Handbook of Mathematical Functions. Cambridge University Press (2010). URL `http://dlmf.nist.gov`
38. Sethian, J.A., Strain, J.: Crystal growth and dendritic solidification. J. Comput. Phys. **98**, 231–253 (1992)
39. Spivak, M., Veerapaneni, S.K., Greengard, L.: The fast generalized Gauss transform. SIAM J. Sci. Comput. **32**, 3092–3107 (2010)
40. Tausch, J.: A fast method for solving the heat equation by layer potentials. J. Comput. Phys **224**(2), 956–969 (2007)
41. Wang, J.: Integral equation methods for the heat equation in moving geometry. Ph.D. thesis, Courant Institute of Mathematical Sciences, New York University, New York (2017)
42. Wang, J., Greengard, L., Jiang, S., Veerapaneni, S.: An efficient bootstrap method for the heat equation in moving geometry (2018). In preparation
43. Wang, S.: Efficient high-order integral equation methods for the heat equation. Ph.D. thesis, Department of Mathematical Sciences, New Jersey Institute of Technology, Newark, New Jersey (2016)
44. Ying, L., Biros, G., Zorin, D.: A kernel-independent adaptive fast multipole algorithm in two and three dimensions. J. Comput. Phys. **196**, 591–626 (2004)