We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter

A Numerical Approach to Solving an Inverse Heat Conduction Problem Using the Levenberg-Marquardt Algorithm

Tao Min, Xing Chen, Yao Sun and Qiang Huang

Abstract

This chapter is intended to provide a numerical algorithm involving the combined use of the Levenberg-Marquardt algorithm and the Galerkin finite element method for estimating the diffusion coefficient in an inverse heat conduction problem (IHCP). In the present study, the functional form of the diffusion coefficient is an unknown priori. The unknown diffusion coefficient is approximated by the polynomial form and the present numerical algorithm is employed to find the solution. Numerical experiments are presented to show the efficiency of the proposed method.

Keywords: parabolic equation, inverse problem, Levenberg-Marquardt

1. Introduction

The numerical solution of the inverse heat conduction problem (IHCP) requires to determine diffusion coefficient from an additional information. Inverse heat conduction problems have many applications in various branches of science and engineering, mechanical and chemical engineers, mathematicians and specialists in many other sciences branches are interested in inverse problems, each with different application in mind [1–15].

In this work, we propose an algorithm for numerical solving an inverse heat conduction problem. The algorithm is based on the Galerkin finite element method and Levenberg-Marquardt algorithm [16–17] in conjunction with the least-squares scheme. It is assumed that no prior information is available on the functional form of the unknown diffusion coefficient in the present study, thus, it is classified as the function estimation in inverse calculation. Run the numerical algorithm to solve the unknown diffusion coefficient which is approximated by the polynomial form. The Levenberg-Marquardt optimization is adopted to modify the estimated values.

The plan of this paper is as follows: in Section 2, we formulate a one-dimensional IHCP. In Section 3, the numerical algorithm is derived. Calculation of sensitivity coefficients will be discussed in Section 4. In order to discuss on some numerical aspects, two examples are given in Section 5. Section 6 ends this paper with a brief discussion on some numerical aspects.

IntechOpen

2. Description of the problem

The mathematical formulation of a one-dimensional heat conduction problem is given as follows:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left[q(x)\frac{\partial u}{\partial x}\right] + f(x,t), \quad (x,t) \in (0,\mathrm{L}) \times (0,\mathrm{T}], \tag{1}$$

with the initial condition

$$u(\mathrm{x},0) = \mathrm{u}_0(x), \quad 0 \le x \le L, \tag{2}$$

and Dirichlet boundary conditions

$$u(0,\mathrm{t}) = \mathrm{g}_1(t), \quad 0 \le t \le T, \tag{3}$$

$$u(1,\mathrm{t}) = \mathrm{g}_2(t), \quad 0 \le t \le T, \tag{4}$$

where $\mathrm{f}(\mathrm{x},\mathrm{t})$, $u_0(\mathrm{x})$, $g_1(\mathrm{t})$, $g_2(\mathrm{t})$ and $q(\mathrm{x})$ are continuous known functions. We consider the problem (1)–(4) as a direct problem. As we all know, if $u_0(\mathrm{x})$, $g_1(\mathrm{t})$, $g_2(\mathrm{t})$ are continuous functions and $q(\mathrm{x})$ is known, the problem (1)–(4) has a unique solution.

For the inverse problem, the diffusion coefficient $q(\mathrm{x})$ is regarded as being unknown. In addition, an overspecified condition is also considered available. To estimate the unknown coefficient $q(\mathrm{x})$, the additional information on the boundary $x = x_0$, $0 < x_0 < L$ is required. Let the $u(\mathrm{x},\mathrm{t})$ taken at $x = x_0$ over the time period $[0,\mathrm{T}]$ be denoted by

$$\mathrm{u}(x_0,t) = \mathrm{g}(t) \quad 0 \le t \le T. \tag{5}$$

It is evident that for an unknown function $q(\mathrm{x})$, the problem (1)–(4) is underdetermined and we are forced to impose additional information (5) to provide a unique solution pair $(\mathrm{u}(\mathrm{x},\mathrm{t}),\mathrm{q}(\mathrm{x}))$ to the inverse problem (1)–(5).

We note that the measured overspecified condition $\mathrm{u}(\mathrm{x}_0,\mathrm{t}) = \mathrm{g}(t)$ should contain measurement errors. Therefore the inverse problem can be stated as follows: by utilizing the above-mentioned measured data, estimate the unknown function $q(\mathrm{x})$.

In this work the polynomial form is proposed for the unknown function $q(\mathrm{x})$ before performing the inverse calculation. Therefore $q(\mathrm{x})$ approximated as

$$q(x) \approx \hat{q(\mathrm{x})} = \mathrm{p}_1 + p_2 x + p_3 x^2 + \cdots + p_{m+1} x^m, \tag{6}$$

where $p_1, p_2, \cdots, p_{m+1}$ are constants which remain to be determined simultaneously. The unknown coefficients $p_1, p_2, \cdots, p_{m+1}$ can be determined by using least squares method. The error in the estimate

$$F(p_1, p_2, \ldots, p_{m+1}) = \sum_{i=1}^{n} \left[u(x_0, t_i, p_1, p_2, \ldots, p_{m+1}) - g(\mathrm{t}_i)\right]^2, \tag{7}$$

is to be minimized. Here, $u(x_0, t_i, p_1, p_2, \ldots, p_{m+1})$ are the calculated results. These quantities are determined from the solution of the direct problem which is given previously by using an approximated $\hat{q(\mathrm{x})}$ for the exact $q(\mathrm{x})$. The estimated values of p_j, $j = 1, 2, \cdots, m+1$ are determined until the value of $F(p_1, p_2, \ldots, p_{m+1})$ is minimum. Such a norm can be written as

$$F(\mathbf{P}) = [\mathbf{U}(\mathbf{P})\text{-}\mathbf{G}]^T[\mathbf{U}(\mathbf{P})\text{-}\mathbf{G}], \tag{8}$$

where $\mathbf{P}^T = [p_1, p_2, \cdots, p_{m+1}]$ denotes the vector of unknown parameters and the superscript T above denotes transpose. The vector $[\mathbf{U}(\mathbf{P})\text{-}\mathbf{G}]^T$ is given by

$$[\mathbf{U}(\mathbf{P})\text{-}\mathbf{G}]^T = [u(x_0, t_1, \mathbf{p}) - g(t_1), u(x_0, t_2, \mathbf{p}) - g(t_2), \cdots, u(x_0, t_n, \mathbf{p}) - g(t_n)]. \tag{9}$$

$F(\mathbf{P})$ is real-valued bounded function defined on a closed bounded domain $D \subset R^{m+1}$. The function $F(\mathbf{P})$ may have many local minimum in D, but it has only one global minimum. When $F(\mathbf{P})$ and D have some attractive properties, for instance, $F(\mathbf{P})$ is a differentiable concave function and D is a convex region, then a local maximum and problem can be solved explicitly by mathematical programming methods.

3. Overview of the Levenberg-Marquardt method

The Levenberg-Marquardt method, originally devised for application to nonlinear parameter estimation problems, has also been successfully applied to the solution of linear ill-conditioned problems. Such a method was first derived by Levenberg (1944) by modifying the ordinary least-squares norm. Later Marquardt (1963) derived basically the same technique by using a different approach. Marquardt's intention was to obtain a method that would tend to the Gauss method in the neighborhood of the minimum of the ordinary least-squares norm, and would tend to the steepest descent method in the neighborhood of the initial guess used for the iterative procedure.

To minimize the least squares norm (8), we need to equate to zero the derivatives of $F(\mathbf{P})$ with respect to each of the unknown parameters $[p_1, p_2, \cdots, p_{m+1}]$, that is

$$\frac{\partial F(\mathbf{P})}{\partial p_1} = \frac{\partial F(\mathbf{P})}{\partial p_2} = \cdots = \frac{\partial F(\mathbf{p})}{\partial p_{m+1}} = 0. \tag{10}$$

Let us introduce the sensitivity or Jacobian matrix, as follows:

$$\mathbf{J}(\mathbf{P}) = \left[\frac{\partial \mathbf{U}^T(\mathbf{P})}{\partial \mathbf{P}}\right]^T = \begin{bmatrix} u_{p_1}(x_0, t_1, \mathbf{p}) & u_{p_2}(x_0, t_1, \mathbf{p}) & \cdots & u_{p_{m+1}}(x_0, t_1, \mathbf{p}) \\ u_{p_1}(x_0, t_2, \mathbf{p}) & u_{p_2}(x_0, t_2, \mathbf{p}) & \cdots & u_{p_{m+1}}(x_0, t_2, \mathbf{p}) \\ \cdots & & & \\ u_{p_1}(x_0, t_n, \mathbf{p}) & u_{p_2}(x_0, t_n, \mathbf{p}) & \cdots & u_{p_{m+1}}(x_0, t_n, \mathbf{p}) \end{bmatrix}, \tag{11}$$

$$\text{or} \quad J_{ij} = u_{p_j}(x_0, t_i, \mathbf{p}) = \frac{\partial u(x_0, t_i, \mathbf{p})}{\partial p_j}, i = 1, 2, \ldots n, j = 1, 2, \ldots m+1. \tag{12}$$

The elements of the sensitivity matrix are called the sensitivity coefficients, the results of differentiation (10) can be written down as follows:

$$-2\mathbf{J}^T(\mathbf{P})[\mathbf{U}(\mathbf{P}) - \mathbf{G}] = 0. \tag{13}$$

For linear inverse problem the sensitivity matrix is not a function of the unknown parameters. The Eq. (13) can be solved then in explicit form:

$$\mathbf{P} = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T\mathbf{G}. \tag{14}$$

In the case of a nonlinear inverse problem, the matrix **J** has some functional dependence on the vector **p**. The solution of Eq. (13) requires an iterative procedure, which is obtained by linearizing the vector $\mathbf{U}(\mathbf{P})$ with a Taylor series expansion around the current solution at iteration k. Such a linearization is given by

$$\mathbf{U}(\mathbf{P}) = \mathbf{U}\left(\mathbf{P}^k\right) + \mathbf{J}^k\left(\mathbf{P}\text{-}\mathbf{P}^k\right), \tag{15}$$

where $\mathbf{U}\left(\mathbf{P}^k\right)$ and $\mathbf{J}^k$ are the estimated temperatures and the sensitivity matrix evaluated at iteration k, respectively. Eq. (15) is substituted into (14) and the resulting expression is rearranged to yield the following iterative procedure to obtain the vector of unknown parameters **P**:

$$\mathbf{P}^{k+1} = \mathbf{P}^k + \left[\left(\mathbf{J}^k\right)^T\mathbf{J}^k\right]^{-1}\left(\mathbf{J}^{\mathbf{k}}\right)^T\left[\mathbf{G}\text{-}\mathbf{U}\left(\mathbf{P}^k\right)\right]. \tag{16}$$

The iterative procedure given by Eq. (16) is called the Gauss method. Such method is actually an approximation for the Newton (or Newton-Raphson) method. We note that Eq. (14), as well as the implementation of the iterative procedure given by Eq. (16), require the matrix $\mathbf{J}^T\mathbf{J}$ to be nonsingular, or

$$\left|\mathbf{J}^T\mathbf{J}\right| \neq 0, \tag{17}$$

where $|\cdot|$ is the determinant.

Formula (17) gives the so called identifiability condition, that is, if the determinant of $\mathbf{J}^T\mathbf{J}$ is zero, or even very small, the parameters p_j, for $j = 1, 2, \cdots, \mathrm{m} + 1$, cannot be determined by using the iterative procedure of Eq. (16).

Problems satisfying $\left|\mathbf{J}^T\mathbf{J}\right| \approx 0$ are denoted ill-conditioned. Inverse heat transfer problems are generally very ill-conditioned, especially near the initial guess used for the unknown parameters, creating difficulties in the application of Eqs. (14) or (16). The Levenberg-Marquardt method alleviates such difficulties by utilizing an iterative procedure in the form:

$$\mathbf{P}^{k+1} = \mathbf{P}^k + \left[\left(\mathbf{J}^k\right)^T\mathbf{J}^k + \mu^k\Omega^k\right]^{-1}\left(\mathbf{J}^k\right)^T\left[\mathbf{G}\text{-}\mathbf{U}\left(\mathbf{P}^k\right)\right], \tag{18}$$

where μ^k is a positive scalar named damping parameter and Ω^k is a diagonal matrix.

The purpose of the matrix term $\mu^k\Omega^k$ is to damp oscillations and instabilities due to the ill-conditioned character of the problem, by making its components large as compared to those of $\mathbf{J}^T\mathbf{J}$ if necessary. μ^k is made large in the beginning of the iterations, since the problem is generally ill-conditioned in the region around the initial guess used for iterative procedure, which can be quite far from the exact parameters. With such an approach, the matrix $\mathbf{J}^T\mathbf{J}$ is not required to be nonsingular in the beginning of iterations and the Levenberg-Marquardt method tends to the steepest descent method, that is, a very small step is taken in the negative gradient direction. The parameter μ^k is then gradually reduced as the iteration procedure advances to the solution of the parameter estimation problem, and then the Levenberg-Marquardt method tends to the Gauss method given by (16). The following criteria were suggested in literature [13] to stop the iterative procedure of the Levenberg-Marquardt method given by Eq. (18):

$$F\left(\mathbf{p}^{k+1}\right) < \varepsilon_1, \tag{19}$$

$$\left\| \left(\mathbf{J}^k\right)\left[\mathbf{G}\text{-}\mathbf{U}\left(\mathbf{p}^k\right)\right] \right\| < \varepsilon_2, \tag{20}$$

$$\left\| \mathbf{p}^{k+1}\text{-}\mathbf{p}^k \right\| < \varepsilon_3, \tag{21}$$

where $\varepsilon_1, \varepsilon_2$ and ε_3 are user prescribed tolerances and $\|\cdot\|$ denotes the Euclidean norm. The criterion given by Eq. (19) tests if the least squares norm is sufficiently small, which is expected in the neighborhood of the solution for the problem. Similarly, Eq. (20) checks if the norm of the gradient of $F(\mathbf{p})$ is sufficiently small, since it is expected to vanish at the point where $F(\mathbf{p})$ is minimum. The last criterion given by Eq. (21) results from the fact that changes in the vector of parameters are very small when the method has converged. Generally, these three stopping criteria need to be tested and the iterative procedure of the Levenberg-Marquardt method is stopped if any of them is satisfied.

Different versions of the Levenberg-Marquardt method can be found in the literature, depending on the choice of the diagonal matrix Ω^k and on the form chosen for the variation of the damping parameter μ^k. In this paper, we choose the Ω^k as

$$\Omega^k = diag\left[\left(\mathbf{J}^k\right)^T\mathbf{J}^k\right]. \tag{22}$$

Suppose that the vector of temperature measurements $\mathbf{G} = [g(t_1), g(t_2), \cdots, g(t_n)]$ are given at times t_i, $i = 1, 2, \cdots, n$ and an initial guess $\mathbf{P}^0$ is available for the vector of unknown parameters $\mathbf{P}$. Choose a value for μ^0, say, $\mu^0 = 0.001$ and $k = 0$. Then,

Step 1. Solve the direct problem (1)–(4) with the available estimate $\mathbf{P}^k$ in order to obtain the vector $\mathbf{U}\left(\mathbf{P}^k\right) = \left[u\left(x_0, t_1, \mathbf{p}^k\right), u\left(x_0, t_2, \mathbf{p}^k\right), \cdots, u\left(x_0, t_n, \mathbf{p}^k\right)\right]$.

Step 2. Compute $F\left(\mathbf{P}^k\right)$ from the Eq. (8).

Step 3. Compute the sensitivity matrix $\mathbf{J}^k$ from (12) and then the matrix Ω^k from (22), by using the current value of $\mathbf{P}^k$.

Step 4. Solve the following linear system of algebraic equations, obtained from (18): $\left[\left(\mathbf{J}^k\right)^T\mathbf{J}^k + \mu^k\Omega^k\right]\Delta\mathbf{P}^k = \left(\mathbf{J}^k\right)^T\left[\mathbf{G}\text{-}\mathbf{U}\left(\mathbf{P}^k\right)\right]$ in order to compute $\Delta\mathbf{P}^k = \mathbf{P}^{k+1}\text{-}\mathbf{P}^k$.

Step 5. Compute the new estimate $\mathbf{P}^{k+1}$ as $\mathbf{P}^{k+1} = \mathbf{P}^k + \Delta\mathbf{P}^k$.

Step 6. Solve the exact problem (1)–(4) with the new estimate $\mathbf{P}^{k+1}$ in order to find $U\left(\mathbf{P}^{k+1}\right)$. Then compute $F\left(\mathbf{P}^{k+1}\right)$.

Step 7. If $F\left(\mathbf{P}^{k+1}\right) \geq F\left(\mathbf{P}^k\right)$ replace μ^k by $10\mu^k$ and return to step 4.

Step 8. If $F\left(\mathbf{P}^{k+1}\right) \leq F\left(\mathbf{P}^k\right)$, accept the new estimate $\mathbf{P}^{k+1}$ and replace μ^k by $0.1\mu^k$.

Step 9. Check the stopping criteria given by (19). Stop the iterative procedure if any of them is satisfied; otherwise, replace k by $k + 1$ and return to step 3.

4. Calculation of sensitivity coefficients

Generally, there have two approaches for determining the gradient; the first is a discretize-then-differentiate approach and the second is a differentiate-then-discretize approach.

The first approach is to approximate the gradient of the functional by a finite difference quotient approximation, but in general, we cannot determine the sensitivities exactly, so this method may led to larger error.

Here we intend to use differentiate-then-discretize approach which we refer to as the sensitivity equation method. This method can be determined more efficiently with the help of the sensitivities

$$u_k = \frac{\partial u}{\partial p_k}, k = 1, 2, \cdots m+1. \tag{23}$$

We first differentiate the flow system (1)–(4) with respect to each of the design parameters $[p_1, p_2, ..., p_{m+1}]$, to obtain the $m+1$ continuous sensitivity systems: for $k = 1, 2, \cdots, m+1$

$$\begin{cases} \dfrac{\partial u_k}{\partial t} = \dfrac{\partial}{\partial x}\left[\left(p_1 + p_2 x + \cdots + p_{m+1}x^m\right)\dfrac{\partial u_k}{\partial x} + x^{k-1}\dfrac{\partial u}{\partial x}\right] \\ u_k(x,0) = 0 \\ u_k(0,t) = 0 \\ u_k(L,t) = 0 \end{cases}. \tag{24}$$

There have $(m+2)$ equations, we can make them in one system equation and use the finite element methods to solve the system of equation. Here, we give the vector form of the equation as follow:

$$(\mathbf{P1})\begin{cases} \dfrac{\partial \vec{\mathbf{U}}}{\partial t} + \nabla \cdot \vec{\mathbf{\Gamma}} = \vec{\mathbf{F}} \\ \vec{\mathbf{U}}(x,0) = \vec{\mathbf{U}}_0(x), \\ \vec{\mathbf{U}}(0,t) = \vec{\mathbf{G}}_1(t) \\ \vec{\mathbf{U}}(L,t) = \vec{\mathbf{G}}_2(t) \end{cases} \tag{25}$$

where

$$\vec{\mathbf{U}} = \begin{bmatrix} u \\ u_1 \\ u_2 \\ \cdots \\ u_{m+1} \end{bmatrix}, \vec{\mathbf{\Gamma}} = \begin{bmatrix} -\left(p_1 + p_2 x + \cdots + p_{m+1}x^m\right)\dfrac{\partial u}{\partial x} \\ -\left(p_1 + p_2 x + \cdots + p_{m+1}x^m\right)\dfrac{\partial u_1}{\partial x} - \left(\dfrac{\partial u}{\partial x}\right) \\ -\left(p_1 + p_2 x + \cdots + p_{m+1}x^m\right)\dfrac{\partial u_2}{\partial x} - x\left(\dfrac{\partial u}{\partial x}\right) \\ \cdots \\ -\left(p_1 + p_2 x + \cdots + p_{m+1}x^m\right)\dfrac{\partial u_{m+1}}{\partial x} - \left(x^m\dfrac{\partial u}{\partial x}\right) \end{bmatrix}, \vec{\mathbf{F}} = \begin{bmatrix} f(\mathrm{x},\mathrm{t}) \\ 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix}. \tag{26}$$

$$\vec{\mathbf{U}}_0(x) = [\mathrm{u}_0(x), 0, 0, \cdots 0]^T, \vec{\mathbf{G}}_1(t) = [g_1(t), 0, 0, \cdots 0]^T, \vec{\mathbf{G}}_2(t) = [g_2(t), 0, 0, \cdots 0]^T. \tag{27}$$

We use the Galerkin finite element method approximation for discretizing problem (25). For this, we multiply the Eq. (25) by a test function $v : [0, L] \to R$, $v \in V_0 := H_0^1(0, L)$ and integrate the obtained equation in space form 0 to L. We obtain the following equation:

$$\int_0^{\mathrm{L}} \frac{\partial \mathbf{U}(x,t)}{\partial t} \cdot v(x)dx - \int_0^L \nabla \mathbf{\Gamma} \cdot v(x)dx = \int_0^L \mathbf{F}(x,t) \cdot v(x)dx, \tag{28}$$

integrating by parts gives

$$\int_0^L \nabla \mathbf{\Gamma} \cdot v(x)dx = (\mathbf{\Gamma} \cdot v(x))\big|_0^L - \int_0^L \mathbf{\Gamma} \cdot \frac{\partial v(x,t)}{\partial x}dx. \tag{29}$$

We can change the first derivative in time and the integral. We have $v(0) = 0 = v(L)$, because $v \in V_0$. This leads to an equivalent problem to (**P**1): $\forall t>0$, find $\mathbf{U}(x,t)$ satisfying

$$\frac{d}{dt}\int_0^{\mathrm{L}} \mathbf{U}(x,t)\cdot v(x)dx + \int_0^L \mathbf{\Gamma}\cdot\frac{\partial v(x,t)}{\partial x}dx = \int_0^L \mathbf{F}(x,t)\cdot v(x)dx, \tag{30}$$

for all $v \in V_0 := H_0^1(0,L)$. To simplify the notation we use the scalar product in $L^2(0,L)$

$$(f,g) = \int_0^L f(x)\cdot g(x)dx. \tag{31}$$

We also can define the following bilinear form:

$$a(\mathbf{U},v) = \int_0^L \mathbf{\Gamma}\cdot\frac{\partial v(x,t)}{\partial x}dx = \begin{cases} \int_0^L -\left(p_1+p_2x+\cdots+p_{m+1}x^m\right)\frac{\partial u}{\partial x}\frac{\partial v}{\partial x}dx \\ \int_0^L\left(-\left(p_1+p_2x+\cdots+p_{m+1}x^m\right)\frac{\partial u_1}{\partial x}\frac{\partial v}{\partial x}-\frac{\partial u}{\partial x}\frac{\partial v}{\partial x}\right)dx \\ \int_0^L\left(-\left(p_1+p_2x+\cdots+p_{m+1}x^m\right)\frac{\partial u_2}{\partial x}\frac{\partial v}{\partial x}-x\frac{\partial u}{\partial x}\frac{\partial v}{\partial x}\right)dx \\ \cdots \\ \int_0^L\left(-\left(p_1+p_2x+\cdots+p_{m+1}x^m\right)\frac{\partial u_{m+1}}{\partial x}\frac{\partial v}{\partial x}-x^{m+1}\frac{\partial u}{\partial x}\frac{\partial v}{\partial x}\right)dx \end{cases}. \tag{32}$$

Finally, we obtain with this notations the weak problem of (**P**1):

$$(\mathbf{P}2)\begin{cases} \frac{d}{dt}(\mathbf{U},v)_{L^2} + a(\mathbf{U},v) = (\mathbf{F},v)_{L^2} \\ \mathbf{U}(x,0) = \mathbf{U}_0(x) \\ \mathbf{U}(0,t) = \mathbf{G}_1(t) \\ \mathbf{U}(L,t) = \mathbf{G}_2(t) \end{cases}. \tag{33}$$

4.1 Space-discretization with the Galerkin method

In this section, we search a semi-discrete approximation of the weak problem (**P**2), using the Galerkin finite element method. This leads to a first order Cauchy-problem in time.

Let V_h be a N_x+1 dimensional subspace of V and $V_{0,h} = V_h \cap V_0$. Then the following problem is an approximation of the weak problem, find $u_h, u_{1,h}, u_{2,h}\cdots u_{m+1.h} \in V_h$ that satisfies:

$$\begin{cases} \frac{d}{dt}(\mathbf{U}_h,v_h) + a(\mathbf{U}_h,v_h) = (\mathbf{F},v_h) \\ \mathbf{U}_h(\mathrm{x},0) = \mathbf{U}_{0,h}(x) \\ \mathbf{U}_h(0,t) = \mathbf{G}_1(t) \\ \mathbf{U}_h(L,t) = \mathbf{G}_2(t) \end{cases}, \tag{34}$$

for all $v_h \in V_{0,h}$. where $\mathbf{U}_h = [u_h, u_{1,h}, u_{2,h}\cdots u_{m+1.h}]^T$.

The choice of V_h is completely arbitrary. So we can choose it the way that for later treatment, it will be as easy as possible. For example, we subdivide the interval $[0, L]$ into partitions of equal distances h:

$$0 = a_1 < a_2 < \cdots < a_{N_x} < a_{N_x+1} = L, , a_i = (i-1) \cdot h \tag{35}$$

$$V_h = \left\{ v_h \in C^0[0, L] : v_h\big|_{[a_i, a_{i+1}]} \in \mathrm{P}_1, \forall i = 1 \cdots N_x \right\}, \tag{36}$$

$$V_{0,h} = \{ v_h \in V_h : v_h(0) = v_h(L) = 0 \}. \tag{37}$$

Note, that the finite dimension allows us to build a finite base for the corresponding space. In the case of $V_{0,h}$, we have: $\{\varphi_i\}_{i=2}^{N_x}$ where $\forall i = 2 \cdots N_x$.

$$\varphi_i(x) = \begin{cases} 0 & x \in [a_0, a_{i-1}] \\ \dfrac{x}{h} - (i-2) & x \in [a_{i-1}, a_i] \\ i - \dfrac{x}{h} & x \in [a_i, a_{i+1}] \\ 0 & x \in [a_{i+1}, a_{N_x+1}] \end{cases}, \tag{38}$$

while we add for V_h the two functions φ_1 and φ_{N_x+1} defined as:

$$\varphi_1(x) = \begin{cases} 1 - \dfrac{x}{h}, & if x \in [a_1, a_2] \\ 0, & if x \in [a_2, a_{N_x+1}] \end{cases}, \tag{39}$$

$$\varphi_{N_x+1}(x) = \begin{cases} 0, & if x \in [a_1, a_{N_x}] \\ \dfrac{x}{h} - N_x + 1, & if x \in [a_{N_x}, a_{N_x+1}] \end{cases}, \tag{40}$$

so that we can write $\mathbf{U}_h$ as a linear combination of the basic elements:

$$\mathbf{U}_h(x,t) = \sum_{j=1}^{N_x+1} \tilde{\mathbf{U}}_j(t) \cdot \varphi_j(x), \tag{41}$$

$$\mathbf{U}_{0,h}(x,t) = \sum_{j=1}^{N_x+1} \mathbf{U}_0\left(x_j\right) \cdot \varphi_j(x), \tag{42}$$

where $\tilde{\mathbf{U}}_1(t) = \mathbf{G}_0(t)$ and $\tilde{\mathbf{U}}_{N_x+1}(t) = \mathbf{G}_1(t)$. Using that $a(\cdot, \cdot)$ is bilinear form and that Eq. (34) is valid for each element of the base $\{\varphi_i\}_{i=2}^{N_x}$, we obtain

$$\sum_{j=1}^{N_x+1} \frac{d}{dt} \tilde{\mathbf{U}}_j(t) \cdot \left(\varphi_j, \varphi_i\right) + \sum_{j=1}^{N_x+1} \tilde{\mathbf{U}}_j(t) \cdot a\left(\varphi_j, \varphi_i\right) = (\mathbf{F}, \varphi_i), \tag{43}$$

$\forall i = 2 \cdots N_x$. This equation can be written in a vector form. For this we define the vectors $\vec{\mathbf{u}}$, $\vec{\mathbf{u_0}}$ and $\vec{\mathbf{F}}$ with components

$$\mathbf{F}_i(t) = (\mathbf{F}, \varphi_i)_{L^2}, \mathbf{u}_j(t) \coloneqq \tilde{\mathbf{u}}_\mathbf{j}(t), \mathbf{u_0}_{,\,\mathbf{j}} = \mathbf{u_0}\left(\mathbf{x_j}\right), \tag{44}$$

and matrices $\mathbf{M}$ and $\mathbf{A}$ as

$$m_{ij} \coloneqq \left(\varphi_i, \varphi_j\right)_{L^2}, a_{ij} \coloneqq a\left(\varphi_i, \varphi_j\right), \tag{45}$$

Note that $\mathbf{M}, \mathbf{A} \in \mathbf{R}^{N_x-1 \times N_x+1}$, $\vec{\mathbf{u}} \in R^{N_x+1}$, *and* $\vec{\mathbf{F}} \in \mathbf{R}^{N_x-1}$. So that (43) is equal to the Cauchy problem

$$\begin{cases} \mathbf{M}\dfrac{d}{dx}\vec{\mathbf{u}}(t) + \mathbf{A}\cdot\vec{\mathbf{u}}(t) = \vec{\mathbf{F}}(t) \\ \vec{\mathbf{u}}(t_0) = \vec{\mathbf{u_0}} \end{cases}, \tag{46}$$

the Crank-Nicolson method can be applied to (46) at time t_k, resulting in

$$\mathbf{M}\cdot\left(\frac{\vec{\mathbf{u}}_{k+1}-\vec{\mathbf{u}}_k}{\Delta t}\right)+\frac{1}{2}\mathbf{A}\cdot\vec{\mathbf{u}}_{\mathbf{k}+\mathbf{1}}+\frac{1}{2}\mathbf{A}\cdot\vec{\mathbf{u}}_{\mathbf{k}}=\frac{1}{2}\left(\vec{\mathbf{F}}_{\mathbf{k}}+\vec{\mathbf{F}}_{\mathbf{k}+\mathbf{1}}\right), \tag{47}$$

where $\vec{\mathbf{u}}_{\mathbf{k}} = \vec{\mathbf{u}}(t_k)$, $\vec{\mathbf{F}}_{\mathbf{k}} = \vec{\mathbf{F}}(t_k), k = 0, 1, \cdots..$
The Eq. (47) can be written in simple form as

$$\left(\mathbf{M}+\frac{\Delta t}{2}\mathbf{A}\right)\cdot\vec{\mathbf{u}}_{\mathbf{k}+\mathbf{1}}=\left(\mathbf{M}-\frac{\Delta t}{2}\mathbf{A}\right)\cdot\vec{\mathbf{u}}_{\mathbf{k}}+\frac{\Delta t}{2}\left(\vec{\mathbf{F}}_{\mathbf{k}}+\vec{\mathbf{F}}_{\mathbf{k}+\mathbf{1}}\right), \tag{48}$$

the algebraic system (48) is solved by Gauss elimination method.

5. Numerical experiment

In this section, we are going to demonstrate some numerical results for $(u(x,t), q(x))$ in the inverse problem (1)–(5). Therefore the following examples are considered and the solution is obtained.

Example 1. Consider (1)–(4) with

$$u(x,0) = \sin x, \quad 0 \le x \le 1, \tag{49}$$

$$u(0,t) = 0, \quad 0 \le t \le 1 \tag{50}$$

$$u(1,t) = \sin(1)e^{-t}, \quad 0 \le t \le 1, \tag{51}$$

$$f(x,t) = \left(\sin x\left(\frac{x^2}{4}+\frac{x}{2}+1\right)\right)e^{-t} - \frac{(x+1)}{2}\cos(x)e^{-t} - \sin x e^{-t}, \quad 0 \le x \le 1 0 \le t \le 1 \tag{52}$$

We obtain the unique exact solution

$$q(x) = 1 + 0.5x + 0.25x^2 \tag{53}$$

And

$$u(x,t) = \sin(x)e^{-t}. \tag{54}$$

We take the observed data g as

$$g(t) = u(0.5,t) = \sin(0.5)e^{-t} 0 \le t \le 1 \tag{55}$$

The unknown function $q(x)$ defined as the following form

$$\hat{q(x)} = p_1 + p_2 x + p_3 x^2, \tag{56}$$

where p_1, p_2, p_3 are unknown coefficients.

Table 1 shows how the Levenberg-Marquardt algorithm can find the best parameters after 12 iterations when it is initialized in four different points.

Figures 1–4 show the fitness of the estimated parameters and the rate of convergence.

Figures 5–8 show the comparison between the inversion results $\hat{q}(x)$ and the exact value $q(x)$:

Table 2 shows the values of $q(\mathrm{j}\Delta\mathrm{x})$ and $u(\mathrm{j}\Delta\mathrm{x}, 0.5)$ in $x = j\Delta x$ with the all the initial values are set 1.

Example 2. Consider (1)–(4) with

$$u(\mathrm{x}, 0) = \mathrm{x}e^{-x}, \quad 0 \le x \le 1, \tag{57}$$

$$u(0, \mathrm{t}) = \mathrm{t}e^{-t}, \quad 0 \le t \le 1 \tag{58}$$

$$u(1, \mathrm{t}) = (1 + \mathrm{t})e^{-1-t}, \quad 0 \le t \le 1 \tag{59}$$

$$f(\mathrm{x}, \mathrm{t}) = e^{-(\mathrm{t}+\mathrm{x})} - (\mathrm{t} + \mathrm{x})e^{-(\mathrm{t}+\mathrm{x})} - e^{x}\left(e^{-(\mathrm{t}+\mathrm{x})} - (\mathrm{t} + \mathrm{x})e^{-(\mathrm{t}+\mathrm{x})}\right) + e^{x}\left(2e^{-(\mathrm{t}+\mathrm{x})} - (\mathrm{t} + \mathrm{x})e^{-(\mathrm{t}+\mathrm{x})}\right), \quad 0 \le x \le 1 \le t \le 1 \tag{60}$$

Starting point	**0.5 0.5 0.5**	**1 1 1**	**10 10 10**	**50 50 50**
Iteration 12	0.999729028233135 0.499885876453067 0.252009862457275	0.999729028233183 0.499885876453056 0.252009862457315	0.999729028233194 0.499885876453057 0.252009862457325	0.999729028307261 0.499885876454169 0.25200986249336
Error F	$8.7564944405 \times 10^{-14}$	$8.7564944427 \times 10^{-14}$	$8.7564944420 \times 10^{-14}$	$8.7564944420 \times 10^{-14}$

Table 1.
Performance of the algorithm when it is run to solve the model using three different parameters guesses.

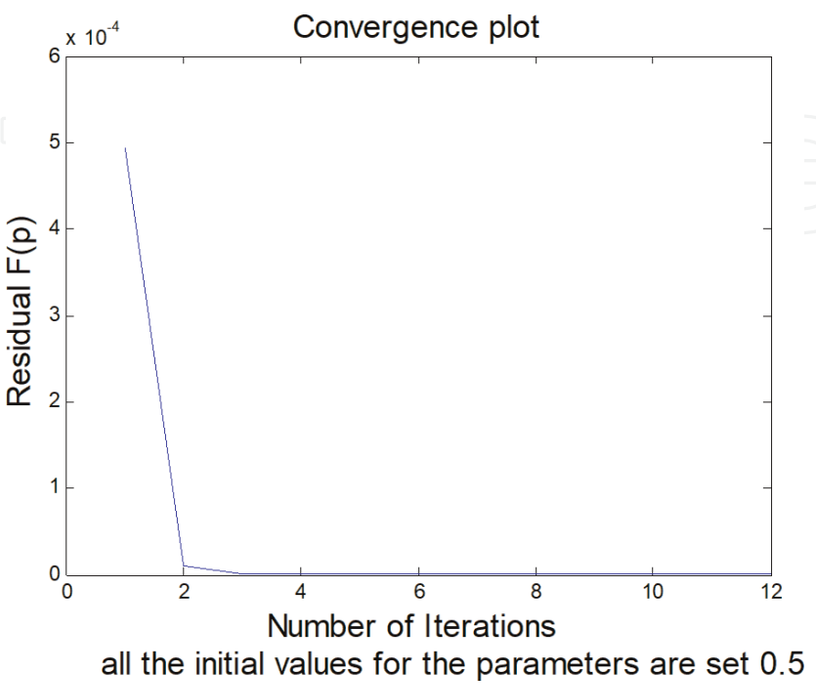


Figure 1.
All the initial values for the parameters are set 0.5.

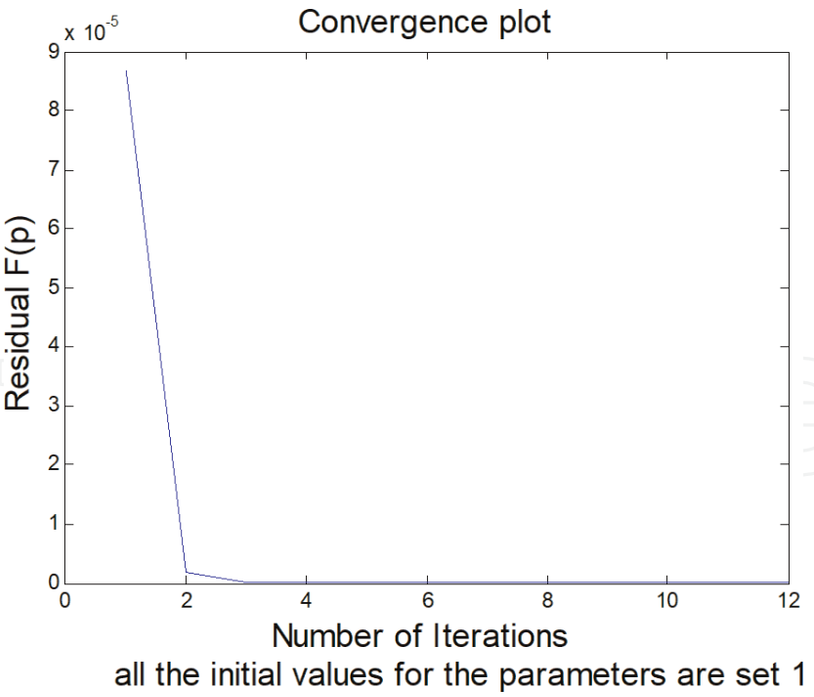


Figure 2.
All the initial values for the parameters are set 1.

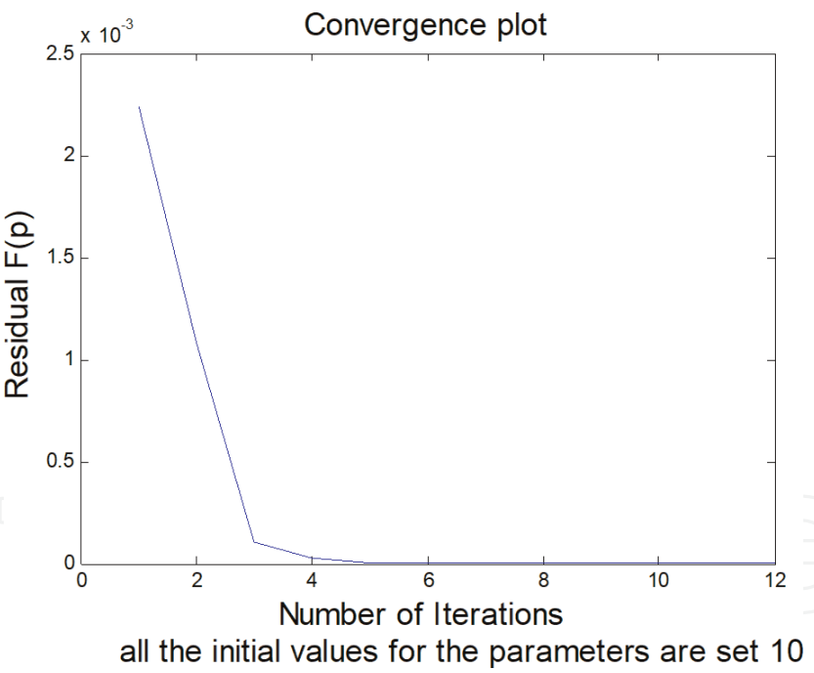


Figure 3.
All the initial values for the parameters are set 10.

We obtain the unique exact solution

$$q(\mathrm{x}) = e^{x}, \tag{61}$$

And

$$u(\mathrm{x}, \mathrm{t}) = (\mathrm{x} + \mathrm{t})\mathrm{e}^{-x-t}. \tag{62}$$

We take the observed data g as

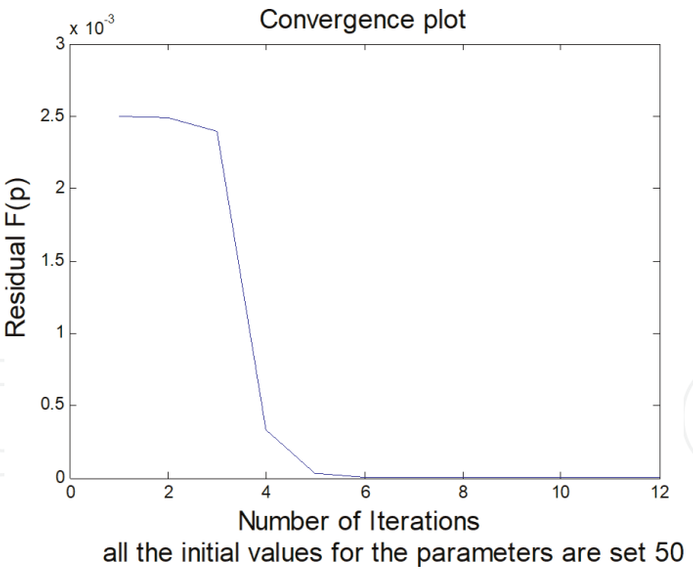


Figure 4.
All the initial values for the parameters are set 50.

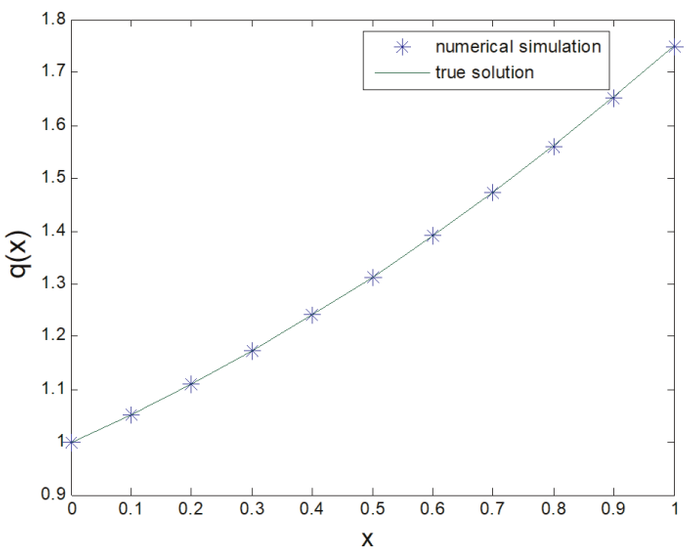


Figure 5.
The comparison chart with all the initial values for the parameters is set 0.5.

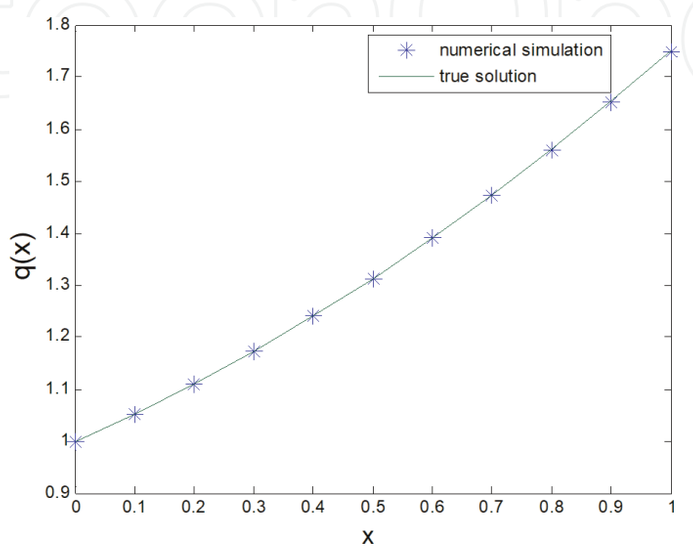


Figure 6.
The comparison chart with all the initial values for the parameters is set 1.

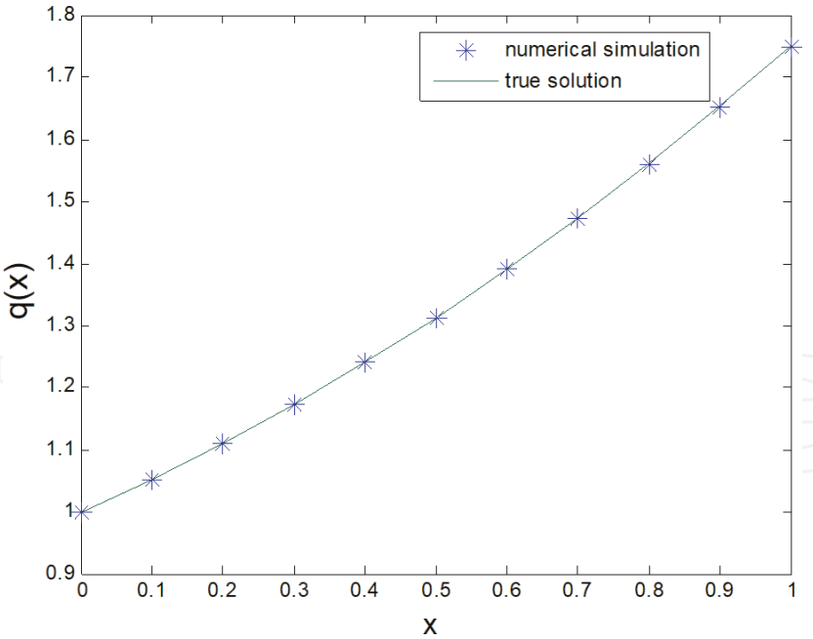


Figure 7.
The comparison chart with all the initial values for the parameters is set 10.

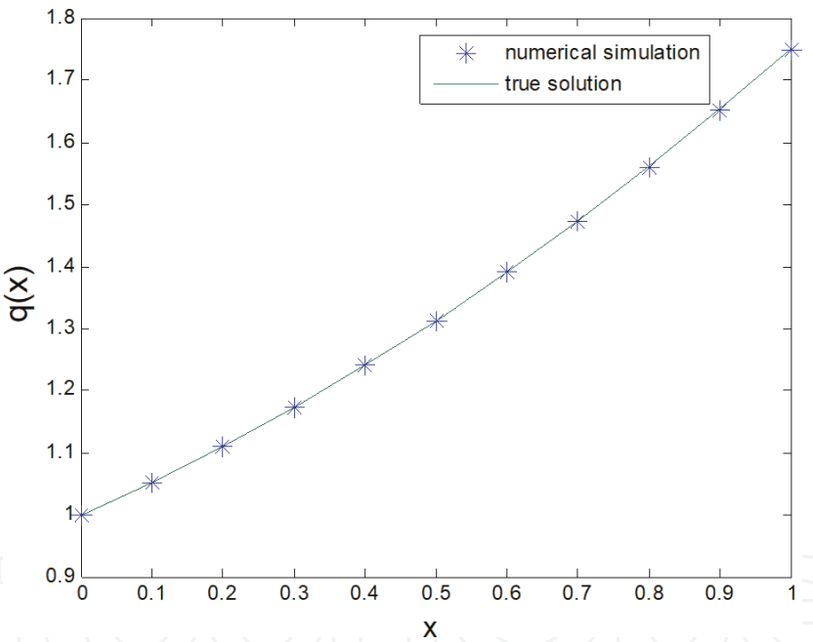


Figure 8.
The comparison chart with all the initial values for the parameters is set 50.

$$g(t) = u(0.5, \mathrm{t}) = (0.5 + \mathrm{t})\mathrm{e}^{(-0.5\text{-t})} 0 \le t \le 1. \tag{63}$$

The unknown function $q(\mathrm{x})$ defined as the following form

$\hat{q(\mathrm{x})} = p_1 + p_2 x + p_3 x^2 + p_4 x^3 + p_5 x^4 + p_6 x^5 + p_7 x^6 + p_8 x^7$, where $p_1, p_2, \cdots, p_7, p_8$ are unknown coefficients.

Table 3 shows how the Levenberg-Marquardt algorithm can find the best parameters after 20 iterations when it is initialized in four different points.

Figures 9–12 show the fitness of the estimated parameters and the rate of convergence.

Figures 13–16 show the comparison between the inversion results $\hat{q}(x)$ and the exact value $q(x)$:

	Numerical	**Exact**	**Numerical**	**Exact**
j	$q(\mathrm{j}\Delta\mathrm{x})$	$q(\mathrm{j}\Delta\mathrm{x})$	$u(\mathrm{j}\Delta\mathrm{x}, 0.5)$	$u(\mathrm{j}\Delta\mathrm{x}, 0.5)$
0	0.999729028233183	1	0	0
1	1.05223771450306	1.0525	0.0605593190239173	0.0605520280601669
2	1.10978659802209	1.11	0.120511797611786	0.120499040271796
3	1.17237567879026	1.1725	0.179257059078521	0.179242065904716
4	1.24000495680758	1.24	0.236207449080344	0.236194164064666
5	1.31267443207404	1.3125	0.290793943250869	0.290786288212692
6	1.39038410458965	1.39	0.342471828361625	0.342472971890064
7	1.47313397435441	1.4725	0.390726114897089	0.390737778838824
8	1.56092404136831	1.56	0.435076630410587	0.435098463062163
9	1.65375430563136	1.6525	0.475082717530532	0.475111787267016
10	1.75162476714355	1.75	0.510347406713368	0.510377951544573

Table 2.
The values of $q(\mathrm{j}\Delta\mathrm{x})$ and $u(\mathrm{j}\Delta\mathrm{x}, 0.5)$ in $x = j\Delta x$ with the all the initial values being set to 1.

Starting point	**0.1 0.1 0.1 0.1** **0.1 0.1 0.1 0.1**	**0.5 0.5 0.5 0.5** **0.5 0.5 0.5 0.5**	**1 1 1 1** **1 1 1 1**	**2 2 2 2** **2 2 2 2**
Iteration 20	1.01536263526644	1.01536263500695	1.01536263525763	1.01536263525905
	0.896348846894057	0.896348850692318	0.896348847022403	0.896348846999736
	0.954285303464511	0.954285278486704	0.954285302637587	0.954285302790922
	–0.890298938193057	–0.890298849338373	–0.890298935334171	–0.890298935876618
	1.40131927153131	1.40131909032315	1.40131926588117	1.40131926696099
	–0.871276408882294	–0.871276197318301	–0.871276402487896	–0.87127640370648
	0.183785623507722	0.183785492186491	0.18378561965322	0.183785620380814
	0.0359103726343979	0.0359104061952515	0.0359103735933848	0.0359103734148875
Error F	$7.89749200363512 \times 10^{-11}$	$7.89749200363504 \times 10^{-11}$	$7.89749200353888 \times 10^{-11}$	$7.8974920035389 \times 10^{-11}$

Table 3.
Performance of the algorithm when it is run to solve the model using four different parameters guesses.

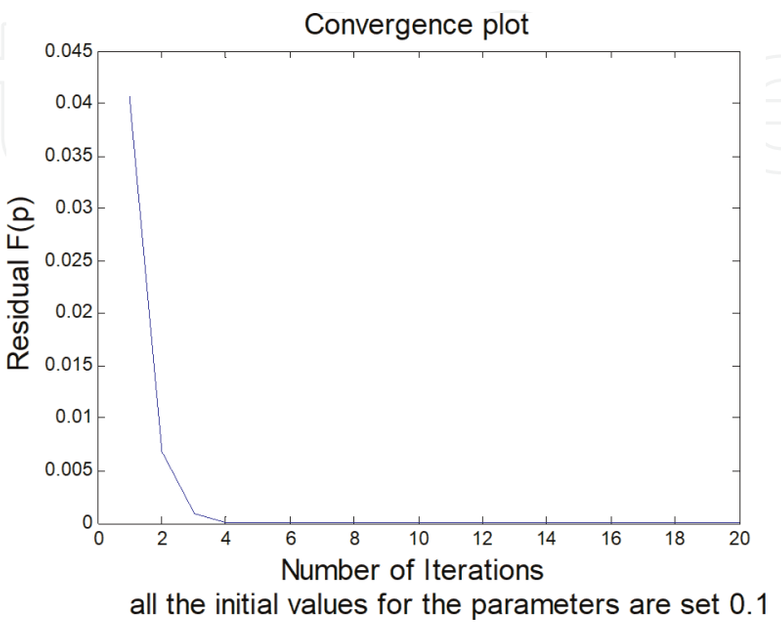


Figure 9.
All the initial values for the parameters are set 0.1.

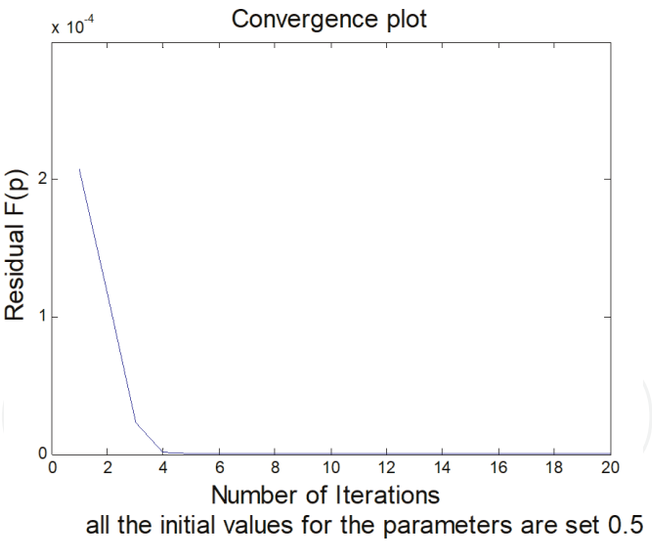


Figure 10.
All the initial values for the parameters are set 0.5.

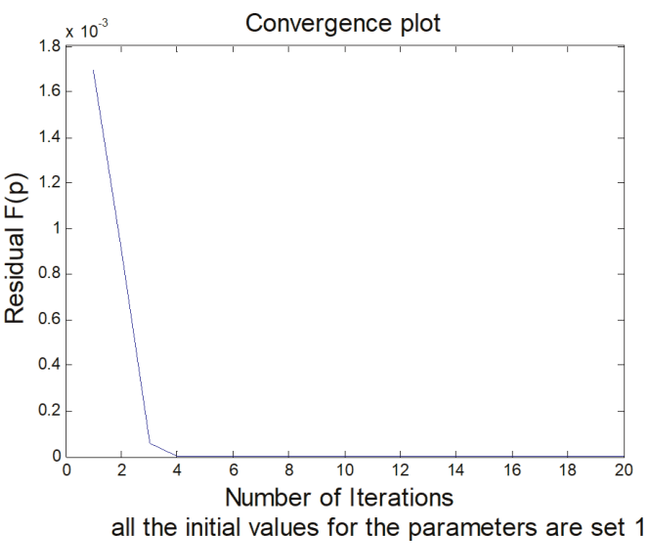


Figure 11.
All the initial values for the parameters are set 1.

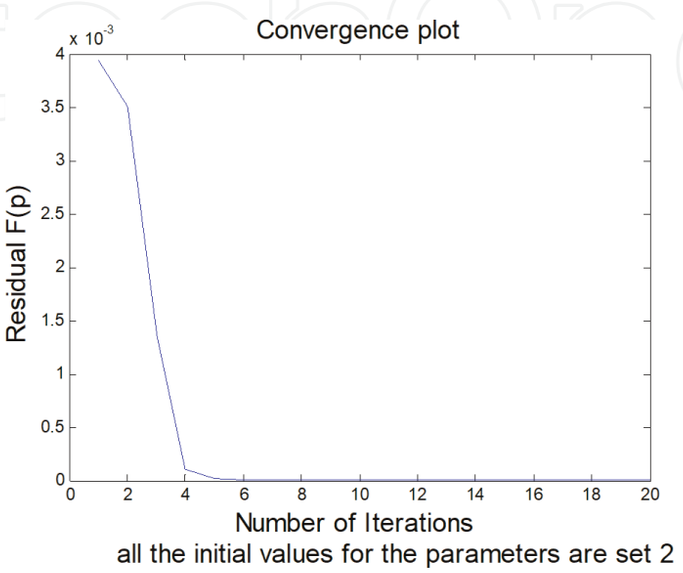


Figure 12.
All the initial values for the parameters are set 2.

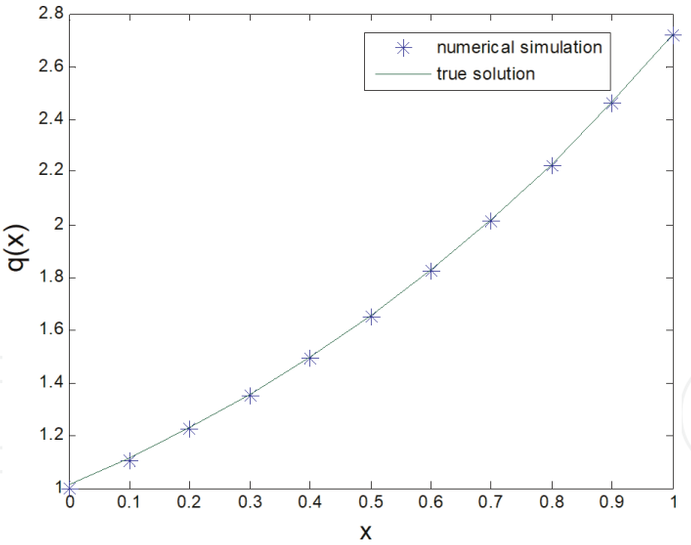


Figure 13.
The comparison chart with all the initial values for the parameters is set 0.1.

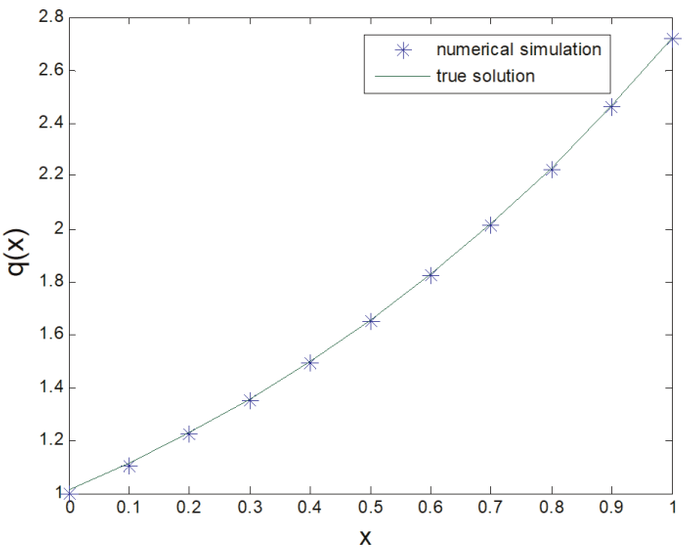


Figure 14.
The comparison chart with all the initial values for the parameters is set 0.5.

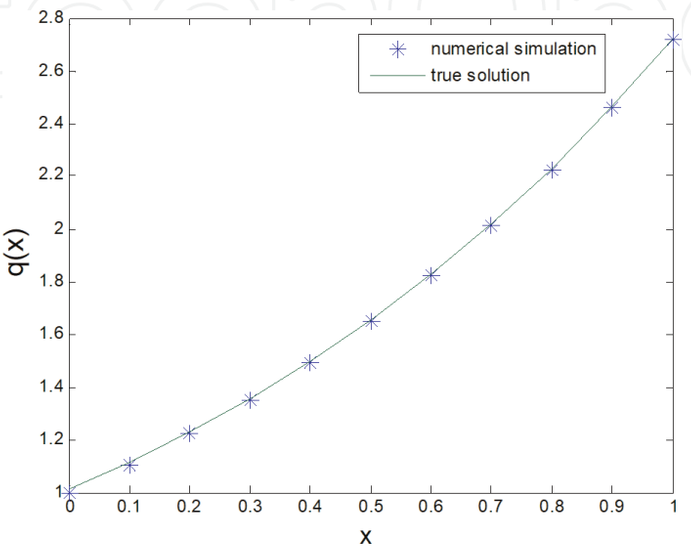


Figure 15.
The comparison chart with all the initial values for the parameters is set 1.

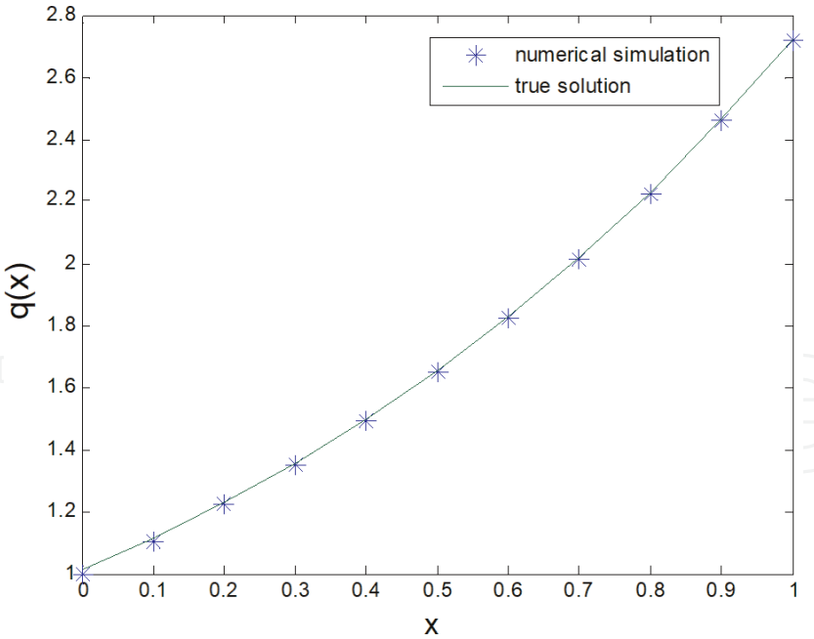


Figure 16.
The comparison chart with all the initial values for the parameters is set 2.

	Numerical	Exact	Numerical	Exact
j	$q(\mathrm{j}\Delta\mathrm{x})$	$q(\mathrm{j}\Delta\mathrm{x})$	$u(\mathrm{j}\Delta\mathrm{x}, 0.5)$	$u(\mathrm{j}\Delta\mathrm{x}, 0.5)$
0	1.01536263525763	1	0.303342962644088	0.303265329856317
1	1.11378168059013	1.10517091807565	0.329201964677126	0.329286981656416
2	1.22765694959399	1.22140275816017	0.347492882224886	0.347609712653987
3	1.35549021305874	1.349858807576	0.359347568702678	0.359463171293777
4	1.4973722149265	1.49182469764127	0.365808711321159	0.365912693766539
5	1.65432828415206	1.64872127070013	0.367792378208857	0.367879441171442
6	1.82785056869393	1.82211880039051	0.366091218454182	0.366158192067887
7	2.01963499046464	2.01375270747048	0.361387761473796	0.361433054294643
8	2.23154102006879	2.22554092849247	0.354267869273063	0.354291330944216
9	2.46579237015687	2.45960311115695	0.345233023618059	0.345235749518249
10	2.72543670622333	2.71828182845905	0.334712604803175	0.334695240222645

Table 4.
The values of $q(\mathrm{j}\Delta\mathrm{x})$ and $u(\mathrm{j}\Delta\mathrm{x}, 0.5)$ in $x = j\Delta x$ with the all the initial values are set 1.

Table 4 shows the values of $q(\mathrm{j}\Delta\mathrm{x})$ and $u(\mathrm{j}\Delta\mathrm{x}, 0.5)$ in $x = j\Delta x$ with the all the initial values are set 1.

6. Conclusions

A numerical method to estimate the temperature $\mathrm{u}(\mathrm{x}, \mathrm{t})$ and the coefficient $q(\mathrm{x})$ is proposed for an IHCP and the following results are obtained.

1. The present study, successfully applies the numerical method involving the Levenberg-Marquardt algorithm in conjunction with the Galerkin finite element method to an IHCP.

2. From the illustrated example it can be seen that the proposed numerical method is efficient and accurate to estimate the temperature $u(x,t)$ and the coefficient $q(x)$.

Acknowledgements

The work of the author is supported by the Special Funds of the National Natural Science Foundation of China (Nos. 51190093 and 51179151). The author would like to thank the referees for constructive suggestions and comments.

Conflict of interests

The authors declare that there is on conflict of interests regarding the publication of this article.

Author details

Tao Min[1,2*], Xing Chen[1], Yao Sun[1] and Qiang Huang[2]

1 School of Science, Xi'an University of Technology, Xi'an Shaanxi, China

2 State Key Laboratory of Eco-Hydraulic Engineering in Shaanxi, Xi'an University of Technology, Xi'an Shaanxi, China

*Address all correspondence to: mintao@xaut.edu.cn

IntechOpen

© 2019 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. CC BY

References

[1] Shidfar A, Karamali GR. Numerical solution of inverse heat conduction problem with nonstationary measurements. Applied Mathematics and Computation. 2005;**168**(1):540-548

[2] Shidfar A, Karamali GR, Damirchi J. An inverse heat conduction problem with a nonlinear source term. Nonlinear Analysis: Theory Methods & Applications. 2006;**65**(3):615-621

[3] Shidfar A, Pourgholi R. Numerical approximation of solution of an inverse heat conduction problem based on Legendre polynomials. Applied Mathematics and Computation. 2006; **175**(2):1366-1374

[4] Shidfar A, Azary H. An inverse problem for a nonlinear diffusion equation. Nonlinear Analysis: Theory Methods & Applications. 1997;**28**(4): 589-593

[5] Kurpisza K, Nowaka AJ. BEM approach to inverse heat conduction problems. Engineering Analysis with Boundary Elements. 1992;**10**(4):291-297

[6] Han H, Ingham DB, Yuan Y. The boundary-element method for the solution of the back ward heat conduction equation. Journal of Computational Physics. 1995;**116**(2): 292-299

[7] Skorek J. Applying the least squares adjustment technique for solving inverse heat conduction problems. In: Taylor C, editor. Proceedings of the 8th Conference on Numerical Methods in Laminar and Turbulent Flow. Swansea: Pineridge Press; 1993. pp. 189-198

[8] Pasquetti R, Le Niliot C. Boundary element approach for inverse heat conduction problems: Application to a bidimensional transient numerical experiment. Numerical Heat Transfer, Part B. 1991;**20**(2):169-189

[9] Ingham DB, Yuan Y. The solution of a nonlinear inverse problem in heat transfer. IMA Journal of Applied Mathematics. 1993;**50**(2):113-132

[10] Hensel E. Inverse Theory and Applications for Engineers. N.J. ISBN: 0135034590: Prentice Hall; 1991

[11] Özisik MN. Inverse Heat Transfer: Fundamentals and Applications. New York, USA: Taylor and Francis. ISBN: 1-56032-838-X; 2000

[12] Pourgholi R, Azizi N, Gasimov YS, Aliev F, Khala HK. Removal of numerical instability in the solution of an inverse heat conduction problem. Communications in Nonlinear Science and Numerical Simulation. 2009;**14**(6): 2664-2669

[13] Shidfar A, Pourgholi R, Ebrahimi M. A numerical method for solving of a nonlinear inverse diffusion problem. Computers and Mathematics with Applications. 2006;**52**(6–7):1021-1030

[14] Wang J, Zabaras N. A Bayesian inference approach to the inverse heat conduction problem. International Journal of Heat and Mass Transfer. 2004;**47**(17–18):3927-3941

[15] Dehghan M. Determination of an unknown parameter in a semi-linear parabolic equation. Mathematical Problems in Engineering. 2002;**8**(2): 111-122

[16] Lihua J, Changfeng M. On L-M method for nonlinear equations. Journal of Mathematics. Wuhan University. 2009;**29**(3):253-259

[17] Chen P. Why not use the Levenberg–Marquardt method for fundamental matrix estimation? The Institution of Engineering and Technology. 2010;**4**(4):286-288