SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 185–206, January 1994

FAST ADAPTIVE METHODS FOR THE FREE-SPACE HEAT EQUATION*

JOHN STRAIN†

Abstract. Standard numerical methods for the heat equation in two or more space dimensions are excellent if it is necessary to follow the evolution in great detail through many small timesteps. This paper presents efficient and accurate new adaptive methods that solve the free-space heat equation with *large* timesteps. These methods combine the fast Gauss transform with an adaptive refinement scheme that represents the solution as a continuous piecewise polynomial, to a user-specified degree of accuracy. The same approach is extended to solve inhomogeneous problems and to solve the heat equation in moving domains with boundaries. In problems with boundaries, it allows the use of accurate boundary representations without requiring difficult product integration formulas or precluding fast evaluation schemes. Numerical experiments in two space dimensions show these methods to be accurate and efficient, especially for highly nonuniform or discontinuous initial data or when substantial accuracy is required.

Key words. heat equation, heat potentials, fast algorithms, adaptive methods, crystal growth

AMS subject classifications. 65M50, 33A65, 35K05, 80A20, 65V05, 65R10

1. Introduction.

It is often necessary to solve the free-space heat equation

$$\begin{aligned} u_t &= \Delta u \quad \text{in} \quad \mathbf{R}^2 \\ u &= f \quad \text{at} \quad t = 0 \end{aligned} \tag{1}$$

with a difficult initial temperature field f: We suppose that f vanishes at infinity, is globally Lipschitz but not necessarily C^1, and varies rapidly over only a small portion of its support. One needs values of u at arbitrary points, not on a regular grid.

In this paper, we present efficient and accurate new numerical methods for solving this problem. These methods are *accurate* in the sense that the user can specify a precision ϵ, and the solution is then produced within error ϵ (relative to $|u|_\infty$). The work required increases as ϵ decreases, as it must, but these methods are *efficient* in the sense that they attempt to represent u with as few degrees of freedom as possible, adapting those degrees of freedom to fit u. These methods are efficient in another sense as well; the work required to go from $u(x,t)$ to $u(x, t+\Delta t)$ is proportional to the number of degrees of freedom required and *decreases* as Δt increases. (For explicit numerical methods, the work required *increases* as Δt increases, and for implicit methods, it is superlinear in the number of degrees of freedom.)

Our methods are based on the exact evolution formula

$$u(x, t+\Delta t) = \frac{1}{\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} u(y,t)\,dy, \qquad \delta = 4\Delta t. \tag{2}$$

We begin by projecting the initial temperature field f onto a finite-dimensional space of piecewise polynomial interpolants, chosen adaptively to resolve f within error ϵ. Then we use Hermite expansions to evaluate $u(x, \Delta t)$ via (2) efficiently, again within error ϵ, and project $u(x, \Delta t)$ onto another adaptively chosen finite-dimensional space that resolves it within error ϵ. This process can be repeated at each timestep, or the Hermite expansion coefficients can be updated to simply evaluate u at any desired time $t \geq \Delta t$. The first approach is more applicable to inhomogeneous problems, the second approach more efficient.

*Received by the editors August 19, 1991; accepted for publication (in revised form) March 17, 1993. This work was supported by Air Force Office of Scientific Research grant AFOSR-91-0165.

†Department of Mathematics, Princeton University, Princeton, New Jersey 08544. Current address: Department of Mathematics, University of California, Berkeley, California 94720 (strain@math.berkeley.edu).

The techniques used here can also be used to solve the more general problem

$$
\begin{aligned}
&u_t = \Delta u + F(x,t) \quad \text{in } \Omega \subset \mathbf{R}^2 \\
(3) \qquad &\alpha u + \beta \frac{\partial u}{\partial n} = g \quad \text{on } \partial\Omega \\
&u = f \quad \text{at } t = 0,
\end{aligned}
$$

with Ω a possibly time-dependent subset of the plane having a bounded C^2 boundary $\partial\Omega$, n the outward unit normal to $\partial\Omega$, and α, β, g, f, and F C^2 functions on their domains.

We first discuss the inhomogeneous free-space problem

$$
(4) \qquad
\begin{aligned}
&u_t = \Delta u + F(x,t) \quad \text{in } \mathbf{R}^2 \\
&u = f \quad \text{at } t = 0
\end{aligned}
$$

as an example of the technique employed on (3). We solve (4) by Duhamel's principle and evaluate the resulting integral by the trapezoidal rule; this is second-order accurate in Δt. Simpson's rule gives a fourth-order method, and higher-order methods are similarly easy to construct.

The organization of the paper is as follows. In §2, we derive the Hermite expansion of the heat kernel and truncation error estimates. Next, in §3, we describe how we project a given function f onto a finite-dimensional subspace, which resolves it within error ϵ relative to $|f|_\infty = \max|f|$. In §4, we describe how to combine these two techniques to solve the homogeneous free-space heat equation (1), and in §5 we extend the method to the inhomogeneous equation (4). In §6 we sketch how to solve (3), using our method and heat potential theory; details will be given in a later publication. Numerical experiments are presented in §7 and conclusions in §8.

2. Hermite expansions.

The purpose of this section is to describe how certain moments of $u(x,t)$ suffice to evaluate $u(x, t+\Delta t)$ at any point x within an error tolerance ϵ. We use the explicit evolution formula [5]

$$
(5) \qquad u(x, t+\Delta t) = \frac{1}{\pi\delta} \int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} u(y,t)\, dy, \qquad \delta = 4\Delta t
$$

for the bounded solution u of (1).

The heat kernel (with $\delta = 1$ for simplicity) is a real-analytic function of $y \in \mathbf{R}^2$, so we can expand it in a two-dimensional Taylor series:

$$
(6) \qquad h(x-y) = e^{-|x-y|^2} = \sum_{\alpha \ge 0} \frac{1}{\alpha!} h_\alpha(x) y^\alpha.
$$

Here $\alpha = (\alpha_1, \alpha_2)$ is a multi-index with integer components, $\alpha! = \alpha_1!\alpha_2!$; we say that $\alpha \ge 0$ if $\alpha_i \ge 0$ for each $i = 1, 2$, and $y^\alpha = y_1^{\alpha_1} y_2^{\alpha_2}$. The Taylor coefficients are given by

$$
(7) \qquad h_\alpha(x) = D_y^\alpha h(x-y)\big|_{y=0} = (-1)^{|\alpha|} D_x^\alpha h(x),
$$

where $|\alpha| = \alpha_1 + \alpha_2$ and

$$
D_y^\alpha = \left(\frac{\partial}{\partial y_1}\right)^{\alpha_1} \left(\frac{\partial}{\partial y_2}\right)^{\alpha_2}.
$$

Since the variables x_1 and x_2 are separated in $h(x)$, we have

$$h_\alpha(x) = h_{\alpha_1}(x_1)h_{\alpha_2}(x_2),$$

where each one-variable function $h_n(x)$ is given by

$$\begin{aligned} h_n(x) &= (-1)^n \left(\frac{d}{dx}\right)^n e^{-x^2} \\ &= H_n(x)e^{-x^2}, \end{aligned} \tag{8}$$

and H_n is the usual Hermite polynomial. We will need two facts from [11]. First, the two-term recursion

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x),$$

with $H_0(x) = 1$ and $H_1(x) = 2x$. Second, Cramer's inequality

$$|H_n(x)| \le K2^{n/2}\sqrt{n!}e^{x^2/2},$$

where K is less than 1.09. Hence

$$\frac{1}{\alpha!}|h_\alpha(x)| \le K^2\frac{2^{|\alpha|/2}}{\sqrt{\alpha!}}. \tag{9}$$

The "tail" left after truncating the series (6) after terms with every $\alpha_i \le p$ is bounded uniformly in x by

$$\sum_{\text{some}\,\alpha_i > p} \frac{1}{\alpha!}|h_\alpha(x)||y^\alpha| \le K^2 \sum_{\text{some}\,\alpha_i > p} \frac{2^{|\alpha|/2}}{\sqrt{\alpha!}} r^{|\alpha|} = O\left(\frac{2er^2}{p+2}\right)^{(p+1)/2} \tag{10}$$

if $|y_i| \le r$ for $i = 1, 2$, by Stirling's formula. This decreases very rapidly as p increases. Thus the kernel $h(x - y)$ can be approximated by a $(p + 1)^2$-term truncated Taylor series when $|y_i| \le r$, with an error that decreases rapidly as p, increases uniformly in x.

This expansion was used in [6] and [10] to evaluate the two-dimensional discrete Gauss transform

$$G_\delta f(t_i) = \sum_{j=1}^{N} f_j e^{-|t_i - s_j|^2/\delta}, \qquad i = 1, 2, 3, \dots, M, \tag{11}$$

with $t_i, s_j \in \mathbf{R}^2$, $\delta > 0$, $|t|^2 = t_1^2 + t_2^2$, in $O(N + M)$ time. (In [6], δ was a constant; [10] extended the algorithm to cover the situation when $\delta = \delta_i$ or $\delta = \delta_j$.)

Now rescale and shift this formula with an arbitrary δ and a center of expansion c; we get

$$h\left(\frac{x-y}{\sqrt{\delta}}\right) = h\left(\frac{x-c-(y-c)}{\sqrt{\delta}}\right) = \sum_{\alpha\ge 0}\frac{1}{\alpha!}h_\alpha\left(\frac{x-c}{\sqrt{\delta}}\right)\left(\frac{y-c}{\sqrt{\delta}}\right)^\alpha, \tag{12}$$

and the error in truncating after terms with $\alpha_i \le p$ is given by (10) if $|y_i - c_i| \le r\sqrt{\delta}$. Hence $e^{-|x-y|^2/\delta}$ is well represented by a truncated Taylor series in y when y lies in a square with center c and side $2r\sqrt{\delta}$.

This suggests a natural way to evaluate the integral

$$G_\delta f(x) = \frac{1}{\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} f(y)\,dy.$$

Suppose f has compact support. Cover the support of f with nonoverlapping square cells C of side $2r\sqrt{\delta}$ and centers c. In each cell, expand the heat kernel about the cell center using (12);

$$G_\delta f(x) = \frac{1}{\pi}\sum_C \sum_{\alpha\ge 0} C_\alpha h_\alpha\left(\frac{x-c}{\sqrt{\delta}}\right), \tag{13}$$

where

$$C_\alpha = \frac{1}{\delta\alpha!}\int_C \left(\frac{y-c}{\sqrt{\delta}}\right)^\alpha f(y)\,dy. \tag{14}$$

The variables x and y are now separated, so we have decomposed the calculation into two simpler pieces; first, we evaluate the moments C_α for each cell C and for $\alpha_i = 0, 1, \dots, p$, and then we sum the Hermite series (13) for each point where $G_\delta f(x)$ is to be evaluated. The numerical error in the approximation of $G_\delta f$ by this procedure comes from two sources: first, truncation of the Hermite series, and second, quadrature error in evaluating C_α numerically. We have already discussed the truncation error; the quadrature error will be discussed in the next section.

We can speed up the evaluation considerably, when δ is small, by dropping negligible contributions from cells distant from the point of evaluation x. The contribution from a distant cell C' is bounded by

$$\frac{1}{\pi\delta}\int_{C'} e^{-|x-y|^2/\delta}|f(y)|\,dy \le \frac{1}{\pi\delta}|f|_\infty |C'| e^{-|x-x_0|^2/\delta},$$

where x_0 is the closest point in C' to x and $|C'|$ is the area of C'. Thus only a fixed number of cells within a distance $O(\sqrt{\delta})$ need to be kept, for any fixed precision. Our next step is to evaluate C_α.

3. Adaptive refinement.

The purpose of adaptive refinement is to know a function within ϵ as efficiently as possible. Numerically, we know a function f within ϵ if we can evaluate it at any point x with an error that is less than $\epsilon|f|_\infty$ and a cost that depends only on ϵ.

The Hermite expansion of §2 enables us to evaluate $u(x, t + \Delta t)$ at any point x, not just on a fixed set of grid points. The cost per evaluation is fixed, because each evaluation is independent of the others. Thus we are in the following situation: We have a function, f say, and we can evaluate it at any point for fixed cost per evaluation. How can we best approximate f by piecewise polynomials within error $\epsilon|f|_\infty$? For our applications, f is not very smooth initially, so refinement in space seems a reasonable approach. Thus we use a fixed low degree d of polynomial interpolation and refine the interpolation grid wherever f is not accurately represented. We use a triangular grid and approximate f by degree-d interpolation over each triangle; the degree is selected by the user. When we evaluate the moments C_α, we have to integrate f times powers over cells C. It is convenient if each triangle of our triangulation lies completely within a single cell C. We begin with the simplest such triangulation, which is the one formed by cutting each cell C into two isosceles right triangles and subdividing until f is accurate within ϵ.

Thus we use the following method to construct the adaptive approximation of f; we begin by dividing each cell C into two right triangles and constructing the degree-d interpolant to f at the $(d+1)(d+2)/2$ nodes shown in Fig. 1.

Now we stack all the triangles and sweep through, testing whether each triangle requires subdivision. To test a triangle, we evaluate f at each node that would be produced by bisecting

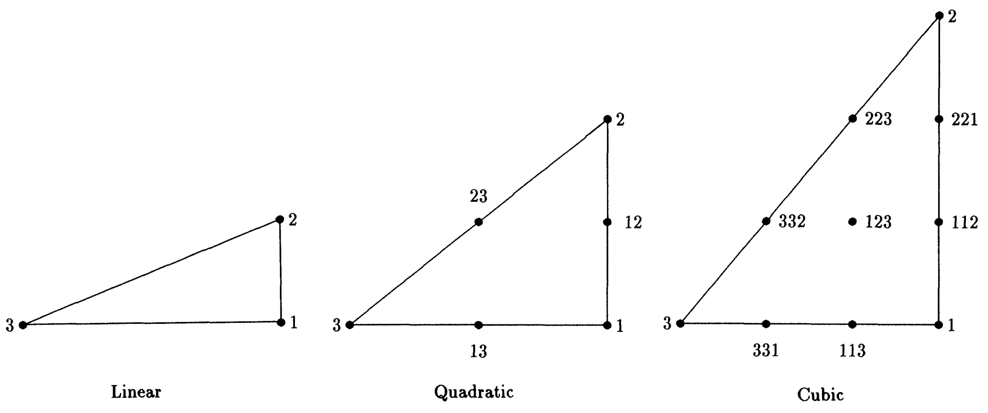

FIG. 1. *Nodes for degree-d interpolation on triangles for $d = 1, 2$, and 3.*

the longest side, as shown in Fig. 2. This requires one evaluation for $d = 1$, three for $d = 2$, and six for $d = 3$; all these values are used if the triangle is subdivided. We also evaluate the degree-d interpolant at the same nodes and compute the maximum difference between the two sets of values. If f is within ϵ (relative to the maximum value of $|f|$ so far encountered) of the interpolant at the new nodes, the triangle is accepted. Otherwise, the triangle is subdivided by Mitchell's newest-node bisection method [8], maintaining compatibility by subdividing neighbors as necessary, and the new triangles are stacked. We then repeat the procedure until the stack is finished.

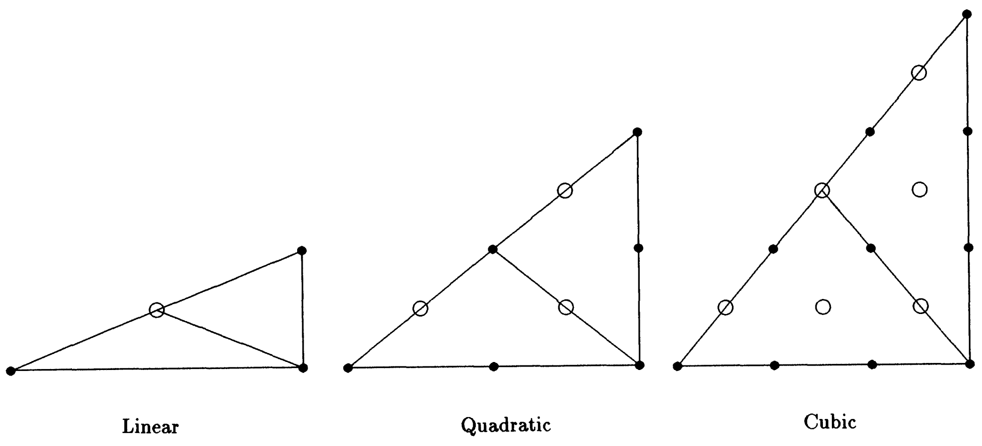

FIG. 2. *New nodes produced when the base is bisected are circled.*

Mitchell's subdivision procedure assigns one vertex of each triangle in the initial triangulation as a "peak" and the side opposite the peak as the base. (In our case, the initial triangulation consists of isosceles right triangles and the peak is the vertex at the right angle, opposite the hypotenuse.) Then it subdivides triangles by dividing the base and the neighboring triangle opposite the peak, with the new vertex being assigned as the peak of each of the four new triangles created by the subdivision. Compatibility is maintained by always subdividing compatible pairs of triangles; if the neighbor opposite the peak is not compatibly subdivisible, it is itself divided recursively until compatibility is maintained. An example is shown in Fig. 3.

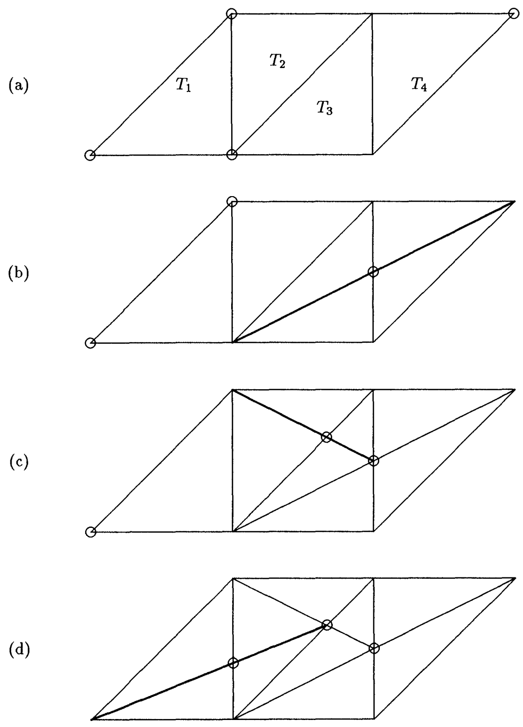

FIG. 3. *An example of Mitchell's recursive newest-node bisection* [8]. *Triangle T_1 is flagged for subdivision, but the peak of its neighbor T_2 (indicated by circle) does not lie opposite T_1. Hence we must refine T_2 and its neighbor T_3. Similarly, the peak of T_3 does not lie opposite T_2, so we must refine T_4 and T_3. The peak of T_4 lies opposite T_3, so the recursion stops here. We then divide triangles backwards in pairs as shown in* (b) *through* (d), *until we have divided the triangle T_1 we originally wanted to divide. The subdivided triangulation is shown in* (d).

Once a triangulation is constructed, we have enough information about f to solve the heat equation to accuracy ϵ after time Δt has elapsed. We need only to use the interpolant of f on the adapted triangulation to evaluate $(p+1)^2$ of the moments C_α within relative error ϵ. This requires integrating the interpolant times $((x-c)/\sqrt{\delta})^\alpha$ over each triangle lying in cell C with center c.

This task is simpler if we put the interpolant, on each triangle, in the shifted and scaled Cartesian form

$$\tilde{f}(x) = \sum_{|\beta| \le d} F_\beta \left(\frac{x-c}{\sqrt{\delta}} \right)^\beta .$$

This requires considerable hand calculation, because $\tilde{f}$ is most naturally expressed in barycentric coordinates as in [3], and the expressions for the Cartesian coefficients F_β can become quite complicated for quadratic or cubic interpolation. The expressions for F_β are given in the Appendix.

Next we need to calculate the approximate moments

$$\begin{aligned} \tilde{C}_\alpha &= \frac{1}{\alpha!\delta} \int_C \left(\frac{y-c}{\sqrt{\delta}} \right)^\alpha \tilde{f}(y)\, dy \\ &= \sum_{T \subset C} \frac{1}{\alpha!\delta} \int_T \left(\frac{y-c}{\sqrt{\delta}} \right)^\alpha \tilde{f}(y)\, dy \end{aligned} \tag{15}$$

using the piecewise polynomial interpolant. Here we have split up the integral over C into a sum of integrals $\tilde{T}_\alpha$, say, over the triangles T contained in C.

Now consider the evaluation of

$$\tilde{T}_\alpha = \frac{1}{\alpha!\delta}\int_T \left(\frac{y-c}{\sqrt{\delta}}\right)^\alpha \tilde{f}(y)\,dy \tag{16}$$

$$= \sum_{|\beta|\le d} F_\beta \frac{1}{\alpha!\delta}\int_T \left(\frac{y-c}{\sqrt{\delta}}\right)^{\beta+\alpha} dy. \tag{17}$$

We need only evaluate triangle moments

$$T_\gamma = \frac{1}{\gamma!\delta}\int_T \left(\frac{x-c}{\sqrt{\delta}}\right)^\gamma dx$$

for $\gamma_i = 0, 1, 2, \ldots, p+d$. Note that while c may not lie in T and δ may be small, we nevertheless have $T \subset C$ and thus $|x_i - c_i| \le r\sqrt{\delta}$ for $x \in T$, where r is a fixed constant. Hence evaluating T_γ and adding up values with coefficients is not a numerically unstable process.

To evaluate T_γ, we first translate and scale T to simplify the notation, then use the Divergence Theorem and recursion. Write $\gamma = (i, j)$ and $x = (x, y)$. Then

$$T_\gamma = T_{ij} = \frac{1}{i!j!}\int_{T'} x^i y^j\,dxdy,$$

where $T' = (T-c)/\sqrt{\delta}$ is a shifted and scaled version of T.

Let v_1, v_2, and v_3 be the vertices of T', arranged in counterclockwise order, and let $v_4 = v_1$. Then by the Divergence Theorem,

$$T_{ij} = \sum_{k=1}^{3} T_{ijk},$$

where explicit parametrization of the line segment $\overline{v_k v_{k+1}}$ gives

$$T_{ijk} = \int_0^1 (y_{k+1} - y_k)\frac{x^{i+1}}{(i+1)!}\frac{y^j}{j!}\,d\theta.$$

Here $(x, y) = v = \theta v_{k+1} + (1-\theta)v_k$ and $v_k = (x_k, y_k)$. Integrating by parts gives a recurrence relation

$$T_{ijk} = \Lambda\left[\frac{x_k^{i+1}}{(i+1)!}\frac{y_k^{j+1}}{(j+1)!}\right] - \frac{\Lambda x_k}{\Lambda y_k} T_{i-1,j+1,k}.$$

Here Λ is the forward difference operator $\Lambda x_k = x_{k+1} - x_k$. We can use this recurrence to compute T_{ijk} for $i, j = 0, 1, 2, \ldots, q = p+d$ if we start with values for T_{0jk} for $j = 0, 1, 2, \ldots, 2q$. Integrating by parts again produces initial values for the recurrence:

$$T_{0jk} = \Lambda\left[x_k \frac{y_k^{j+1}}{(j+1)!}\right] - \frac{\Lambda x_k}{\Lambda y_k}\Lambda\left[\frac{y_k^{j+2}}{(j+2)!}\right].$$

Note that (a) it is efficient to precompute and store the values $x_k^i/i!$ and $y_k^j/j!$, which are used repeatedly, and (b) $T_{ijk} = 0$ when $\Lambda y_k = 0$ (so there is no difficulty with dividing by zero). This concludes the evaluation of T_{ijk} and hence of C_α.

4. The homogeneous problem. We now combine the tools described in §§2 and 3 to construct methods for solving the homogeneous free-space heat equation

$$u_t = \Delta u \quad \text{in } \mathbf{R}^2, \tag{18}$$

$$u(x,0) = f(x), \tag{19}$$

with f a bounded function that vanishes at infinity. The unique bounded solution u satisfies the semigroup property [5]

$$u(x,t+\Delta t) = \frac{1}{\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} u(y,t)\,dy, \tag{20}$$

with $\delta = 4\Delta t$.

Method 1. Taking $t = 0$ and $\Delta t = t$ gives

$$u(x,t) = \frac{1}{\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} f(y)\,dy, \qquad \delta = 4t, \tag{21}$$

which represents u at any time t in terms of the convolution of a Gaussian with scale $\delta = 4t$ with the initial temperature field f. Let us suppose that we want to know $u(x,t)$ within relative error ϵ at a sequence of times $t = 0, \Delta t, 2\Delta t, \ldots, N\Delta t = T$, and Δt is not too small. (If Δt is very small and we wish to know u at every step, explicit adaptive finite difference or finite element methods are excellent.)

Let $\delta = 4\Delta t$ and let us first evaluate $u(x, \Delta t)$. We replace f by a piecewise polynomial $\tilde{f}$ constructed, as in §3, to approximate f within $\epsilon|f|_\infty/2$. Then we use $\tilde{f}$ to evaluate the moments $\tilde{C}_\alpha$ defined in (15) for $0 \le \alpha_i \le p$ for each cell. We can now evaluate $u(x, \Delta t)$ by (21) within error ϵ relative to $|f|_\infty$, by a truncated Hermite series

$$u(x,t+\Delta t) = \frac{1}{\pi}\sum_C \sum_{\alpha\ge 0}^{p} C_\alpha h_\alpha\left(\frac{x-c}{\sqrt{\delta}}\right) + E,$$

where the error E comes from several sources; first, the error due to replacing f by $\tilde{f}$, which is bounded by $\epsilon|f|_\infty/2$ by the maximum principle [5]; second, the error due to truncating each Hermite series at $\alpha_i = p$, and third, the error due to ignoring distant cells. These add up to

$$\frac{\epsilon|f|_\infty}{2} + O\left(\frac{2er^2}{p+2}\right)^{(p+1)/2} |f|_\infty + O\left(e^{-4r^2 l^2}\right)|f|_\infty, \tag{22}$$

which can be easily made less than ϵ by choice of p and the cutoff distance once r and N are fixed.

Thus we can evaluate $u(x, \Delta t)$ at any point x. Method 1 is distinguished by the way in which we evaluate u at later times. In principle, a separate computation should be required for each time at which we wish to evaluate u, because the moments C_α depend on t through δ. However, C_α is homogeneous of degree $-1 - |\alpha|/2$ in δ;

$$C_\alpha = \frac{1}{\alpha!\delta}\int_C \left(\frac{y-c}{\sqrt{\delta}}\right)^\alpha f(y)\,dy.$$

Thus $C_\alpha(2\delta) = 2^{-(2+|\alpha|)/2} C_\alpha(\delta)$ if we make the dependence on δ explicit. Once we have $C_\alpha(\delta)$, therefore, we can evaluate $u(x,t)$ with relative error less than ϵ at any time $t \ge \Delta t$, simply by scaling the coefficients and evaluating. More precisely,

$$u(x,t) = \frac{1}{\pi} \sum_{C} \sum_{\alpha \geq 0}^{p} \left(\frac{t}{\Delta t}\right)^{-(2+|\alpha|)/2} C_\alpha h_\alpha \left(\frac{x-c}{\sqrt{4t}}\right) + E,$$

where $E \leq \epsilon |f|_\infty$. The need to include more cells C in the sum as t increases, due to spreading of the Gaussians, is more than counterbalanced by the fewer coefficients required: C_α is effectively reduced exponentially as $\delta = 4t$ increases.

The scaling of coefficients used in Method 1 was developed in [10] for evaluating the discrete Gauss transform with target-dependent scales.

Method 1 is highly efficient and accurate. It requires a substantial investment of effort at the beginning of the calculation to construct the triangulation and approximate f (especially if f is not very smooth), but evaluation of u gets less and less expensive as time goes by. Moreover, the space required for the triangulation can be reused for other purposes as soon as the coefficients are evaluated and stored. Method 1 is useful whenever the homogeneous free-space heat equation is to be solved for rough data and not too small times.

Two fundamental properties of the heat equation influence this method. First, the increasing smoothness and decreasing variation of $u(x,t)$ as t increases means that it takes fewer and fewer degrees of freedom to represent u within a fixed relative precision ϵ. A typical example is shown in Figs. 4 and 5. In Fig. 4, we display the triangulation needed to resolve

$$f(x) = \sum_{k_1=1}^{10} \sum_{k_2=1}^{10} \cos(k_1(x_1 - \bar{x}_1)) \cos(k_2(x_2 - \bar{x}_2)) e^{-(k_1(x_1-\bar{x}_1))^2 - (k_2(x_2-\bar{x}_2))^2}$$

(where $\bar{x}_1$ and $\bar{x}_2$ are randomly chosen, for each k_1 and k_2, on $[-1, 1]$) within error $\epsilon = 10^{-2}$ relative to $|f|_\infty$, using linear interpolation. Figure 5 displays the triangulation needed to resolve the solution $u(x,t)$ with initial data $f(x)$, at $t = 0.1$, to the same relative precision. The triangulation began with a 6×6 grid of square cells on $[-3, 3]^2$ in both cases; f required 2910 triangles while u required only 608, a reduction in the number of degrees of freedom by almost 5. The smallest triangle for f is 2^{-6} on a side, while at $t = 0.1$, the smallest triangle is 2^{-3}. Thus smoothing by heat flow speeds up later stages of the calculation.

The other property of the heat equation, the spreading support of its solutions, is less benign though hardly malignant. Even though f has compact support, $u(x,t)$ will not have compact support for any $t > 0$, because heat flows instantaneously (though in small amounts) to infinity. Thus if we begin by representing f by a piecewise polynomial on $B = [-R, R]^2$, say, where $|f| \leq \epsilon$ outside B, the support of u will spread and $|u|$ will eventually be greater than ϵ outside B. No fixed region can be used for computing u for arbitrarily long times, because eventually significant heat flows to infinity.

Method 2. We now describe another method for solving (18), with slightly different aims. Method 2 is better suited to solving inhomogeneous problems and extends more easily to solving problems on domains with boundaries and variable-coefficient and nonlinear problems.

The basic idea of Method 2 is simply to restart Method 1 at each step. (More generally, we could restart every few steps.) The advantage in this is that data like inhomogeneous terms and boundaries can more easily enter the evolution.

More precisely, we carry out Method 1 to construct the coefficients C_α of $u(x, \Delta t)$. Then we apply the adaptive refinement strategy again, with $u(x, \Delta t)$ in place of $f(x)$. Thus we construct a new triangulation on which $u(x, \Delta t)$ is equal to a piecewise polynomial of degree d, within error $\epsilon |u(\cdot, \Delta t)|_\infty$. We then repeat the construction of the C_α's with $u(x, \Delta t)$ in place of f, and we then can evaluate $u(x, 2\Delta t)$. We repeat this process, going from real-space

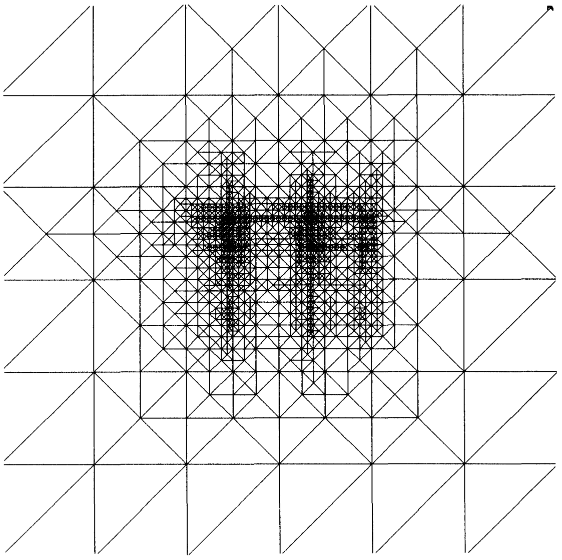

FIG. 4. *Initial triangulation for sum of* 100 *scaled and randomly shifted terms of*

$$u_1(x,t) = \frac{e^{1/4T-1/4}}{\sqrt{T}} \cos\left(\frac{x}{T}\right) e^{-x^2/T},$$

where $T = 1 + 4t$, *resolved to error* $\epsilon = 10^{-2}$ *relative to* $|u_1|_\infty$ *with linear interpolation. The triangulation began with a* 6×6 *grid of square cells on* $[-3, 3]$. *This required* 2910 *triangles and* 1468 *nodes, with minimum side length* $2^{-6.5}$ *times the maximum side length.*

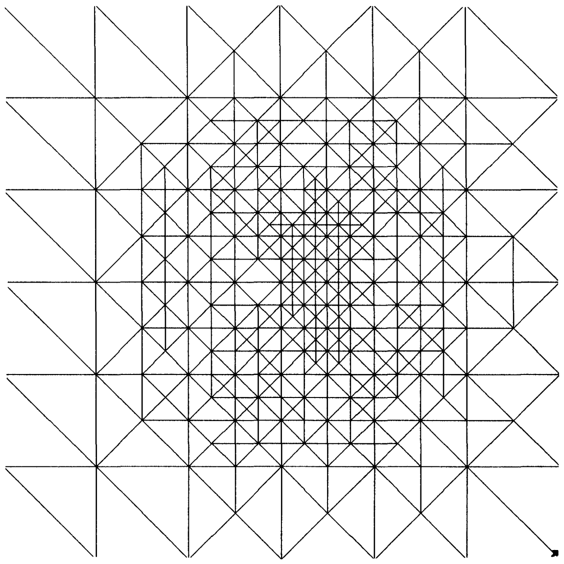

FIG. 5. *Triangulation for* 100 *terms of* u_1 *at* $t = 0.1$, *with* $\epsilon = 10^{-2}$ *and linear interpolation. The increasing smoothness of the solution is reflected in the larger triangles used to represent it; only* 608 *triangles are needed here, with* 317 *nodes and minimum side length* $2^{-3.5}$ *times the maximum side length.*

values sufficient to resolve $u(x, n\Delta t)$ to the coefficients for $u(x, (n+1)\Delta t)$ and back again, at each timestep.

A disadvantage of Method 2 is that in the worst case we commit error ϵ *at each step*; thus if we want to go N steps to time T, and *guarantee* a final error $\leq \epsilon$, we have to commit error $\epsilon' = \frac{\epsilon}{N}$ at each step. Fortunately, this estimate is quite pessimistic, because the error committed at one time is high frequency and therefore decays rapidly under the heat flow. Hence errors do not accumulate to this extent in practice.

There are several ways to deal with the problem of spreading support and they lead to a subdivision of Method 2.

Method 2.1 is the simplest. Here we plan to compute up to some final time T, determine R so that $|u(x,t)| \leq \epsilon$ outside $[-R, R]^2$ for $0 \leq t \leq T$, then compute on the fixed domain $[-R, R]^2$. Either R can be determined experimentally or the exact representation (18) for u can be used to bound R a priori. Suppose $|f| \leq \epsilon |f|_\infty$ when $|x| \geq \rho$. Then within error $\epsilon |f|_\infty$,

$$u(x,t) = \frac{1}{\pi\delta} \int_{|y|\leq\rho} e^{-|x-y|^2/\delta} f(y)\, dy, \qquad \delta = 4\Delta t.$$

If $|x| \geq R$ where $R \geq \rho$, then

$$|u(x,t)| \leq \frac{\rho^2}{\delta} e^{-(R-\rho)^2/\delta} |f|_\infty.$$

We can make this $\leq \epsilon$ for $0 \leq t \leq T$ with $R = \rho + \mathrm{O}(\sqrt{T \log(\epsilon T)})$, so R grows rather slowly with T. This a priori bound is rather crude, however, and it is usually far cheaper to run the code once with $\Delta t = T$ and large ϵ (which costs very little), measure the maximum of $|u|$ near the boundary, and increase R until a suitable value is found. Note that increasing R adds to the outer regions of the computational domain, where u is small and smooth and therefore needs little refinement, hence adds very little to the cost of the calculation.

Method 2.2 is almost as simple as Method 2.1; here, we compute on a changing computational box $B(t) = [a_1(t), b_1(t)] \times [a_2(t), b_2(t)]$, outside of which $|f| \leq \epsilon$ initially. At each step, we expand $B(t)$ adaptively to include all regions in which $|u| > \epsilon$. This is done by computing the max of $|u|$ over each side and adding one cell at a time until satisfied.

This method is adaptive in a way that fits well with our general approach. It is not foolproof (as Method 2.1 with an a priori estimate would be), but in practice works very well. It achieves a small savings over Method 2.1 at a small additional cost in algorithmic complexity.

Method 2.3, like 2.2, expands the domain as necessary. But Method 2.2 adds cells, so the cost goes up; in Method 2.3, we simply scale up the size of each cell by increasing δ until cells of size $2r\sqrt{\delta}$ cover the region on which $|u| > \epsilon$. Thus we look at the solution on a larger spatial scale to take advantage of its increased smoothness and take larger timesteps accordingly. The spatial and temporal scalings are related by the natural scaling law of the heat equation; time = space squared.

This method achieves roughly constant cost, but at the price of varying the timestep. Thus we have to use interpolation in time if we want to know u at a specific time. Another disadvantage is that Method 2.3 does not easily extend to inhomogeneous problems, where the solution does not necessarily become smoother as time goes by.

Method 3 speeds up the evaluation of u by transforming the sum of $(2I+1)^2$ Hermite series into a single Taylor series for each cell, using a technique developed in [6]. We have

$$u(x,t) = \sum_C \sum_{\alpha\geq 0} C_\alpha h_\alpha \left(\frac{x-c}{\sqrt{\delta}}\right).$$

We can transform this to a single Taylor series in the cell B in which x lies, say,

$$u(x,t) = \sum_{\beta \geq 0} B_\beta \left(\frac{x-b}{\sqrt{\delta}} \right)^\beta,$$

where b is the center of B. To do this, we simply calculate the Taylor coefficients

$$\begin{aligned} B_\beta &= \frac{(\sqrt{\delta})^{|\beta|}}{\beta!} D_b^\beta u(b,t) \\ &= \frac{1}{\beta!} \sum_C \sum_{\alpha \geq 0} C_\alpha (\sqrt{\delta})^{|\beta|} D_b^\beta h_\alpha \left(\frac{b-c}{\sqrt{\delta}} \right). \end{aligned}$$

But $h_\alpha = (-D)^\alpha e^{-|x|^2}$, by definition, so

$$B_\beta = \frac{(-1)^{|\beta|}}{\beta!} \sum_C \sum_{\alpha \geq 0} C_\alpha h_{\alpha+\beta} \left(\frac{b-c}{\sqrt{\delta}} \right). \tag{23}$$

This sum can be truncated with an error bound similar to that for the original series. Thus u can be expressed as a local Taylor series in each cell, with coefficients depending on the Hermite coefficients C_α for nearby cells and the Hermite functions of the center-to-center vectors.

The advantage of this is that each evaluation of u costs only $O(p+1)^2$ arithmetic operations, rather than $O((2I+1)(p+1))^2$, which can be one or two orders of magnitude larger. Since most of the computational effort is spent on evaluating u, this is a considerable savings.

As in [6], there is a break-even point, a number of evaluations per cell below which it is not efficient to transform u to a Taylor series. This point depends on the degree of refinement necessary in the given cell, hence is not known a priori. However, it can be estimated from the behavior of u in the cell in the previous timestep or directly from the rate of decay of the Hermite coefficients.

However, another advantage is that when u is given by a truncated Taylor series, the coefficents C_α for the next time level are trivial to compute. For cell C, we have

$$\begin{aligned} C_\alpha &= \int_C \left(\frac{x-c}{\sqrt{\delta}} \right)^\alpha \sum_{\beta \leq p} B_\beta \left(\frac{x-c}{\sqrt{\delta}} \right)^\beta dx \\ &= \sum_{\beta \leq p} B_\beta \frac{r^{|\alpha+\beta|+2}(1-(-1)^{\alpha_1+\beta_1+1})(1-(-1)^{\alpha_2+\beta_2+1})}{(\alpha_1+\beta_1+1)(\alpha_2+\beta_2+1)} \end{aligned} \tag{24}$$

since the sides of C are $2r\sqrt{\delta}$ long. Thus it is unnecessary, as far as u is concerned, to retriangulate. Of course, inhomogeneous terms and layer potentials will still need to be approximated on a triangulation.

By substituting (23) into (24), we also get an expression for the new C_α's in terms of the old C_α's, which allows us to evolve u completely in transform space and makes it unnecessary to return to real space at every step.

5. The inhomogeneous case.

We now generalize one of the methods, Method 2.1, to solve the inhomogeneous free-space heat equation

$$\begin{aligned} u_t &= \Delta u + F(x,t) \quad \text{in } \mathbf{R}^2, \\ u(x,0) &= f(x), \end{aligned} \tag{25}$$

where F has compact support in x for each t. The evolution formula corresponding to (20) is Duhamel's principle [5, p. 196]

$$u(x,t+\Delta t) = \frac{1}{\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} u(y,t)\,dy + \int_t^{t+\Delta t} \frac{1}{\pi\sigma}\int_{\mathbf{R}^2} e^{-|x-y|^2/\sigma} F(y,s)\,dy\,ds \tag{26}$$

with $\delta = 4\Delta t$ as usual and $\sigma = 4(t+\Delta t - s)$.

Let us begin by discretizing the time integral. The integrand is a smooth function of s, so the trapezoidal rule

$$\int_t^{t+\Delta t} g(s)\,ds = \frac{\Delta t}{2}(g(t+\Delta t)+g(t)) + O(\Delta t^3)$$

is globally second-order accurate. Moreover, when $s \to t+\Delta t$, the integrand approaches $F(x,t+\Delta t)$. Thus we have

$$u(x,t+\Delta t) = \frac{1}{\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta}\left[u(y,t) + \frac{\Delta t}{2}F(y,t)\right] dy + \frac{\Delta t}{2}F(x,t+\Delta t) + O(\Delta t^3).$$

Thus $u(x,t+\Delta t)$ can be found by Method 2.1, if we simply replace $u(y,t)$ by $u(y,t) + \frac{1}{2}\Delta t F(y,t)$ in the construction of the triangulation and the evaluation of the moments, and afterward add $\frac{1}{2}\Delta t F(x,t+\Delta t)$. Since we commit $O(\Delta t^3)$ error at each of $N = \frac{T}{\Delta t}$ steps, the final error in u up to time T is $O(\Delta t^2)$.

Higher-order methods are easy to construct, but slightly more expensive per timestep. Several sets of coefficients must be stored, and updated (by the technique of Method 1) after each step. These higher-order methods are important, however, because our approach is efficient when Δt is *large*; thus we need a higher order of accuracy to capture variations in F. Hence this approach is suited to F varying rapidly in x but not in t.

A fourth-order method, for example, can be based on Simpson's rule:

$$\int_{t-\Delta t}^{t+\Delta t} \frac{\Delta t}{3}\left[g(t-\Delta t) + 4g(t) + g(t+\Delta t)\right] + O(\Delta t^5).$$

This leads to two possible methods, depending on whether or not it is convenient to evaluate F at half timesteps. If not, for example, we get

$$u(x,t+\Delta t) = \frac{1}{2\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/2\delta}\left[u(y,t-\Delta t) + \frac{\Delta t}{3}F(y,t-\Delta t)\right] dy + \frac{4\Delta t}{3\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} F(y,t)\,dy + \frac{\Delta t}{3}F(x,t+\Delta t) + O(\Delta t^5).$$

The final error after N steps of this method is $O(\Delta t^4)$.

6. Domains with boundaries.

6. Domains with boundaries. The methods we are developing in this paper find perhaps their best application in the solution of the heat equation on domains with (possibly time-dependent) boundaries. To do this, we use the classical theory of layer potentials for the heat equation described in [9] and more recently in [2]. The key operators in this theory are the layer potentials

$$Sg(x,t) = \int_0^t \int_{\Gamma(t')} G(x-x', t-t') g(x,t')\, dx'\, dt',$$

$$Dg(x,t) = \int_0^t \int_{\Gamma(t')} \frac{\partial G}{\partial n'}(x-x', t-t') g(x,t')\, dx'\, dt',$$

$$D^* g(x,t) = \int_0^t \int_{\Gamma(t')} \frac{\partial G}{\partial n}(x-x', t-t') g(x,t')\, dx'\, dt',$$

taken over a time-dependent boundary $\Gamma(t)$. Here G is the heat kernel $G(x,t) = (4\pi t)^{-1} e^{-|x|^2/4t}$ and g is a function defined on $\Gamma(t)$ for each t. Various boundary conditions for the heat equation can be transformed into Volterra integral equations on $\Gamma(t)$, with operators like S, D, or D^*, by representing the solution of the heat equation as a sum of layer potentials.

In this section, we describe briefly how our methods can be used to evaluate Sg. The basic idea is that layer potentials are solutions of inhomogeneous problems like the one treated in the previous section, but with distributions rather than smooth functions on the right-hand side as forcing terms. Sg, for example, solves

$$\begin{aligned} u_t &= \Delta u + \mu, \\ u(x,0) &= 0, \end{aligned} \tag{27}$$

where μ is the *measure* that assigns density g concentrated on $\Gamma(t)$. The double layer potential Dg has the normal derivative of a measure on the right-hand side.

Clearly this observation does not allow one to evaluate layer potentials with standard numerical methods that mostly require point values. Our methods, however, require only the ability to integrate the inhomogeneous term against a piecewise polynomial. The singularity in time of the kernel G turns out to complicate the integration slightly, requiring a straightforward asymptotic calculation.

Let us consider the single layer potential $u = Sg$. Since u satisfies (27), the evolution formula (26) gives

$$\begin{aligned} u(x,t) &= \frac{1}{\pi\delta} \int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} u(y, t-\Delta t)\, dy \\ &\quad + \int_{t-\Delta t}^t \int_{\Gamma(t')} \frac{e^{-|x-y|^2/4(t-t')}}{4\pi(t-t')} g(y,t')\, dy\, dt'. \end{aligned} \tag{28}$$

The integrand in the time integral is singular; it becomes infinite as t' approaches t. Thus a standard integration formula like the trapezoidal rule will not achieve its usual order of accuracy. A standard technique for eliminating this difficulty is *product integration*; to apply product integration, one extracts an analytic form for the singularity of

$$\tilde{g}(x,t') = \int_{\Gamma(t')} \frac{e^{-|x-y|^2/4(t-t')}}{4\pi(t-t')} g(y,t')\, dy$$

as t' approaches t from below. Say, $\tilde{g}(x,t') \sim \phi(x,t')$ as $t' \uparrow t$. Then we construct an integration rule by interpolating $\tilde{g}(x,t')/\phi(x,t')$ and integrating the resulting polynomial times $\phi(x, t')$ exactly. Such a rule will achieve the order of accuracy of the interpolation used. Of course, ϕ must be simple enough that ϕ times a polynomial can be integrated exactly or very accurately, by some trick.

Thus we need the asymptotic behavior of $\tilde{g}$ as $t' \uparrow t$. When x lies on $\Gamma(t)$, a simple calculation shows that

$$\tilde{g}(x, t') \sim \frac{1}{\sqrt{4\pi(t-t')}} g(x, t) \quad \text{as } t' \uparrow t.$$

This square-root singularity is consistent with the usual parabolic scaling of the heat equation. It is derived by approximating $\Gamma(t')$ by its tangent line, estimating the error, and Taylor expanding. The technique is fairly standard [1].

When x does not lie on $\Gamma(t)$, however, approximation by the tangent line gives the wrong answer, because it neglects the curvature of $\Gamma(t)$. A better approach is to approximate $\Gamma(t)$ by the osculating circle or parabola; the result depends only on the curvature of $\Gamma(t)$, so either will do. We omit the details and state the result.

Assume x_0 is the closest point on $\Gamma(t)$ to x and let $D = |x - x_0|$. Let R be the radius of curvature of $\Gamma(t)$ at x_0 and ρ be the distance from x to the center of curvature. Then, if $\rho > 0$,

$$\tilde{g}(x, t') \sim \sqrt{\frac{R}{\rho}} \frac{1}{\sqrt{4\pi(t-t')}} e^{-D^2/4(t-t')} g(x_0, t)$$

as $t' \uparrow t$.

We now have the asymptotic behavior of the integrand. The next step is to write the time integral as

$$\int_{t-\Delta t}^{t} \frac{e^{-D^2/4(t-t')}}{\sqrt{4\pi(t-t')}} h(x, t')\, dt'$$

$$h(x, t') = \int_{\Gamma(t')} \frac{e^{(D^2-|x-y|^2)/4(t-t')}}{\sqrt{4\pi(t-t')}} g(y, t')\, dy$$

so that $h(x, t') \to \sqrt{R/\rho}\, g(x_0, t)$ as $t' \uparrow t$. Now h is a nice smooth function, so we replace it by its linear interpolant

$$\tilde{h}(x, t') = \frac{t - t'}{\Delta t} h(x, t - \Delta t) - \frac{t - \Delta t - t'}{\Delta t} h(x, t)$$

and carry out the time integral exactly. This gives

$$\begin{aligned} &\int_{t-\Delta t}^{t} \frac{e^{-D^2/4(t-t')}}{\sqrt{4\pi(t-t')}} \tilde{h}(x, t')\, dt' \\ (29) \quad &= \sqrt{\frac{\Delta t}{4\pi}} e^{-D^2/\delta} \left[\Gamma_0\left(-\frac{1}{2}, \frac{D^2}{\delta}\right) - \Gamma_0\left(-\frac{3}{2}, \frac{D^2}{\delta}\right) \right] \sqrt{\frac{R}{\rho}} g(x_0, t) \\ &+ \sqrt{\frac{\Delta t}{4\pi}} \Gamma_0\left(-\frac{3}{2}, \frac{D^2}{\delta}\right) \int_{\Gamma(t-\Delta t)} \frac{e^{-|x-y|^2/\delta}}{\sqrt{\pi\delta}} g(y, t - \Delta t)\, dy + O(\Delta t^{5/2}). \end{aligned}$$

Here Γ_0 is the normalized incomplete gamma function defined by

$$\Gamma_0(a, \xi) = e^{\xi} \xi^{-a} \int_{\xi}^{\infty} e^{-z} z^{a-1}\, dz \qquad a < 0.$$

We now have an approximate evolution formula

$$u(x, t+\Delta t) = \frac{1}{\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} u(y,t)\,dy$$
$$+ W_1(x,t)g(x_0, t+\Delta t) + W_2(x,t)\int_{\Gamma(t)} \frac{e^{-|x-y|^2/\delta}}{\sqrt{\pi\delta}} g(y,t)\,dy,$$

accurate to $O(\Delta t^{5/2})$. Here W_0 and W_1 are given in (29) as functions of x, Δt, and $\Gamma(t)$. To evaluate u by this formula, we need to find the nearest point x_0 on $\Gamma(t+\Delta t)$ to x and estimate the curvature and normal of $\Gamma(t+\Delta t)$ at x_0. Finding the nearest point can be done efficiently by binning pieces of $\Gamma(t+\Delta t)$ and searching bins.

To evaluate u, we also need to evaluate the "Gauss transform on a curve" defined by

$$\Gamma_\delta g(x) = \int_\Gamma \frac{e^{-|x-y|^2/\delta}}{\sqrt{\pi\delta}} g(y)\,dy.$$

This can be approximated adaptively by the Hermite expansion technique used for the homogeneous case. We begin by representing Γ and g as piecewise polynomials of degree d, within accuracy ϵ. To do this, we first lay down a coarse grid on Γ (parametrized by $s \in [0,1]$, say), and construct piecewise polynomial interpolants to the coordinate functions of Γ and to g at meshpoints s_j equispaced in $[0,1]$. In the style of §3, we now refine our representation of Γ adaptively whenever the polynomial interpolation fails to represent either Γ or g accurately. The result is a representation

$$\Gamma = \cup_j \Gamma_j + O(\epsilon),$$
$$g = \tilde{g} + O(\epsilon),$$

where Γ_j is an element; if we use linear interpolation, for example, Γ_j is a line segment.

Now we apply the expansion technique of §2. Lay down cells C of size $2r\sqrt{\delta}$ to cover Γ, and expand the contribution of $\Gamma \cap C$ as a series about the cell center c. We find

$$\Gamma_\delta g(x) = \sum_C \sum_{\alpha\geq 0} C_\alpha h_\alpha\left(\frac{x-c}{\sqrt{\delta}}\right) + E$$

with $E = O(\epsilon)$ and

$$C_\alpha = \frac{1}{\sqrt{\pi\delta}}\int_{\Gamma\cap C}\left(\frac{y-c}{\sqrt{\delta}}\right)^\alpha \tilde{g}(y)\,dy$$
$$= \frac{1}{\sqrt{\pi\delta}}\sum_{\Gamma_j\subset C}\int_{\Gamma_j}\left(\frac{y-c}{\sqrt{\delta}}\right)^\alpha \tilde{g}(y)\,dy,$$

where $\Gamma = \cup\Gamma_j$. (In order to have Γ divided into elements each of which lies in a single cell C, we perform a further subdivision of Γ by cutting any element crossing more than one cell.) This Hermite series can be truncated with the usual bounds.

We now have to evaluate integrals of the form

$$\int_{\Gamma_j}\left(\frac{y-c}{\sqrt{\delta}}\right)^\alpha \tilde{g}(y)\,dy,$$

where $\tilde{g}$ is a polynomial. If we use linear interpolation, the recurrence relation for doing this was developed in §3. Otherwise, similar but more complicated recurrence relations can be developed, or Gauss–Legendre integration of sufficiently high order can be used.

A major advantage of the present approach is that we are able to use high-order approximations of Γ without having to integrate Gaussians over such approximations. The difficulty of carrying out these calculations has been a major stumbling block in the construction of high-order product integration schemes, both for the heat equation and the Laplace equation. With the current approach, one needs only to integrate powers over polynomial curve elements, a technique that can always be carried out. This advantage was observed but not developed in [6]. The fast multipole method [7] can be used in a similar way to eliminate the necessity of product integration in potential theory for the Laplace equation. This approach seems likely to be particularly useful in three-dimensional problems, where product integration is more difficult.

The remainder of the calculation is straightforward; we present some preliminary numerical results in §7. We have described only the single layer potential, but the analysis of the double layer potential is quite similar. The jump in the double layer potential across the boundary complicates matters surprisingly little.

7. Numerical experiments.

In this section, we describe the results of codes written for the homogeneous problem and the single layer potential. The codes were written in standard FORTRAN 77 and run on a SUN SPARCstation 1+ with the optimizer flag -O, using double-precision arithmetic. The timings reported are usually reproducible within 1 or 2%, which is sufficient for our purposes.

First, we implemented Method 2.1 for the homogeneous problem, checking the numerical results against exact solutions for three sets of exact temperature fields; each is produced by shifting, scaling, and summing a basic solution. We put

$$u(x, y, t) = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} u_j(k_1(x - x_{k_1}), k_1^2 t) u_j(k_2(y - y_{k_2}), k_2^2 t),$$

where x_{k_1} and y_{k_2} are random uniformly distributed on $[-1, 1]$. The three basic solutions are given by

$$u_1(x, t) = \frac{e^{1/4T} e^{-1/4}}{\sqrt{T}} \cos\left(\frac{x}{T}\right) e^{-x^2/T}$$

where $T = 1 + 4t$,

$$u_2(x, t) = \begin{cases} \max(1 - |x|, 0) & \text{if } t = 0, \\ \Lambda^2[\frac{1}{2}(x - 1)\text{erf}((x - 1)/\sqrt{4t}) + \sqrt{t/\pi} e^{-(x-1)^2/4t}] & \text{if } t > 0, \end{cases}$$

where $\Lambda f(x) = f(x + 1) - f(x)$, and

$$u_3(x, t) = \begin{cases} \chi_{[-1,1]}(x) & \text{if } t = 0, \\ \frac{1}{2}[\text{erf}((x + 1)/\sqrt{4t}) - \text{erf}((x - 1)/\sqrt{4t})] & \text{if } t > 0, \end{cases}$$

where $\chi_{[-1,1]}(x)$ is zero for $|x| > 1$ and 1 otherwise. The first solution is smooth with K^2 sharp peaks of scales from 1 to $\frac{1}{K}$, and decays exponentially at infinity. The second is piecewise linear, continuous, Lipschitz, and piecewise smooth but not C^1 at $t = 0$. The third is discontinuous in x at $t = 0$ but smooth and sharply varying for $t > 0$. We used our method to compute u for ten timesteps equispaced from 0 to 1, beginning each triangulation with a 14×14 grid of square cells on the domain $[-7, 7]^2$ and using linear, quadratic, and cubic interpolation with various error tolerances ϵ. We took $K = 10$, so each solution u_j varies over scales from 1 to 0.1. We used $p = 6$ and $I = 2$ to achieve error $\epsilon = 10^{-2}$ relative to $|u|_\infty$ at each step.

Table 1 reports the errors and times produced by Method 2.1 on these three temperature fields. Several conclusions can be drawn from Table 1. The method achieves the requested accuracy in every case. The time required for accuracy ϵ scales roughly like ϵ^{-1} for linear, $\epsilon^{-2/3}$ for quadratic, and $\epsilon^{-1/2}$ for cubic interpolation, as it should.

For purposes of comparison, the standard explicit nonadaptive finite difference method

$$u_{ij}^{n+1} = u_{ij}^n + \frac{\Delta t}{h^2}(\Lambda_i^2 + \Lambda_j^2)u_{i-1,j-1}^n$$

on a square $N \times N$ grid requires time $O(\epsilon^{-2})$ to attain accuracy ϵ. Indeed, this method has error $O(\Delta t + h^2)$ and is stable if $\Delta t \le h^2/4$, so decreasing the error by a factor of 4 requires halving h and reducing Δt by 4. Since this method requires $O(h^{-2})$ work per step, we see that decreasing the error by a factor of 4 requires 16 times as much work, corresponding to $O(\epsilon^{-2})$. A fourth-order explicit method would need $O(\epsilon^{-1})$ work to achieve error less than ϵ.

Our method is relatively unfazed by nonsmooth or discontinuous initial temperature fields as long as they are bounded. It resolves the discontinuity as well as it can, and ends up with a very good approximation except on a very small area. The small area where the interpolant is inaccurate does not affect the total error, essentially because the heat equation is stable in L^1 as well as L^∞.

TABLE 1

Times and errors for the homogeneous free-space heat equation. Column N_0/N_1 displays the number of triangles in the triangulation at $t = 0$ and at $t = 1$, T is the total computing time for ten steps from $t = 0$ to $t = 1$, and E is the maximum error, divided by the maximum of the solution. Results for lower-degree interpolation with small ϵ, shown with dashes, were too time consuming and were omitted.

u_1	Linear			Quadratic			Cubic		
ϵ	N_0/N_1	T	E	N_0/N_1	T	E	N_0/N_1	T	E
10^{-1}	3198/784	52	.96−1	1076/394	66	.81−1	728/392	115	.76−1
10^{-2}	27634/5260	752	.43−2	3382/590	276	.18−2	1550/396	317	.17−2
10^{-3}	264454/49028	8812	.44−3	14460/1888	1079	.20−3	3948/532	628	.21 − 3
10^{-4}	–	–	–	64056/8126	5619	.20−4	11096/1236	1898	.14−4
10^{-5}	–	–	–	–	–	–	33496/3604	11178	.22−5
u_2	Linear			Quadratic			Cubic		
ϵ	N_0/N_1	T	E	N_0/N_1	T	E	N_0/N_1	T	E
10^{-1}	4548/784	26	.98−1	2188/394	32	.82−1	1410/392	55	.77−1
10^{-2}	54664/5242	568	.46−2	27340/588	318	.19−2	21746/396	446	.18−2
10^{-3}	399988/48982	7834	.48−3	316904/1884	2973	.22−3	249236/534	3833	.22−3
10^{-4}	–	–	–	399988/8152	7350	.21−4	399988/1242	7764	.14−4
10^{-5}	–	–	–	–	–	–	399988/3654	16443	.23−5
u_3	Linear			Quadratic			Cubic		
ϵ	N_0/N_1	T	E	N_0/N_1	T	E	N_0/N_1	T	E
10^{-1}	5106/782	25	.10−0	5290/394	41	.84−1	5344/392	80	.75−1
10^{-2}	14632/5266	457	.86−2	15032/590	213	.52−2	15060/394	309	.42−2
10^{-3}	115852/49016	6476	.95−3	115852/1878	1430	.18−2	115852/530	1779	.60−3

One feature that is not apparent from the tables is that evaluation of u is more costly than evaluation of the coefficients. Typically the code spends 80% of its time evaluating u and only 20% evaluating coefficients. This is partly due to inefficiency; the refinement test we use wastes many evaluations of u when the grid is almost completed. We plan to address this admittedly minor point in future improvements.

Finally, we present some preliminary numerical results for the evaluation of the single layer potential

$$Sg(x,t) = \int_0^t \int_{\Gamma(t')} \frac{e^{-|x-y|^2/4(t-s)}}{4\pi(t-s)} g(y,s)\,dy\,ds.$$

In order to compute the error, we took a very simple case with $g = 1$ and $\Gamma(t)$ a stationary circle with center $(0,0)$ and radius 1.1. (Of course, we coded the method for a general curve and density.) We computed Sg on an adaptive grid using N steps until $t = 1$, setting $\epsilon = 10^{-1}$ initially and reducing ϵ by a factor of 4 for each successive calculation. The triangulation with $\epsilon = 10^{-1}$ is shown in Fig. 6 at $t = 1$; the triangulation for this problem changes little over time.

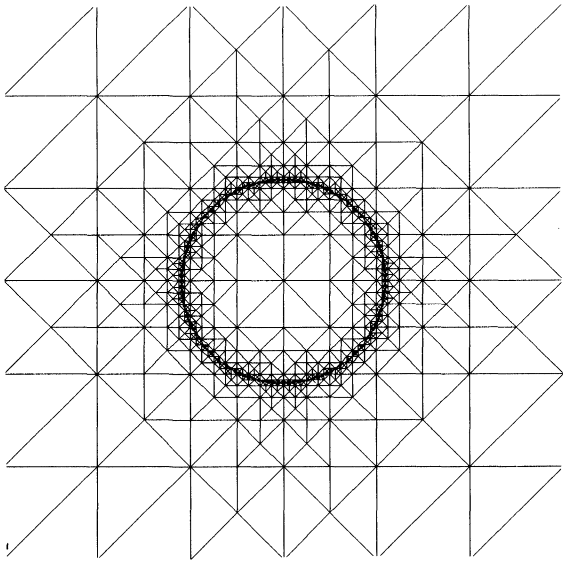

FIG. 6. *Triangulation for the single layer potential of unit density on a circle with center* $(0,0)$ *and radius* 1.1. *Here* $\epsilon = 10^{-2}$ *and* $t = 0.1$. *The smallest triangle has sides of length* 2^{-5} *times the maximum side length and there are* 1232 *triangles with* 5581 *nodes in the box* $[-3,3]$. *The maximum level of subdivision was* 10. *The dotted circle is the curve* $\Gamma(t)$; *it is resolved to accuracy* $\epsilon' = 10^{-3}$ *with* 120 *line segments, subdivided further to have each line segment lie in a single cell.*

Table 2 presents numerical results for this calculation with $N = 10, 20, 40$ steps and cubic interpolation. T is the total computing time, E is the maximum error, divided by the maximum of the solution, and N_T is the number of triangles at $t = 1$.

TABLE 2
Times and errors for the single-layer heat potential; T *is the total computing time for* N *steps from* $t = 0$ *to* $t = 1$, *and* E *is the maximum error, divided by the maximum of the solution.*

Δt	h	p	l	N_T	T	E
0.1	1.33	8	1	218	79	.58−1
0.05	0.923	10	2	520	1104	.15−1
0.025	0.632	11	2	1220	6694	.37−2

We conclude that the method is expensive but accurate; perhaps its best feature is the elimination of product integration.

8. Conclusions. We have described an efficient adaptive approach to several heat flow problems that arise in physics. The simplest is the homogeneous free-space heat equation, for which we constructed several methods. All are based on the Hermite expansion of the heat kernel, combined with an adaptive triangulation scheme that represents a given function f as a piecewise polynomial within an error $\epsilon|f|_\infty$. Another key ingredient is the idea of writing the solution as a *function* that can be evaluated at any point, rather than a set of values to be interpolated.

We then demonstrated how to construct methods for the inhomogeneous case, using Duhamel's principle, and how to evaluate layer potentials. We treated layer potentials as inhomogeneous problems with distributional right-hand sides and treated the time singularities by product integration. The Hermite expansion is highly useful here, because it makes it unnecessary to carry out product integration in space. Thus even for high-order curve representations such as cubic splines, we still need only to evaluate integrals of *monomials* over the curve; we do not have to integrate Gaussians over a cubic spline curve, which is very difficult or impossible to do exactly. The Hermite series approach also combines the accuracy of product integration with the speed of the fast algorithm; usually product integration and fast algorithms do not marry well, because fast algorithms depend on processing the source independently of the location of the target then evaluating. In this case, the product integration simply *postmultiplies* the output of the fast summation technique.

Numerical results show these methods to be efficient and accurate. They perform particularly well in spatially rough problems where considerable accuracy is required.

The present techniques are formulated for the heat equation. It is shown in [4] how to extend fast techniques for the heat equation to nonlinear parabolic problems, and a similar technique works for variable-coefficient linear problems. Thus the techniques presented in this paper seem likely to be broadly applicable.

Appendix A. Cartesian coefficients. The polynomial of degree d that interpolates given values of a function f at the nodes shown in Fig. 1 (for $d = 1, 2, 3$) is usually expressed in terms of the barycentric coordinates $\lambda_1(x, y), \lambda_2(x, y), \lambda_3(x, y)$ defined by

$$\begin{aligned} \lambda_1 + \lambda_2 + \lambda_3 &= 1, \\ x_1\lambda_1 + x_2\lambda_2 + x_3\lambda_3 &= x, \\ y_1\lambda_1 + y_2\lambda_2 + y_3\lambda_3 &= y, \end{aligned} \tag{30}$$

with $(x_i, y_i) = v_i$ the vertices of the triangle. The barycentric coordinates of a point v can be computed in terms of the components x and y of v, by

$$\lambda_i = a_i x + b_i y + c_i, \tag{31}$$

by solving (30).

The $(d + 1)(d + 2)/2$ nodes required for interpolation of degree d are shown in Fig. 1. The interpolants of degrees $d = 1, 2, 3$ at these nodes are given by [3]

$$\begin{aligned} f(x) &= \sum_{i=1}^{3} \lambda_i f(v_i) \qquad (d = 1), \\ f(x) &= \sum_{i=1}^{3} \lambda_i(2\lambda_i - 1) f(v_i) + \sum_{i<j} 4\lambda_i\lambda_j f(v_{ij}) \qquad (d = 2), \\ f(x) &= \frac{1}{2}\sum_{i=1}^{3} \lambda_i(3\lambda_i - 1)(3\lambda_i - 2) f(v_i) + \frac{9}{2}\sum_{i\neq j} \lambda_i\lambda_j(3\lambda_i - 1) f(v_{iij}) \\ &\quad + 27 \sum_{i<j<k} \lambda_i\lambda_j\lambda_k f(v_{ijk}) \qquad (d = 3). \end{aligned}$$

The numbering of the nodes is shown in Fig. 1 and follows [3].

In the linear case, it is straightforward to transform from barycentric to Cartesian coordinates:

$$f(x,y) = (a\cdot f)x + (b\cdot f)y + (c\cdot f),$$

where $a\cdot f = \sum_{i=1}^{3} a_i f(v_i)$, and so forth.

For the quadratic case, we find after some tedious calculations that

$$f(x,y) = \sum_{|\alpha|\le 2} F_\alpha x^\alpha = F_{20}x^2 + F_{11}xy + F_{02}y^2 + F_{10}x + F_{01}y + F_{00},$$

where the Cartesian coefficients F_{ij} are given by

$$F_{20} = \sum_{i=1}^{3} 2a_i^2 f_i + \sum_{i<j} 4a_i a_j f_{ij},$$

$$F_{11} = \sum_{i=1}^{3} 4a_i b_i f_i + \sum_{i<j} 4(a_i b_j + a_j b_i) f_{ij},$$

$$F_{02} = \sum_{i=1}^{3} 2b_i^2 f_i + \sum_{i<j} 4b_i b_j f_{ij},$$

$$F_{10} = \sum_{i=1}^{3} 2a_i(4c_i - 1) f_i + \sum_{i<j} 4(a_i c_j + a_j c_i) f_{ij},$$

$$F_{01} = \sum_{i=1}^{3} 2b_i(4c_i - 1) f_i + \sum_{i<j} 4(b_i c_j + b_j c_i) f_{ij},$$

$$F_{00} = \sum_{i=1}^{3} 2c_i(2c_i - 1) f_i + \sum_{i<j} 4c_i c_j f_{ij}.$$

Here we have abbreviated $f_i = f(v_i)$, $f_{ij} = f(v_{ij})$, and so forth.

The calculations for the cubic case are even more tedious and lead to

$$\begin{aligned} f(x,y) &= \sum_{|\alpha|\le 3} F_\alpha x^\alpha \\ &= F_{30}x^3 + F_{21}x^2y + F_{12}xy^2 + F_{03}y^3 \\ &\quad + F_{20}x^2 + F_{11}xy + F_{02}y^2 + F_{10}x + F_{01}y + F_{00}, \end{aligned}$$

with

$$F_{30} = \frac{9}{2}\sum_{i=1}^{3} a_i^3 f_i + \frac{27}{2}\sum_{i\ne j} a_i^2 a_j f_{iij} + 27a_1a_2a_3 f_{123},$$

$$\begin{aligned} F_{21} &= \frac{27}{2}\sum_{i=1}^{3} a_i^2 b_i f_i + \frac{27}{2}\sum_{i\ne j} a_i(a_i b_j + 2b_i a_j) f_{iij} \\ &\quad + 27(a_1a_2b_3 + a_1b_2a_3 + b_1a_2a_3) f_{123}, \end{aligned}$$

$$\begin{aligned} F_{12} &= \frac{27}{2}\sum_{i=1}^{3} b_i^2 a_i f_i + \frac{27}{2}\sum_{i\ne j} b_i(b_i a_j + 2a_i b_j) f_{iij} \\ &\quad + 27(b_1b_2a_3 + b_1a_2b_3 + a_1b_2b_3) f_{123}, \end{aligned}$$

$$F_{03} = \frac{9}{2}\sum_{i=1}^{3} b_i^3 f_i + \frac{27}{2}\sum_{i\neq j} b_i^2 b_j f_{iij} + 27 b_1 b_2 b_3 f_{123},$$

$$F_{20} = \frac{9}{2}\sum_{i=1}^{3} a_i^2(3c_i - 1) f_i + \frac{9}{2}\sum_{i\neq j} a_i(3a_i c_j + (6c_i - 1)a_j) f_{iij}$$

$$+ 27(a_1 a_2 c_3 + a_1 c_2 a_3 + c_1 a_2 a_3) f_{123},$$

$$F_{11} = 9\sum_{i=1}^{3} a_i b_i (3c_i - 1) f_i + \frac{9}{2}\sum_{i\neq j}(6a_i b_i c_j + a_i(6c_i - 1)b_j + b_i(6c_i - 1)a_j) f_{iij}$$

$$+ 27(c_1(a_2 b_3 + a_3 b_2) + c_2(a_1 b_3 + a_3 b_1) + c_3(a_1 b_2 + a_2 b_1)) f_{123},$$

$$F_{02} = \frac{9}{2}\sum_{i=1}^{3} b_i^2(3c_i - 1) f_i + \frac{9}{2}\sum_{i\neq j} b_i(3b_i c_j + (6c_i - 1)b_j) f_{iij}$$

$$+ 27(b_1 b_2 c_3 + b_1 c_2 b_3 + c_1 b_2 b_3) f_{123},$$

$$F_{10} = \frac{1}{2}\sum_{i=1}^{3} a_i(27c_i^2 - 18c_i + 2) f_i + \frac{9}{2}\sum_{i\neq j}(a_i(6c_i - 1)c_j + c_i(3c_i - 1)a_j) f_{iij}$$

$$+ 27(a_1 c_2 c_3 + c_1 a_2 c_3 + c_1 c_2 a_3) f_{123},$$

$$F_{01} = \frac{1}{2}\sum_{i=1}^{3} b_i(27c_i^2 - 18c_i + 2) f_i + \frac{9}{2}\sum_{i\neq j}(b_i(6c_i - 1)c_j + c_i(3c_i - 1)b_j) f_{iij}$$

$$+27(b_1 c_2 c_3 + c_1 b_2 c_3 + c_1 c_2 b_3) f_{123},$$

$$F_{00} = \frac{1}{2}\sum_{i=1}^{3} c_i(3c_i - 1)(3c_i - 2) f_i + \frac{9}{2}\sum_{i\neq j} c_i(3c_i - 1)c_j f_{iij} + 27 c_1 c_2 c_3 f_{123}.$$

REFERENCES

[1] N. Bleistein and R. A. Handelsman, *Asymptotic Expansions of Integrals*, Dover, New York, 1986.

[2] R. Brown, *Layer Potentials and Boundary Value Problems for the Heat Equation on Lipschitz Cylinders*, Ph.D. thesis, Dept. of Mathematics, University of Minnesota, Duluth, 1987.

[3] P. G. Ciarlet, *Numerical Analysis of the Finite Element Method*, Seminaire de Mathématique Supérieures Été 1975, Université de Montréal, Montréal, 1976.

[4] J. Epperson, *Semi-group linearization for nonlinear parabolic equations*, preprint, 1991.

[5] G. B. Folland, *Introduction to Partial Differential Equations*, Mathematical Notes Number 17, Princeton University Press, Princeton, NJ, 1976.

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

[7] L. G. J. Carrier and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 669–686.

[8] W. F. Mitchell, *A comparison of adaptive refinement techniques for elliptic problems*, ACM Trans. Math. Software, 15 (1989), pp. 326–347.

[9] W. Pogorzelski, *Integral Equations and Their Applications*, Pergamon Press, Oxford, 1966.

[10] J. Strain, *The fast Gauss transform with variable scales*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1131–1139.

[11] G. Szegö, *Orthogonal Polynomials*, American Mathematical Society Colloquium Publications Vol. XXIII, American Mathematical Society, Providence, RI, 1939.