SIAM J. NUMER. ANAL.
Vol. 29, No. 2, pp. 332–364, April 1992

ANALYSIS OF A ONE-DIMENSIONAL MODEL FOR THE IMMERSED BOUNDARY METHOD*

R. P. BEYER† AND R. J. LEVEQUE‡

Abstract. Numerical methods are studied for the one-dimensional heat equation with a singular forcing term,

$$u_t = u_{xx} + c(t)\delta(x - \alpha(t)).$$

The delta function $\delta(x)$ is replaced by a discrete approximation $d_h(x)$ and the resulting equation is solved by a Crank–Nicolson method on a uniform grid. The accuracy of this method is analyzed for various choices of d_h. The case where $c(t)$ is specified and also the case where c is determined implicitly by a constraint on the solution at the point α are studied. These problems serve as a model for the immersed boundary method of Peskin for incompressible flow problems in irregular regions. Some insight is gained into the accuracy that can be achieved and the importance of choosing appropriate discrete delta functions.

Key words. numerical analysis, immersed-boundary method, error analysis, discrete delta function

AMS(MOS) subject classifications. 65M10, 65N10, 76-08, 76D05, 76Z05

1. Introduction.

We study numerical methods for time-dependent partial differential equations in one space variable of the form

$$u_t = u_{xx} + f(x, t, u) \tag{1.1}$$

where f is a singular forcing term, or source term, typically a linear combination of delta functions. Here we only consider a single delta function with variable strength c and position α,

$$f(x, t, u) = c(t)\,\delta(x - \alpha(t)), \tag{1.2}$$

although the results would extend to more general f.

One simple example is the heat equation with a variable strength delta function source at the fixed point α,

$$u_t = u_{xx} + c(t)\delta(x - \alpha). \tag{1.3}$$

The strength $c(t)$ might be given a priori, or it might be unknown, and determined as part of the solution by imposing some additional constraint such as

$$u(\alpha, t) = \bar{u}(t) \tag{1.4}$$

where $\bar{u}(t)$ is given.

If the point $\alpha(t)$ is moving then this location may be specified or may also be an unknown which is to be determined, perhaps through the specification of a coupled ordinary differential equation of the form

$$\alpha'(t) = \psi(t, u). \tag{1.5}$$

* Received by the editors September 24, 1990; accepted for publication (in revised form) May 17, 1991.

† Department of Applied Mathematics and the Center for Bioengineering, University of Washington, Seattle, Washington 98195.

‡ Department of Mathematics, University of Washington, Seattle, Washington 98195.

Such problems arise, for example, in free-boundary solidification or melting problems where $\alpha(t)$ is the boundary between phases and the latent heat release provides a delta function heat source at this boundary. Although the techniques discussed here can be extended to this problem, we will not study this case in the present paper but will restrict our attention to the case where $\alpha(t)$ is specified.

Our interest in such equations was initially motivated by the desire to study Peskin's immersed boundary method for incompressible fluid dynamics. This numerical method applies to two or three dimensional versions of (1.1) in which u represents the fluid velocity. The point $\alpha(t)$ then becomes a curve in two dimensions or a surface in three dimensions, which represents a boundary immersed in the fluid. The boundary exerts forces on the fluid, which leads to a singular forcing term in the Navier–Stokes equations, with support only on the immersed boundary. In addition, the boundary moves at the local fluid velocity giving a coupled equation for motion of the boundary analogous to the ODE (1.5). If the boundary is stationary then the force at the boundary is determined by the constraint that the fluid velocity should be zero along the boundary, analogous to (1.4). These equations, and Peskin's immersed boundary method for their solution, will be described in more detail in the next section.

The one-dimensional analogue of this numerical method, for (1.1), would take the following form. We specify a uniform fixed grid with grid points $x_j = jh$, $j = 0, 1, \cdots, N$ with $h = 1/N$. We also choose a timestep $k = \Delta t$ and set $t_n = nk$. The solution $u(x,t)$ is approximated by the discrete values $U_j^n \approx u(x_j, t_n)$.

The idea in an immersed boundary method is to solve difference equations only on the uniform grid, obtaining the solution value at the point $\alpha(t)$ (if required) via some form of interpolation. Typically the point $\alpha(t)$ will not lie exactly at a grid point x_j, and so the delta function forcing function must be replaced by inhomogeneous terms in the difference equations on the uniform grid. One way to do this is to replace the delta function $\delta(x)$ by a discrete approximation $d_h(x)$. Some examples: the "hat function" with support $(-h, h)$,

$$d_h^{(1)}(x) = \begin{cases} (h - |x|)/h^2, & |x| \le h, \\ 0 & \text{otherwise}, \end{cases} \tag{1.6}$$

a wider hat function with support $(-2h, 2h)$,

$$d_h^{(2)}(x) = \begin{cases} (2h - |x|)/4h^2, & |x| \le 2h, \\ 0 & \text{otherwise}, \end{cases} \tag{1.7}$$

or perhaps a smoother version, such as

$$d_h^{(3)}(x) = \begin{cases} (\cos(\pi x/2h) + 1)/4h, & |x| \le 2h, \\ 0 & \text{otherwise}. \end{cases} \tag{1.8}$$

This last delta function was introduced by Peskin [6] because of certain translation invariance properties.

Once any delta functions appearing in the function f in (1.1) are replaced by suitable discrete delta functions, the resulting equation is discretized using standard finite difference methods on the uniform grid. For example, the heat equation (1.3) would be replaced by

$$u_t = u_{xx} + c(t)\, d_h(x - \alpha)$$

and then discretized, using perhaps the standard Crank–Nicolson method,

$$\frac{1}{k}(U_j^{n+1} - U_j^n) = \frac{1}{2}(D_x^2 U_j^n + D_x^2 U_j^{n+1}) + c(t_{n+1/2}) d_h(x_j - \alpha). \tag{1.9}$$

Here D_x^2 represents the centered approximation to the second derivative:

$$D_x^2 U_j^n = \frac{1}{h^2}(U_{j+1}^n - 2U_j^n + U_{j-1}^n). \tag{1.10}$$

If the location α is varying with time then we can replace α in (1.9) by $\alpha(t_{n+1/2})$. Alternatively, the source term in (1.9) can be replaced by

$$\tfrac{1}{2}(c(t_n) d_h(x_j - \alpha(t_n)) + c(t_{n+1}) d_h(x_j - \alpha(t_{n+1}))).$$

This latter form turns out to be easier to analyze.

Note that each of the discrete delta functions above has the property that

$$h \sum_j d_h(x_j - \alpha) = 1 \tag{1.11}$$

for any choice of α. Hence the correct total source is transferred to the discrete grid; it is simply shifted from the point α to the neighboring grid points.

For the one-dimensional problem there are clearly more sophisticated ways to handle this problem, e.g., using a moving mesh or front tracking algorithm to capture the location of the boundary exactly at a grid point in each step. However, in two or more dimensions there are real advantages to the present approach: very efficient PDE solvers can be used on the uniform rectangular grid and complex immersed boundaries can be handled by using discrete delta functions to spread the singular source terms to the uniform grid.

A related method for elliptic equations in two space dimensions has been developed by Mayo [5], and also involves embedding the irregular region into a rectangle and then solving discretized equations on a uniform grid. The extended solution over the rectangle has jumps in certain derivatives along the embedded boundary that can be determined by solving an integral equation. These jumps are then used to modify the finite difference equations near the embedded boundary so that the resulting discrete solution has the correct discontinuities. These modifications are analogous to the discrete delta function term in (1.9), which modifies the standard Crank–Nicolson method for the heat equation only near the point α where u is not smooth.

In fact, we will see that this viewpoint is very useful in analyzing (1.9) and related methods. The discrete delta function is a correction to the centered difference operator D_x^2 that allows us to compute accurate approximations to u_{xx} from discrete grid values even when the underlying function u has discontinuous derivatives between the grid points (see Lemma 6.1 and Corollary 6.2 below).

Peskin's immersed boundary method for fluid dynamics [6] is described in more detail in the next section to motivate further some of the one-dimensional model problems studied here. The remainder of the paper is devoted to the study of numerical methods on the model problems. In particular, we investigate the accuracy that can be achieved and the effect that the choice of discrete delta function can have on these results.

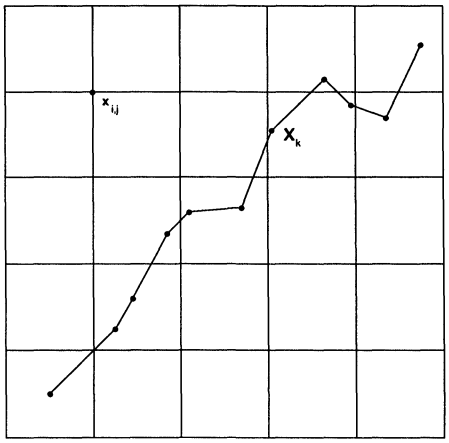


FIG. 1. *Uniform fluid grid and immersed boundary points.*

2. Peskin's immersed boundary method. The immersed boundary method is a numerical method for solving the incompressible Navier–Stokes equations in domains with geometrically complicated boundaries. This method was originally introduced for studying blood flow in the heart [6], but has since been used to study other problems, e.g., [2]–[4]. Even in a two-dimensional model, the heart wall has a complicated shape that varies with time as the heart goes through each pumping cycle. Contractions of the heart muscle exert forces at the wall that interact with the fluid dynamical forces, so that the resulting boundary configuration is a part of the desired solution. Due to the complicated shape and the time-dependent nature, it would be very difficult to impose boundary conditions on a numerical method in standard ways.

Peskin's approach is to immerse the entire structure in a rectangular box of fluid. This box is discretized using a uniform rectangular grid. For simplicity, periodic boundary conditions are imposed at the boundaries of the box. Solving the incompressible Navier–Stokes equations on this regular domain is easily accomplished with a variety of numerical methods. The use of periodic boundary conditions leads to additional efficiencies since fast Fourier transform techniques can be used.

Of course, some mechanism must be introduced to simulate the original boundary, which is now an immersed boundary with fluid on either side. This immersed boundary is modeled by an additional set of points $X_k(t) \in \mathbb{R}^2$ (in two space dimensions) that indicate the location of the boundary at each time t. (See Fig. 1.) Since fluid should not flow across this boundary, these points are moved in each timestep according to the local fluid velocity, i.e., by solving the ODEs

$$X_k'(t) = u(X_k(t), t). \tag{2.1}$$

In the numerical method, the fluid velocity at the point $X_k(t)$ must be determined in each timestep from velocity values on the uniform grid, using some form of interpolation.

Conversely, the immersed boundary must have some effect on the fluid dynamics. In the immersed boundary method, the boundary affects the fluid not by the introduction of boundary conditions in the Navier–Stokes equations, but rather by the introduction of a inhomogeneous forcing term in the equations. The incompressible

Navier–Stokes equations then take the form

$$u_t + u \cdot (\nabla u) = \mu u_{xx} + f, \tag{2.2}$$

$$\nabla \cdot u = 0 \tag{2.3}$$

where the forcing term f is a singular function with support only along the immersed boundary. In the discrete method, the support of f is only at the discrete points $X_k(t)$, and f takes the form

$$f(x,t) = \sum_k F_k(t)\delta(x - X_k(t))$$

where $F_k \in \mathbb{R}^2$ is the force vector at the point X_k and δ is the two-dimensional delta function. This is further discretized by replacing the true delta functions by discrete versions, two-dimensional analogues of the functions d_h discussed above.

When the Navier–Stokes equations are discretized on the Cartesian grid, the equation for the velocity u_{ij} at grid point (i,j) involves the forcing term

$$f_{ij} = \sum_k F_k(t)\, d_h(x_{ij} - X_k(t)). \tag{2.4}$$

By introducing this discrete delta function, each point source $F_k(t)\delta(x - X_k(t))$ is spread to several neighboring grid points on the rectangular fluid grid. As in one dimension, an appropriate choice of d_h will give the correct total force, i.e., a two-dimensional version of (1.11) should hold.

Peskin also uses the discrete delta function to perform the interpolation operation described earlier, which is needed to find $U_k(t) \approx u(X_k(t),t)$ from the grid values of velocity u_{ij}. The interpolation formula is

$$U_k = \sum_{ij} u_{ij}\, d_h(x_{ij} - X_k). \tag{2.5}$$

The forces F_k are determined in some manner based on the correct physical boundary conditions. This might involve the configuration of the boundary (e.g., how much it is stretched, or how far it is from some equilibrium position), the fluid velocity at the boundary, external forces (e.g., contractile forces of the heart muscle), or other effects. In each timestep these forces are calculated based on the current configuration, velocities, external forces, etc.

Note that we use upper-case letters (e.g., X_k and F_k) to represent values on the immersed boundary and lower-case letters (e.g., u_{ij}, f_{ij}) to represent values on the regular grid.

To summarize, the immersed boundary method involves the following steps:

1. Given velocities u_{ij}^n and boundary locations X_k^n at time t_n, compute the forces F_k^n at the immersed points.
2. Spread the forces F_k^n to values f_{ij}^n on the uniform grid using (2.4).
3. Solve the inhomogeneous Navier–Stokes equations (2.2) on the regular grid with periodic boundary conditions to obtain u_{ij}^{n+1}.
4. Interpolate the resulting velocities u_{ij}^{n+1} to the immersed boundary using (2.5) to obtain U_k^{n+1}.
5. Move the boundary points X_k^n to X_k^{n+1} using the velocities U_k^n, U_k^{n+1} and some discrete approximation to the ODE (2.1).

6. Return to Step 1 for the next timestep.

In practice, the algorithm is complicated by the fact that the forces F_k must be calculated in some implicit or at least semi-implicit manner in order to maintain stability with reasonable timesteps. See [6] for a more complete description of the method.

This method is perhaps most natural in situations where specification of the force exerted by the immersed boundary is the natural boundary condition for the problem, as in the heart model where forces exerted by the heart wall can be directly calculated from the boundary configuration and the stage of the heart cycle. In other problems the natural boundary conditions might take a different form. For example, if we consider flow around a solid stationary body then the boundary condition is simply $u(x,t) \equiv 0$ at the solid wall. However, this can also be reformulated in terms of an inhomogeneous forcing term so that the immersed boundary method can be used. In this case the force $f(x,t)$ is again a singular force at the boundary, and is determined by the requirement that the resulting fluid velocity $u(x,t)$ should satisfy the boundary condition at the solid wall. The solid wall exerts just enough force on the fluid to maintain the no-flow boundary condition. (This is analogous to using the constraint (1.4) in the one-dimensional problem to determine the source strength $c(t)$.) In the discrete method we can choose F_k^n in Step 1 by an implicit scheme, requiring that the resulting velocity U_k^{n+1} obtained in Step 4 satisfy $U_k^{n+1} = 0$. This gives a large system of equations to be solved for the forces F_k^n. Another approach (which requires less computational work) would be to view each point X_k as being attached to some equilibrium position X_k^0 (the correct location of the fixed boundary) by a very stiff spring with spring constant $s \gg 1$. Then the force F_k^n might be given by

$$F_k^n = -s(X_k^n - X_k^0)$$

in the simplest explicit case. This would of course cause severe stability problems (exponentially growing oscillations of the boundary) unless very small timesteps were used, but a semi-implicit version of this can be successfully used in some cases.

Work on both this semi-implicit stiff spring approach and the fully implicit approach is reported in Beyer [2]. An efficient and high-order accurate version of this method for solid wall boundaries, or more general boundary conditions, would be very useful for solving a variety of fluid dynamics problems in complicated geometries. Work is continuing in this direction.

Accuracy issues. Any high-quality method for the incompressible Navier–Stokes equations can be used to solve the discrete equations on the rectangular grid. For example, the method of Bell, Colella, and Glaz [1] has been used in [2]. This method is second-order accurate on smooth flows and also deals well with steep gradients in the flow. However, it is not at all clear how the error behaves when such a method is applied in the context of the immersed boundary method. Standard error estimates assume the inhomogeneous term is smooth and fixed as $h \to 0$. Here the inhomogeneous term is obtained by discretizing delta functions and becomes singular as $h \to 0$.

Doing a full error analysis in two or three space dimensions would be valuable. In this paper we take an initial step in this direction by introducing a one-dimensional model problem and doing some careful error analysis. Already from this model problem it is possible to gain some insight into a variety of issues that arise. In particular, we compare several choices of discrete delta functions for use in spreading forces to the uniform grid. We will also see that it may be advantageous to use quite different interpolation procedures to compute the boundary velocity in situations where the velocity is not smooth near the boundary.

3. One-dimensional models. In one space dimension the equations of incompressible flow are not very interesting, and so it is not simple to write a direct one-dimensional analogue of the immersed boundary method that captures all of the features. However, the method described above for (1.1), with f given by (1.2), coupled with the evolution equation (1.5) for the immersed boundary point $\alpha(t)$, has much the same flavor as the immersed boundary method.

We will start with a simple version of this, the one-dimensional heat equation with a singular source,

$$u_t = u_{xx} + c(t)\delta(x-\alpha), \qquad 0 \le x \le 1, \tag{3.1}$$

with Dirichlet boundary conditions $u(0) = u(1) = 0$. Here α is fixed and the strength $c(t)$ is a given function of t.

At first glance the heat equation may seem far removed from the Navier–Stokes equations of primary interest, but in fact (3.1) can be viewed as a special case of the immersed boundary method described in the previous section. Consider a viscous incompressible fluid in the channel $0 \le x \le 1$, $-\infty < y < \infty$ (or, alternatively, in $0 \le y \le 1$ with periodic boundary conditions in y). Suppose there is a vertical plate running through the length of this channel at the position $x = \alpha$, and that this plate moves vertically, exerting some specified force $f(t)$ on the fluid. This will accelerate the fluid in the vertical direction due to viscous stress at the plate. We expect no motion in the x-direction and so all terms in the Navier–Stokes equations involving the horizontal velocity drop out. There should also be no variation in the y-direction, and so all terms involving y-derivatives drop out. In particular, all the nonlinear convective terms drop out and the Navier–Stokes equations (2.2) reduce to a single equation for the vertical velocity v, which takes the form

$$v_t = \mu v_{xx} + f(t)\delta(x-\alpha), \qquad 0 \le x \le 1. \tag{3.2}$$

At the edges of the channel the fluid velocity should be zero, so we have the boundary conditions $v(0,t) = v(1,t) = 0$. This is of the form (3.1). Hence an analysis of the one-dimensional immersed boundary method on (3.1) gives us direct information about the behavior of the two-dimensional immersed boundary method in one special case.

A more realistic situation would be to specify the velocity $\bar{v}(t)$ of the plate rather than the force $f(t)$. We could solve this by decoupling the problem into two separate channels, giving the heat equation $v_t = \mu v_{xx}$ on $0 \le x \le \alpha$ with $v(0,t) = 0$, $v(\alpha,t) = \bar{v}(t)$ and the heat equation on $\alpha \le x \le 1$ with $v(\alpha,t) = \bar{v}(t)$, $v(1,t) = 0$. This would avoid the need for a singular forcing term. However, in the spirit of the immersed boundary method we would instead again solve (3.2) with $f(t)$ now determined implicitly by the requirement that the resulting velocity $v(x,t)$ satisfy

$$v(\alpha,t) = \bar{v}(t). \tag{3.3}$$

This leads to the constraint (1.4) in the one-dimensional problem. This serves as a model for the immersed boundary method in contexts where we wish to impose a solid wall boundary condition, either stationary or moving at some prescribed velocity, and will be studied later in this paper.

We return to the simplest case (3.1) with $c(t)$ specified, and consider the Crank–Nicolson method (1.9). The error in the numerical solution depends on our choice of discrete delta function $d_h(x)$. Before analyzing the error we first make some comments about these functions.

4. Interpolation properties of discrete delta functions. The discrete delta function $d_h(x)$ is used to spread the delta function source terms to the uniform grid. At a later stage we will also use discrete delta functions to interpolate grid values $u(x_j, t)$ to the point α in order to approximate $u(\alpha, t)$. In this section we recall a few fundamental facts about interpolation rules in one space dimension. If $f(x)$ is any function of one variable, we can approximate

$$f(\alpha) \approx h \sum_j f(x_j) d_h(x_j - \alpha). \tag{4.1}$$

This is a discrete form of the statement $f(\alpha) = \int_{-\infty}^{\infty} f(x)\delta(x-\alpha)\,dx$. Any discrete delta function defines an interpolation formula via (4.1), and conversely, any translation invariant interpolation rule can be used to define a discrete delta function.

If $f(x)$ is a smooth function of x, then we can expand $f(x_j)$ in a Taylor series about the point α to obtain

$$f(x_j) = f(\alpha) + \sum_{m=1}^{\infty} \frac{1}{m!} f^{(m)}(\alpha)(x_j - \alpha)^m$$

where $f^{(m)}$ represents the mth derivative of f. Inserting this expansion in (4.1) gives an expression for the error,

$$\begin{aligned} f(\alpha) - h \sum_j f(x_j) d_h(x_j - \alpha) &= f(\alpha) - h f(\alpha) \sum_j d_h(x_j - \alpha) \\ &\quad - \sum_{m=1}^{\infty} \frac{1}{m!} f^{(m)}(\alpha) h \sum_j (x_j - \alpha)^m d_h(x_j - \alpha). \end{aligned} \tag{4.2}$$

The first two terms cancel due to the consistency condition (1.11). We will assume in general that d_h has support only over a few mesh widths, say

$$d_h(x) = 0 \quad \text{for } |x| > Mh. \tag{4.3}$$

This, combined with (1.11), shows that

$$h \sum_j (x_j - \alpha)^m d_h(x_j - \alpha) = O(h^m) \quad \text{as } h \to 0.$$

Condition (1.11) then guarantees that the interpolation error (4) will be at most $O(h)$ as $h \to 0$. The error will be smaller if additional terms in the error expansion (4) are identically zero. We see that the error is $O(h^p)$ if (1.11) is satisfied and the next $p-1$ discrete moments of d_h are zero. By using a Taylor series with remainder in the above expansion we see that we only require continuity of p derivatives of f for this result to hold. We summarize this in the next lemma, where δ_{m0} is the Kronecker delta.

LEMMA 4.1. *Suppose d_h satisfies* (4.3) *and also the discrete moment condition*

$$h \sum_j (x_j - \alpha)^m d_h(x_j - \alpha) = \delta_{m0} \tag{4.4}$$

for $m = 0, 1, \cdots, p-1$. If $f \in C^{(p-1)}([\alpha - Mh, \alpha + Mh])$ and $f^{(p-1)}$ is Lipschitz continuous on this interval, then

$$f(\alpha) - h \sum_j f(x_j) d_h(x_j - \alpha) = O(h^p) \quad \text{as } h \to 0.$$

We can easily verify that the hat functions $d_h^{(1)}$ and $d_h^{(2)}$ from (1.6) and (1.7) satisfy (4.4) for $m = 0, 1$ while $d_h^{(3)}$ from (1.8) only satisfies (4.4) for $m = 0$. The hat functions give second-order accurate interpolation while the cosine function only gives first-order interpolation.

The above result assumes f is smooth. In our applications $u(x,t)$ will typically have a kink at the point α where one or more derivatives are discontinuous. This arises from the delta function singularity in the differential equation for u. If we wish to interpolate to the point α we may obtain less accuracy than the above result indicates. For example, the hat function $d_h^{(1)}$ corresponds to linear interpolation between the two neighboring grid points. If f is smooth then this gives $O(h^2)$ accuracy, but if f has a kink at α then we only obtain $O(h)$ accuracy.

Let $[f^{(m)}]_\alpha = f^{(m)}(\alpha+) - f^{(m)}(\alpha-)$ be the jump in the mth derivative of f at the point α. We can expand $f(x_j)$ about $f(\alpha)$ as before if we are careful to use derivatives on the correct side:

$$f(x_j) = \begin{cases} f(\alpha) + \sum_{m=1}^{\infty} \frac{1}{m!} f^{(m)}(\alpha-)(x_j - \alpha)^m & \text{for } x_j < \alpha, \\ f(\alpha) + \sum_{m=1}^{\infty} \frac{1}{m!} f^{(m)}(\alpha+)(x_j - \alpha)^m & \text{for } x_j > \alpha. \end{cases}$$

These can be combined to give

$$f(x_j) = f(\alpha) + \sum_{m=1}^{\infty} \frac{1}{m!} f^{(m)}(\alpha-)(x_j - \alpha)^m + \sum_{m=1}^{\infty} \frac{1}{m!} H(x_j - \alpha)[f^{(m)}]_\alpha (x_j - \alpha)^m$$

where H is the Heaviside function. Using this expansion in the expression for the interpolation error now gives

$$f(\alpha) - h\sum_j f(x_j) d_h(x_j - \alpha) = -\sum_{m=1}^{\infty} \frac{1}{m!} \left(f^{(m)}(\alpha) + H(x_j - \alpha)[f^{(m)}]_\alpha \right) \times h \sum_j (x_j - \alpha)^m d_h(x_j - \alpha). \tag{4.5}$$

This gives a modified version of Lemma 4.1 for functions f that are smooth except at the point α.

LEMMA 4.2. *Suppose* (4.4) *is satisfied for* $m = 0,\ 1,\ \cdots,\ p-1$ *and in addition the one-sided discrete moment condition*

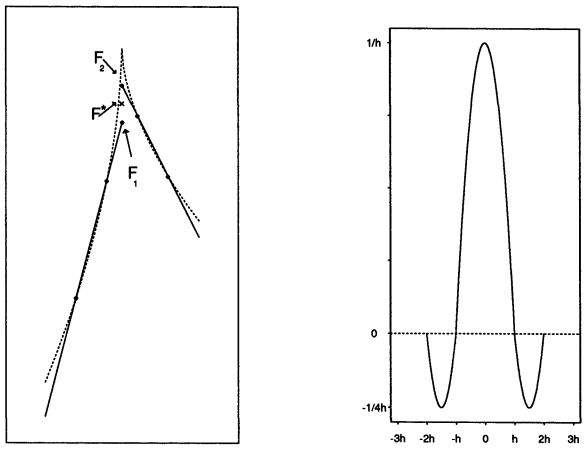
$$h\sum_j H(x_j - \alpha)(x_j - \alpha)^m d_h(x_j - \alpha) = 0 \tag{4.6}$$

is satisfied for $m = 1,\ 2, \cdots,\ p-1$. *Let* f *be a continuous function with* $f \in C^{p-1}([\alpha - Mh, \alpha) \cap (\alpha, \alpha + Mh])$, *with* $f^{(p-1)}$ *Lipschitz continuous on each half interval. Then*

$$f(\alpha) - h\sum_j f(x_j) d_h(x_j - \alpha) = O(h^p) \quad \text{as } h \to 0.$$

The hat functions $d_h^{(1)}$ and $d_h^{(2)}$ fail to satisfy (4.6) for $m = 1$ and hence are only first-order accurate if $[f']_\alpha \neq 0$.

We can obtain a second-order accurate interpolation formula based on the grid values by using linear extrapolation from each side of α. Figure 2(a) illustrates the



(a) (b)

FIG. 2. (a) *Linear extrapolation to approximate $f(\alpha)$ based on grid values in the case where f has a kink at α. A convex combination of the two one-sided extrapolation values is used.* (b) *The discrete delta function $d_h^{(4)}(x)$ obtained by this procedure.*

idea. If $x_J < \alpha < x_{J+1}$ then we extrapolate linearly from x_{J-1} and x_J to α obtaining a value F_1 and also from x_{J+1} and x_{J+2} to α obtaining a value F_2. We then approximate $f(\alpha)$ by a weighted average of F_1 and F_2,

$$f(\alpha) \approx ((x_{J+1} - \alpha)F_1 + (\alpha - x_J)F_2)/h.$$

Since F_1 and F_2 are themselves linear combinations of the grid values $f(x_j)$, this gives a rule of the form (4.1). Working out the details shows that this discrete delta function, which we will call $d_h^{(4)}$, is given by

$$d_h^{(4)}(x) = \frac{1}{h}\begin{cases} 1 - (x/h)^2, & |x| \le h, \\ 2 - |3x/h| + (x/h)^2, & h \le |x| \le 2h, \\ 0 & \text{otherwise}, \end{cases} \tag{4.7}$$

as illustrated in Fig. 2(b). This delta function satisfies (4.4) for $m = 0$, 1 and also (4.6) for $m = 1$, so that it gives a second-order accurate approximation for any f that is smooth on each side of α.

5. Overview of results. Before presenting all of our analysis and numerical results, we will give a brief overview of our findings. Some of these results are rigorously proved in later sections. In other cases we do not have a complete proof but do have a good understanding of the behavior that is seen numerically. We have presented this incomplete analysis in later sections as well since we feel that it is essentially correct and provides valuable insight into the behavior and some of the subtle cancellation properties that come into play. These results indicate that good accuracy can be obtained with immersed boundary methods but that a certain amount of care must be exercised in designing these methods in order to achieve this accuracy.

5.1. The steady state case. The simplest case is the steady state equation

$$\hat{u}''(x) = -c\delta(x-\alpha), \qquad \hat{u}(0) = \hat{u}(1) = 0$$

with the piecewise linear solution

$$\hat{u}(x) = \begin{cases} cx(1-\alpha), & x \le \alpha, \\ c\alpha(1-x), & x \ge \alpha. \end{cases} \tag{5.1}$$

Applying the Crank–Nicolson method (1.9) and setting $\hat{U}_j^n = \hat{U}_j^{n+1}$, we find that the numerical steady state $\hat{U}_j$ will satisfy

$$D_x^2 \hat{U}_j = -cd_h(x_j - \alpha), \qquad \hat{U}_0 = \hat{U}_N = 0. \tag{5.2}$$

The numerical solution depends on the choice of the discrete delta function $d_h(x)$. For any reasonable choice we will have first-order accuracy at all grid points. If the moment condition (4.4) is satisfied with $m = 1$ then we obtain considerably more for this simple steady state problem: the numerical solution will give the *exact* solution, except perhaps at a few points near α. We have the following result.

LEMMA 5.1. *Suppose $\hat{U}_j$ satisfies the difference equations* (5.2). *Let $\hat{u}(x)$ be the true solution to* (5.1). *If the discrete delta function $d_h(x)$ satisfies* (1.11) (*i.e.,* (4.4) *with $m = 0$*) *then*

$$\hat{U}_j = \hat{u}(x_j) + O(h) \quad \textit{for all } x_j.$$

If, in addition, (4.4) *is satisfied with $m = 1$ and the support of d_h has radius M, as in* (4.3), *then*

$$\hat{U}_j = \hat{u}(x_j) \quad \textit{for } |x_j - \alpha| \ge (M-1)h,$$

i.e., the numerical steady state solution agrees with the true solution away from the point α.

A proof of this based on discrete Green's functions is given in §6.1.

In particular, note that in the case $M = 1$ we obtain $\hat{U}_j = \hat{u}(x_j)$ for all j, for example, when $d_h^{(1)}$ from (1.6) is used. When $d_h^{(2)}$ from (1.7) is used, $M = 2$ and the numerical solution differs from the true solution at only two points. At the points where the solution is in error, the error is $O(h)$ as $h \to 0$, and hence the method is first-order accurate in the max-norm,

$$\|\hat{U}_j - \hat{u}(x_j)\|_\infty = \max_j |\hat{U}_j - \hat{u}(x_j)| = O(h) \quad \text{as } h \to 0.$$

On the other hand, since the solution is exact except at two points, we would have second-order accuracy in the 1-norm,

$$\|\hat{U}_j - \hat{u}(x_j)\|_1 = h\sum_j |\hat{U}_j - \hat{u}(x_j)| = O(h^2) \quad \text{as } h \to 0.$$

The cosine delta function (1.8) does not satisfy condition (4.4) with $m = 1$, and the numerical solution is in error at all points. However, the lemma shows that the solution is still $O(h)$ accurate at each point, and so the error is now $O(h)$ in either the max-norm or the 1-norm.

5.2. $c(t)$ specified and α fixed. For the time-dependent problem

$$u_t = u_{xx} + c(t)\delta(x-\alpha) \tag{5.3}$$

we study the Crank–Nicolson method (1.9) and show that the accuracy again depends on the choice of discrete delta function d_h. We cannot expect to obtain the exact solution as in the steady state case, but we can achieve second-order accuracy by requiring the moment condition for d_h. We assume that $c(t)$ is smooth and that the initial data is consistent with the equation, as discussed further in §6.2. We then have the following result.

THEOREM 5.2. *Suppose U_j^n satisfies the difference equations* (1.9). *Let $u(x,t)$ be the true solution to* (3.1). *If the discrete delta function $d_h(x)$ satisfies* (1.11) *then*

$$U_j^n = u(x_j, t_n) + O(h) \quad \text{for all } x_j.$$

If, in addition, (4.4) *is satisfied with $m = 1$ and the support of d_h has radius M, as in* (4.3), *then*

$$U_j^n = u(x_j, t_n) + O(h^2) \quad \text{for } |x_j - \alpha| \geq (M-1)h.$$

This is proved in §6.2. Note in particular that for the hat function $d_h^{(1)}$ the moment condition is satisfied and $M = 1$, so that we obtain second-order accuracy at all grid points. When $d_h^{(2)}$ is used we expect second-order accuracy except at the grid points adjacent to the point α.

As an illustration, we present some numerical results on one sample problem with a variety of choices for d_h. This is a case where the exact solution is known, although similar results have been observed on other problems where we specify $c(t)$ at will and compare the computed solution to a fine grid solution.

The exact solution we use is

$$u(x,t) = \begin{cases} \sin(3\pi x)\exp\left(-9\pi^2 t\right), & 0 \leq x \leq \frac{1}{3}, \\ \sin(\frac{3}{2}\pi(1-x))\exp(-\frac{9}{4}\pi^2 t), & \frac{1}{3} \leq x \leq 1, \end{cases} \tag{5.4}$$

with $\alpha = \frac{1}{3}$. Note that $u(\alpha, t) = 0$ for all t and that

$$c(t) = -[u_x]_\alpha = -\frac{3}{2}\pi \exp\left(-\frac{9}{4}\pi^2 t\right) - 3\pi \exp(-9\pi^2 t).$$

We tested grids with $n = 10,\ 20,\ 40, \cdots,\ 320$ grid points and looked at the error in the infinity norm $\|E\|_\infty$, and also at the error if we ignore the two grid points closest to the point α, which we denote by $\|\tilde{E}\|_\infty$.

We choose $\alpha = \frac{1}{3}$ so that this point always lies one third of the way between grid points on each grid. This leads to numerical results that clearly show the expected asymptotic behavior. Other choices of α give results that are harder to interpret since the truncation error depends on the location of α relative to the grid.

Table 1 shows the results obtained with three choices of d_h. We display the error norm and also the ratio between the errors for successive values of n. With a first-order method this ratio should be 2 asymptotically, while for a second-order method the ratio should be 4. We observe exactly the behavior we expect. The delta function $d_h^{(1)}$ gives second-order accuracy while $d_h^{(2)}$ gives second-order accuracy away from α but only first-order accuracy near α. The use of $d_h^{(3)}$ gives first-order accuracy at all points.

TABLE 1

d_h	n	$\|E\|_\infty$	ratio	$\|\tilde{E}\|_\infty$	ratio
$d_h^{(1)}$	10	0.219×10^{-1}		0.219×10^{-1}	
	20	0.637×10^{-2}	3.4	0.637×10^{-2}	3.4
	40	0.155×10^{-2}	4.1	0.155×10^{-2}	4.1
	80	0.393×10^{-3}	3.9	0.393×10^{-3}	3.9
	160	0.978×10^{-4}	4.0	0.978×10^{-4}	4.0
	320	0.245×10^{-4}	4.0	0.245×10^{-4}	4.0
$d_h^{(2)}$	10	0.928×10^{-1}		0.163×10^{-1}	
	20	0.553×10^{-1}	1.7	0.111×10^{-1}	1.5
	40	0.302×10^{-1}	1.8	0.285×10^{-2}	3.9
	80	0.155×10^{-1}	1.9	0.725×10^{-3}	3.9
	160	0.787×10^{-2}	2.0	0.180×10^{-3}	4.0
	320	0.396×10^{-2}	2.0	0.452×10^{-4}	4.0
$d_h^{(3)}$	10	0.697×10^{-1}		0.795×10^{-2}	
	20	0.393×10^{-1}	1.8	0.887×10^{-2}	0.9
	40	0.213×10^{-1}	1.8	0.254×10^{-2}	3.5
	80	0.109×10^{-1}	2.0	0.111×10^{-2}	2.3
	160	0.552×10^{-2}	2.0	0.452×10^{-3}	2.4
	320	0.278×10^{-2}	2.0	0.211×10^{-3}	2.1

5.3. $u(\alpha,t)$ specified and α fixed. If we specify the value $u(\alpha,t)$ rather than $c(t)$ then we must supplement (1.9) by an equation for $c_{n+1/2} \approx c(t_{n+1/2})$. The equation we use is a discrete form of (1.4) using some discrete delta function $\tilde{d}_h$ to interpolate the grid values U_j^{n+1} to the point α,

$$h\sum_j U_j^{n+1}\tilde{d}_h(x_j-\alpha) = \bar{u}(t_{n+1}). \tag{5.5}$$

This can be combined with (1.9) to uniquely determine $c_{n+1/2}$ and the new solution vector U^{n+1}, as described further in §6.5. The discrete delta function $\tilde{d}_h$ need not be the same as d_h in (1.9). The best choice for d_h is to use $d_h^{(1)}$, since Theorem 5.2 shows that this is the only choice that gives second-order accuracy at all grid points in the case where $c(t)$ is known. On the other hand, we use $\tilde{d}_h$ to interpolate the function $u(t_{n+1})$ to the point α, where u has a kink, and so Lemma 4.2 indicates that we should use a discrete delta function satisfying the one-sided moment condition in order to obtain second-order accuracy in this step. For example, $d_h^{(4)}$ might be used for $\tilde{d}_h$.

Numerical results indicate that with this choice ($d_h = d_h^{(1)}$ and $\tilde{d}_h = d_h^{(4)}$) we obtain overall second-order accuracy in both the solution U^n and in the values $c_{n+1/2}$. However, with any other choice of d_h this second-order accuracy appears to be lost. Even a choice such as $d_h^{(2)}$, which in the case where $c(t)$ is specified would cause a loss of accuracy only at two grid points neighboring α, will cause a global degradation to first-order accuracy in the present case. The reason is that a first-order error near α

causes an error in the value interpolated to α and hence an error in the value $c_{n+1/2}$. This in turn causes a change in the overall solution.

The fact that the combination $d_h^{(1)}$, $d_h^{(4)}$ gives second-order accuracy has not been fully proved. However, in §6.5 we provide some analysis of this situation that gives strong support for this belief. Choices of $\tilde{d}_h$ other than $d_h^{(4)}$ should also be allowed provided the one-sided moment condition is satisfied.

As a numerical example, we use the same test case as before, (5.4), but now specify $u(1/3,t) = 0$ for all t instead of specifying $c(t)$. Table 2 shows the results obtained with various choices of d_h and $\tilde{d}_h$. Here we also show the error in the computed $c_{n+1/2}$. The combination $d_h^{(1)}$, $d_h^{(4)}$ gives second-order accuracy for each of the errors while with any other combination the error deteriorates to first order. Note also that if we do not use $\tilde{d}_h = d_h^{(4)}$ then the approximation $c_{n+1/2}$ does not converge to the true value $c(t_{n+1/2})$. We have an $O(1)$ error in $c_{n+1/2}$ even though U^n still converges with first order accuracy. This nonconvergence of $c_{n+1/2}$ is also discussed in §6.5.

5.4. $c(t)$ and $\alpha(t)$ specified. When the point $\alpha(t)$ is not constant, a natural generalization of the Crank–Nicolson method takes the form

(5.6)
$$\left(1-\frac{1}{2}kD_x^2\right)U_j^{n+1} = \left(1+\frac{1}{2}kD_x^2\right)U_j^n + \frac{k}{2}[c(t_n)d_h^{(1)}(x_j-\alpha(t_n)) + c(t_{n+1})d_h^{(1)}(x_j-\alpha(t_{n+1}))].$$

In many computations this method appears to give second-order accurate results. However, this is the result of fortuitous cancellation. A careful analysis shows that some additional terms are needed to insure second-order accuracy in general. With these correction terms, the method takes the form

(5.7)
$$\begin{aligned}\left(1-\frac{1}{2}kD_x^2\right)U_j^{n+1} &= \left(1+\frac{1}{2}kD_x^2\right)U_j^n \\ &\quad + \frac{k}{2}[c(t_n)d_h^{(1)}(x_j-\alpha(t_n)) + c(t_{n+1})d_h^{(1)}(x_j-\alpha(t_{n+1}))] \\ &\quad + \frac{k}{2}(Y_j^n + Y_j^{n+1}) + W_j^n.\end{aligned}$$

The correction terms are given by

(5.8)
$$Y_j^n = \frac{h^2}{2}\alpha'(t_n)c(t_n)\operatorname{sgn}(x_j-\alpha(t_n))\left(d_h^{(1)}(x_j-\alpha(t_n))\right)^2$$

and

(5.9)
$$W_j^n = -(t_{n+1/2}-\tau_j^n)|\alpha'(\tau_j^n)|c(\tau_j^n).$$

Here τ_j^n represents the time at which the immersed point $\alpha(t)$ crosses the grid line x_j, if this happens during the nth timestep. Otherwise we set $\tau_j^n = t_{n+1/2}$ so that $W_j^n = 0$. So

$$\tau_j^n = \begin{cases} \text{time } \tau \text{ at which } \alpha(\tau) = x_j & \text{if } t_n \le \tau \le t_{n+1}, \\ t_{n+1/2} & \text{otherwise.} \end{cases}$$

TABLE 2

d_h	$\tilde{d}_h$	n	$\|E\|_\infty$	ratio	$\|\tilde{E}\|_\infty$	ratio	$\|c(t_{n+1/2}) - c_{n+1/2}\|_\infty$	ratio
		10	0.504×10^{-1}		0.109×10^{-1}		0.244×10^{0}	
		20	0.125×10^{-1}	4.0	0.649×10^{-2}	1.7	0.736×10^{-1}	3.3
$d_h^{(1)}$	$d_h^{(4)}$	40	0.206×10^{-2}	6.0	0.144×10^{-2}	4.5	0.207×10^{-1}	3.6
		80	0.504×10^{-3}	4.1	0.439×10^{-3}	3.3	0.490×10^{-2}	4.2
		160	0.108×10^{-3}	4.7	0.996×10^{-4}	4.4	0.124×10^{-2}	4.0
		320	0.268×10^{-4}	4.0	0.258×10^{-4}	3.9	0.304×10^{-3}	4.1
		10	0.921×10^{-1}		0.150×10^{-1}		$0.307\times10^{+1}$	
		20	0.354×10^{-1}	2.6	0.240×10^{-1}	0.6	$0.274\times10^{+1}$	1.1
$d_h^{(2)}$	$d_h^{(4)}$	40	0.157×10^{-1}	2.3	0.125×10^{-1}	1.9	$0.262\times10^{+1}$	1.0
		80	0.742×10^{-2}	2.1	0.710×10^{-2}	1.8	$0.258\times10^{+1}$	1.0
		160	0.362×10^{-2}	2.0	0.350×10^{-2}	2.0	$0.256\times10^{+1}$	1.0
		320	0.179×10^{-2}	2.0	0.178×10^{-2}	2.0	$0.256\times10^{+1}$	1.0
		10	0.165×10^{0}		0.784×10^{-1}		0.199×10^{0}	
		20	0.842×10^{-1}	2.0	0.598×10^{-1}	1.3	0.151×10^{0}	1.3
$d_h^{(1)}$	$d_h^{(1)}$	40	0.422×10^{-1}	2.0	0.391×10^{-1}	1.5	0.125×10^{0}	1.2
		80	0.212×10^{-1}	2.0	0.202×10^{-1}	1.9	0.741×10^{-1}	1.7
		160	0.106×10^{-1}	2.0	0.105×10^{-1}	1.9	0.399×10^{-1}	1.9
		320	0.531×10^{-2}	2.0	0.526×10^{-2}	2.0	0.206×10^{-1}	1.9
		10	0.251×10^{0}		0.987×10^{-1}		0.406×10^{0}	
		20	0.138×10^{0}	1.8	0.111×10^{0}	0.9	$0.157\times10^{+1}$	0.3
$d_h^{(3)}$	$d_h^{(3)}$	40	0.706×10^{-1}	2.0	0.593×10^{-1}	1.9	$0.160\times10^{+1}$	1.0
		80	0.363×10^{-1}	1.9	0.351×10^{-1}	1.7	$0.157\times10^{+1}$	1.0
		160	0.180×10^{-1}	2.0	0.174×10^{-1}	2.0	$0.151\times10^{+1}$	1.0
		320	0.914×10^{-2}	2.0	0.908×10^{-2}	1.9	$0.148\times10^{+1}$	1.0

A natural timestep restriction requires that α cross no more than one grid line per timestep (i.e., $|k\alpha'(t)/h| < 1$) and so W_j^n is nonzero for at most one value of j. Also note that Y_j^n is nonzero for at most two values of j.

These correction terms are derived in §6.6. Here we show two numerical examples. In the first case we let $c(t) = 1$ and $\alpha(t) = \alpha(0) + t$, with $\alpha(0) = \frac{1}{3}$. This particular choice of $c(t)$ and $\alpha(t)$ maximizes the magnitude of the correction terms and allows us to examine the effect of including either Y or W. Figure 3 shows the resulting errors for various values of h on a log-log plot. With no correction terms we appear to already have second-order accuracy, although including the correction terms gives an order of magnitude decrease in the error constant. It is very interesting to observe, however, what happens if we introduce only one of the corrections Y or W but not the other. In either case the error becomes $O(h)$ rather than $O(h^2)$. This shows that the two sources of error corrected for by Y and W are in fact $O(h)$ errors, but that these errors happen to cancel out so that the Crank–Nicolson method (5.4) without

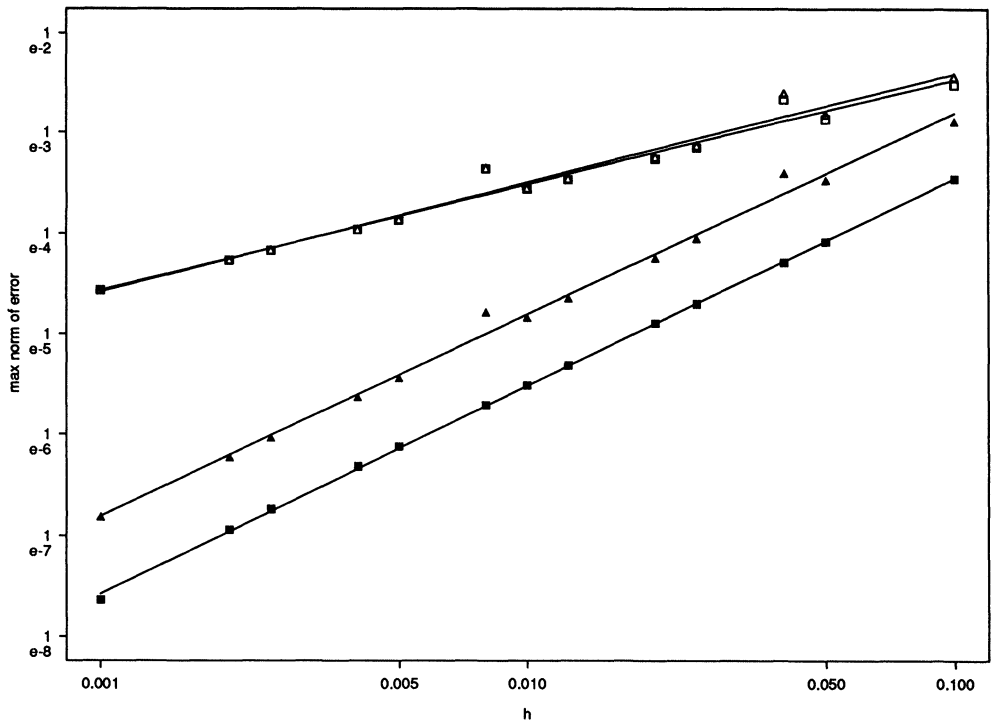


FIG. 3. $||E||_\infty$ *for various values of* h. *Solid triangles: no correction terms. Open boxes: Y correction terms. Open triangles: W correction terms. Solid boxes: both correction terms.*

correction terms is also second order. For this particular problem this cancellation can be analyzed since $\alpha'(t) = 1$ and the discontinuity hits each grid cell in precisely the same way. More generally there is not such complete cancellation, although some cancellation still occurs. The method (5.4) appears to be better than first order but not fully second-order accurate.

As another example, we compute an approximation to the true solution

$$u(x,t) = \begin{cases} \sin(\omega_1 x)\exp(-\omega_1^2 t), & x \le \alpha(t), \\ \sin(\omega_2(1-x))\exp(-\omega_2^2 t), & x \ge \alpha(t) \end{cases} \tag{5.10}$$

for some choice of ω_1 and ω_2. The point $\alpha(t)$ is found by solving the scalar equation

$$\sin(\omega_1\alpha)\exp(-\omega_1^2 t) = \sin(\omega_2(1-\alpha))\exp(-\omega_2^2 t)$$

for α. This equation has a unique solution if we take, for example, $\pi < \omega_1 < 2\pi$ and also $\pi < \omega_2 < 2\pi$. We compute $\alpha(t)$ to high precision using a zero-finding routine and also specify $c(t) = -[u_x]_\alpha$, which is easily computed from the exact solution. Figure 4 shows the true solution and the approximate solution with $\omega_1 = 5\pi/4$ and $\omega_2 = 7\pi/4$ for $n = 25$ at $t = 0.1$. From this figure we can see that even with a grid as coarse as $n = 25$, we get excellent resolution of the solution even near the point α. Figure 5 shows the error as a function of h together with least squares line fits to estimate the order of accuracy. On this example, the method (5.4) appears to have order 1.7 while the full method (5.7) has order 2.0.

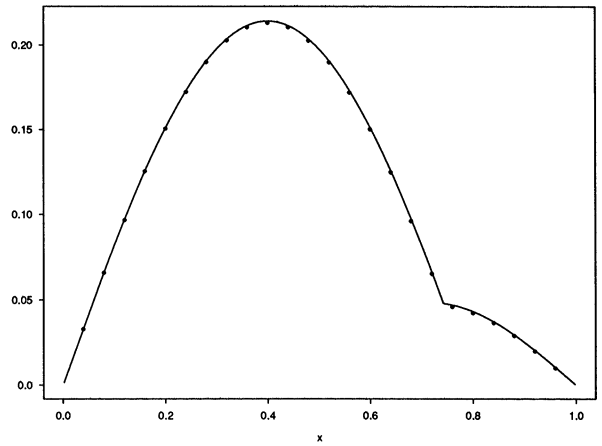


FIG. 4. *True solution and approximate solution for* $n = 25$ *at* $t = 0.1$. *Since the timestep equalled the space step, the approximate solution is for step* 25.

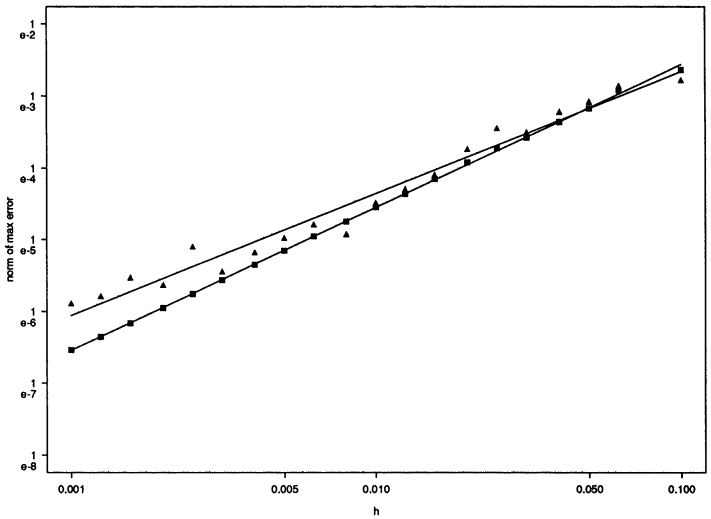


FIG. 5. $||E||_\infty$ *for the case* $c(t)$ *given and* $\alpha(t)$ *given for various values of* h. *Triangles: without correction terms. Boxes: with correction terms.*

5.5. $u(\alpha(t), t)$ **and** $\alpha(t)$ **specified.** When $u(\alpha(t), t)$ is specified rather than $c(t)$, then $c(t)$ has to be calculated implicitly. In this case we replace $c(t_n)$ and $c(t_{n+1})$ in (5.7) by approximations c_n and c_{n+1}, respectively. Using (5.5), we can solve for c_{n+1} to get

$$(5.11)\quad c_{n+1} = \left(\frac{k}{2} r_{n+1}^T A^{-1} d_h^{n+1}\right)^{-1} \left(\bar{u}(t_{n+1}) - r_{n+1}^T A^{-1} B U^n - \frac{k}{2} r_{n+1}^T A^{-1} c_n d_h^n\right),$$

where $r_{n+1}^T U^{n+1} = \bar{u}(t_{n+1})$ and d_h is the vector whose components are $d_h^{(1)}(x_j - \alpha(t_n))$. Once we know c_{n+1}, we can calculate U^{n+1} with (5.7).

We compute an approximation to the same true solution as was done in §5.4. As before we only look at two cases, that with no correction terms and that including

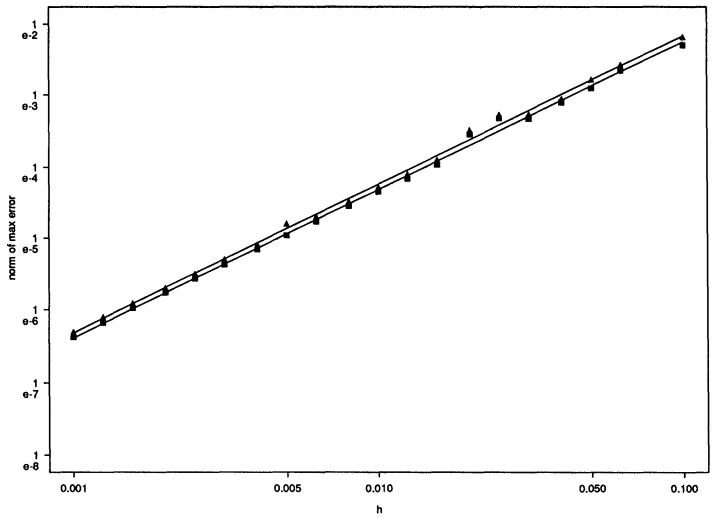


FIG. 6. $||E||_\infty$ *for the case* $u(\alpha(t),t)$ *given and* $\alpha(t)$ *given for various values of* h. *Triangles: without correction terms. Boxes: with correction terms.*

both correction terms. Shown in Fig. 6 are the results for these two cases. As we can see from the figure, including the correction terms does not change the results to any significant degree. In both cases the order is 2.0.

6. Analysis. We begin with an analysis of the steady state case and the case where c and α are both constant. These cases can be analyzed by considering a discrete Green's function corresponding to a delta function source concentrated at a single grid point, for which the accuracy is easily computed. The case where α lies between grid points can then be analyzed by viewing the numerical solution as an interpolation between the grid point Green's functions, using the discrete delta function d_h as the interpolation rule. The accuracy results then follow from the results on interpolation properties of Green's functions from §4.

6.1. The steady state case. We prove Lemma 5.1. Let $G(x;\beta)$ be the Green's function for this problem with a unit strength source at β,

$$G(x;\beta) = \begin{cases} x(1-\beta), & x \le \beta, \\ \beta(1-x), & x \ge \beta. \end{cases}$$

The function G satisfies $G_{xx} = -\delta(x-\beta)$. It is easy to check that G also satisfies

$$D_x^2 G(x_j;x_i) = -\frac{1}{h}\delta_{ij} \tag{6.1}$$

where δ_{ij} is the Kronecker delta. This shows that if the source location is at a grid point ($\beta = x_i$) then we obtain the true solution by solving the difference equation (6.1) with the entire source concentrated at that grid point. Note that this corresponds to the use of $d_h^{(1)}$ in this case, since $d_h^{(1)}(x_j - x_i) = \delta_{ij}/h$.

Now consider the original problem which has the source at α. The right-hand side of (5.2) can be rewritten as

$$-cd_h(x_j - \alpha) = -ch \sum_i d_h(x_i - \alpha)\delta_{ij}/h.$$

By linearity, the solution $\hat{U}_j$ of (5.2) can be expressed as a linear combination of the Green's function at each x_i,

$$\hat{U}_j = ch \sum_i d_h(x_i - \alpha) G(x_j; x_i). \tag{6.2}$$

This can be viewed as an interpolatory approximation to $cG(x_j;\alpha) = \hat{u}(x_j)$ based on grid values $cG(x_j; x_i)$. For x fixed, the function $G(x;\beta)$ is continuous and piecewise linear in β and hence Lemma 4.1 applies to show that

$$\hat{U}_j = \hat{u}(x_j) + O(h) \quad \text{for all } j.$$

If, in addition, (4.4) is satisfied with $m = 1$ then $d_h(x)$ interpolates linear functions exactly. If $|x_j - \alpha| \geq (M-1)h$ then (6.2) is sampling $G(x_j; x_i)$ over a set of grid points x_i where G is linear, and hence (6.2) recovers the exact value. So $\hat{U}_j = \hat{u}(x_j)$ for these values of j. This completes the proof.

6.2. c and α constant. Now consider the time-dependent problem

$$u_t = u_{xx} + c\delta(x - \alpha) \tag{6.3}$$

with c and α constant, with data $u(x,0) = u_0(x)$. Suppose we use the Crank–Nicolson method (1.9). In order to analyze this method with different choices of d_h, we introduce the function

$$v(x,t) = u(x,t) - \hat{u}(x)$$

where $\hat{u}$ is the steady state solution (5.1). Then it is easy to verify that v satisfies the homogeneous heat equation

$$v_t = v_{xx} \tag{6.4}$$

with data

$$v(x,0) = u_0(x) - \hat{u}(x).$$

If we also introduce the grid function $V_j^n = U_j^n - \hat{U}_j$, where the discrete steady state approximation $\hat{U}_j$ satisfies (5.2), then combining (5.2) and (1.9) shows that V_j^n satisfies

$$\frac{1}{k}(V_j^{n+1} - V_j^n) = \frac{1}{2} D_x^2 (V_j^n + V_j^{n+1}).$$

This is simply the Crank–Nicolson method for (6.4). To analyze the error in U_j^n, we express

$$\begin{aligned} U_j^n - u(x_j - t_n) &= (U_j^n - \hat{U}_j) + (\hat{U}_j - \hat{u}(x_j)) + (\hat{u}(x_j) - u(x_j, t_n)) \\ &= (V_j^n - v(x_j, t_n)) + (\hat{U}_j - \hat{u}(x_j)). \end{aligned}$$

The error is composed of two parts. The first part is the error in the Crank–Nicolson method on the homogeneous heat equation (6.4). The second part is the error in the steady state solution.

Crank–Nicolson on the heat equation is second-order accurate provided the solution is smooth, which it will be if the initial data $v_0(x)$ is smooth. Since $v_0(x) = u_0(x) - \hat{u}(x)$, this will be the case provided u_0 is of the form $\hat{u} + v_0$ for some smooth function v_0. Note that this requires that u_0 have an appropriate kink at the point α,

$$[u_0']_\alpha = -c. \tag{6.5}$$

If u_0 is not of this form then v_0 will have a kink at α and we will lose accuracy.

The second part of the error, the error in the steady state solution, was analyzed above. In particular, we saw that if we use the hat function $d_h^{(1)}$ then there is no error at the grid points, $\hat{U}_j = \hat{u}(x_j)$ for all j. In this case the approximation U_j^n will agree with the true solution of the time-dependent equation (6.3) to $O(h^2)$ at the grid points. If the wider hat function $d_h^{(2)}$ is used then there will be an $O(h)$ error introduced at two grid points but otherwise the time-dependent solution will be unchanged. If the cosine delta function $d_h^{(3)}$ is used then an $O(h)$ error will be introduced at all points.

6.3. Truncation error analysis. In the analysis of the time-dependent error when the strength or location of the delta function varies, it will not be possible to decompose this error into homogeneous and steady state parts as we have just done. Before presenting additional analysis we introduce some new notation and discuss the local truncation error and the evolution of the global error.

To simplify some of our discussions below, we introduce matrix-vector notation for the difference schemes. Let U^n be the vector with components U_j^n and $u(t_n)$ the vector with values $u(x_j, t_n)$ from the true solution. Let d be the vector with components $d_h(x_j - \alpha)$. The Crank–Nicolson method (1.9) may then be written as

$$AU^{n+1} = BU^n + kc(t_{n+1/2})d \tag{6.6}$$

where A and B are the tridiagonal matrices representing the operators $I - \frac{1}{2}kD_x^2$ and $I + \frac{1}{2}kD_x^2$, respectively. We define the local truncation error by

$$T^n = Au(t_{n+1}) - Bu(t_n) - kc(t_{n+1/2})d. \tag{6.7}$$

The global error E^n is defined by

$$E^n = U^n - u(t_n),$$

and combining (6.6) and (6.7) shows that E^n evolves according to

$$AE^{n+1} = BE^n - T^n. \tag{6.8}$$

The Crank–Nicolson method is unconditionally stable, and hence the global error will be $O(h^p)$ accurate provided that $E^0 = O(h^p)$ and $T^n = O(h^{p+1})$. This loss of one order of accuracy between T^n and E^n is a reflection of the fact that we have not divided by k in our definition (6.7) of T^n, contrary to some definitions of the local truncation error. In the case $c \equiv 0$ a standard expansion in Taylor series shows that $T^n = O(h^3)$, and hence the method is second-order accurate. (Note that we always assume that k/h is constant as $h \to 0$ so that $O(kh^2) = O(h^3)$, for example.)

It will be important to observe that in some cases it is possible to achieve second-order accuracy even if T^n is not $O(h^3)$. Most notably, this occurs if all components of T^n are $O(h^3)$ except for a few components for which the error is $O(h^2)$ (with the number of such components being bounded as $h \to 0$). To see this, consider the case where such an error is confined to a single point x_i. (The more general case is easily handled by linear superposition.) Then

$$T^n = \hat{T}^n + \tau^n e_i$$

where $\hat{T}^n = O(h^3)$, τ^n is an $O(h^2)$ scalar, and e_i is the ith unit vector. We can rewrite $\tau^n e_i$ as

$$\tau^n e_i = k\left(\tau^n \frac{h}{k}\right)\left(\frac{1}{h} e_i\right).$$

Note that e_i/h has components δ_{ij}/h and is a discrete delta function corresponding to a point source concentrated at x_i. The error evolution equation (6.8) then takes the form

$$AE^{n+1} = BE^n - \hat{T}^n - k\left(\tau^n \frac{h}{k}\right)\left(\frac{1}{h} e_i\right). \tag{6.9}$$

The last term can be interpreted as a delta function source term with magnitude $\tau^n h/k = O(h^2)$ being introduced into the Crank–Nicolson method (compare (6.9) with (6.6)). Just as the introduction of a delta function with unit strength in our original problem gives an $O(1)$ solution, the introduction of a delta function truncation error with $O(h^2)$ strength gives only an $O(h^2)$ contribution to the global error, and hence we maintain second-order accuracy.

This observation is important since some of our methods will introduce $O(h^2)$ truncation errors at a few grid points near the point α.

We now return to the analysis of the method (1.9) in the case where c and α are constant. We have already analyzed the accuracy of this method by decomposing the error into the steady state error and the error in the Crank–Nicolson method on the homogeneous heat equation. We now present an alternative analysis based directly on the truncation error that extends more easily to the case where $c(t)$ varies.

Consider the truncation error (6.7) in the case where c is constant. The jth component of this vector is

$$\begin{aligned} T_j^n &= u(x_j, t_{n+1}) - u(x_j, t_n) - \frac{k}{2}[D_x^2 u(x_j, t_n) + D_x^2 u(x_j, t_{n+1})] \\ &\quad - kcd_h(x_j - \alpha). \end{aligned} \tag{6.10}$$

If we assume that α lies between grid points, then u is smooth in t at each grid point x_j. Hence a Taylor series expansion shows that

$$u(x_j, t_{n+1}) - u(x_j, t_n) = \frac{k}{2}(u_t(x_j, t_n) + u_t(x_j, t_{n+1})) + O(k^3).$$

Using this in (6.10) and rearranging gives

$$\begin{aligned} T_j^n &= \frac{k}{2}[u_t(x_j, t_n) - D_x^2 u(x_j, t_n) - cd_h(x_j - \alpha)] \\ &\quad + \frac{k}{2}[u_t(x_j, t_{n+1}) - D_x^2 u(x_j, t_{n+1}) - cd_h(x_j - \alpha)]. \end{aligned} \tag{6.11}$$

Since α lies between grid points we have $u_t(x_j,t) = u_{xx}(x_j,t)$ and so the truncation error is clearly $O(h^3)$ provided that

$$u_{xx}(x_j,t_n) = D_x^2 u(x_j,t_n) + cd_h(x_j-\alpha) + O(h^2). \tag{6.12}$$

Away from the point α, u is smooth and so $D_x^2 u(x_j,t_n)$ is a second-order accurate approximation to u_{xx}. We can view the term $cd_h(x_j-\alpha)$ in (6.12) as the modification to $D_x^2 u(x_j,t_n)$ required to approximate u_{xx} to second order near the point α. We claim that the correct modification is obtained by using the hat function $d_h^{(1)}(x)$. With this choice of d_h, the condition (6.12) will be satisfied and the method will be second-order accurate.

LEMMA 6.1. *Suppose $u(x,t)$ satisfies (6.3). Then at any time $t > 0$ we can approximate $u_{xx}(x_j,t)$ to second-order accuracy based only on grid values of u using*

$$u_{xx}(x_j,t) = D_x^2 u(x_j,t) + cd_h^{(1)}(x_j-\alpha) + O(h^2). \tag{6.13}$$

Proof. To see that (6.13) is valid, consider the typical situation as illustrated in Fig. 2(a). The function $u(x,t)$ is smooth except at the point α where one or more derivatives of u have jump discontinuities due to the delta function source. Suppose that we want to approximate $u_{xx}(x_J,t)$ based on the three grid values $u(x_j,t)$, $j = J-1,\ J,\ J+1$. Clearly the standard centered difference approximation $D_x^2 u(x_J,t)$ will not work, e.g., in Fig. 2(a) this would predict a negative second derivative while the true second derivative is positive. To find the correct approximation we expand each of the values $u(x_j,t)$ ($j = J-1,\ J,\ J+1$) about the point $x = \alpha$, being careful to use limiting values of the derivatives as $x \to \alpha$ from the appropriate side. For shorthand we write

$$\partial_x^m u^{\pm} = \lim_{x\to\alpha\pm} \frac{\partial^m}{\partial x^m} u(x,t).$$

Taylor series expansion gives

$$u(x,t) = \begin{cases} u(\alpha,t) + \sum_{m=1}^{\infty} \frac{1}{m!}(x-\alpha)^m \partial_x^m u^- & \text{if } x \le \alpha, \\ u(\alpha,t) + \sum_{m=1}^{\infty} \frac{1}{m!}(x-\alpha)^m \partial_x^m u^+ & \text{if } x \ge \alpha. \end{cases} \tag{6.14}$$

These can be combined into

$$u(x,t) = u(\alpha,t) + \sum_{m=1}^{\infty} \frac{1}{m!}(x-\alpha)^m \partial_x^m u^- + H(x-\alpha) \sum_{m=1}^{\infty} \frac{1}{m!}(x-\alpha)^m [\partial_x^m u]_\alpha, \tag{6.15}$$

which is valid for any x. Now consider the centered second difference at x_J,

$$D_x^2 u(x_J,t) = \frac{1}{h^2}(u(x_{J+1},t) - 2u(x_J,t) + u(x_{J-1},t)).$$

Using (6.3) in each member on the right gives

$$D_x^2 u(x_J,t) = \sum_{m=1}^{\infty} \frac{1}{m!}(D_x^2(x_J-\alpha)^m)\partial_x^m u^- + \frac{1}{h^2}\sum_{m=1}^{\infty} \frac{1}{m!}(x_{J+1}-\alpha)^m [\partial_x^m u]_\alpha, \tag{6.16}$$

in the case $x_J < \alpha < x_{J+1}$. Since $(x-\alpha)^m$ is a smooth function, we have

$$D_x^2(x_J-\alpha)^m = m(m-1)(x_J-\alpha)^{m-2} + O(h^2)$$

and so the first sum on the right-hand side of (6.16) gives, to $O(h^2)$, the Taylor series expansion for $\partial_x^2 u(x_J,t)$:

$$\begin{aligned}(6.17)\quad \sum_{m=1}^{\infty} \frac{1}{m!}(D_x^2(x_J-\alpha)^m)\partial_x^m u^- &= \sum_{m=2}^{\infty} \frac{1}{(m-2)!}(x_J-\alpha)^{m-2}\partial_x^m u^- + O(h^2) \\ &= \partial_x^2 u(x_J,t) + O(h^2).\end{aligned}$$

To evaluate the second sum in (6.16), we need to know the jumps $[\partial_x^m u]_\alpha$ in derivatives of u at the point α. A key observation is that these jumps can be determined directly from the original equation (6.3) a priori, and need not be estimated from the approximate solution.

If the initial data u_0 satisfies (6.5), then the solution $u(x,t)$ will be continuous in both space and time and the singularity $c\delta(x-\alpha)$ in (6.3) must be balanced at each time by a jump in u_x of strength $-c\delta(x-\alpha)$ (since u_t is not singular),

$$[u_x]_\alpha = -c.$$

The time derivative u_t will be continuous, and hence $[u_t]_\alpha = 0$. We also find from (3.1) that

$$\begin{aligned}[u_t]_\alpha &= [u_{xx}]_\alpha + [c\delta(x-\alpha)]_\alpha \\ &= [u_{xx}]_\alpha,\end{aligned}$$

and hence

$$[u_{xx}]_\alpha = 0.$$

To compute $[u_{xxx}]_\alpha$ we differentiate (6.3) with respect to x, giving

$$u_{tx} = u_{xxx} + c\delta'(x-\alpha),$$

which yields

$$[u_{xxx}]_\alpha = [u_{tx}]_\alpha.$$

To compute the latter quantity we differentiate the relation

$$(6.18)\qquad u_x(\alpha+,t) - u_x(\alpha-,t) = -c \quad \forall t$$

with respect to t, which yields

$$[u_{xt}]_\alpha = 0,$$

and hence $[u_{xxx}]_\alpha = 0$. We can show by further differentiation that all higher derivatives are also continuous at α.

To summarize, we have

$$[u_x]_\alpha = -c \quad \text{and} \quad [\partial_x^m u]_\alpha = 0 \quad \text{for } m > 1.$$

Using this, together with (6.17), in (6.16) gives

$$D_x^2 u(x_J,t) = \partial_x^2 u(x_J,t) - \frac{c}{h^2}(x_{J+1}-\alpha) + O(h^2).$$

This is simply (6.13) at the point $j = J$, since $d_h^{(1)}(x_J - \alpha) = (x_{J+1} - \alpha)/h^2$.

By exactly the same procedure we can verify that (6.13) holds at $j = J+1$. For $j \neq J, J+1$, u is smooth for $|x - x_j| \leq h$ and (6.13) reduces to

$$u_{xx}(x_j,t) = D_x^2 u(x_j,t) + O(h^2),$$

which is true for smooth u. The proof is complete.

In deriving (6.13) we used (6.3) to compute the jumps $[\partial_x^m u]_\alpha$. The proof, however, clearly extends to give the following result which will be useful below.

COROLLARY 6.2. *Suppose $f \in C^3((-\infty,\alpha) \cup (\alpha,\infty))$ and f''' is Lipschitz continuous on each half line. Here α is an arbitrary point, say $\alpha \in [x_J, x_{J+1}]$. Then we can approximate $f''(x_j)$ to second-order accuracy based only on values of f at the grid points and the jumps in derivatives at the point α,*

$$f''(x_j) = \tilde{D}_x^2 f(x_j) + O(h^2),$$

where $\tilde{D}_x^2$ is a modification to the centered difference operator D_x^2 defined by

$$\tilde{D}_x^2 f(x_j) = D_x^2 f(x_j) + \begin{cases} -\frac{1}{h^2}\sum_{m=1}^{3} \frac{1}{m!}(x_{J+1}-\alpha)^m [f^{(m)}]_\alpha, & j = J, \\ \frac{1}{h^2}\sum_{m=1}^{3} \frac{1}{m!}(x_J-\alpha)^m [f^{(m)}]_\alpha, & j = J+1, \\ 0 & \text{otherwise.} \end{cases} \tag{6.19}$$

These formulas can be combined to give

(6.20)

$$\tilde{D}_x^2 f(x_j) = D_x^2 f(x_j) - \frac{\operatorname{sgn}(\alpha - x_j)}{h^2} \sum_{m=1}^{3} \frac{1}{m!} \left(h^2 \operatorname{sgn}(\alpha - x_j) d_h^{(1)}(x_j - \alpha) \right)^m [f^{(m)}]_\alpha.$$

Note that we need only keep three terms in the sum since higher-order terms are $O(h^2)$.

6.4. $c(t)$ given and α fixed. For (3.1), in which the strength $c(t)$ is time-dependent, exactly the same approach can be used to obtain a second-order accurate method. The analysis proceeds just as before, except that formulas for the jumps $[\partial_x^m u]_\alpha$ must be rederived. We now have

$$[u_x]_\alpha = -c(t)$$

and $[u_t]_\alpha = 0$ so that

$$[u_{xx}]_\alpha = 0$$

still holds. However, in computing $[u_{xxx}]_\alpha$ we must replace (6.18) by

$$u_x(\alpha+,t) - u_x(\alpha-,t) = -c(t) \quad \forall t,$$

and hence differentiating with respect to t gives $[u_{xt}]_\alpha = c'(t)$. This gives

$$[u_{xxx}]_\alpha = -c'(t).$$

To obtain a second-order accurate approximation to $u_{xx}(x_j, t)$ we must now use, according to Corollary 6.2,

$$\tilde{D}_x^2 u(x_j,t) = D_x^2 u(x_j,t) + c(t)d_h^{(1)}(x_j - \alpha) + \frac{1}{6}c'(t)h^4\left(d_h^{(1)}(x_j-\alpha)\right)^3. \tag{6.21}$$

The final term here is an $O(h)$ correction term since $d_h^{(1)} = O(\frac{1}{h})$ at the points where it is nonzero. However, it turns out that we can ignore this correction term in estimating u_{xx} for the Crank–Nicolson method and still maintain second-order accuracy in the global error, i.e., we claim that the method (1.9) with $d_h^{(1)}$ used for d_h gives a second-order accurate approximation to the true solution.

Using the fact that

$$c(t_{n+1/2}) = \frac{1}{2}(c(t_n) + c(t_{n+1})) + O(k^2)$$

we can write the local truncation error for this method as in (6.11), obtaining

$$\begin{aligned} T_j^n &= \frac{k}{2}[u_{xx}(x_j,t_n) - D_x^2 u(x_j,t_n) - c(t_n)d_h^{(1)}(x_j-\alpha)] \\ &\quad + \frac{k}{2}[u_{xx}(x_j,t_{n+1}) - D_x^2 u(x_j,t_{n+1}) - c(t_{n+1})d_h^{(1)}(x_j-\alpha)]. \end{aligned} \tag{6.22}$$

We now have

$$u_{xx}(x_j,t) - D_x^2 u(x_j,t) - c(t)d_h^{(1)}(x_j-\alpha) = \frac{1}{6}c'(t)h^4\left(d_h^{(1)}(x_j-\alpha)\right)^3 + O(h^2),$$

and hence

$$T_j^n = O(h^3) + \frac{k}{12}(c'(t_n) + c'(t_{n+1}))h^4\left(d_h^{(1)}(x_j-\alpha)\right)^3.$$

The latter term is $O(h^2)$ but is nonzero at only a few points near α. By our comments in §6.3, it follows that the method is second-order accurate (assuming that $c'(t)$ is uniformly bounded).

This shows that the use of $d_h^{(1)}$ gives a second-order accurate method. What happens if we use a different function d_h? We can investigate this by again viewing d_h as an interpolation rule applied to discrete Green's functions. Let $G(x,t;\beta,c(t))$ be the true solution to the problem with source strength $c(t)$ at $x = \beta$,

$$G_t = G_{xx} + c(t)\delta(x-\beta).$$

Although we assumed above that α lies between grid points, it can be verified that the results are uniformly valid as α approaches a grid point, and hence we obtain a second-order accurate approximation to $G(x,t;x_i,c(t))$ by using the numerical method

$$AG^{n+1} = BG^n + kc(t_{n+1/2})\left(\frac{1}{h}e_i\right)$$

where e_i is again the ith unit vector with components $\delta_{ij} = hd_h^{(1)}(x_i - x_j)$. Denote this grid function by $G_j^n(x_i, c(t))$.

Now suppose we solve the original problem using (1.9) with any discrete delta function d_h. Just as in our analysis of the steady state equation, we can view the nonhomogeneous term as a linear combination of sources located at the grid points,

$$c(t_{n+1/2})d_h(x_j - \alpha) = h\sum_i c(t_{n+1/2})d_h(x_i - \alpha)\delta_{ij}/h.$$

By linearity, the solution U_j^n can be written as a linear combination of the functions $G_j^n(x_i, c(t))$,

$$U_j^n = h\sum_i G_j^n(x_i, c(t))d_h(x_i - \alpha).$$

Since $G_j^n(x_i, c(t)) = G(x_j, t_n; x_i, c(t)) + O(h^2)$, we obtain

$$U_j^n = h\sum_i G(x_j, t_n;\, x_i, c(t))d_h(x_i - \alpha) + O(h^2).$$

But this is just an approximation to $G(x_j, t_n;\, \alpha, c(t))$ using the interpolation rule defined by d_h. Using the results of Lemma 4.1, we thus obtain Theorem 5.2.

Note that this result assumes that (6.13) holds at each time t_n. Lemma 6.1 guarantees that this holds for $n > 0$ but we must also require consistency of the initial data $u_0(x)$, i.e., we require that $[u_0']_\alpha = c(0)$.

6.5. $u(\alpha, t)$ given and α fixed. So far we have assumed that $c(t)$ is specified a priori. Now suppose that we wish to determine $c(t)$ as part of the solution so that the constraint (1.4) is satisfied. We will use the Crank–Nicolson method

$$AU^{n+1} = BU^n + kc_{n+1/2}d \tag{6.23}$$

augmented by the constraint (5.5), which we rewrite in vector notation as

$$r^T U^{n+1} = \bar{u}(t_{n+1}). \tag{6.24}$$

Here the row vector r^T has components $h\tilde{d}_h(x_j - \alpha)$. We choose the symbol r to indicate "restriction," since $r^T U^{n+1}$ computes the restriction of the vector U^{n+1} to the point α.

Solving (6.23) for U^{n+1} gives

$$U^{n+1} = A^{-1}BU^n + c_{n+1/2}z \tag{6.25}$$

where

$$z = kA^{-1}d. \tag{6.26}$$

Applying the constraint (6.24) to U^{n+1} in (6.25) allows us to solve for $c_{n+1/2}$:

$$c_{n+1/2} = \frac{1}{r^T z}(\bar{u}(t_{n+1}) - r^T A^{-1}BU^n). \tag{6.27}$$

If we use this to eliminate $c_{n+1/2}$ in (6.25) then we obtain

$$U^{n+1} = \left(I - \frac{zr^T}{r^T z}\right)A^{-1}BU^n + \left(\frac{\bar{u}(t_{n+1})}{r^T z}\right)z. \tag{6.28}$$

To simplify notation below we define

$$Q \equiv \left(I - \frac{zr^T}{r^T z}\right). \tag{6.29}$$

The restriction operator r^T is determined by our choice of discrete delta function $\tilde{d}_h$ in (5.5). As demonstrated by numerical results in §5.2, we obtain the best results by using $\tilde{d}_h = d_h^{(4)}$ in conjunction with $d_h = d_h^{(1)}$ in (6.25). We now wish to give an indication of why this method should produce second-order accurate results.

We define the local truncation error as usual by

$$T^n = Au(t_{n+1}) - Bu(t_n) - kc(t_{n+1/2})d. \tag{6.30}$$

Note that we use both the exact solution u and the exact function $c(t)$ in defining T^n. Consequently, $T^n = O(h^3)$, just as in the case where c is given. What changes now is the evolution equation for the global error E^n. If we solve (6.30) for $A^{-1}Bu(t_n)$ and use this in (6.28) then (6.28) becomes

$$\begin{aligned} U^{n+1} &= QA^{-1}B(u(t_n) + E^n) + \left(\frac{\bar{u}(t_{n+1})}{r^T z}\right) z \\ &= Q\left(u(t_{n+1}) - A^{-1}T^n - c(t_{n+1/2})z + A^{-1}BE^n\right) + \left(\frac{\bar{u}(t_{n+1})}{r^T z}\right) z \end{aligned}$$

so that

$$E^{n+1} = Q(A^{-1}BE^n - A^{-1}T^n) + \frac{1}{r^T z}(\bar{u}(t_{n+1}) - r^T u(t_{n+1}))z. \tag{6.31}$$

Note that the matrix Q anihilates the vector z so that the term involving $c(t_{n+1/2})$ drops out.

As usual, we must consider two effects in order to analyze this error: the local error introduced in each timestep, which comes partly from the truncation error T^n and partly from the interpolation error $\bar{u}(t_{n+1}) - r^T u(t_{n+1})$, and the stability properties needed to ensure that the accumulated errors remain small, which depends on properties of the iteration matrix $QA^{-1}B$.

The new error introduced in each step according to (6.31) is

$$-QA^{-1}T^n + \frac{1}{r^T z}(\bar{u}(t_{n+1}) - r^T u(t_{n+1}))z. \tag{6.32}$$

The vector z defined by (6.26) can be viewed as the result of applying a single step (with $\Delta t = \frac{k}{2}$) of the backward Euler method for the heat equation to the initial data $kd_h(x_j - \alpha)$. We therefore expect z to approximate a smooth but sharply peaked Gaussian centered about α with magnitude $O(k^{1/2})$, as is easily confirmed by numerical experiments. The magnitude of this Gaussian can be approximated by interpolating the vector z to the point α, for example by the value $r^T z$. It follows that $z/r^T z$ is a scaled version of z with $O(1)$ norm. The vector r^T also has an $O(1)$ norm since it is an interpolation operator. So the matrix Q has an $O(1)$ norm. Moreover, since A^{-1} is backward Euler, we have $\|A^{-1}T^n\| \le \|T^n\| = O(h^3)$ and so

$$\left\|QA^{-1}T^n\right\| = O(h^3).$$

The other component of the local error (6.32) has magnitude roughly given by the interpolation error

$$|\bar{u}(t_{n+1}) - r^T u(t_{n+1})|.$$

Recall that $\bar{u}(t_{n+1})$ is the exact value $u(\alpha, t_{n+1})$ whereas $r^T u(t_{n+1})$ is the value interpolated to α from the exact solution at the mesh points, $u(x_j, t_{n+1})$. If $d_h^{(4)}$ is used to define r^T then this error is $O(h^2)$ even though u has a kink at the point α. This is better than the $O(h)$ error which would result from using other choices of d_h, but at first glance does not appear to be enough to guarantee second-order global accuracy. However, notice that this $O(h^2)$ error is essentially confined to a few mesh points near the point α, since z is sharply peaked and decays exponentially. Hence our previous comments from §6.3 apply, and an $O(h^2)$ error is allowed at these points.

Moreover, notice that this error is always in the direction z. This may be significant in further reducing the effect of this error. The accumulated error E^n is multiplied by $QA^{-1}B$ in forming E^{n+1}, and the matrix Q anihilates z. This does not completely eliminate the $O(h^2)$ error introduced in the previous step because the error is first multiplied by $A^{-1}B$, but $A^{-1}B$ corresponds to a step of Crank–Nicolson on the homogeneous heat equation and its effect on the Gaussian vector z is to cause some decay but relatively little change in shape.

In analyzing the global error we must also consider the stability of the iteration matrix $QA^{-1}B$. We know that the Crank–Nicolson operator $A^{-1}B$ is unconditionally stable, and a standard von Neumann analysis shows that $\|A^{-1}B\|_2 = 1$. To show stability of the present method it would suffice to show that $\|QA^{-1}B\|_2 \leq 1$, for which it would suffice to show that $\|Q\|_2 \leq 1$. Unfortunately, this is not strictly true: we will display later a vector y for which $\|Qy\|_2 > \|y\|_2$. However, this seems to be a very unusual situation because of the fact that Q is nearly a projection matrix. Recall that if v is any vector with $\|v\|_2 = 1$ then $I - vv^T$ is a projection matrix with norm 1. Here v is replaced by $z/r^T z$ in one instance and by r in the second, but each of these vectors has norm roughly equal to 1 and can be viewed as an interpolation operator. For example, $r^T y \approx y(\alpha)$ for smooth vectors y. The vector $z/r^T z$ has a similar effect, since it is sharply peaked near α. Hence we might expect Q to behave much like a projection matrix.

More specifically, applying Q to any vector y gives

$$\hat{y} = y - r^T y \left(\frac{z}{r^T z}\right).$$

The last term here is sharply peaked with amplitude roughly $r^T y \approx y(\alpha)$. Most components of this vector are nearly zero and so the corresponding components of y are unchanged. If the components of y near α are smoothly varying, then modifying these components by subtracting out a Gaussian with peak value $y(\alpha)$ will almost certainly decrease the norm of the vector. See Fig. 7 for a typical example. However, because $z \neq r$, it is possible to construct examples where $\|\hat{y}\|_2 > \|y\|_2$. For example, if all components of y are nonnegative and near α we have

$$y_{J-1} = 1, \quad y_J = 0, \quad y_{J+1} = 0, \quad y_{J+2} = 1$$

where $x_J < \alpha < x_{J+1}$, then $r^T y$ will be negative, since r is based on the extrapolatory delta function $d_h^{(4)}$. The vector $r^T y(z/r^T z)$ will then have strictly negative entries and subtracting this from y gives a vector $\hat{y}$ with $\|\hat{y}\|_2 > \|y\|_2$.

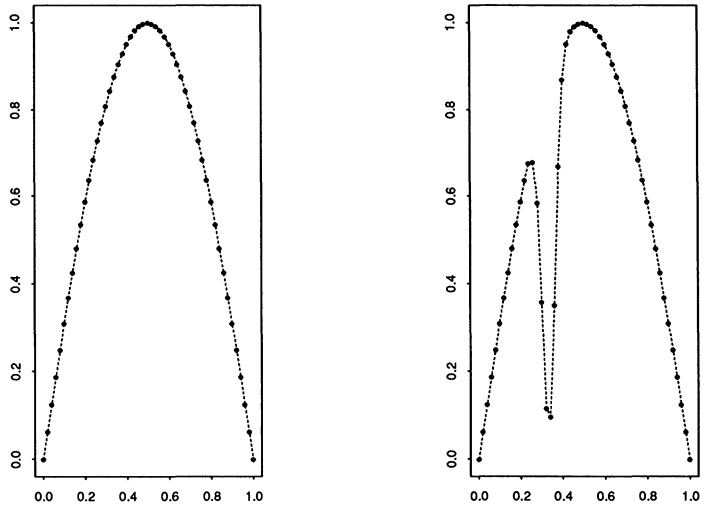


FIG. 7. *The effect of multiplying a vector y with smooth components by the matrix Q.* (a) *Components of y.* (b) *Components of Qy.*

This situation is very unusual. For almost all vectors the matrix Q is norm decreasing. Moreover this matrix is multiplied by $A^{-1}B$ to form the actual iteration matrix, and $A^{-1}B$ is itself a stable matrix. We do not expect stability to be any problem and have not observed any in practical applications.

Now consider the error in $c_{n+1/2}$ as given by (6.27). From the definition (6.30) of the local truncation error we find that

$$c(t_{n+1/2})z = u(t_{n+1}) - A^{-1}Bu(t_n) - A^{-1}T^n.$$

Applying r^T to this, we can solve for

$$c(t_{n+1/2}) = \frac{1}{r^T z}\left[r^T u(t_{n+1}) - r^T A^{-1}Bu(t_n) - r^T A^{-1}T^n\right].$$

Subtracting this from (6.27) gives

$$\begin{aligned} c_{n+1/2} - c(t_{n+1/2}) &= \frac{1}{r^T z}\left[(\bar{u}(t_{n+1}) - r^T u(t_{n+1}))\right. \\ &\qquad \left. - r^T A^{-1}B(U^n - u(t_n)) - r^T A^{-1}T^n\right]. \end{aligned} \tag{6.33}$$

We have $r^T A^{-1}T^n = O(h^3)$ while the other two terms in the brackets are each $O(h^2)$. Unfortunately, $r^T z = O(h^{1/2})$, so that this suggests an $O(h^{3/2})$ error in $c_{n+1/2}$. However, notice that

$$\begin{aligned} r^T(U^n - u(t_n)) &= \bar{u}(t_n) - r^T u(t_n) \\ &= \bar{u}(t_{n+1}) - r^T u(t_{n+1}) + O(h^3), \end{aligned}$$

using the fact that $\bar{u}(t) - r^T u(t) = O(h^2)$ and assuming $\bar{u}$ is smooth in time. This suggests that some cancellation occurs between the two $O(h^2)$ terms in (6.33), although

a complete analysis is complicated by the presence of the $A^{-1}B$ in the second term. Apparently this results in cancellation only to $O(h^{2.5})$ rather than to $O(h^3)$, so that dividing by $r^T z$ then gives an $O(h^2)$ error in $c_{n+1/2}$.

We can also see from (6.33) what happens if we use a different choice for d_h or $\tilde{d}_h$. If we keep $d_h = d_h^{(1)}$ but change $\tilde{d}_h$, then T^n remains the same but r is changed. For example, if we take $\tilde{d}_h = d_h^{(1)}$, the interpolation error $\bar{u}(t_{n+1}) - r^T u(t_{n+1})$ will be $O(h)$ rather than $O(h^2)$. We expect the error in $c_{n+1/2}$ to be one order lower as a result, as is confirmed by Table 2.

If we change d_h then T^n changes. For example, suppose we take $d_h = d_h^{(2)}$. Then from (6.22) we see that T^n is modified by the addition of a term

$$\frac{k}{2}(c(t_n) + c(t_{n+1}))(d_h^{(1)}(x_j - \alpha) - d_h^{(2)}(x_j - \alpha)).$$

Note that $d_h^{(1)}(x) - d_h^{(2)}(x)$ is $O(h^{-1})$ in regions where it is nonzero, so that T^n acquires a term of the form

$$k\beta d + [O(1) \text{ terms localized at nearby meshpoints}].$$

Here β is an $O(1)$ constant. This causes (6.33) to be modified by adding the term

$$\frac{1}{r^T z}[r^T A^{-1}(k\beta d + \cdots)] = \beta + \text{ other } O(1) \text{ terms}$$

since $kA^{-1}d = z$. This introduces an error in $c_{n+1/2}$ that is $O(1)$, as confirmed by the results in Table 2.

6.6. $c(t)$ and $\alpha(t)$ given. We next consider the case where the source location $\alpha(t)$ is varying in a known manner and study the method (5.7),

$$\begin{aligned}\left(1 - \frac{1}{2}kD_x^2\right) U_j^{n+1} &= \left(1 + \frac{1}{2}kD_x^2\right) U_j^n \\ &+ \frac{k}{2}\left[c(t_n)d_h^{(1)}(x_j - \alpha(t_n)) + c(t_{n+1})d_h^{(1)}(x_j - \alpha(t_{n+1}))\right] \\ &+ \frac{k}{2}(Y_j^n + Y_j^{n+1}) + W_j^n.\end{aligned} \tag{6.34}$$

We will compute the local truncation error of this method for general Y_j^n and W_j^n. By setting $Y_j^n = W_j^n = 0$ we can obtain the truncation error for the method (5.4) and we can also derive the correct expressions for Y_j^n and W_j^n to give second-order accuracy.

Computing the local truncation error of this method gives

$$\begin{aligned}T_j^n &= [u(x_j, t_{n+1}) - u(x_j, t_n)] \\ &- \frac{k}{2}[D_x^2 u(x_j, t_n) + c(t_n)d_h(x_j - \alpha(t_n)) + Y_j^n \\ &+ D_x^2 u(x_j, t_{n+1}) + c(t_{n+1})d_h(x_j - \alpha(t_{n+1})) + Y_j^{n+1}] - W_j^n.\end{aligned} \tag{6.35}$$

Since $[u_x]_\alpha = -c(t)$, we can use Corollary 6.2 to obtain

$$\begin{aligned} & D_x^2 u(x_j,t) + c(t) d_h^{(1)}(x_j - \alpha(t)) \\ & = u_{xx}(x_j,t) - \frac{h^2}{2}[u_{xx}]_\alpha \mathrm{sgn}(x_j - \alpha(t)) \left(d_h^{(1)}(x_j - \alpha(t))\right)^2 \\ & \quad + \frac{h^4}{6}[u_{xxx}]_\alpha \left(d_h^{(1)}(x_j - \alpha(t))\right)^3 + O(h^2). \end{aligned} \tag{6.36}$$

If we assume that $\alpha(t) \neq x_j$ at t_n or t_{n+1}, then

$$u_{xx}(x_j,t) = u_t(x_j,t) \tag{6.37}$$

at each time level. If, moreover, $\alpha(t) \neq x_j$ for $t_n \leq t \leq t_{n+1}$, then $u_t(x_j,t)$ will be a smooth function of t in the timestep and we have

$$u(x_j,t_{n+1}) - u(x_j,t_n) = \frac{k}{2}[u_t(x_j,t_n) + u_t(x_j,t_{n+1})] + O(k^3). \tag{6.38}$$

Using (6.36), (6.37), and (6.38) in (6.35) gives

(6.39)

$$\begin{aligned} T_j^n = \frac{k}{2} \Bigg\{ & \frac{h^2}{2}[u_{xx}(\cdot,t_n)]_\alpha \mathrm{sgn}(x_j - \alpha(t_n)) \left(d_h^{(1)}(x_j - \alpha(t_n))\right)^2 - Y_j^n \\ & + \frac{h^2}{2}[u_{xx}(\cdot,t_{n+1})]_\alpha \mathrm{sgn}(x_j - \alpha(t_{n+1})) \left(d_h^{(1)}(x_j - \alpha(t_{n+1}))\right)^2 - Y_j^{n+1} \\ & - \frac{h^4}{6}[u_{xxx}(\cdot,t_n)]_\alpha \left(d_h^{(1)}(x_j - \alpha(t_n))\right)^3 \\ & - \frac{h^4}{6}[u_{xxx}(\cdot,t_{n+1})]_\alpha \left(d_h^{(1)}(x_j - \alpha(t_{n+1}))\right)^3 \Bigg\} - W_j^n + O(h^3). \end{aligned}$$

The terms involving $[u_{xxx}]_\alpha$ have magnitude $O(h^2)$. However, these are localized near the point α, since $(d_h^{(1)})^3$ is nonzero at only two points. Hence these terms do not lead to a degradation in accuracy.

However, the terms involving $[u_{xx}]_\alpha$ are $O(h)$ and so even though these are also spatially localized, we will see some degradation in accuracy if $[u_{xx}]_\alpha \neq 0$ and $Y_j^n = 0$. The correction terms Y_j^n are thus chosen to cancel out these error terms. (Recall that in the previous cases analyzed α is fixed and consequently $[u_{xx}]_\alpha = 0$.)

To compute the jump in u_{xx}, first note that $[u_{xx}]_\alpha = [u_t]_\alpha$ as before, from the original partial differential equation. The solution u is still continuous and so we have

$$u(\alpha(t)+,t) - u(\alpha(t)-,t) = 0 \quad \forall\, t.$$

Differentiating this with respect to t gives

$$\alpha'(t)[u_x]_\alpha + [u_t]_\alpha = 0,$$

and hence

$$\begin{aligned} [u_{xx}]_\alpha = [u_t]_\alpha &= -\alpha'(t)[u_x]_\alpha \\ &= \alpha'(t)c(t). \end{aligned} \tag{6.40}$$

We see that when α is moving this jump is nonzero.

To eliminate this error term we take

$$Y_j^n = \frac{h^2}{2}[u_{xx}(\cdot,t_n)]_\alpha \mathrm{sgn}(x_j - \alpha(t_n)) \left(d_h^{(1)}(x_j - \alpha(t_n))\right)^2.$$

Using (6.40) in this gives (5.8).

With this choice of Y_j^n, (6.6) becomes

$$T_j^n = [O(h^2) \text{ terms localized near } \alpha] - W_j^n + O(h^3). \tag{6.41}$$

Taking $W_j^n = 0$ should then give second-order global accuracy. But recall that we derived (6.6) under the assumption that $\alpha(t) \neq x_j$ during the timestep. We now consider the case where α does cross a grid line, and see that a new error is introduced that can be eliminated by the proper choice of W_j^n.

We have assumed that the timestep is small enough that α does not cross more than one grid line in any given timestep, so $\alpha(t) \neq x_j$ for all j, except perhaps at one value x_i. If $\alpha(\tau) = x_i$ for some τ with $t_n < \tau < t_{n+1}$, then (6.35), (6.36), and (6.37) are still valid for $j = i$, but (6.38) is not since $u(x_i,t)$ is not smooth in time. We will derive the correct expression by expanding in Taylor series about the time τ since u is smooth on either side of τ. We have

$$u(x_i,t_n) = u(x_j,\tau) + (t_n - \tau)u_t^- + O(k^2),$$

where $u_t^- \equiv u(x_j,\tau-)$. We can also expand

$$\begin{aligned} u(x_i,t_{n+1}) &= u(x_i,\tau) + (t_{n+1} - \tau)u_t^+ + O(k^2) \\ &= u(x_i,\tau) + (t_{n+1} - \tau)u_t^- + (t_{n+1} - \tau)[u_t]_\tau + O(k^2), \end{aligned}$$

where

$$[u_t]_\tau = u_t(x_i,\tau+) - u_t(x_i,\tau-).$$

Combining these expressions gives

$$u(x_i,t_{n+1}) - u(x_i,t_n) = ku_t^- + (t_{n+1} - \tau)[u_t]_\tau. \tag{6.42}$$

On the other hand, we also have

$$\begin{aligned} u_t(x_i,t_n) &= u_t^- + O(k), \\ u_t(x_i,t_{n+1}) &= u_t^- + [u_t]_\tau + O(k) \end{aligned}$$

so that

$$\frac{k}{2}[u_t(x_i,t_n) + u_t(x_i,t_{n+1})] = ku_t^- + \frac{k}{2}[u_t]_\tau. \tag{6.43}$$

Combining (6.42) and (6.43) shows that (6.38) must now be replaced by

$$\begin{aligned} u(x_i,t_{n+1}) - u(x_i,t_n) &= \frac{k}{2}[u_t(x_i,t_n) + u_t(x_i,t_{n+1})] \\ &\quad + (t_{n+1/2} - \tau)[u_t]_\tau + O(k^2). \end{aligned} \tag{6.44}$$

Since the $O(k^2)$ term is localized at the single grid point x_i, we do not need to be concerned about its effect on the global error.

Using the expression (6.44) in (6.35) and following through the rest of the analysis as before, we find that (6.41) becomes, for this value of $j = i$,

$$T_j^n = [O(h^2) \text{ terms localized near } \alpha] + (t_{n+1/2} - \tau)[u_t]_\tau - W_i^n + O(h^3).$$

The new term introduced appears only at the one point near α, but has magnitude $O(h)$ and hence may cause a degradation in accuracy unless we choose the correction term W_i^n to eliminate this error. We want

$$W_i^n = (t_{n+1/2} - \tau)[u_t]_\tau. \tag{6.45}$$

Note that

$$\begin{aligned}
[u_t]_\tau &= u_t(x_i, \tau+) - u_t(x_i, \tau-) \\
&= -\text{sgn}(\alpha'(\tau))(u_t(x_i+, \tau) - u_t(x_i-, \tau)) \\
&= -\text{sgn}(\alpha'(\tau))[u_t(\cdot, \tau)]_{\alpha(\tau)} \\
&= -\text{sgn}(\alpha'(\tau))\alpha'(\tau)c(\tau) \\
&= -|\alpha'(\tau)|c(\tau)
\end{aligned}$$

where we have used (6.40). Using this in (6.45) gives (5.9).

7. Conclusions. We have examined the capabilities of an immersed boundary method for some one-dimensional model problems. We find that it is possible to achieve good accuracy with this approach if we are careful to choose appropriate discrete representations of the delta function. Our analysis guides this choice and explains the success of the method. It is illuminating to view the discrete delta function as a correction term that is needed to adjust the standard centered approximation to u_{xx} in the case where u has discontinuous derivatives.

When $\alpha(t)$ is not constant, we see that it is sometimes necessary to add correction terms to the naive approximation to the delta function in order to maintain second-order accuracy. Our analysis shows how such correction terms can be derived.

We believe that similar techniques can be used to investigate the immersed boundary method of Peskin for fluid dynamical problems in two (or even three) space dimensions. Our results suggest that a careful choice of delta functions will be required for optimal accuracy and that it may even be necessary to incorporate additional correction terms, particularly in the case of a moving boundary.

REFERENCES

[1] J. B. Bell, P. Colella, and H. M. Glaz, *A second-order projection method for the incompressible Navier–Stokes equations*, J. Comput. Phys., 85 (1989), pp. 257–283.

[2] R. P. Beyer, *A computational model of the cochlea using the immersed boundary method*, Ph.D. thesis, University of Washington, Seattle, WA, 1989.

[3] L. Fauci and C. S. Peskin, *A computational model of aquatic animal locomotion*, J. Comput. Phys., to appear.

[4] A. L. Fogelson, *A mathematical model and numerical method for studying platelet adhesion and aggregation during blood clotting*, J. Comput. Phys., 56 (1984), pp. 111–134.

[5] A. Mayo, *The fast solution of Poisson's and the biharmonic equations on irregular regions*, SIAM J. Numer. Anal., 21 (1984), pp. 285–299.

[6] C. S. Peskin, *Numerical analysis of blood flow in the heart*, J. Comput. Phys., 25 (1977), pp. 220–252.