UNIVERSITY OF CINCINNATI

Nov. 21 , 19 94

I, Yinghui Liu ,

hereby submit this as part of the requirements for the degree of:

Ph.D.

in Mathematics

It is entitled Mollification Method for 2-D IHCP On Bounded Domains

Approved by:

Professor D. A. Murio

Professor C.W. Groetsch

Professor A. Leung

Mollification Method for 2-D IHCP on Bounded Domains

A dissertation submitted to the
Division of Research and Advanced Studies
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in the Department of Mathematical Sciences
of the College of Arts and Sciences

1994

by

Yinghui Liu

B.S., Nankai University (1982)
M.S., University of Cincinnati (1992)

Advisor: Diego A. Murio

Abstract

In this dissertation, different space marching implementations of the Mollification method are introduced to numerically recover the temperature and heat flux histories on a bounded two–dimensional rectangular body when the initial sample data are collected on one side of the body. We combined the mollification method with a singular perturbation scheme to obtain a stable algorithm. A reliable set of parameter values is experimentally determined by numerical tests to guarantee the accuracy and stability of the algorithm.

Acknowledgements

I would like to express my deepest gratitude to my dissertation advisor, Professor Diego A. Murio, for his help and kindness in all these years. It would be hard to image that this research could have been completed without his guidence and encouragement.

Professor C.W. Groetsch and Professor A. Leung, the members of my committee, read the manuscripts carefully and gave many comments. I would like to thank them for their patience and time.

I would also like to thank Professor J.K. King and D. French for their encouragement during my stay at the University of Cincinnati.

Contents

List of Tables

List of Figures

1 Introduction

Theoretical concepts and computer implementations related to the inverse heat conduction problem (IHCP) have been mostly restricted to one-dimensional models. The difficulties of the two-dimensional IHCP are more pronounced and very few results are available in this case.

The first analytical solution – requiring exact data – that is applicable to two-dimensional conduction systems for geometries of arbitrary shape was introduced by M. Imber [5]. Most of the literature related to the numerical treatment of the two-dimensional IHCP is based on different ways of combining finite elements realizations with the Future Temperatures Method of J. V. Beck [3]. For some of the early applications of these ideas, see B. R. Bass and L. J. Ott [1]. More numerical experimentation can be found in T. Yoshimura and K. Ituka [7]. An elaborated and comprehensive exposition of the method was presented later by J. Baumeister and H. J. Reinhardt [2], and, more recently, by N. Zabaras and J. C. Liu [8]. See D. A. Murio [6] for a historical perspective.

The fundamental part of this study is to numerically test the space-marching implementation of the mollification method for the two dimensional

IHCP, which was proposed by L. Guo and D. A. Murio [4] as a consistent and stable method to unbounded domain. A review of this implementation, as well as the corresponding numerical analysis, will be presented in Section 2.

We will test the proposed algorithm for two-dimensional IHCP in section 3. The space domain in the (x, y) plane will be restricted to the boundary prototype rectangle $R = [0, 0.5] \times [0, 1]$, and the time domain will also be restricted to the interval [0,1]. Therefore we will work on the cube $D = R \times [0, 1]$ in the (x, y, t) space. Our goal is, starting from the sample data collected on the initial surface $x = 0$, to recover the temperature and heat-flux history for the whole solid cube, especially, for the other side of the cube $x = 0.5$. Random noise up to certain magnitude will be added in the initial data sample to test the stability of the implementation. It will turn out that the mollification method indeed stabilizes the numerical procedure and gives a consistent recovery of the temperature and heat-flux functions.

We will also introduce other techniques, such as singular perturbation schemes, in our numerical experiments to enhance the stability and consistency of the numerical procedure. All these improvements will be presented in section 4.

For the implementation of the mollification method on the cube D, there is a so called "lost boundary" problem. The temperature and heat-flux functions can only be recovered on an "interior" domain in the y, t directions, and the information on a substantial portion of the boundary can not be obtained by direct application of the algorithm. We will investigate some strategies for the recovery of both functions on the "lost boundary" in section 5.

Five different model examples will be used to illustrate the results of the numerical experiments in all cases. We will also assume that all the functions involved are L_2 functions in R^2 suitably extended whenever is necessary for the analysis.

2 The Two-D IHCP

We consider a two-dimensional IHCP when the space domain in the (x,y) plane is restricted to the rectangle R=[0,0.5]×[0, 1] and the time domain is also restricted to the interval [0,1]. Consequently, we will be working with the cube D=[0,0.5]×[0, 1] × [0, 1] in the (x,y,t) space. The temperature and heat flux on the 2-surface $x = 0$ (the initial surface) are approximately measurable. It is desired to recover the temperature history $u(x, y, t)$ and

the heat flux history $u_x(x, y, t)$ on the whole cube, especially, on the other side of the cube $x = 0.5$.

2.1 The problem

The unknown temperature $u(x, y, t)$ satisfies:

$$u_t = u_{xx} + u_{yy} \tag{1}$$

with the boundary conditions:

$$u(0, y, t) = F(y, t), \qquad u_x(0, y, t) = Q(y, t), \tag{2}$$

where $F(y, t)$ is the exact initial temperature function and $Q(y, t)$ is the exact initial heat-flux function.

Let $F_m(y, t)$ and $Q_m(y, t)$ be the approximated data functions corresponding to $F(y, t)$ and $Q(y, t)$ respectively. We will assume that they satisfy the L_2 data error bounds

$$\|F - F_m\| \leq \varepsilon, \qquad \|Q - Q_m\| \leq \varepsilon. \tag{3}$$

Let

$$f(y,t) = u(0.5,y,t), \qquad q(y,t) = u_x(0.5,y,t), \tag{4}$$

be the desired unknown temperature and heat-flux functions on the other side of the cube at $x = 0.5$. The Fourier analysis of system $(1) - (4)$, presented in L. Guo and D. A. Murio [4], shows that, when solving for $f(y,t)$ and $q(y,t)$ from $F(y,t)$ and $Q(y,t)$, $(-\infty < y < +\infty)$, the errors in the high frequency components will be amplified by the factor

$$\exp\left[\left(\sqrt{s^4+w^2}+s^2\right)^{1/2}\right],$$

where s and w represent the Fourier transform variables associated with y and t respectively. Thus, the inverse problem is highly ill-posed in the high frequency components.

2.2 The mollification scheme

Introducing the two-dimensional Gaussian kernel

$$\rho(y,t,\delta_y,\delta_t) = \frac{1}{\pi\delta_y\delta_t}\exp\left[-\left(\frac{y^2}{\delta_y^2}+\frac{t^2}{\delta_t^2}\right)\right]$$

and denoting $\rho(y,t,\delta_y,\delta_t)$ by $\rho_\delta(y,t)$ if $\delta_y = \delta_t = \delta$, the two-dimensional convolution of any locally integrable function $g(y,t)$ with the Gaussian kernel

$\rho_\delta(y,t)$ – the mollification of the function $g(y,t)$ – is written as

$$J_\delta g(y,t) = (\rho_\delta * g)(y,t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \rho_\delta(y',t')g(y-y',t-t')dy'dt'.$$

Mollifying system $(1)-(4)$, we obtain

$$(J_\delta u)_t = (J_\delta u)_{xx} + (J_\delta u)_{yy} \tag{5}$$

with the boundary conditions

$$J_\delta u(0,y,t) = J_\delta F(y,t), \qquad J_\delta u_x(0,y,t) = J_\delta Q(y,t). \tag{6}$$

Let $J_\delta f_m(y,t) = J_\delta u(0.5,y,t)$ and $J_\delta q_m(y,t) = J_\delta u_x(0.5,\ y,t)$ when we use the approximated data functions $F_m(y,t)$ and $Q_m(y,t)$ instead of $F(y,t)$ and $Q(y,t)$. We have:

Theorem 1. *Suppose that* $\|F - F_m\| \le \epsilon$ *and* $\|Q - Q_m\| \le \epsilon$. *Then,*

1. *Problem* (5)*-(6) is formally stable with respect to noises in the data.*

2. *If the exact temperature function* $f(y,t)$ *and heat flux function* $q(y,t)$ *have uniformly bounded first order partial derivatives on the bounded domain* D *, then* $J_\delta f_m$ *and* $J_\delta q_m$ *satisfy*

$$\|f - J_\delta f_m\|_D \le O(\delta) + 2\epsilon \exp[7\delta^{-2}]$$

and

$$\|q - J_\delta q_m\|_D \leq O(\delta) + 2\epsilon \exp[7\delta^{-2}].$$

The proof of this statement can be found in L. Guo and D. A. Murio [4].

2.3 Space marching scheme

With $v = J_\delta u$ and $w = v_x$, equation (5) becomes

$$v_t = w_x + v_{yy}$$

and the boundary condition (6) is now

$$v(0, y, t) = J_\delta F_m(y, t), \qquad w(0, y, t) = J_\delta Q_m(y, t).$$

To find the mollified functions $J_\delta f_m(y, t)$ and $J_\delta q_m(y, t)$, one basically has to mollify the initial data samples, and then solve the heat-conduction equation by a space marching scheme.

Consider a uniform grid in the (x, y, t) space:

$$\{(\xi_i = ih,\ y_j = j\triangle y,\ t_n = n\ \triangle t) \in D : i, j, n \in Z^+\},$$

where $\triangle y$, $\triangle t$ are the sample step sizes in the y and t directions and h is the space marching step size in the x direction. We call it the **Starting grid.**

One sees that the indices i, j, n are restricted by:

$$0 \leq i \leq \frac{0.5}{h} = N_x,\ 0 \leq j \leq \frac{1}{\triangle y} = N_y,\ 0 \leq n \leq \frac{1}{\triangle t} = N_t,$$

since we work only on the cube D.

Let the grid functions $\ddot{V}$ and W be defined by

$$V_{i,j}^n = v(\xi_i, y_j, t_n),\ \ W_{i,j}^n = w(\xi_i, y_j, t_n).$$

On the surface $x = 0$,

$$V_{0,j}^n = J_\delta F_m(y_j, t_n), \qquad W_{0,j}^n = J_\delta Q_m(y_j, t_n).$$

We approximate the system of partial differential equations (5) with the consistent finite difference schemes

$$W_{i+1,j}^n = W_{i,j}^n + \tfrac{h}{2\triangle t}(V_{i,j}^{n+1} - V_{i,j}^{n-1}) - \tfrac{h}{\triangle y^2}(V_{i,j-1}^n - 2V_{i,j}^n + V_{i,j+1}^n), \tag{7}$$

$$V_{i+1,j}^n = V_{i,j}^n + hW_{i-1,j}^n.$$

The stability of the finite difference scheme (7) and the convergence of the numerical solution to the mollified problem restricted to the grid points are shown in L. Guo and D. A. Murio [4], with $W_{i-1,j}^n$ replaced by $W_{i,j}^n$ in the last equation.

Remarks:

1. The radius of mollification, δ, can be selected automatically as a function of the level of noise in the data. In fact, for a given $\epsilon > 0$, there is a unique $\delta > 0$, such-that

$$\|J_\delta F_m - F_m\|_D = \epsilon.$$

2. It is also possible to replace the discrete two-dimensional mollification of the data functions by two successive one-dimensional mollifications in time and y-space. In this manner, the data filtering task can be executed as a parallel process.

3 The Mollification Method

In this section we begin the numerical testing of the fully explicit space marching implementation of the mollification method. The numerical process, as well as the results of the recovery, depends on a set of parameters that includes the sample step-sizes, the radii of mollification and some others. One of the major tasks of this numerical study is to show practical stability, and to find an optimal set of parameters for the algorithm. We test

the algorithm on five different model examples. The overall behavior of the numerical process does not depend on specific model examples .

3.1 The list of examples

Here we list the five examples that are used to illustrate the results of our numerical experiments . As mentioned before, the performance of our algorithm is "problem independent". Thus, most of the time, we will only present the numerical results associated with **example 1.**

Example 1:

$$u(x,y,t) = e^{5-2t}\sin x \sin y$$

$$u_x(x,y,t) = e^{5-2t}\cos x \sin y$$

Example 2:

$$u(x,y,t) = e^{5-4.25t}\sin\frac{1}{2}x \sin 2y$$

$$u_x(x,y,t) = \frac{1}{2}e^{5-4.25t}\ \cos\frac{1}{2}x \sin 2y$$

Example 3:

$$u(x,y,t) = e^{t+\frac{1}{\sqrt{2}}(x+y)}$$

$$u_x(x,y,t) = \frac{1}{\sqrt{2}}e^{t+\frac{1}{\sqrt{2}}(x+y)}$$

Example 4:

$$u(x,y,t) = \frac{1}{3}x^3y + 3txy + \frac{1}{6}xy^3$$

$$u_x(x,y,t) = x^2y + 3ty + \frac{1}{6}y^3$$

Example 5:

$$u(x,y,t) = 50x - 100x^2 + 50y^2 - 100t$$

$$u_x(x,y,t) = 50 - 200x$$

All these functions satisfy the heat-conduction equation (1).

3.2 The parameter set

The performance of the mollification algorithm depends on a set of parameters. They are as follows:

i). Noise level ε and mollification constant δ.

The IHCP is an ill-posed problem because it is not stable with respect to noises in the collected data. Consequently, we add random noise to the initial data sample to test the algorithm. The magnitude of the noise is measured by the noise level constant ε. To stabilize the numerical procedure, we have to mollify the initial data sample before performing the space-marching. The optimal choice of the radius of mollification, δ, depends on the noise level ε.

ii). Grid constants $\triangle y, \triangle t$ and h.

It is desirable to make the grids denser in order to get more detailed information about the computed temperature and heat-flux functions. However, there is a trade off between the step-size and the accuracy of the recovery since the IHCP is an ill-posed problem. Keeping the accuracy at a certain level, we try to make the sample step-sizes $\triangle y, \triangle t$ and space-marching step-size h as small as possible. The performance of the algorithm is mostly justified by the consistency and the magnitude of these parameters.

We will also use the step numbers $N_y = \frac{1}{\triangle y}, N_t = \frac{1}{\triangle t}$ and $N_x = \frac{0.5}{h}$ in our analysis and tables.

iii). The grid refine constant k.

After the random noise is added, the mollification of the initial data sample is crucial to stabilize the algorithm. According to our experiments, it is not good enough to filter the noise by only using the sample data on the starting grid. In our implementation, we refine the starting grids on the initial 2-surface $x = 0$ by dividing each grid into $k \times k$ finer grids. We call this refined grid the **Mollification Grid**. The parameter k is called the grid refine constant.

The mollification is performed as follows: we first generate the initial

data sample on the mollification grid, add noise randomly, then compute the mollified initial data on the starting grid using all the data on the refined mollification grid. After that, we will restrict ourself to the starting grid to perform the space-marching scheme.

Remarks:

a). For a fixed grid point P on the surface $x = 0$, data mollification at P requires all the sample data in the square centered at P with width 6δ. Hence, the data mollification can not be carried out to a grid point whose t or y distance from the boundary of the unit square $[0,1] \times [0,1]$ is less than 3δ. This shrinks the grid we will actually work on from the starting grid on $[0, 0.5] \times [0,1] \times [0,1]$ to the grid on $[0, 0.5] \times [3\delta, 1 - 3\delta] \times [3\delta, 1 - 3\delta]$. We will call this latter grid the **Working Grid**.

For the moment, we will only consider the recovery of the temperature and heat-flux functions on the working grid. However, for most applications, the value of δ is usually not small, typically, $\delta = \frac{1}{15}$. Hence the shrinking of the original domain will cause a substantial loss of the information on and near the boundaries in the y and t directions. We will deal with this "lost boundary" in section 5.

b). As we march forward in the x-direction, we must drop the estima-

tion of V_{ij}^n associated with the boundary of the working grid in the y and t directions. This is again a "lost boundary" problem. However, this "lost boundary" is easier to recover. For each n, a one-dimensional 3-rd order extrapolation scheme is introduced in the algorithm to recover the lost y-boundary values. The extrapolation is accurate enough and causes no trouble in the stability of the numerical process.

3.3 The numerical procedure

We have two basic options to add the random noise to the initial data sample. One is example dependent and the other is example independent. For the first option, the noise is proportional to the temperature and heat-flux function values on the initial surface $x = 0$. In the second option, the noise is always selected randomly from a given range for every individual application. We call the first option the "relative noise adding" and the second one the "absolute noise adding".

In real applications, the initial data samples are collected by certain instruments and measurement schemes, and the magnitude of the noise slipped into the collected data sample depends mainly on the accuracy of the instru-

ment and the merit of the measurement scheme; the noise level is usually independent of the individual application. For this reason, we consider the "absolute noise adding" option in this study.

We would like to test the stability of the algorithm under moderately high levels of noise. However, higher noise level means that a larger mollification constant is required to stabilize the numerical process, and, therefore, we have a smaller inner cube to recover.

Two things we should worry most for the implementation and the testing of the algorithm: accuracy and stability. The accuracy of a recovery is measured by the l_2 error norm of the recovered function **over the whole cube.**

Since the IHCP is an ill-posed problem, we should pay special attention to the stability of the algorithm. This is indirectly measured by the size of the working grid, i.e., by the sample step-size and the space-marching step-size of the recovery. At the same level of accuracy, an algorithm is considered more stable if it allows for a denser working grid, i.e., smaller sample step-size and space marching step-size.

To put ourself in a comparable contention, we consider (most of the time) the noise level constant, ε, up to 0.005 in our experiments. The corresponding

mollification constant δ is 0.0667. A recovery is considered as accurate if the l_2 error norm of the recovered temperature function is less than 0.01 and that of the heat flux function is less than 1. An algorithm is **"quasi-stable"** if it allows an accurate recovery with $h, \triangle y, \triangle t \leq 0.1$.

No specific requirements are imposed on the grid refine constant k. We will take the smallest integer which stabilizes the numerical process. In most cases, $k = 10$.

3.4 The mollification result

For the mollification method, with $\varepsilon = 0.005$ and $\delta = 0.0667$, the optimal combination of the parameter values, according to our numerical experience, are $N_x = 10$, $N_y = N_t = 40$, $k = 10$.

For this specific choice of the parameter set, the l_2 error norms of the recovered temperature and heat-flux functions on the whole inner cube are shown in table 1.

Example 1	Temperature	Heat-flux
l_2 Error	0.0079	0.4490
Relative l_2 Error	0.0009	0.01153

Table 1: The l_2 error norms. Example 1.

$$N_x = 10,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k = 10.$$

Here and in what follows,

$$\text{Relative } l_2 \text{ error norm} = \frac{l_2 \text{ error norm}}{l_2 \text{ norm of the function}}.$$

Table 2 contains the l_2 error norms for a list of 2-surfaces for fixed values of x.

x	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
0.1	0.0153	0.0050	0.1776	0.0058
0.2	0.0573	0.0094	0.2514	0.0084
0.3	0.1295	0.0143	0.3246	0.0111
0.4	0.2310	0.0193	0.4367	0.0154
0.5	0.3612	0.0245	1.0794	0.0400

Table 2: The l_2 error norms for 2-surfaces for fixed values of x. Example 1.

$$N_x = 10,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k = 10.$$

We recall that for all the figures and tables above, the parameter set is: $\varepsilon = 0.005,\ \delta = 0.0667,\ N_x = 10,\ N_y = N_t = 40,\ k = 10.$

Next we will take these values as a reference to test the performance of the algorithm under a different choice of parameters. At each experience, only one of the parameters is changed.

Table 3 lists the dependency between the two parameters ε and δ. All the error norms are for the temperature function.

ε	δ	l_2 error	R. l_2 error
0.0005	0.0417	0.0061	0.0006
0.0010	0.0417	0.0061	0.0006
0.0020	0.0500	0.0069	0.0007
0.0030	0.0583	0.0071	0.0008
0.0040	0.0583	0.0071	0.0008
0.0050	0.0667	0.0079	0.0009

Table 3: The optimal δ for different ε's. Example 1.

$$N_x = 10,\ N_y = N_t = 40,\ k = 10.$$

Table 4 shows the performance of the algorithm when the space marching step-size h changes. Table 5 is for the changes of the sample step-sizes $\triangle y$ and $\triangle t$.

N_x	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
5	0.0138	0.0015	0.5242	0.0179
10	0.0079	0.0009	0.4490	0.0153
15	0.0058	0.0007	1.7076	0.0580
20	0.0060	0.0007	7.2921	0.2477
25	0.0093	0.0011	26.6089	0.9035

Table 4: The change of the space marching step-size. Example 1.

$N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k = 10.$

k	$N_y = N_t$	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
10	10	0.0301	0.0033	0.3188	0.0105
10	20	0.0151	0.0017	0.4476	0.0150
10	40	0.0079	0.0009	0.4490	0.0153
10	60	0.0053	0.0010	0.5186	0.0177
8	80	0.0040	0.0005	0.3826	0.0131
6	100	0.0032	0.0004	0.5165	0.0177

Table 5: The change of the initial sample step-size. Example 1.

$N_x = 10,\ \varepsilon = 0.005,\ \delta = 0.0667.$

Again, remember that all the tables and figures listed above are for example 1. The performance of the algorithm is the same when applied to other examples. The l_2 error norms of the temperature and heat flux functions for the other examples are show in table 6. For all the examples, we set $\varepsilon = 0.005,\ \delta = 0.0667,\ h = 0.05,\ \triangle y = \triangle t = 0.025,\ k = 10.$

Example	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
1	0.0079	0.0009	0.5242	0.0179
2	0.0042	0.0012	0.3857	0.0332
3	0.0028	0.0009	0.3307	0.1571
4	0.0017	0.0063	0.3306	0.3569
5	0.0917	0.0024	0.4842	0.0153

Table 6: The l_2 error norm, all the examples

$N_x = 10,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k = 10.$

Remarks: a). As we mentioned earlier, one would like to make the number of steps of the space marching, N_x, as big as possible to get more detailed information of the temperature and heat-flux functions on the cube D. However, due to the ill-posedness of the problem, the accuracy of the recovery is very sensitive to N_x. As illustrated in table 4, the accuracy drops dramatically on every increment of five units in N_x. According to our experience, N_x is one of the most important stability measurements. To keep the accuracy at the same level, allowing an increment of ten in N_x should be

regarded as a substantial improvement on the stability of a given algorithm.

b). As illustrated in table 5, the accuracy of the recovery seems to be less sensitive to the initial sample sizes N_t and N_y. This is indeed not true in general. If we change N_x from 10 to 15, as illustrated in the next table, the sensitivity of the accuracy on N_t and N_y becomes clear.

k	$N_t = N_y$	l_2 for $u(x,y,t)$	l_2 for $u_x(x,y,t)$
10	40	0.0058	1.7076
10	60	0.0036	1.3039
8	80	0.0059	4.2234
6	100	0.0027	32.1356

Table 7: The change of the initial sample size. Example 1.

$$N_x = 15,\ \varepsilon = 0.005,\ \delta = 0.0667.$$

Hence, the grid sizes N_t and N_y are also important measurements for the stability of the algorithm.

The major objective for a quasi-stable algorithm is to make $\mathbf{N_x}$,

as well as N_y and N_t, as large as possible while keeping the same level of accuracy.

c). For our example 1, the exact temperature function on the initial surface is identically zero. Nevertheless, we do add randomly generated noise to the **zero data**. The perturbed "zero data" causes specific trouble on the stability and accuracy of the recovery. As illustrated in table 6, the accuracy is only related to the magnitude of the noise added into the initial data sample, but not to the individual example model. The performance of the mollification method, when applied to different problems, is consistent.

4 The Stable Algorithm

In this section we describe different ways to improve the stability of the mollification algorithm. In particular, we will combine the mollification algorithm with a singular perturbation method. The addition of a suitably chosen term to the heat conduction equation will greatly improve the stability of the mollification algorithm.

4.1 The perturbation scheme

The discussion in this section is motivated by the realization that it is possible to modify the previous numerical algorithm in such a way as to allow for some extra filtering of the high frequency components of the noise. In particular, we would like to improve the stability related with the approximation of the second order partial derivative of the solution temperature with respect to the y-variable as we march in the x-direction. What we have in mind is a potential perturbation of the differential operator itself, so that the finite-difference implementation of the algorithm is not substantially changed.

We recall that in the frequency space, the relationship

$$\widehat{u_{xx}}(x,s,w) = (iw + s^2)\hat{u}(x,s,w), \tag{8}$$

where s and w are the frequency variables corresponding to the real variables y and t respectively, leads to the general solution

$$\hat{u}(x,s,w) = A(s,w)e^{\sqrt{iw+s^2}x} + B(s,w)e^{-\sqrt{iw+s^2}x}.$$

We propose to replace the solution of equation (8) by the function

$$\widehat{v}(x,s,w) = (A(s,w)e^{\sqrt{iw+s^2}x} + B(s,w)e^{-\sqrt{iw+s^2}x})e^{-\lambda s^2}, \qquad (9)$$

where λ is a small positive constant. We observe that in real space, *the function v is simply the convolution of the function u with a Gaussian kernel that acts only in the y-variable* and, consequently, the analysis of the high frequency components of section 2 applies directly to this situation. The fundamental task is now reduced to finding the governing differential equation for v.

To this end, we proceed as follows. Introducing $\alpha = \sqrt{iw + s^2}$, taking partial derivatives with respect to x and using (9), we have

$$\widehat{v_x}(x,s,w) = (\alpha - \lambda s^2)A(s,w)^{(\alpha-\lambda s^2)x} - (\alpha + \lambda s^2)B(s,w)e^{-(\alpha+\lambda s^2)x}$$

$$= \alpha(A(x,s,w)e^{\alpha x} - B(x,s,w)e^{-\alpha x})e^{-\lambda s^2 x} - \lambda s^2 \widehat{v}(x,s,w). \qquad (10)$$

Similarly, utilizing (10),

$$\widehat{v_{xx}}(x,s,w) = \alpha^2 \widehat{v}(x,s,w) - 2\alpha\lambda s^2(A(x,s,w)e^{\alpha x} - B(x,s,w)e^{-\alpha x})e^{-\lambda s^2 x}$$

$$-\lambda^2 s^4 \widehat{v}(x,s,w)$$

$$= \widehat{v_t}(x,s,w) - \widehat{v_{yy}}(x,s,w) - 2\lambda s^2 \widehat{v_x}(x,s,w) - \lambda^2 s^4 \widehat{v}(x,s,w).$$

By the linearity of the Fourier transformations, the function $v(x,y,t)$ satisfies the linear partial differential equation

$$v_{xx}(x,y,t) = v_t(x,y,t) - v_{yy}(x,y,t) + 2\lambda v_{xyy}(x,y,t) - \lambda^2 v_{yyyy}(x,y,t),$$

a singular perturbation of the original diffusion equation, that depends on the small positive parameter λ.

In actual implementations, assuming the solution is sufficiently smooth, we only retain the first order term in λ , i.e., numerically we use the approximation

$$v_{xx}(x,y,t) \cong v_t(x,y,t) - v_{yy}(x,y,t) + 2\lambda v_{xyy}(x,y,t).$$

Thus, instead of solving the equation

$$u_{xx} = u_t - u_{yy},$$

we will work with the perturbed equation

$$u_{xx} = u_t - u_{yy} + 2\lambda u_{xyy}, \tag{11}$$

where λ is a small positive constant which measures the magnitude of the singular perturbation.

For the numerical implementation, we use the following consistent finite difference schemes to obtain the fundamental formulas for the space-marching algorithm:

$$W^n_{i+1,j} = W^n_{i,j} + \frac{h}{2\triangle t}(V^{n+1}_{i,j} - V^{n-1}_{i,j}) - \frac{h}{\triangle y^2}(V^n_{i,j-1} - 2V^n_{i,j} + V^n_{i,j+1})$$

$$+2\lambda\frac{h}{\triangle y^2}(W^n_{i,j-1} - 2W^n_{i,j} + W^n_{i,j+1}),$$

$$V^n_{i+1,j} = V^n_{i,j} + hW^n_{i-1,j}.$$

4.2 Combined with mollification

To add the singular perturbation into the mollification algorithm, we only change the corresponding space-marching formulas. Everything else is kept the same, as in section 3. We again consider $\varepsilon = 0.005$ and $\delta = 0.0667$.

To see the improvement, we set $N_y = N_t = 40$ and $\lambda = 0.010$. One can compare Table 8 with Table 4 to see the differences made by the singular perturbation term.

N_x	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
5	0.0047	0.0013	0.1602	0.0138
10	0.0023	0.0007	0.5075	0.0437
15	0.0015	0.0004	0.1927	0.0166
20	0.0015	0.0004	0.2544	0.0219
25	0.0017	0.0005	0.3531	0.0304

Table 8: The l_2 error norms: With perturbation. Example 1.

$N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k = 10,\ \lambda = 0.010.$

Table 8 illustrates a significant improvement in the stability of the algorithm. Much smaller space marching step size is now allowed, and all the results satisfy our accuracy criterion.

We provide one more comparison. Table 9 contains the results without singular perturbation and table 10, those with singular perturbation. The parameter values are $\varepsilon = 0.005$, $\delta = 0.0667$, $N_y = N_t = 60$ and $\lambda = 0.004$.

N_x	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
5	0.0093	0.0010	0.5186	0.0177
10	0.0053	0.0006	0.3436	0.0117
15	0.0036	0.0004	1.3039	0.0445
20	0.0037	0.0004	9.9876	0.3405
25	0.0134	0.0015	78.3677	2.6714

Table 9: The l_2 error norms: Without perturbation. Example 1.

$N_y = N_t = 60,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k = 10.$

N_x	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
5	0.0091	0.0010	0.4455	0.0125
10	0.0051	0.0006	0.2363	0.0081
15	0.0035	0.0004	0.2495	0.0085
20	0.0027	0.0003	0.5511	0.0188
25	0.0024	0.0003	1.3109	0.0447

Table 10: The l_2 error norms: With perturbation. Example 1.

$N_y = N_t = 60,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k = 10,\ \lambda = 0.004.$

Figures 1 to 6 illustrate the qualitative behavior of the reconstructed temperatures and heat fluxes at $x = 0.5$, corresponding to Example 1 and the set of parameters $N_x = 20$, $N_y = N_t = 40$, $\varepsilon = 0.005$, $\delta = 0.0667$, $k = 10$ and $\lambda = 0.01$.

Figures 1 and 2, show the computed temperature and heat flux functions, respectively, for y and t between 0.2 and 0.8.

The discrete relative error functions associated with the temperature and heat flux functions $\left(\frac{V_{i,j}^n - u(ih,j\Delta y,n\Delta t)}{u(ih,j\Delta y,n\Delta t)} \text{ and } \frac{W_{i,j}^n - u_x(ih,j\Delta y,n\Delta t)}{u_x(ih,j\Delta y,n\Delta t)}\right)$, respectively, are displayed in Figures 3 and 4 for y and t between 0.2 and 0.8.

The transient grid relative errors for the temperature and heat flux functions, corresponding to the space locations $x = y = 0.5$ and $x = 0.5$, $y = 0.8$, respectively, are shown in Figures 5 and 6, for $0.2 \leq t \leq 0.8$.

Figure 1: Reconstructed temperature function at $x = 0.5$. Example 1.

$N_x = 20,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_y = k_t = 10,\ \lambda = 0.010$.

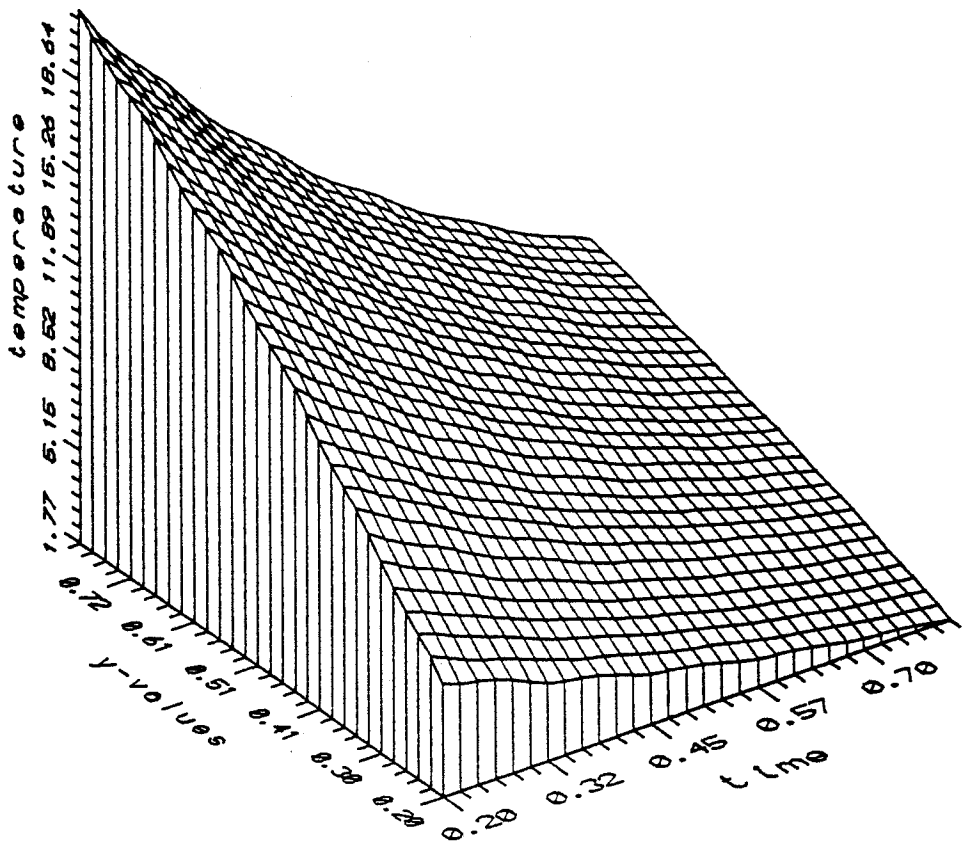


Figure 2: Reconstructed heat flux function at $x = 0.5$. Example 1.

$N_x = 20,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_y = k_t = 10,\ \lambda = 0.010.$

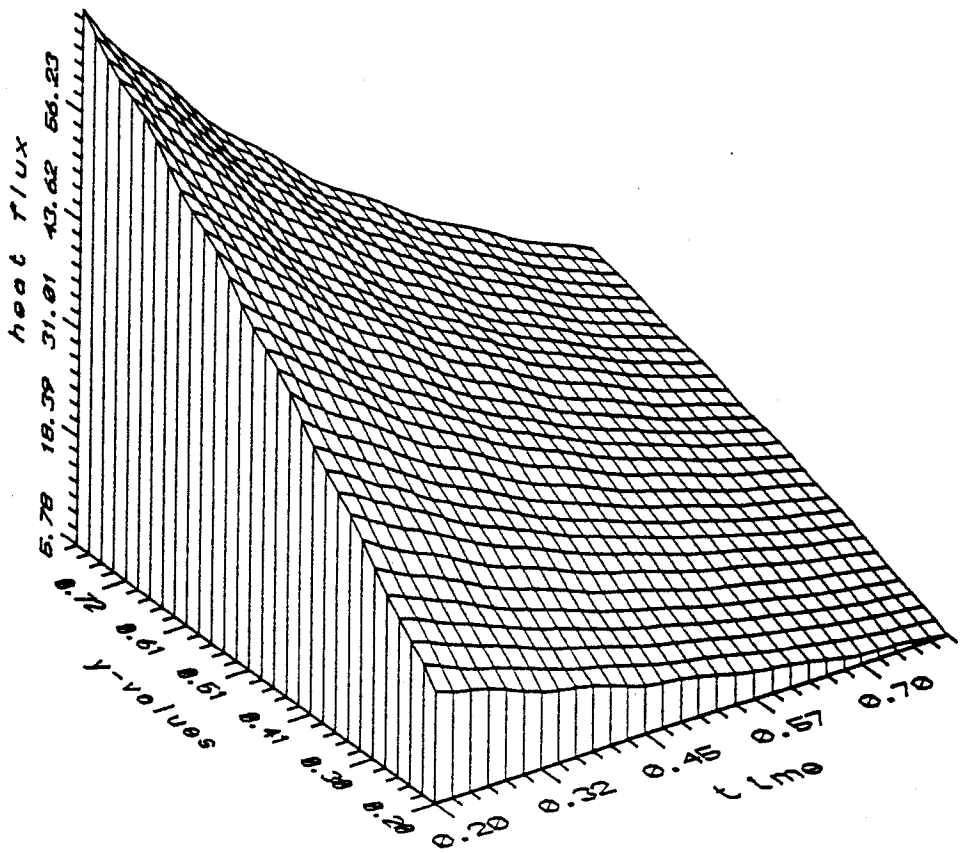


Figure 3: Relative errors for the temperature function at $x = 0.5$.

Example 1.

$N_x = 20,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_y = k_t = 10,\ \lambda = 0.010.$

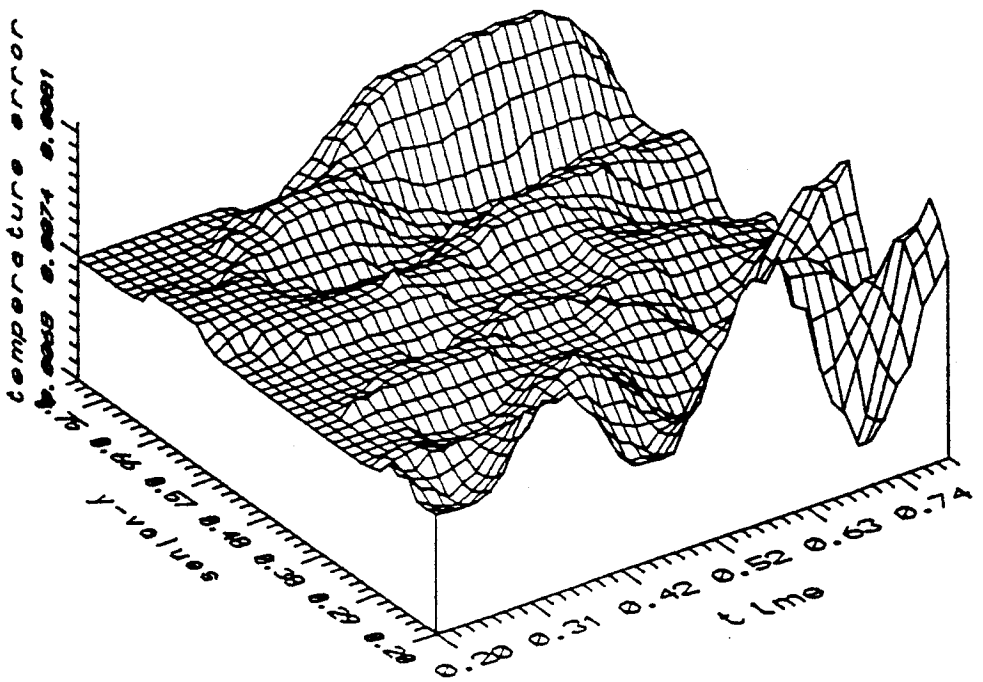


Figure 4: Relative errors for the heat flux function at $x = 0.5$. Example 1.

$N_x = 20,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_y = k_t = 10,\ \lambda = 0.010.$

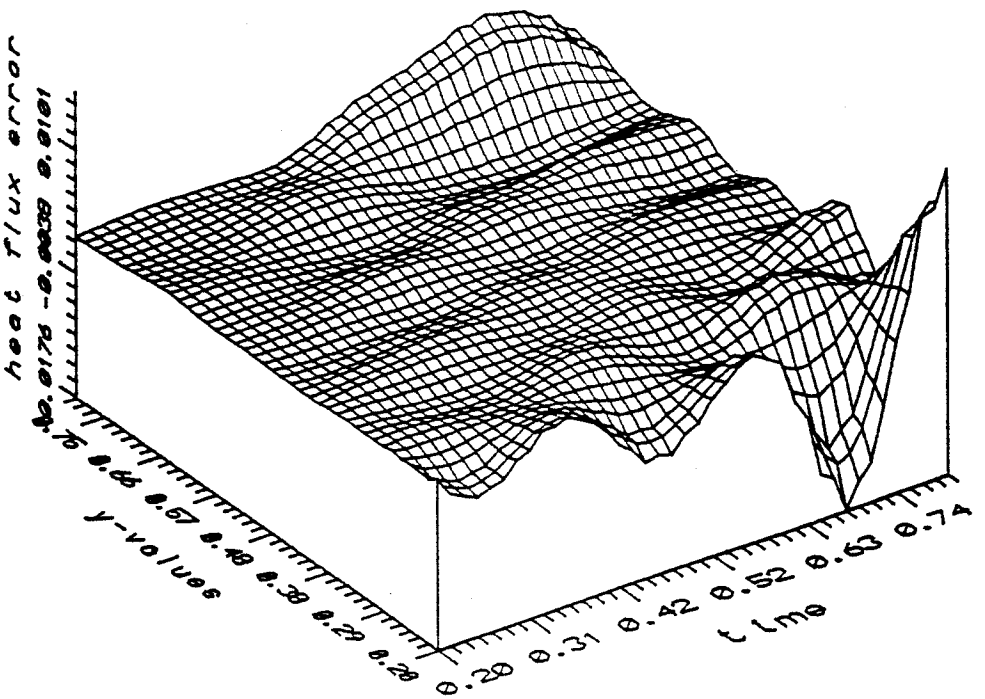


Figure 5: Transient relative errors for the temperature (*solid like*) and heat flux functions (*dashed like*) at $x = y = 0.5$. Example 1. $N_x = 20,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_y = k_t = 10,\ \lambda = 0.010.$

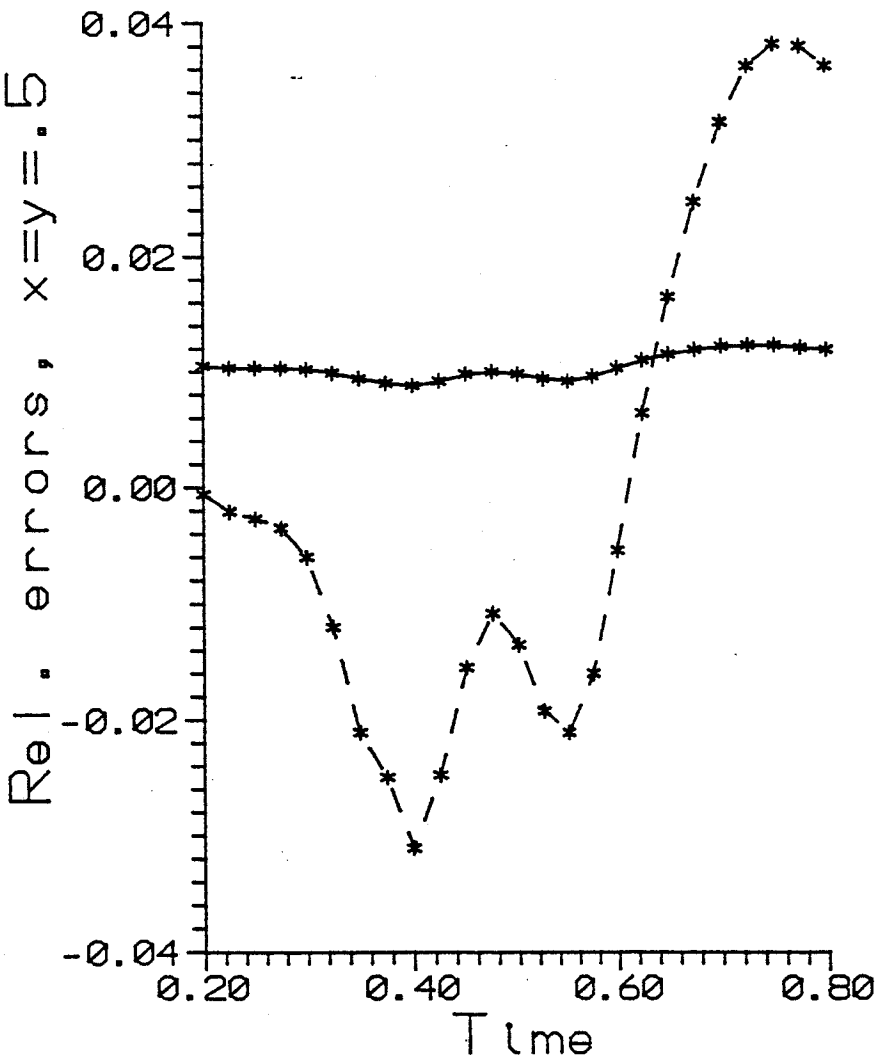
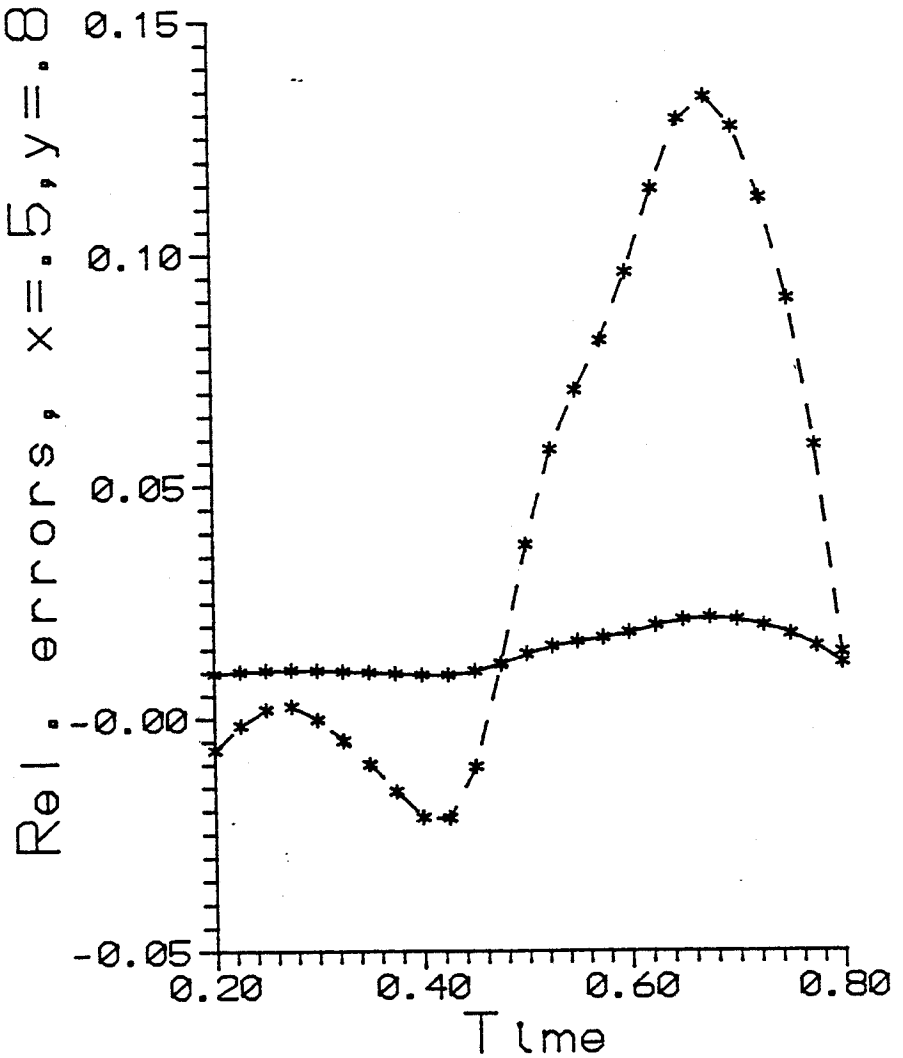


Figure 6: Transient relative errors for the temperature (*solid like*) and heat flux (*dashed like*) functions at $x = 0.5$, $y = 0.8$. Example 1. $N_x = 20,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_y = k_t = 10,\ \lambda = 0.010.$



4.3 Performance on dense grids

As illustrated in table 7, the accuracy of the recovery is also sensitive to the density of the grid for the mollification algorithm when $N_x > 10$. Adding the singular perturbation term also improves the performance of the algorithm on dense grids. To see the improvement, we test the algorithm on the grids with $N_y = N_t > 60$. N_x is set to 20, and again, $\varepsilon = 0.005$ and $\delta = 0.0667$ for all the cases considered.

Tables 11 and 12 are for $N_y = N_t = 80$. All the model examples are tested and the l_2 error norms are presented. Table 11 contains the results without the singular perturbation, and table 12 those with the singular perturbation. $\lambda = 0.003$ for the results illustrated in table 12.

Example	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
1	0.0059	0.0007	******	3.6241
2	0.0059	0.0017	******	9.2092
3	0.0059	0.0020	******	50.6177
4	0.0059	0.0224	******	******
5	0.0183	0.0005	******	3.5097

Table 11: The l_2 error norm: Without singular perturbation.

$N_x = 20, N_y = N_t = 80,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k = 8.$

Example	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
1	0.0021	0.0002	2.5275	0.0863
2	0.0015	0.0004	2.5108	0.2195
3	0.0013	0.0004	2.5258	1.2032
4	0.0013	0.0049	2.5258	2.7430
5	0.0186	0.0005	2.5259	0.0834

Table 12: The l_2 error norm: With singular perturbation.

$N_x = 20,\ N_y = N_t = 80,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k = 8,\ \lambda = 0.003$

Table 13 illustrates the results for the even denser grid $N_y = N_t = 100$ with singular perturbation. The testing of the mollification algorithm without singular perturbation for the same grid size gives unacceptable data output and no corresponding table is provided.

Example	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
1	0.0031	0.0004	31.4977	1.0773
2	0.0031	0.0009	31.5609	2.7675
3	0.0031	0.0010	31.4842	15.0037
4	0.0031	0.0116	31.4842	34.2225
5	0.0146	0.0004	31.4847	1.0399

Table 13: The l_2 error norms: With singular perturbation

$N_x = 20,\ N_y = N_t = 100,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k = 6,\ \lambda = 0.001.$

Remarks:

a). As mentioned earlier, the performance of the algorithm is consistent when applied to different example models. Table 11 to 13 can also be taken as evidence of such a strong consistency of the algorithm. The absolute l_2 error norms for the temperature and heat flux functions are kept almost constant for different examples.

b). The performance of the combined singular perturbation scheme also depends on the perturbation constant λ. For different grid sizes, one has to

chose a proper value of λ to get the expected improvement. Table 14 contains the corresponding optimal values of λ for different grid sizes $N_y = N_t$.

$N_y = N_t$	λ
40	0.010
60	0.004
80	0.003
100	0.001

Table 14: The relationship between $N_y = N_t$ and λ.

4.4 Correction For Heat Flux

According to tables 12 and 13, the recovery is good for the temperature functions on the dense grid. However, the results for the heat-flux functions do not meet our accuracy standard with $N_y = N_t = 80$, and are not acceptable at all with $N_y = N_t = 100$. To look at the recovery functions more closely, we take a list of 2-surfaces with x fixed and observe the recovered heat-flux function. The results with $N_x = 20$, $N_y = N_t = 100$ are listed in table 15.

x	l_2 for $u_x(x,y,t)$	R. l_2 for $u_x(x,y,t)$
0.05	0.1004	0.0033
0.10	0.1321	0.0044
0.15	0.1473	0.0049
0.20	0.1621	0.0054
0.25	0.1776	0.0060
0.30	0.2166	0.0074
0.35	0.2683	0.0094
0.40	3.5576	0.1266
0.45	19.2814	0.7021
0.50	*******	4.2664

Table 15: The l_2 error norms with fixed x for heat flux. Example 1.

$N_x = 20,\ N_y = N_t = 100,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k = 6,\ \lambda = 0.001.$

One can see that the large l_2 error norm is mainly caused by the last couple of steps when marching in the x direction. This explains that, although the earlier recovered heat flux function has to be used to compute the temperature in the current marching step, the high error did not contribute to

the recovery of the temperature function. The wild last step values of the heat flux function are not used to compute the temperature.

Consequently, a natural way to fix the error in the recovery of the heat-flux function is to use the recovered temperature to re-compute the heat flux function at the same level. This will indeed substantially improve the accuracy for the heat flux function. The l_2 error norms, for the algorithm with correction, are in table 16 and 17. Table 16 is for the case of $N_y = N_t = 80$ and table 17 for $N_y = N_t = 100$. One can compare table 16 to 12 and table 17 to 13 to see the improvement.

Note that the formula used for the correction of the heat-flux functions is

$$W_{i,j}^n = \frac{V_{i+1,j}^n - V_{i-1,j}^n}{2h},$$

and the function values at the very end of the space marching ($x = 0.5$) are re-computed by extrapolation. Example 5 (with $u_{xx} = -200$ and $u_{yy} = 100$) is an exception. The correction scheme does not improve the l_2 norm for the heat flux function in this case.

Example	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
1	0.0021	0.0002	0.7596	0.0259
2	0.0015	0.0007	0.6358	0.0556
3	0.0013	0.0004	0.6202	0.2955
4	0.0013	0.0049	0.6184	0.6715
5	0.0186	0.0005	7.2472	0.2394

Table 16: The l_2 error norm: With correction

$N_x = 20,\ N_y = N_t = 80,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k = 8,\ \lambda = 0.003.$

Example	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
1	0.0031	0.0004	1.1888	0.0407
2	0.0031	0.0009	1.1147	0.0977
3	0.0031	0.0010	1.0949	0.5218
4	0.0031	0.0116	1.0937	1.1889
5	0.0146	0.0004	7.3011	0.2411

Table 17: The l_2 error norm: With correction.

$N_x = 20,\ N_y = N_t = 100,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k = 6,\ \lambda = 0.001.$

5 The "lost boundary"

The first step of the numerical procedure for the mollification scheme is to filter the high frequency components of the initial sample data collected on the 2-surface $x = 0$. To compute the value of the mollified function at a given point P at $x = 0$, all the initial sample data in the interior of the square centered at P and width 6δ are required. Since the initial sample data are collected on the unit square $[0,1] \times [0,1]$, the mollified function can only be computed on the set $(y,t) \in [3\delta, 1-3\delta] \times [3\delta, 1-3\delta]$. Therefore, the

domain of the recovered temperature and heat-flux functions is restricted to the "inner cube" $[0, 0.5] \times [3\delta, 1 - 3\delta] \times [3\delta, 1 - 3\delta]$.

It is unfortunate that, for most of the applications, the value of δ is usually not "very small". For example, corresponding to our choice of $\varepsilon = 0.005$, δ has to be at least $\frac{1}{15}$ to stabilize the numerical process. Consequently, instead of working on the whole cube $[0, 0.5] \times [0, 1] \times [0, 1]$, we have only worked, so far, in the "inner cube" $[0, 0.5] \times [0.2, 0.8] \times [0.2, 0.8]$. The domain of the recovered functions has been substantially shrinked.

In this section, we try to extend the domain of the recovered functions from the "inner cube" to the "whole cube" to resolve this "lost boundary" problem. Based on the fact that the t-direction is less ill-posed than the y-direction – the parabolic operator involves first partial derivatives with respect to time and second partials derivatives with respect to y – we propose completely different extension techniques for the individual directions. We first introduce the "dynamical radius of mollification" in the mollification of the initial data to extend the domain in the t-direction and, later, the space marching scheme is again called in to extend the computed solutions in the y-direction.

5.1 The differences between y and t-directions

In all the discussions so far, we have only consider the case $N_y = N_t$. However, since only the first derivative for the variable t is involved in the heat conduction equation, one could expect that the t-direction is "less ill-posed" than the y-direction. Hence, the stability of the algorithm should be more sensitive to the magnitude of N_y than that of N_t.

The numerical results for $N_y = 40$ and $N_t = 100$ are illustrated in table 18, and those for $N_y = 100$ and $N_t = 40$ in table 19. We use k_y and k_t to represent the grid refine constants for the y and t-directions respectively. We observe that the accuracy remains almost the same when N_t is increased, but drops rapidly when N_y is increased. For table 18 we set $\lambda = 0.01$, and for table 19, $\lambda = 0.002$. The choice of these values are optimal for the corresponding cases.

N_x	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
5	0.0085	0.0009	0.3381	0.0116
10	0.0046	0.0005	0.2921	0.0100
15	0.0030	0.0003	0.0951	0.0033
20	0.0023	0.0003	0.1180	0.0040
25	0.0018	0.0002	0.1757	0.0060

Table 18: The l_2 error norms. Example 1.

$N_y = 40,\ N_t = 100,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_y = 10,\ k_t = 6,\ \lambda = 0.010.$

N_x	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
5	0.0087	0.0010	0.4791	0.0163
10	0.0049	0.0006	1.0079	0.0343
15	0.0036	0.0004	6.5758	0.2238
20	0.0056	0.0006	92.0814	3.1322
25	0.0090	0.0010	*******	9.3764

Table 19: The l_2 error norms. Example 1.

$N_y = 100,\ N_t = 40, \varepsilon = 0.005,\ \delta = 0.0667,\ k_y = 6,\ k_t = 10,\ \lambda = 0.002.$

For the same reason, the data mollification in the y and t-directions can also be treated differently. For instance, instead of using

$$\rho_\delta(y,t) = \frac{1}{\pi\delta^2}\exp\left(-\frac{t^2+y^2}{\delta^2}\right),$$

we could use the more general Gaussian kernel

$$\rho(y,t,\delta_y,\delta_t) = \frac{1}{\pi\delta_y\delta_t}\exp\left[-\left(\frac{y^2}{\delta_y^2}+\frac{t^2}{\delta_t^2}\right)\right]$$

to mollify the two-dimensional initial data. The radius of mollification δ_t can be made smaller than its counterpart δ_y to obtain larger recovered regions for the solution functions.

Table 20 illustrates some results involving these ideas. Here $\delta_y = 0.0667$ and $\delta_t = 0.050$.

N_x	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
5	0.0119	0.0013	0.3114	0.0102
10	0.0063	0.0007	0.5655	0.0185
15	0.0041	0.0004	0.1422	0.0046
20	0.0030	0.0003	0.2303	0.0075
25	0.0025	0.0003	0.3552	0.0116

Table 20: The l_2 error norms for the general Gaussian Kernel. Example 1.

$N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta_y = 0.0667,\ \delta_t = 0.050,\ k = 10,\ \lambda = 0.010.$

5.2 The extension in t-direction

Up to now, in all the theoretical and numerical studies, the mollification radii δ_y and δ_t have always been regarded as constants. One observes that this assumption is, indeed, not essential for the mollification method. The data mollification is in fact a point by point process. For a given point (y_0, t_0) on the surface $x = 0$, one applies the Gaussian kernel to calculate $J(\delta_y, \delta_t, y_0, t_0)$ by using all the sample data in the square $[y_0-3\delta_y, y_0+3\delta_y]\times[t_0-3\delta_t, t_0+3\delta_t]$.

Practically, one can regard δ_y and δ_t as functions of (y_0, t_0). We will call a mollification of this kind (with non-constant mollification radii) "dynamical mollification". The former method with constant mollification radii will be referred as the "uniform mollification".

The change from the uniform mollification to the dynamical mollification makes the "lost boundary" problem somehow disappear. For a point (y_0, t_0) closer to the boundary of the square $[0,1] \times [0,1]$, smaller radii of mollification will be assigned to this point to perform the required integration to mollify the data functions there.

We observe that as (y_0, t_0) approaches the boundary, δ_y and/or δ_t get smaller. This means that the sample data closer to the boundary are less mollified, hence the high frequency components of the noise are probably not filtered enough by the mollification procedure. However, according to the fact that the diffusion problem is less ill-posed in the t-direction than in the y-direction, dynamical mollification could be used to extend the domain of the recovered functions in the t-direction and, therefore, help to recover part of the "lost boundary".

Let δ be the radius of mollification corresponding to uniform mollification. We define the dynamical mollification radius δ_y and δ_t on the set $(y_0, t_0) \in$

$[3\delta, 1-3\delta] \times [\,0, 1]$ by

$$\delta_y(y_0, t_0) = \delta$$

and

$$\delta_t(y_0, t_0) = \begin{cases} \delta, & (y_0, t_0) \in [3\delta, 1-3\delta] \times [3\delta, 1-3\delta], \\ \frac{1}{3}(1-t_0), & (y_0, t_0) \in [3\delta, 1-3\delta] \times [1-3\delta, 1], \\ \frac{1}{3}t_0, & (y_0, t_0) \in [3\delta, 1-3\delta] \times [0, 3\delta]. \end{cases}$$

Dynamical mollification with the given mollification radii $\delta_y(y_0, t_0), \delta_t(y_0, t_0)$ produces the mollified initial data on the subset $(y, t) \in [3\delta, 1-3\delta] \times [0, 1]$ of the surface $x = 0$.

At this point, we merely apply the space marching scheme in the x-direction to recover the temperature and heat-flux functions in the domain $(x, y, t) = [0, 0.5] \times [3\delta, 1-3\delta] \times [0, 1]$. By following this simple procedure, the part of the lost boundary in the t-direction is actually recovered.

We recall that for those points that are very close to the lines $t = 0$ and $t = 1$, the dynamical mollification approach does not filter the original sample data. Consequently, for the last three-points next to the line $t = 0$ and $t = 1$, a 4-th order extrapolation polynomial is introduced to create the filtered initial data.

Table 21 illustrates the l_2 error of the recovery in the t-direction by dynamical mollification for example 1. We again set $N_y = N_t = 40$. The mollification radius is $\delta = 0.0667$ and the noise level constant is set as $\varepsilon = 0.005$.

N_x	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
5	0.0137	0.0010	1.9262	0.0425
10	0.0080	0.0006	1.1449	0.0252
15	0.0048	0.0004	0.7141	0.0157
20	0.0042	0.0003	0.7852	0.0173
25	0.0066	0.0005	1.2742	0.0280

Table 21: The recovery in the t-direction. Example 1.

$N_t = N_y = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_t = k_y = 10,\ \lambda = 0.01.$

5.3 Recovery for the whole cube

We have extended the domain of the recovered functions from the inner cube $[0, 0.5] \times [3\delta, 1 - 3\delta] \times [3\delta, 1 - 3\delta]$ to $[0, 0.5] \times [3\delta, 1 - 3\delta] \times [0, 1]$. Dynamical

mollification can not be used in the y-direction because this direction is more ill-posed than the t-direction. However, starting from the 2-surface $(y, x, t) = \{3\delta\} \times [0, 0.5] \times [0, 1]$, we can now solve the inverse heat conduction problem by marching in the y-direction. We only need to march a short distance, 3δ, up to the surface $y = 0$ to recover one side of the lost boundary.

The formula for the implementation of the space marching scheme now is a little different from what was presented in section 2.3. We have to compute $u_y(x, y, t)$ together with $u_x(x, y, t)$. Setting $v = J_\delta u,\ w = v_x(x, y, t)$ and $f = v_y(x, y, t)$, the grid functions are now defined by

$$V^n_{i,j} = v(\xi_i, y_j, t_n), \qquad W^n_{i,j} = w(\xi_i, y_j, t_n), \qquad F^n_{i,j} = f(\xi_i, y_j, t_n).$$

The corresponding formulas for the space marching scheme read

$$F^n_{i,j} = \frac{V^n_{i,j} - V^n_{i,j-2}}{2\triangle y},$$

$$F^n_{i,j+1} = F^n_{i,j} + \frac{\triangle y}{2\triangle t}\left(V^{n+1}_{i,j} - V^{n-1}_{i,j}\right) - \frac{\triangle y}{2h}\left(W^n_{i+1,j} - W^n_{i-1,j}\right),$$

$$V^n_{i,j+1} = V^n_{i,j} + F^n_{i,j+1}\triangle y,$$

$$W^n_{i,j+1} = \frac{V^n_{i+1,j+1} - V^n_{i-1,j+1}}{2h}.$$

The discrete error norms of the recoveries for example 1 are presented in table 22. We again set $N_t = N_y = 40$, $\varepsilon = 0.005$ and $\delta = 0.667$. The

perturbation constant is $\lambda = 0.01$.

N_x	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for $u_x(x,y,t)$
5	0.0120	0.0006	2.8108	0.0459
10	0.0249	0.0014	2.7965	0.0456
15	0.0023	0.0001	1.8666	0.0304
20	0.0177	0.0010	1.4889	0.0243
25	0.0420	0.0023	2.9487	0.0480

Table 22: The recovery of the whole cube. Example 1.

$N_t = N_y = 40$, $\varepsilon = 0.005$, $\delta = 0.0667$, $k_t = k_y = 10$, $\lambda = 0.01$.

Figures 7 and 8 show the computed temperature and heat flux functions for example 1, at the 2-surface $x = 0.5$ for $0 \leq t, y \leq 1$. Figures 9 and 10 illustrate the qualitative behavior of the relative grid error functions for the temperature and the heat flux functions respectively for example 1. Normal sections of the previous discrete error functions at $y = 0.5$ and $y = 0.0$, are indicated in Figures 11 and 12, respectively.

Figure 7: Reconstructed temperature function at $x = 0.5$.

Whole cube, including the boundarry. Example 1.

$N_x = 20,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_y = k_t = 10,\ \lambda = 0.010.$

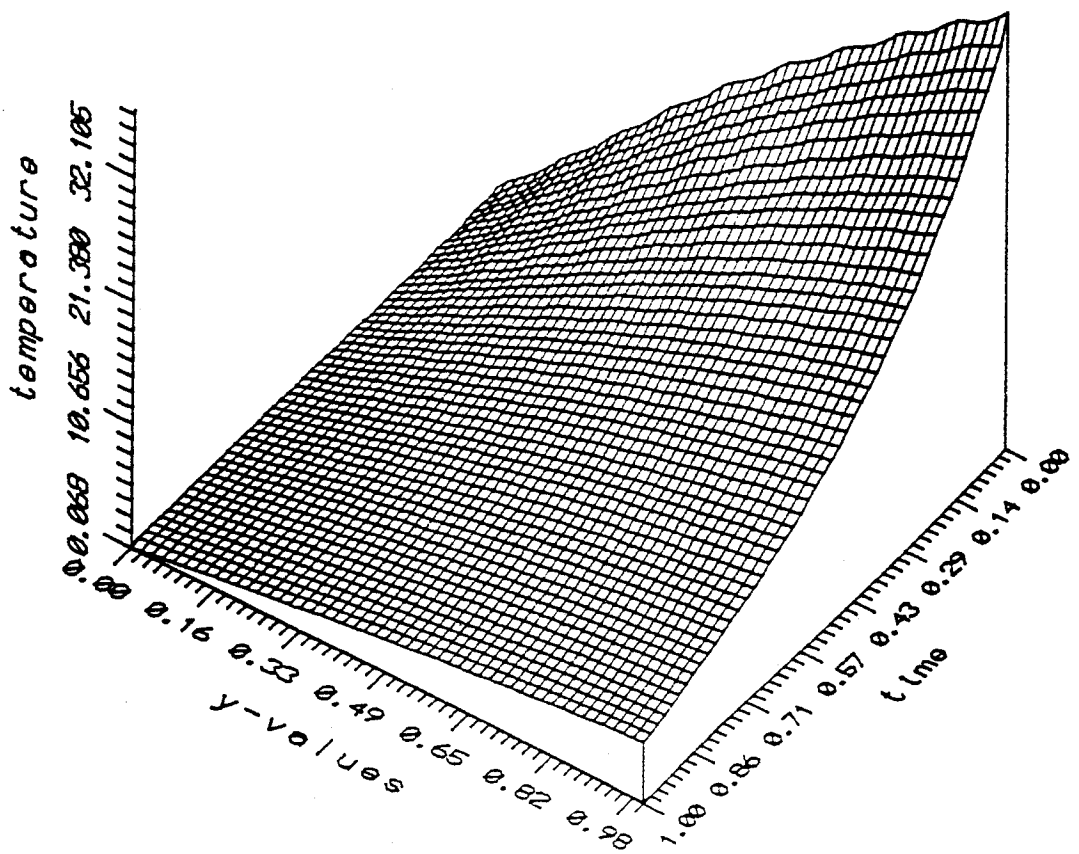


Figure 8: Reconstructed heat flux function at $x = 0.5$.

Whole cube, including the boundary. Example 1.

$N_x = 20,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_y = k_t = 10,\ \lambda = 0.010.$

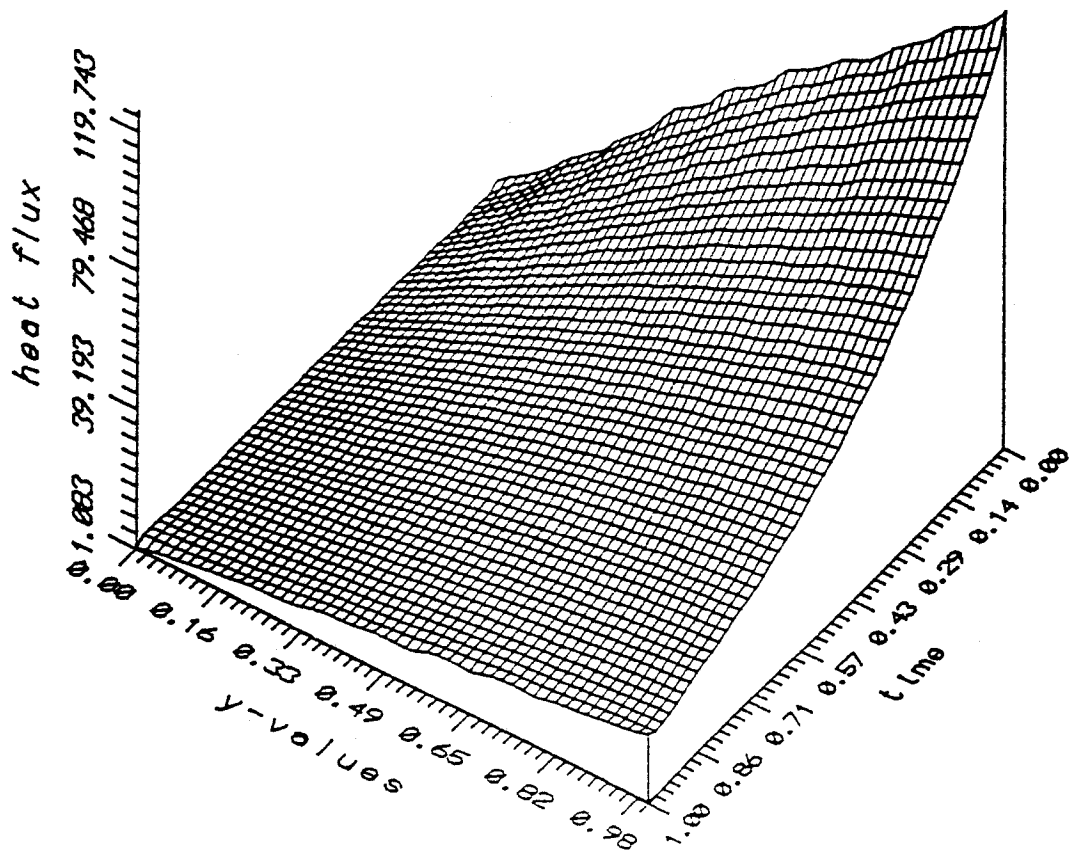


Figure 9: Relative errors for the temperature function at $x = 0.5$.

Whole cube, including the boundary. Example 1.

$N_x = 20,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_y = k_t = 10,\ \lambda = 0.010.$

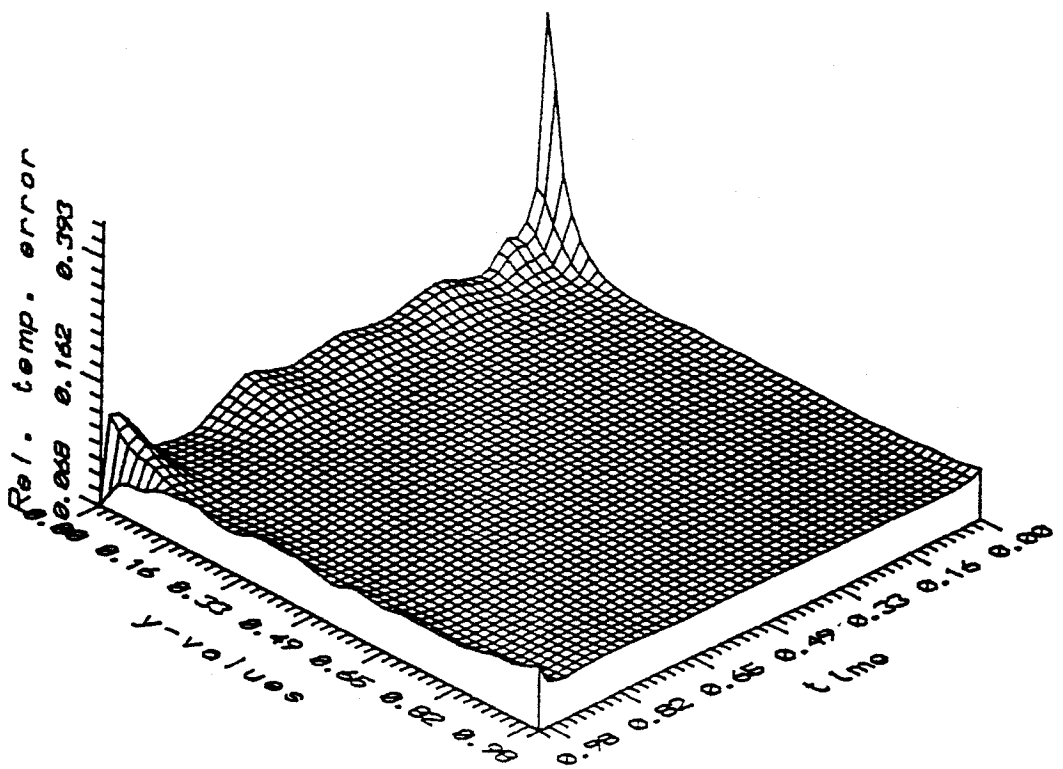


Figure 10: Relative errors for the heat flux function at $x = 0.5$.

Whole cube, including the boundary. Example 1.

$N_x = 20,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_y = k_t = 10,\ \lambda = 0.010.$

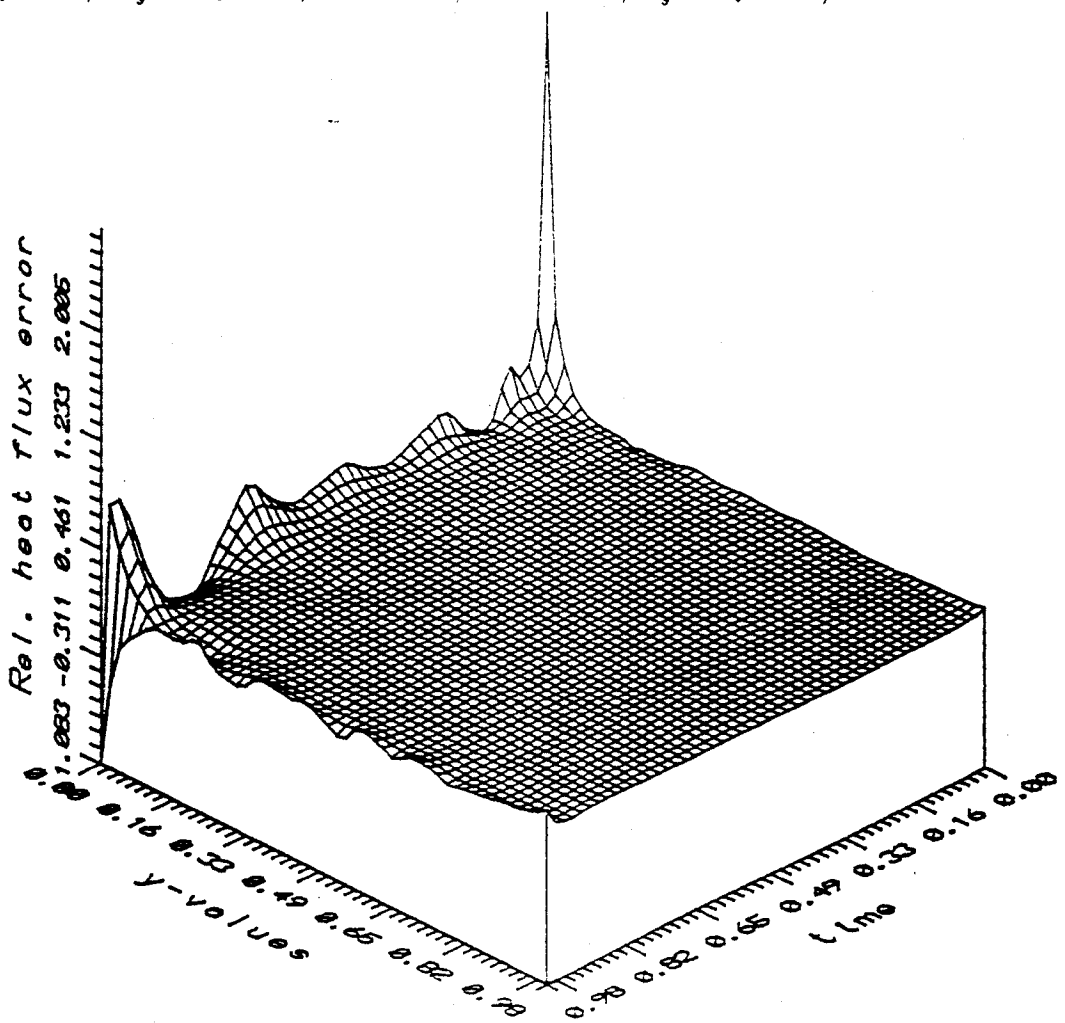


Figure 11: Transient relative errors for the temperature (*solid like*) and heat flux functions (*dashed like*) at $x = y = 0.5$. Example 1. $N_x = 20,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_y = k_t = 10,\ \lambda = 0.010.$

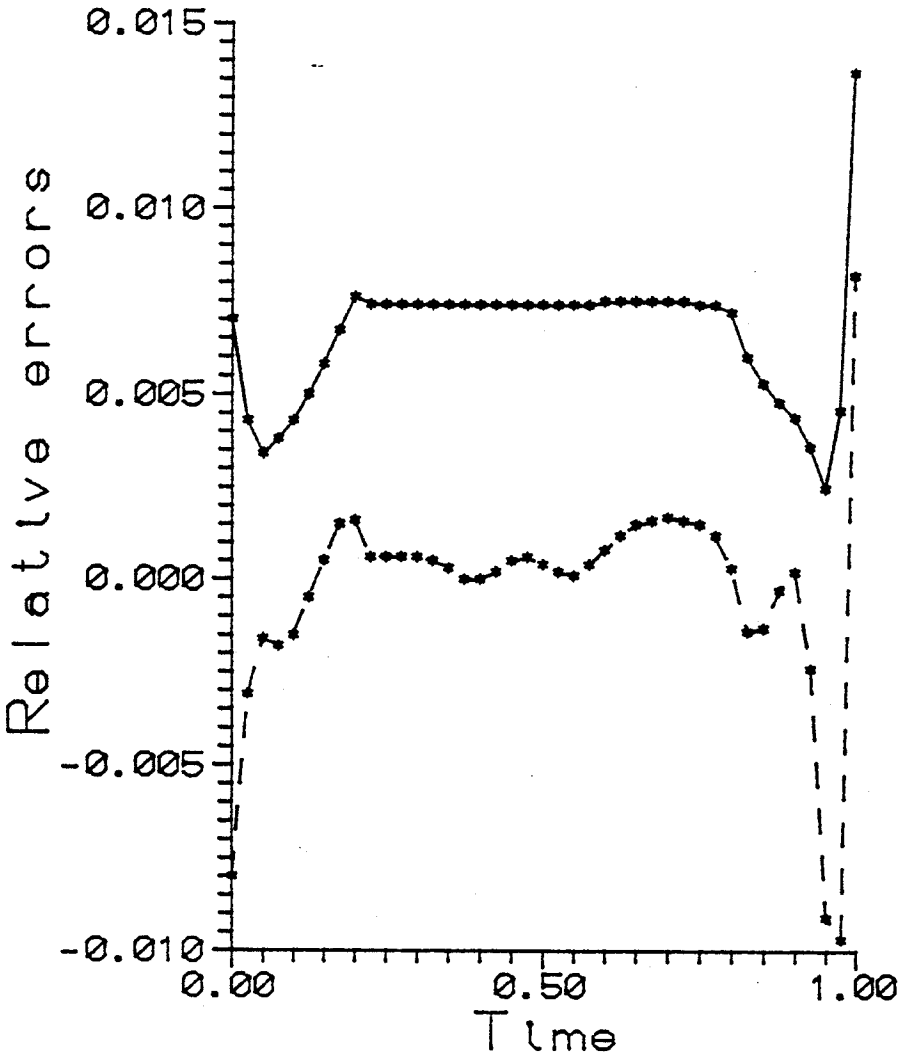
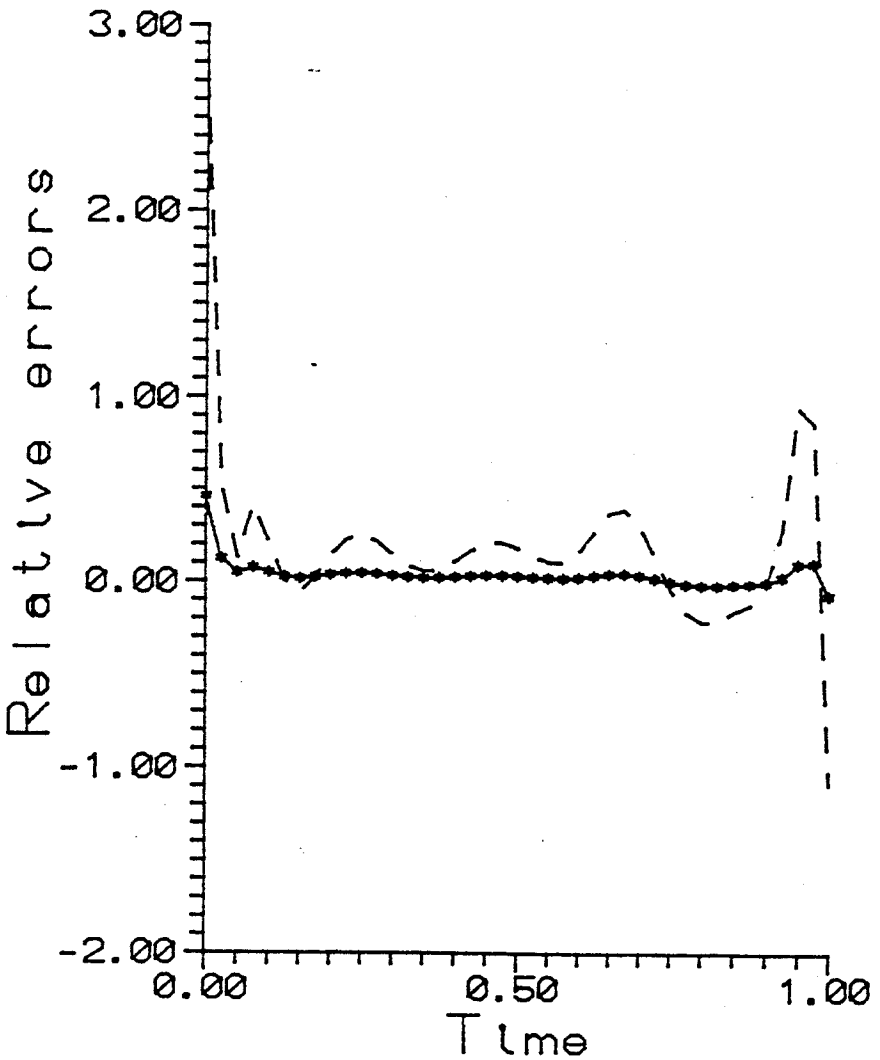


Figure 12: Transient relative errors for the temperature (*solid like*) and heat flux (*dashed like*) functions at $x = 0.5$, $y = 0.0$. Example 1. $N_x = 20,\ N_y = N_t = 40,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_y = k_t = 10,\ \lambda = 0.010$.



To show the consistency of the numerical process, we present the error norms of the recovered functions for all the examples in table 23. Note that the last step of the correction for the heat flux function is not applied for example 5, according to the previous result in section 4.4.

Ex.	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
1	0.0177	0.0010	1.4889	0.0243
2	0.0156	0.0016	1.5711	0.0476
3	0.0142	0.0027	1.3212	0.3538
4	0.0146	0.0278	1.3069	0.7185
*5	0.0366	0.0005	7.8020	0.1571

Table 23: The recovery of the whole cube

$N_t = N_y = 40,\ N_x = 20,\ \varepsilon = 0.005,\ \delta = 0.0667,\ k_t = k_y = 10.$

*without the last step of correction on the heat flux function.

Remarks:

(i). As expected, the accuracy of the recovery on the boundary is not as good as that on the inner cube. However, the algorithm performs in a stable and consistent manner for all the examples.

(ii). To explain the source of the larger error norms in all these tables, we drop the last **one** value of the recovered functions in the y and t-directions. The corresponding results are illustrated in table 24. We observe that the magnitude of the error norms drops by more than one-half.

N_x	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
5	0.0052	0.0003	2.4861	0.0442
10	0.0009	0.0001	1.5601	0.0277
15	0.0025	0.0001	0.9879	0.0175
20	0.0026	0.0002	0.8345	0.0148
25	0.0055	0.0003	0.9768	0.0173

Table 24: Example 1. The recovery on the shrinked cube

D: $[0, 0.5] \times [0.025, 0.975] \times [0.025, 0.975]$,

$N_x = N_y = 40, \varepsilon = 0.005, \delta = 0.0667, k_t = k_y = 10.$

(iii). $N_y = 40$ is the maximum value that we could take for the recovery of the lost boundary. However, we can refine the grids in the t-direction at least up to $N_t = 100$. The results are illustrated in table 25, for example 1.

N_x	l_2 for u	R. l_2 for u	l_2 for u_x	R. l_2 for u_x
5	0.0001	0.0000	2.5426	0.0439
10	0.0052	0.0003	1.5164	0.0262
15	0.0054	0.0003	1.0112	0.0174
20	0.0052	0.0003	0.8842	0.0152
25	0.0078	0.0005	1.0074	0.0174

Table 25: The recovery with fine grid in the t-direction. Example 1.

Shrinked cube D: $[0, 0.5] \times [0.025, 0.975] \times [0.025, 0.975]$,

$N_t = 100, N_y = 40, \varepsilon = 0.005, \delta = 0.0667, k_t = 6, k_y = 10.$

6 Summary and Conclusions

A combined mollification-singular perturbation regularization technique is derived and investigated. The method is implemented as a space-marching-fully-explicit finite differences algorithm, providing computational flexibility and overall efficiency.

Several test cases are investigated for a finite three-dimensional cube in the (x, y, t) space when the temperature and heat flux data functions are measured only on the square $[0, 1] \times [0, 1]$ in the (y, t) space at $x = 0$. The required temperature and heat flux functions are numerically approximated on the cube $\Omega = [0, 0.5] \times [0, 1] \times [0, 1]$ of the (x, y, t) space.

A number of parameters are varied, including radii of mollification, amount of noise in the data, step marching and grid sizes. A reliable set of parameters values is experimentally determined to guarantee the accuracy and good stability of the algorithm.

A general conclusion is that the numerical procedure presented in this work remains a viable procedure for the numerical solution of the 2-D IHCP in bounded domains.

References

[1] B. R. Bass and L. J. Ott, "A finite element formulation of the two-dimensional inverse heat conduction problem", Adv. Comput. Technol., Vol. 2, 1980, pp. 238-248.

[2] J. Baumeister and H. J. Reinhardt, "On the approximate solution of a two-dimensional inverse heat conduction problem", Inverse and Ill-posed Problems, H. G. Engl and C. W. Groetsch, (Eds.), Academic Press, Orlando, Florida, 1987, pp. 325-344.

[3] J. V. Beck, "Nonlinear estimation applied to the nonlinear inverse heat conduction problem", Int. J. Heat Mass Transfer, Vol. 13, 1970, pp. 703-716.

[4] L. Guo and D. A. Murio, "A mollified space-marching finite difference algorithm for the two-dimensional inverse heat conduction problem with slab symmetry", Inverse Problems, Vol. 7, 1991, pp. 247-259.

[5] M. Imber, "Temperature extrapolation mechanism for two-dimensional heat flow", AIAA J., Vol. 12, 1974, pp. 1089-1093.

[6] D. A. Murio, *The Mollification Method and the Numerical Solution of Ill-Posed Problems,* John Wiley & Sons, Interscience, New York, New York, 1993.

[7] T. Yoshimura and K. Ituka, "Inverse heat conduction problem by finite element formulation", Int. J. Systems Sci., Vol. 16, 1985, pp. 1365-1376.

[8] N. Zabaras and J. C. Liu, "An analysis of two-dimensional linear inverse heat transfer problems using an integral method", Numer. Heat Transfer, Vol. 13, 1988, pp. 527-533.