DOMAIN DECOMPOSITION BASED ALGORITHMS FOR SOME INVERSE PROBLEMS

CHARAKA JEEWANA PALANSURIYA

A thesis submitted in partial fulfilment of the requirements of the University of Greenwich for the Degree of Doctor of Philosophy

August 2000

Abstract

The work presented in this thesis develop algorithms to solve inverse problems where source terms are unknown. The algorithms are developed on frameworks provided by domain decomposition methods and the numerical schemes use finite volume and finite difference discretisations.

Three algorithms are developed in the context of a metal cutting problem. The algorithms require measurement data within the physical body in order to retrieve the temperature field and the unknown source terms. It is shown that the algorithms can retrieve both the temperature field and the unknown source accurately.

Applicability of the algorithms to other problems is shown by using one of the algorithms to solve a welding problem.

Presence of untreated noisy measurement data can severely affect the accuracy of the retrieved source. It is illustrated that a simple noise treatment procedure such as a least squares method can remedy this situation.

The algorithms are implemented on parallel computing platforms to reduce the execution time. By exploiting domain and data parallelism within the algorithms significant performance improvements are achieved. It is also shown that by exploiting mathematical properties such as change of nonlinearity further performance improvements can be made.

Acknowledgements

The author wishes to express his gratitude to the University of Greenwich who is sponsoring this study.

I would like to thank my supervisors Dr Choi-Hong Lai, Dr Cos Ierotheou and Professor Koulis Pericleous for their concern and guidance at the various stages of the research process.

I would particularly like to thank Professor David Keyes for advice and discussions on the algorithms and their implementations.

I would also like to thank Professor X.-C. Tai for various advice on sensitivity analysis and Dr Loise McInnes for introducing PETSc (along with Professor David Keyes).

Last but not least, I would like to thank my parents for their support.

Charaka J. Palansuriya.

Nomenclature

A	Cell face area; amplitude
c	Specific heat capacity
c^e	Effective specific heat capacity
D	Domain
D_i	Subdomain i
h_{eff}	Effective heat transfer
J	Jacobian
k	Thermal conductivity
L	Differential operator; latent heat
m	Iterative step
n	Time step; Schwarz iterative step
Q_c, Q_w	Source term
Q_p	Predicted heat source
T_a	Ambient temperature
T, u	Temperature
T_i, U_i	Initial Conditions
T^*, U^*	Sensor temperature

T_p, U_p	Predicted temperature
Tol	Tolerance
T_s	Solidus temperature; measured temperature
T_l	Liquidus temperature
t	Time
x, y	Cartesian coordinates
x_c	Cutter point
x_s	Sensor point
U	Temperature vector
u_i	Temperature in subdomain i
V	Correction vector
Δt	Regular time step spacing
Δx, Δy	Regular Cartesian grid spacing
ρ	Density

List of Figures

Contents

Chapter 1

Introduction

1.1 Inverse problems

If a problem is specified by a differential equation, its boundary data and its domain, then the problem is a *direct problem.* However, if part of the differential equation is unknown and has to be determined then it is an *inverse problem* (Colton et al., 1990). For example, if the source term is an unknown in a heat conduction equation then the resulting inverse problem has to be solved by using some internal temperature measurements. Typically such inverse problems require either the retrieval of sources/sinks or conductivity/permeability in the relevant partial differential equations. However, there are other inverse problems where the aim is to recover the unknown heat fluxes across a boundary or to retrieve the unknown temperature at the boundary or, in some situations, to retrieve an unknown boundary.

A great boost to the research in inverse problems was given by the space program started around 1950s (Beck et al., 1985). The main interest there was the calculation of surface temperature and heat flux across the outer skin of a rocket nozzle when it is re-entering the earth's atmosphere. Sensors cannot be placed on the outer skin since it is exposed to extremely high temperatures. A direct problem cannot be formulated in such situations. A practical solution available to the scientist was to attach the sensors inside the rocket nozzle, between the inner and outer skin (see Figure 1.1), and then to retrieve the temperature at the outer skin (and/or the heat flux across it). Therefore one needs to solve an inverse problem.

There are many other industrially vital problems that require solutions of inverse problems. One of them is a oil reservoir simulation. In oil reservoir simulations the

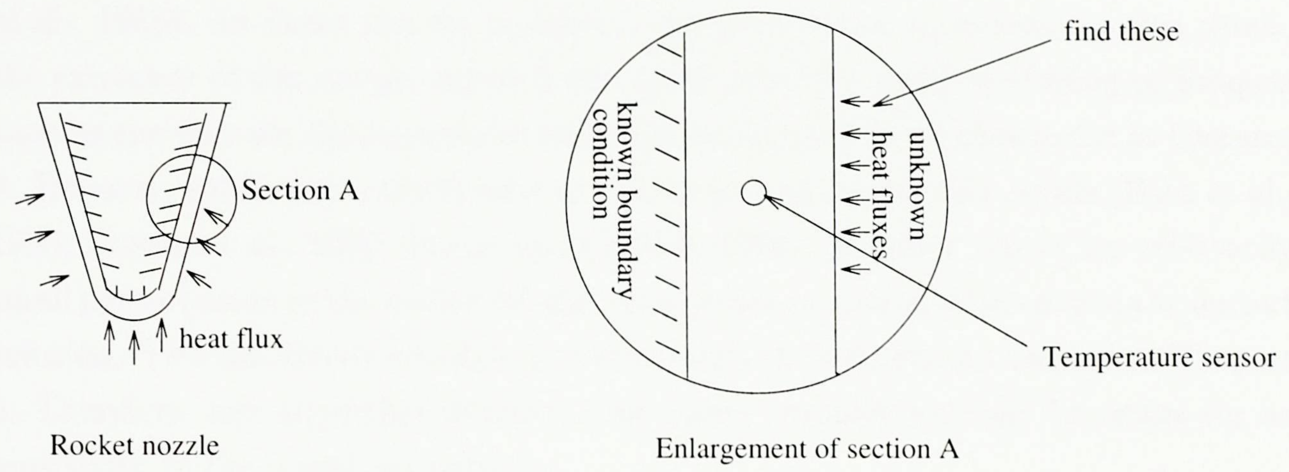


Figure 1.1: Sensors are placed between the inner and outer skin of the rocket nozzle.

permeability of the oil fields are unknown and therefore the resulting inverse problem is to retrieve this unknown coefficient. Another example from industry is the casting process simulation where the unknown conductivity or the heat transfer coefficient, during the solidification process, has to be retrieved (Kim and Lee, 1997). Metal cutting and welding are also widely occurring industrial processes. Such activities require careful control of temperature to guarantee the quality of the cut or the weld, to prolong the life of the cutting tool. In order to control the temperature, the strength of the heat source has to be determined and regulated. Hence, the inverse problem in these processes is to recover the unknown source. Last, but not least, thermal imaging is widely used for carrying out non-destructive testing (Bryan and Caudill, 1994; Patel et al., 1992). Here, the technique is used to recover information about the internal condition of an object by applying a heat flux to its boundary and observing the resulting temperature response of the object's surface. The information is used to retrieve the internal thermal properties of the object, or the shape of some unknown portion of the boundary. A typical application of this technique is to detect damage or corrosion in aircraft.

Many more fields of science and technology, such as astronomy, chemistry and medicine require solutions to inverse problems. The above applications require the development of accurate, fast and efficient algorithms to solve the relevant inverse problems.

Development of algorithms to solve inverse problems is complicated due to the fact that such problems are often ill-posed (not well-posed). A problem is said to be well-posed provided that the solution (1) exists, (2) is unique, and (3) is stable (Beck et al., 1985). In many inverse problems, including those considered in this thesis, the existance of the unique solution can be satisfied by using modelling techniques such as the domain decomposition methods as explained and illustrated in Chapter 3. However, solutions to the inverse problems are not necessarily stable (Beck et al., 1985; Colton et al., 1990; Bryan and Caudill, 1994). In other words, an arbitrarily small perturbation in the measured data may produce a large difference in the output solution. This sensitivity problem is more clearly illustrated and tackled in Chapter 5. Therefore, any algorithm developed for inverse problems should be tested for its sensitivity to the measured data.

While there is much research work done in the inverse determination of conductivity properties and coefficients (Beck et al., 1985; Colton et al., 1990; Kunisch and Tai, 1996), the inverse determination of sources is not well documented in the literature. The type of inverse problems considered in this thesis is the retrieval of unknown source terms. The algorithms developed to solve such problems are tested and validated on nonlinear applications such as metal cutting and welding. The applications considered may require on-line processing (for example in real-time control) which in turn requires real time computation. Therefore, various performance enhancement schemes are considered to speedup the computation. The domain and data parallelism within algorithms are exploited to speedup the calculations. The computer simulation of parallel algorithms can be carried out virtually in any research institute due to the wide availability of distributed computing platforms such as networks of various workstation clusters. In this thesis, an existing networked workstation cluster is adapted to carry out this type of computer simulation. Further performance enhancement is also considered on such a network of workstations.

This thesis examines the concept of domain decomposition techniques for the solution of some inverse problems. A number of novel ideas including a simplification of mathematical models, numerical procedures, exploitation of parallel properties, and various performance enhancement techniques are discussed in Chapters 6 and 7.

1.2 Methods used to solve inverse problems

Inverse problems are more difficult to solve analytically than direct problems. Exact solution techniques have been proposed by Burggraf (Burggraf, 1964), Imber and

Khan (Imber and Khan, 1972), Langford (Langford, 1967) and others. These techniques have only limited use for realistic problems. Therefore, various approximate methods have been developed instead to solve such problems. These include graphical(Stolz, 1960), polynomial (Frank, 1963), Laplace transform (Krzysztof et al., 1981), dynamic programming (Trujillo, 1978), finite difference (D'Souza, 1975), finite elements (Krutz et al., 1978). A finite volume (FV) method is used in numerical experiments carried out in this thesis. Although the above methods are used to solve different types of inverse problems, only the methods used for solving inverse source problems are discussed in this section.

Attempts to reformulate the inverse problems at the partial differential equation level and to eliminate the unknown source terms have not been successful in solving realistic nonlinear problems numerically (Cannon et al., 1990). These methods use the overspecified conditions, such as extra measurements at a boundary, to eliminate the unknown source functions from the partial differential equation. The resulting reformulated problem has the form of a standard boundary value or an initial boundary value problem but containing sources which are functionals of the *unknown solution*. This recursive nature of the reformulated problem is known to worsen the stability of the inverse source problem and therefore is not considered in this thesis.

Domain decomposition techniques were used with various polynomial interpolations (e.g., Lagrange) to approximate the partial differential equations and then to retrieve the unknown source term (Preziosi, 1993). However, such schemes do not conserve the physical properties of a problem at the discretised level. The schemes are neither validated nor shown to be stable. The numerical schemes described there do not have any error or noise treatment procedures.

The use of least squares method to solve nonlinear problems dates back to 1940s (Levenberg, 1944). The method minimizes the difference between the measured value and the true function value. The least squares method is used with some success to retrieve unknown source function in linear partial differential equations (Chow et al., 1999). An advantage of the method is that it smooths the noise in the measurement data to produce stable results. However, the way the method is used is a trade off between accuracy and stability, that is, a stable solution to the inverse problem is achieved with less accuracy. Typically, the least squares method is used to solve the entire inverse problem without considering the physical nature (e.g., discontinuities) of a problem and this is a significant contribution to the inaccuracy in the solution. As illustrated in this thesis, the use of the method alongside domain decomposition and FV discretisation, which conserve the physical properties of the problem even at discretised level (Patankar, 1980; Versteeg and Malalasekera, 1995), can produce an accurate and stable solution.

A class of methods called regularization methods that modifies the least squares method is to damp the fluctuation in the unknown function (e.g., source term) (Beck et al., 1985). The idea behind these methods is that by varying the regularization parameter the fluctuations in the unknown function can be controlled. These fluctuations may be due to ill-posedness of the problem. One major question that has yet to be answered is the choice of regularisation parameters, particularly when nothing is known about the function to be retrieved, it is not clear which value of the regularization parameter is to be used (Tai et al., 1997; Neumaier, 1998; Frommer and Maass, 1999).

Although all the above methods are computationally intensive, parallel properties of algorithms for inverse source problems are not fully exploited. This thesis attempts to fill the gap with a systematic way of extracting parallel properties through different levels of hierarchy of the problems. The exploitation of such properties is linked with the use of appropriate mathematical models and numerical techniques. Although the literature has not shown a significant use of parallel computing to solve inverse source problems, the author feels that the investigation and development of parallel algorithms for inverse source problems are vital to industrial real time simulation. As illustrated in Chapter 6, the performance improvement gained by the use of parallel computing is considerable.

1.3 Objectives

The main objectives of this thesis are as follows:

1. Development of accurate and efficient algorithms, based on a domain decomposition concept, to solve nonlinear time dependent inverse source problems; given the time dependent temperature measurements at some interior locations of a physical domain.

2. Examination of suitable techniques to reduce the sensitivity of the algorithms

to measurement errors. These techniques should be able to handle a reasonable level of noise in sensor data.

3. Development of fast algorithms by exploiting various levels of parallelism in the algorithms.

1.4 Outline of contents

While this chapter does not serve as a comprehensive review of inverse problems, it does give a summary of a few industrial related examples as an illustration to an increasingly important field of modelling. It underlines the importance of parallel numerical methods for such problems. As stated in the first objective, much of this thesis discusses the development of efficient numerical algorithms based on a domain decomposition method. Therefore, Chapter 2 provides a brief review to the domain decomposition method. The main aim is to bring in a more general concept of domain decomposition at different levels of a physical problem and to relate the concept to the algorithm and software development. A typical engineering problem, namely the metal cutting problem, is being considered in Chapter 3. A simplified mathematical model is derived resulting in an unsteady nonlinear parabolic problem with an unknown source term. It shows the use of a domain decomposition method for partitioning the physical domain into simpler subdomains. Also, the formulation of the source retrieval is explained. By using the framework provided by the domain decomposition, three algorithms are developed based on a finite volume discretisation. Their accuracy and the uni-processor performance are illustrated. Another industrial application, electric arc welding is examined in Chapter 4. This is a moving source problem where the source strength is unknown. The domain decomposition method explained in Chapter 3 is used for partitioning the physical domain and one of the algorithms developed is used to solve this problem. The accuracy of the temperature and source retrieval are shown. The sensitivity of the algorithms to measurement errors are examined in Chapter 5. It is shown that the use of a least squares method to smooth the errors makes the algorithms less sensitive to their presence. The exploitation of various levels of parallelism in order to develop fast algorithms is examined in Chapter 6. A parallel performance model developed for one of the algorithms is used to explain the performance of the algorithm in a distributed computing environment. Further performance enhancement techniques for the algorithms are considered in Chapter 7. It shows some preliminary results for such a technique. Finally, Chapter 8 makes various conclusions regarding the algorithms, discusses how the original objectives were met and makes some recommendations for further research work.

Chapter 2

Domain decomposition methods

Domain decomposition can be described as the division of a problem domain into a number of subdomains, each containing a complete subproblem. The resulting subproblems can be solved separately and then combined to give the solution to the original problem. It is effectively a *divide-and-conquer* and then *re-combine* strategy. During the last decade, domain decomposition based methods were used extensively to solve a large variety of scientific problems (Chan et al., 1988; Keyes et al., 1991). Recently such methods were used to solve inverse problems related to the estimation of coefficients of partial differential equations (Kunisch and Tai, 1996). In this thesis, as explained in the previous chapter, the development of domain decomposition based methods to solve inverse problems related to the retrieval of unknown source terms is examined.

Domain decomposition can be applied at the physical problem level and/or the discretised problem level. At the physical problem level, the regions governed by different mathematical models or some other criteria such as different material properties and/or conductivity are identified and decomposed into different subdomains. At the discretised level, the resulting system of equations are rearranged as a collection of smaller systems which can be solved independently.

The concept of domain decomposition has been evolving for over a century. The earliest known domain decomposition algorithm at the physical problem level to solve elliptic problems was due to Schwarz (Schwarz, 1869). Schwarz considered an elliptic boundary value problem posed on an irregular domain that was made up by two regular subdomains and the solutions for each of these subdomains can be obtained readily. The first algorithm based on the concept of domain decomposition to treat a variety of large scale computational engineering problems including elasticity, electrical network, and incompressible and compressible flow was due to Kron (Kron, 1963).

2.1 Why use domain decomposition methods ?

Domain decomposition simplifies the solution of complex problems, that is, the subproblems are much simpler to solve than the original problem. The complexity of a problem could be due to the original domain containing material inhomogeneities, rapid change of nonlinearity in a subregion, different physical regions such as solid and fluid, different mathematical models, etc. Such complexities can be removed by applying a suitable decomposition of the problem domain. For example, in the case of material inhomogeneities, domain decompositioning can be applied at points where material properties change to generate subdomains with homogeneous material properties. Similarly, as presented by Lai (Lai et al., 1998), if a domain contains different mathematical models such as Euler and Navier-Stokes models then domain decomposition can be carried out to generate subdomains with homogeneous mathematical models. Therefore, in applying domain decomposition we are trying to generate homogeneous and therefore simpler subproblems. An advantage of generating homogeneous subdomains is that the subproblems can be solved using existing numerical methods and software.

Algorithms developed using domain decomposition are suitable for sequential (uni-processor) as well as parallel (multi-processor) computation. The data localities of such algorithms contribute to various efficiencies at the implementation level, for example efficient use of cache. Considering parallel computing requirements (Chapter 6), access of non-local data (i.e., data in other subdomain) can be identified at the algorithm development phase (Bjørstad and Karstad, 1995). This helps to analyse the communication overheads and therefore to design and implement suitable software with minimal overheads to solve various scientific problems.

2.2 Some domain decomposition methods

The classical Schwarz alternating method (Schwarz, 1869) is still widely used in solving contemporary scientific problems (Chan and Mathew, 1994; Smith et al., 1995; Keyes et al., 1995; Sirotkin, 1997). The method is used to solve overlap subdomains. The idea is to partition an irregular domain into regular subdomains (see Figure 1) and then use an iterative coupling technique for the coupling of subdomains. The following algorithm describes the Schwarz alternating method for the differential problem $Lu = f$ defined in D which is partitioned into two overlapped subdomains D_1 and D_2 (see Figure 1) with $f|_{D_i} = f_i$, $i = 1, 2$, and the problem is prescribed with Dirichlet boundary conditions g_1 and g_2 along $\partial D_1 \bigcap \partial D$ and $\partial D_2 \bigcap \partial D$ respectively. Here the superscript (n) denotes the number of Schwarz iterations.

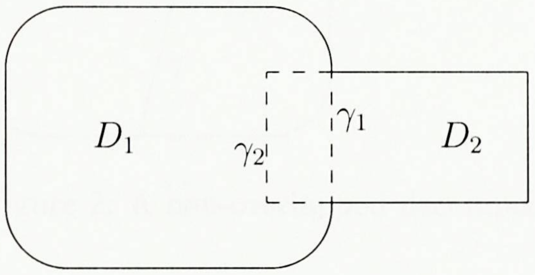


Figure 1: A flask-shaped region similar to that used by Schwarz.

```
The Schwarz Alternating Method (Schwarz, 1869):
begin {
n := 0; $u_2^{(0)}|_{\gamma_1}$ := initial approximation;
repeat {
n := n + 1;
$u_1^{(n)}$ := { solve $Lu_1^{(n)} = f_1$ in $D_1$
          subject to
            $u_1^{(n)} = g_1$ on $\partial D_1 \bigcap \partial D$
            $u_1^{(n)}|_{\gamma_1} = u_2^{(n-1)}|_{\gamma_1}$ };
$u_2^{(n)}$ := { solve $Lu_2^{(n)} = f_2$ in $D_2$
          subject to
            $u_2^{(n)} = g_2$ on $\partial D_2 \bigcap \partial D$
            $u_2^{(n)}|_{\gamma_2} = u_1^{(n)}|_{\gamma_2}$ };
} until converge; } end.
```

The overlapped approach may not be suitable in situations such as when the meshes in different subdomains do not match each other or one mesh is finer than the other

(Lai, 1994). Such problems may be best solved by using a non-overlapped approach (Figure 2). Various variants of the Schwarz alternating method suitable for solving non-overlapped subdomains exist (Lai, 1994; Quarteroni, 1995). A gradient variant of the Schwarz alternating method is described below.

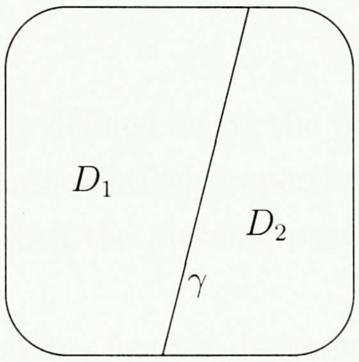


Figure 2: A non-overlapped decomposition.

```
A Gradient Variant of the
Schwarz Alternating Method (Lai, 1994):
begin {
$n := 0$; $\lambda^{(0)} :=$ initial approximation;
repeat {
$u_1^{(n)} :=$ { solve $Lu_1^{(n)} = f_1$ in $D_1$
                subject to
                    $u_1^{(n)} = g_1$ on $\partial D_1 \bigcap \partial D$
                    $\frac{\partial u_1^{(n)}}{\partial n_1} = \lambda^{(n)}$ on $\gamma$ };
$u_2^{(n)} :=$ { solve $Lu_2^{(n)} = f_2$ in $D_2$
                subject to
                    $u_2^{(n)} = g_2$ on $\partial D_2 \bigcap \partial D$
                    $\frac{\partial u_2^{(n)}}{\partial n_2} = \lambda^{(n)}$ on $\gamma$ };
$\lambda^{(n+1)} := \lambda^{(n)} + \alpha(u_2^{(n)} - u_1^{(n)})|_\gamma$ };
$n := n + 1$;
} until converge; } end.
```

Here $D = D_1 \cup D_2$, γ denotes the interface of the two non-overlapped subdomains as depicted in Figure 2 and n_1 and n_2 denote the outward normals along γ with respect to D_1 and D_2. An important advantage of the non-overlap approach is its inherent suitability to parallel computing. That is each subdomain can be independently solved before updating the interface.

2.3 Problem partitioning and decomposition hierarchy

The problem partition is defined to be the partitioning of a problem by just considering the physical/mathematical properties. For example, if a problem contains solid and fluid regions then the problem partition can be applied at the solid-fluid interface.

Within each subdomain, further decompositioning can be carried out for the discretised problem. If different discretisations are used in a subdomain then partitioning could be carried out where the discretisation changes.

Additional decomposition can be carried out for the data in a subdomain. For example, to solve a problem in parallel, mesh partitioning may be required which means data decomposition needs to be carried out within each subdomain.

The concept that is evolving here is a hierarchy of decomposition (see Figure 2.1). As can be seen, the decomposition hierarchy is suitable for parallel computation as well as for sequential computation. An example of the concept can be seen from the problem partitioning carried out in Chapter 3 and the data partitioning carried out within the resultant subdomains in Chapter 6. The decomposition hierarchy is a concept which facilitates the development of clear and simple algorithms that may be easier to implement in terms of software development. The software developed, as a result, is usually memory and cache efficient. In parallel computing the decomposition provides a framework to develop algorithms which are suitable for fine-grain as well as coarse-grain parallel computers (Ierotheou et al., 1998).

2.4 Domain decomposition software

As the hierarchical model suggests, domain decomposition may occur at several levels in software. At the highest level, subproblems resulting from a problem partitioning may be of different mathematical models and can be solved by using dedicated software. If the problems are continuous and homogeneous, then existing

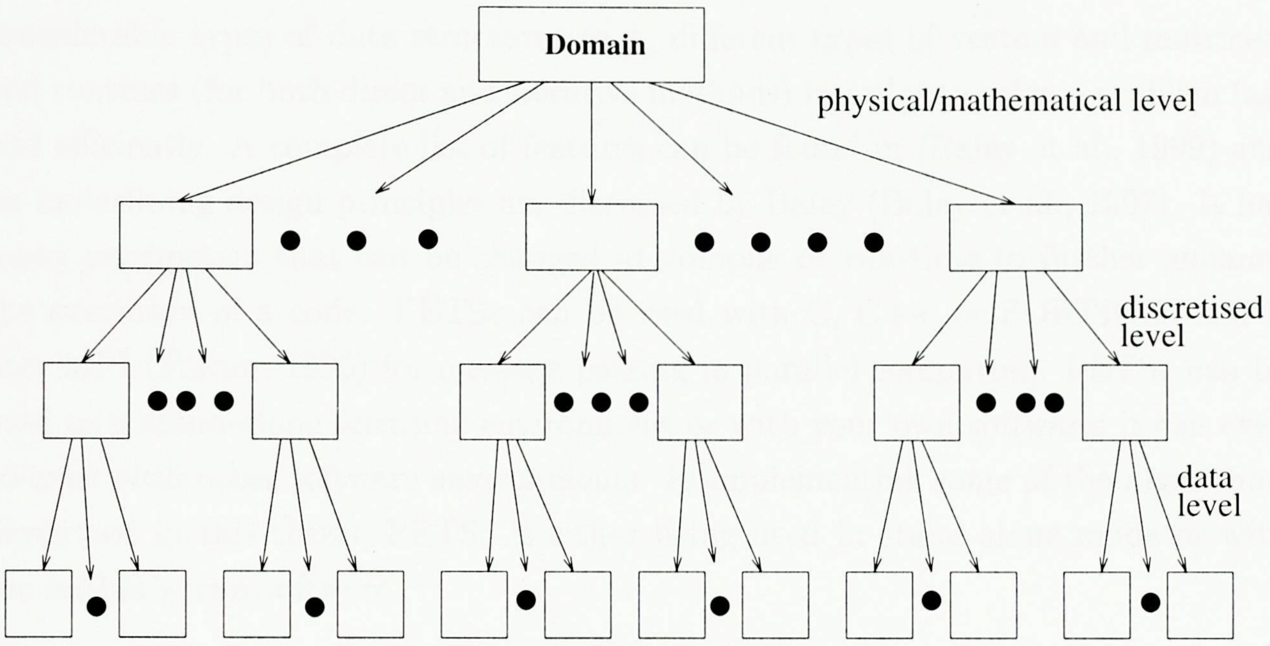


Figure 2.1: A hierarchy of decomposition.

software may be used to solve the subproblems. However, if there are discontinuities or inhomogeneities in the resulting "first-level" subproblems then further domain decompositioning may be applied (domain decompositioning may be applied until continuous and homogeneous subproblems are produced). At each level in the hierarchy a dedicated software may be employed to solve the corresponding subproblems.

If existing software is used to solve some or all of the subdomains then those software themselves may employ domain decomposition methods. Software such as PSPARSLIB (Saad et al., 1998), BlockSolve95 (Jones and Plassmann, 1995) and PETSc (Balay et al., 1999) are used to solve general sparse matrix systems (such as the ones dealt in this thesis) in sequential and parallel computing environments. The above software use Schwarz methods as preconditioners and data partitioning in parallel routines. The PETSc (Balay et al., 1999) software is used to implement some of the algorithms developed in this thesis.

PETSc is a public domain software developed at Argonne National Laboratory. Initially developed in large part to aid the research in parallel domain decomposition methods it is now a powerful scientific computing environment which is suitable for both sequential and parallel computing. It is being used extensively by the scientific computing community to solve large classes of significant problems. It provides considerable types of data structures (e.g., different types of vectors and matrices) and routines (for both direct and iterative methods) in order to solve a problem fast and efficiently. A complete list of features can be found in (Balay et al., 1999) and its underlining design principles are discussed by Balay (Balay et al., 1997). It has many parameters that can be changed at compile or run-time to further enhance the execution of a code. PETSc can be used with C, C++ or FORTRAN and it uses MPI (Forum, 1995) for message passing in parallel computing. PETSc can be used as a stand-alone scientific environment or with your own software; it can even co-exist with other software environments. In implementing some of the algorithms developed in this thesis, PETSc is either being used in stand-alone mode or with the author's own software.

Chapter 3

A metal cutting problem

The properties of a piece of metal may alter and the quality of the cut may degrade considerably due to high temperatures generated in a cutting process. Therefore, the determination of the temperature distribution due to the application of a cutting tool is of industrial interest. An accurate simulation of the temperature distribution of the metal, subject to cutting, is vital in order to lengthen the life time of the cutting tool and to guarantee the quality of the cutting. Particularly, the real-time simulation of the temperature distribution is vital in order to control the cutter speed and the coolant application. It is important to be able to regulate the cutter speed and coolant application in order to keep the temperature (especially at the cutter points) below a threshold. When the temperature rises above the threshold this will cause deformation of the metal or it may become fatigued. In reality, the accurate measurement of temperature at the cutter points is not possible. Therefore, a direct problem cannot be formulated; inverse methods can be used to retrieve the temperature at these points. It has been shown that accurate estimates can be obtained using such methods (Beck et al., 1985), but the inverse problem is more difficult to solve analytically than direct problems. Therefore, various approximation methods have been developed to solve such inverse problems. These include graphical (Stolz, 1960), polynomial (Frank, 1963), Laplace transform (Krzysztof et al., 1981), dynamic programming (Trujillo, 1978), finite difference (D'Souza, 1975), finite elements (Krutz et al., 1978). Here we use finite volume based methods.

One assumption that has been made in the present investigation concerning cutting is that the application of a cutter at a point is equivalent to the application of an external heat source at that point. This assumption is physically sensible and effectively reduces the problem into the inverse determination of heat sources. A simplified mathematical model is discussed and formulated. To determine the heat source, certain assumptions, based on physical reasoning, have been made. The heat source strength is then derived. In the development of numerical algorithms for the cutting problem, a domain decomposition method is applied to the mathematical model. As will be explained in the following section (section 3.1), domain decomposition generates well-defined, homogeneous and continuous subproblems whereas the original problem is ill-posed and discontinuous.

Three algorithms are developed to solve the metal cutting problem and are described in the following section. One algorithm is based on an explicit method (Palansuriya et al., 1998) and the other two are implicit methods (Palansuriya et al., 1999). Domain decomposition (DD) is first used to partition the original domain into subdomains, each containing a properly connected, well-formulated and continuous subproblem. The implementations are carried out using FORTRAN 77. One implicit algorithm is implemented by coupling the state-of-the-art PETSc (Portable, Extensible Toolkit for Scientific Computation) (Balay et al., 1999) software with in-house (author's own) software in order to solve the subproblems. The second implicit algorithm is implemented completely within PETSc. A 2D example is used to test the algorithms and various comparisons are made.

The following novel contributions are made in this chapter. Firstly, a domain decomposition method is proposed based on the problem partition concept explained in Chapter 2. Secondly, a unique and accurate source retrieval method is developed. Finally three new algorithms are developed to solve nonlinear time dependent inverse source problems.

It should be pointed out that the domain decomposition method, as will be shown in the following sections, not only facilitates the use of existing numerical method but also promotes the software re-usability.

3.1 The dimensionless 2d nonlinear metal cutting problem

The metal cutting problem considered here is a 2d thin sheet of metal defined in the domain $D = \{(x, y) : 0 < x < 1 \text{ and } 0 < y < 1\}$. The material properties are assumed to be homogeneous across the domain of interest and the following assumptions are made for idealised cutting :

(a) the application of a cutting tool at the cutter points is equivalent to the application of a heat source at these points,
(b) no phase changes occur during cutting and
(c) the thickness of the cutter is negligible.

The cutting is considered to be applied along a line parallel to the y-axis at $x = x_c$. These assumptions lead to the following dimensionless 2d nonlinear, unsteady, parabolic, heat conduction equation,

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}(k(u)\frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}(k(u)\frac{\partial u}{\partial y}) + Q_c(y,t)\delta(x - x_c) \in D, \tag{3.1}$$

subject to the initial condition $u(x, y, 0) = U_i(x, y)$, boundary conditions $u(0, y, t) = B_0(y, t)$, $u(1, y, t) = B_1(y, t)$, $u(x, 0, t) = C_0(x, t)$ and $u(x, 1, t) = C_1(x, t)$. Here $u(x, y, t)$ is a dimensionless temperature distribution, $k(u)$ is the conductivity of the metal, $Q_c(y, t)$ is the unknown source being applied at $x = x_c$, $\delta(x - x_c)$ is the Dirac delta function and U_i, B_0, B_1, C_0 and C_1 are known functions.

Equation (3.1) has two unknowns, namely $u(x, y, t)$ and $Q_c(y, t)$. The boundary conditions and initial conditions would allow $u(x, y, t)$ to be determined if $Q_c(y, t)$ is known. Therefore, one has to extract certain information from the temperature field as discussed in Chapter 1. One possible method is to include temperature sensors at $x = x_s$, such that $0 < x_s < x_c < 1$ in order to obtain the temperature at that point. Let the temperature measured by means of the temperature sensors be $u(x_s, y, t) = u^*(y, t)$. The sensors effectively introduce the known function $u(x_s, y, t) = u^*(y, t)$ and may be used in a problem partitioning of the mathematical model.

Note that, it is not necessary to have $x_s < x_c$. As the measured temperatures are only used to retrieve temperatures at the cutting points, similar problem partitioning can be generated for $0 < x_c < x_s < 1$. The measured temperatures are used to retrieve temperatures at the cutting points. Such inverse methods avoid the basic difficulties of a direct method since remote temperatures can be measured more easily and accurately.

3.2 Problem partitioning

Problem partitioning was defined in the previous chapter as a domain decomposition method applied at the mathematical/physical problem level (Lai, 1994). In order to solve the inverse problem given in (3.1) with the additional condition available at $x = x_s$, problem partitioning is carried out to produce three subdomains, such that each subproblem may be solved using a different numerical algorithm. The three subdomains are:

$$D_1 = \{(x, y) : 0 < x < x_s \text{ and } 0 < y < 1\}$$
$$D_2 = \{(x, y) : x_s < x < x_c \text{ and } 0 < y < 1\}$$
$$D_3 = \{(x, y) : x_c < x < 1 \text{ and } 0 < y < 1\}$$

Figure 3.1 shows a visual representation of the subdomains. This problem partition-

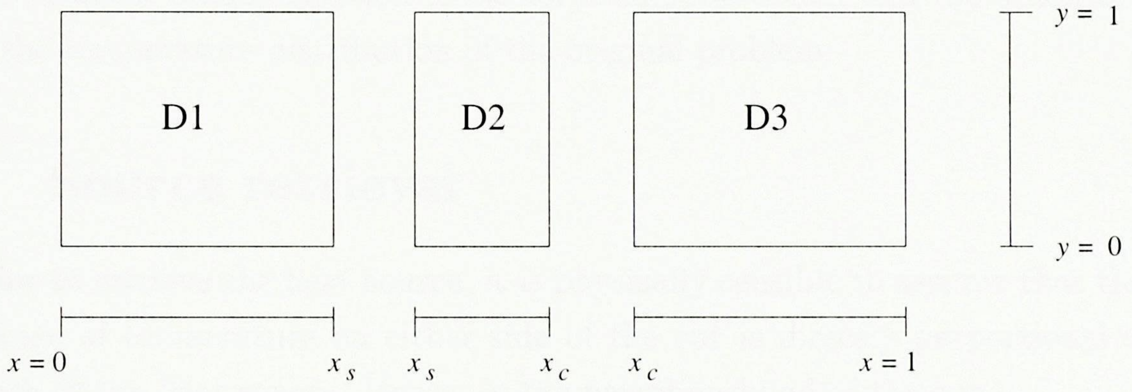


Figure 3.1: A visual representation of the problem partitioning

ing is able to remove the unknown source term $Q_c(y, t)$ and the Dirac delta function which are associated with the differential equation. The three subproblems (SPs) can be written as follows:

SP_1: $\frac{\partial u_1}{\partial t} = \frac{\partial}{\partial x}(k(u_1)\frac{\partial u_1}{\partial x}) + \frac{\partial}{\partial y}(k(u_1)\frac{\partial u_1}{\partial y}) \in D_1$
subject to $u_1(x, y, 0) = U_i(x, y)$, $u_1(0, y, t) = B_0(y, t)$,
$u_1(x_s, y, t) = u^*(y, t)$, $u_1(x, 0, t) = C_0(x, t)$, $u_1(x, 1, t) = C_1(x, t)$.

SP_2: $\frac{\partial u_2}{\partial t} = \frac{\partial}{\partial x}(k(u_2)\frac{\partial u_2}{\partial x}) + \frac{\partial}{\partial y}(k(u_2)\frac{\partial u_2}{\partial y}) \in D_2$
subject to $u_2(x, y, 0) = U_i(x, y)$, $u_2(x_s, y, t) = u^*(y, t)$,
$\frac{\partial u_2(x_s,y,t)}{\partial x} = \frac{\partial u_1(x_s,y,t)}{\partial x}$, $u_2(x, 0, t) = C_0(x, t)$, $u_2(x, 1, t) = C_1(x, t)$.

SP_3: $\frac{\partial u_3}{\partial t} = \frac{\partial}{\partial x}(k(u_3)\frac{\partial u_3}{\partial x}) + \frac{\partial}{\partial y}(k(u_3)\frac{\partial u_3}{\partial y}) \in D_3$
subject to $u_3(x, y, 0) = U_i(x, y)$, $u_3(x_c, y, t) = u_2(x_c, y, t)$,
$u_3(1, y, t) = B_1(y, t)$, $u_3(x, 0, t) = C_0(x, t)$, $u_3(x, 1, t) = C_1(x, t)$.

Since the temperature values are given at $y = 0$, $y = 1$, $x = 0$ and there are temperature sensors located at $x = x_s$, Dirichlet boundary conditions are defined at the boundary of D_1. The solution of the differential equation provides the required data to calculate the heat flux $\frac{\partial u_1}{\partial x}(x_s, y, t)$. Therefore, with the knowledge of the temperatures $u_2(x_s, y, t)$ acquired by the temperature sensors at $x = x_s$, an initial value problem can be formulated in D_2. $u_2(x_c, y, t)$ values may be obtained by solving this initial value problem, marching along the x direction. Finally, with the calculated temperatures $u_2(x_c, y, t)$, another Dirichlet problem can be formulated in D_3. The above three subproblems are well-defined (Beck et al., 1985) (Zwillinger, 1989). Hence a unique solution exists for each subproblem and the union of these gives the temperature distribution of the original problem.

3.3 Source retrieval

In order to retrieve the heat source, it is physically sensible to assume that the rate of change of temperature on either side of the cut is directly proportional to the strength of the heat source. Hence, in the neighbourhood of the cut,

$$\frac{\partial u}{\partial t} = \alpha(y,t)Q_c(y,t)\delta(x - x_c) \tag{3.2}$$

where $\alpha > 1$ is a time dependent function that also depends on y. The condition $\alpha > 1$ is to ensure an increase in temperature at the cut. Integrating (3.2) across the cut at a given value of y gives

$$\int_{x_c^-}^{x_c^+} \frac{\partial u}{\partial t} dx = \alpha(y,t)Q_c(y,t) \tag{3.3}$$

Here x_c^- denotes a spatial point just to the left of x_c and x_c^+ denotes a spatial point just to the right of x_c. Similarly, integrating (3.1) across the cut and equating the result to (3.3) leads to

$$k(u)\frac{\partial u}{\partial x}|_{x_c^+} - k(u)\frac{\partial u}{\partial x}|_{x_c^-} + \frac{\partial}{\partial y}(k(u)\frac{\partial u}{\partial y})(x_c^+ - x_c^-) + Q_c(y,t) = \alpha(y,t)Q_c(y,t) \tag{3.4}$$

Assumption (c) (see Section 3.1) suggests that equation (3.4) can be simplified to:

$$k(u)\frac{\partial u}{\partial x}|_{x_c^+} - k(u)\frac{\partial u}{\partial x}|_{x_c^-} = (\alpha(y,t) - 1)Q_c(y,t) \tag{3.5}$$

Define the predicted heat source, Q_p, as

$$Q_p(y,t) = k(u)\frac{\partial u}{\partial x}|_{x_c^+} \; - \; k(u)\frac{\partial u}{\partial x}|_{x_c^-} \; = \; \beta(y,t)Q_c(y,t) \tag{3.6}$$

where $\beta = \alpha - 1$. Then substitute Q_p into (3.1) to replace Q_c and solve the equation as a direct problem with u_p being the corresponding temperature distribution. It is clear that the ratio of $Q_p : Q_c$ must be equivalent to the ratio between their corresponding temperature distributions, i.e.,

$$\frac{Q_p(y,t)}{Q_c(y,t)} \; = \; \frac{u_p(y,t)}{u(x_c,y,t)} \; = \; \beta(y,t) \tag{3.7}$$

Therefore, in order to determine the proportional factor β, one needs to solve (3.1) as a direct problem by using the predicted source $Q_p(y,t)$ in order to obtain the corresponding temperature distribution $u_p(y,t)$, then use β as determined in (3.6) to retrieve the true source Q_c.

3.4 Algorithms

If the development of a numerical method relates to the use of specialised software dedicated to a particular problem, it is possible to have a more effective algorithm. In the present problem, the original mathematical model is divided into three subproblems. Two of which belong to the same class (Dirichlet problems), while one belongs to an initial value problem along the x direction. This thesis follows such a concept of using specialised (but standard) software to solve homogeneous subproblems.

The following three algorithms are developed using the problem partition described in section 3.2 as a framework. In the algorithms, the derivatives are discretised using a first-order forward difference approximation of the temporal derivative and a second-order Finite Volume (FV) approximation of the spatial derivatives (Versteeg and Malalasekera, 1995). It is an aim of this thesis to find out if such a discretisation scheme is accurate enough to retrieve the unknown temperature field and, particularly, the source term.

3.4.1 Algorithm 1

To solve the problems in SP_1 and SP_3, a first-order forward difference approximation of the temporal derivative and a second-order FV approximation of the spatial derivatives leads to a five-point explicit scheme. Dropping the subscript used in denoting the subdomains, the explicit scheme for the subdomains D_1 and D_3 can be written as,

$$u_{i,j}^{(n+1)} = r_x b_i^{(n)} u_{i-1,j}^{(n)} + r_x a_i^{(n)} u_{i+1,j}^{(n)} + (1 - r_x a_i^{(n)} - r_x b_i^{(n)} - r_y c_j^{(n)} - r_y d_j^{(n)}) u_{i,j}^{(n)} + r_y d_j^{(n)} u_{i,j-1}^{(n)} + r_y c_j^{(n)} u_{i,j+1}^{(n)} \tag{3.8}$$

where (i, j) denotes the (i, j)-th grid point,

$$r_x = \frac{\Delta t}{(\Delta x)^2} \tag{3.9}$$

$$r_y = \frac{\Delta t}{(\Delta y)^2} \tag{3.10}$$

$$a_i^{(n)} = \frac{k_{i+1,j}^{(n)} + k_{i,j}^{(n)}}{2} \tag{3.11}$$

$$b_i^{(n)} = \frac{k_{i-1,j}^{(n)} + k_{i,j}^{(n)}}{2} \tag{3.12}$$

$$c_j^{(n)} = \frac{k_{i,j+1}^{(n)} + k_{i,j}^{(n)}}{2} \tag{3.13}$$

$$d_j^{(n)} = \frac{k_{i,j-1}^{(n)} + k_{i,j}^{(n)}}{2} \tag{3.14}$$

(n) denotes the time-step, Δt is the step size along the temporal axis and Δx, Δy are the grid spacings along the spatial axis x, y, respectively.

The initial value problem in SP_2 is solved by employing a second-order Euler Predictor-Corrector (P-C) method along the x-axis for each time-step. Again, the spatial derivatives are discretised using second-order FV approximations and the time derivative with a first-order finite difference approximation. The two step P-C method can be written as:

$$\begin{pmatrix} u \\ v \end{pmatrix}^{*} = \begin{pmatrix} u \\ v \end{pmatrix} + \Delta x \underline{f}\ , \quad \begin{pmatrix} u \\ v \end{pmatrix}^{\text{new}} = \begin{pmatrix} u \\ v \end{pmatrix} + \frac{\Delta x}{2} \left\{ \underline{f} + \underline{f}^{*} \right\}\ , \tag{3.15}$$

where,

$$v = \frac{\partial u}{\partial x} \tag{3.16}$$

$$\underline{f} = \underline{f}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} v \\ \frac{1}{k(u)}\left(\frac{\partial u}{\partial t} - \frac{\partial}{\partial y}\left(k(u)\frac{\partial u}{\partial y}\right) - k'(u)v^2\right) \end{pmatrix} \tag{3.17}$$

$$\underline{f}^* = \underline{f}\begin{pmatrix} u \\ v \end{pmatrix}^* \tag{3.18}$$

A second order spatially accurate solution may be obtained for each of the three subproblems. Therefore, it is expected to have a second order spatially accurate global solution for the inverse problem (3.1). The effect of the local truncation error for SP_2 is minimized because of the small size of the subdomain which usually consists of only a few Euler P-C steps. All experiments give stable results as long as the CFL condition $r_x, r_y \leq 0.25$ is satisfied.

3.4.2 Algorithm 2

Newton's method is applied to D_1 and D_3 and leads to an implicit method. Using Newton's method, the iterative scheme for solving an equation of the type $F(U) = 0$ may be written as,

```
{ Solve
    $J(U_m)V_m = -F(U_m)$
  Update
    $U_{m+1} = U_m + V_m$
} Repeat until $||V|| < Tol$
```

For the metal cutting problem,

$$F(U) = \frac{\partial}{\partial x}(k(U)\frac{\partial U}{\partial x}) + \frac{\partial}{\partial y}(k(U)\frac{\partial U}{\partial y}) - \frac{\partial U}{\partial t} \tag{3.19}$$

and Jacobian, $J(U)$, defined as $J(U) = F'(U)$ is,

$$J(U) = k''(U)(\tfrac{\partial U}{\partial x})^2 + 2k'(U)\tfrac{\partial U}{\partial x}\tfrac{\partial}{\partial x} + k'(U)\tfrac{\partial^2 U}{\partial x^2} + k(U)\tfrac{\partial^2}{\partial x^2}$$

$$+ k''(U)(\frac{\partial U}{\partial y})^2 + 2k'(U)\frac{\partial U}{\partial y}\frac{\partial}{\partial y} + k'(U)\frac{\partial^2 U}{\partial y^2} + k(U)\frac{\partial^2}{\partial y^2} - \frac{\partial}{\partial t} \tag{3.20}$$

The matrix generated due to the discretisation of $J(U)$ is a penta-diagonal sparse matrix. Hence, any existing (preferably robust and efficient) software for solving such a matrix may be employed to solve D_1 and D_3. PETSc is used to solve the linearised systems in D_1 and D_3. Subdomain D_2 uses the same Euler P-C method as in Algorithm 1. This algorithm is fully implicit.

3.4.3 Algorithm 3

All three subdomains are solved by using the Newton's method as described in algorithm 2. This algorithm leads to a global linearised system for all three subdomains and it is solved by using PETSc.

The implementation of the above three algorithms highlights the fact that the generation of homogeneous and continuous subproblems due to problem partitioning facilitates the use of general purpose libraries, as well as the coupling of subdomain-specialized software solutions.

3.5 Numerical examples

Numerical results are obtained for equation (3.1) with $x_s = 0.5$, $x_c = 0.6$, $U_i(x, y) = 0$, $B_0(y, t) = 0$, $B_1(y, t) = 0$, $C_0(x, t) = 0$ and $C_1(x, t) = 0$. Sensor points are modelled as $u^*(y, t) = \sigma y(y - 1)^2 \sin(\omega t)$, with $\sigma = 0.1$ and $\omega = 2\pi$. Non-linear heat conductivity is given by $k(u) = \frac{1}{1+u^2}$. Number of grid points along x-axis is 21 and along y-axis is 11. The resulting temperature distributions are shown for time $t = 0$ to $t = 0.5$ in Figures 3.2 to 3.4. The retrieved source/sink strength is shown in Figure 3.5, it reflects the shape of the function used in the modelling of sensor temperatures, i.e. a sinusoidal function in time. Figure 3.6(a) shows the comparison of temperature distribution at $y = 0.4$,$t = 0.1$ between the three Algorithms. The tolerance, Tol, in Algorithms 2 and 3 has been chosen as 10^{-8}. The numerical quality of all the three algorithms are the same. The sequential implementation of the algorithms was tested for performance using a Sun Sparc 5 workstation. The run times are shown in Figure 3.6(b). Algorithm 1 does not perform well because of a very small $\triangle t$. It is necessary to use a small $\triangle t$ in order to satisfy the CFL condition. There is no significant difference between the performance of Algorithm 2 and Algorithm 3. The latter is slightly more efficient since it solves the subproblems using one system matrix.

3.6 Validation

In this Section the algorithms are to be validated by using a more difficult source term, one which contains discontinuities as well as a gentle slope in the source.

To compare the accuracy of the retrieved temperature field and the source term, an exact source such as the one given below is used. Equation (3.1) is then solved as a direct problem to obtain the noise-free sensor temperature as well as the temperature distribution. Then an inverse problem is formulated by assuming that the noise-free sensor temperatures are given, the source term being unknown. The temperature distribution from the direct problem is compared with the temperature distribution obtained from the inverse problem.

The problem considered here is as defined in Section 3.5 but with the following source term,

$$Q(x_c, y, t) = \begin{cases} 0, & t < \frac{y}{v} \\ 3\pi(t - \frac{y}{v}), & \frac{y}{v} \le t < 0.05 + \frac{y}{v} \\ 0.15\pi, & 0.05 + \frac{y}{v} \le t \le 0.09 + \frac{y}{v} \\ 0, & t > 0.09 + \frac{y}{v} \end{cases}$$

where v is the forward velocity of the cutter. For comparison of accuracy, only the unsteady source at $y = 0$ is considered. The source term at $y = 0$ is,

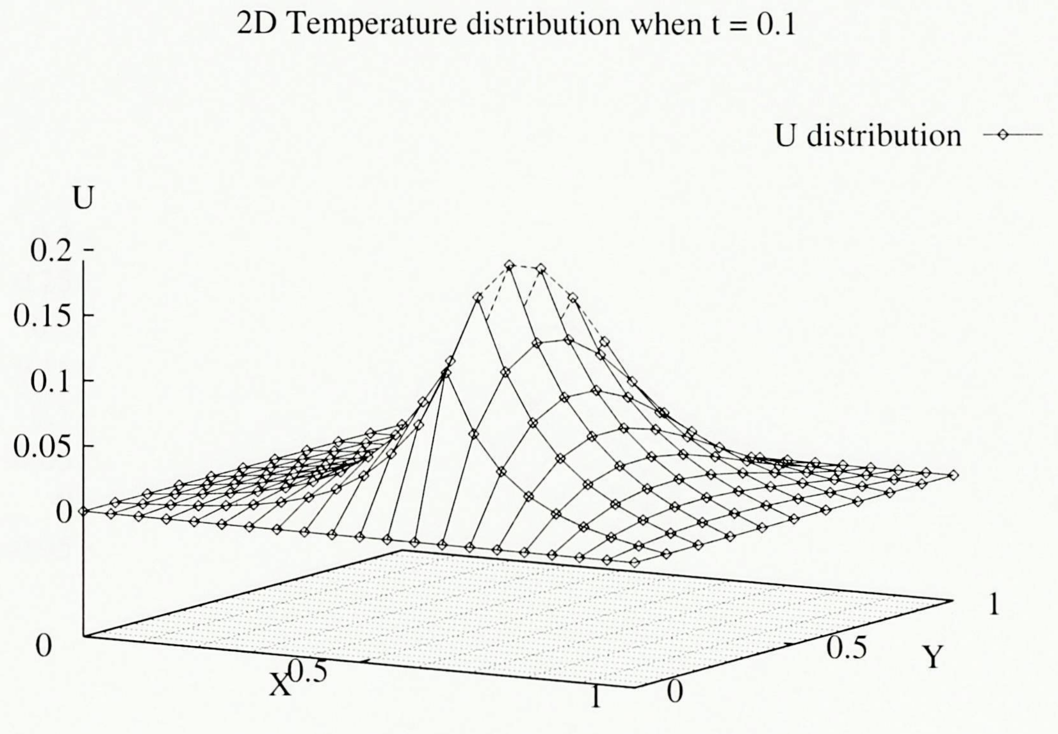
$$Q(x_c, 0, t) = \begin{cases} 3\pi t, & t < 0.05 \\ 0.15\pi, & 0.05 \le t \le 0.09 \\ 0, & t > 0.09 \end{cases}$$

The following results are obtained using the mesh configuration 40×40. Figures 3.7 and 3.8 shows the accuracy of the retrieved temperature field at various time steps using Algorithm 1 and Figure 3.9 shows its accuracy on retrieving the source term. Figures 3.10 and 3.11 show the accuracy of the retrieved temperature field when Algorithm 2 is used. Figure 3.12 shows the accuracy of source retrieval using Algorithm 2. Figures 3.13 to 3.15 illustrate the accuracy of Algorithm 3. The above Figures show that the three algorithms give temperature distributions that are almost the same as the exact answer, even for relatively coarse grid. The heat source with a discontinuity is retrieved accurately. There is a slight "shock effect" in capturing the discontinuity in the source term which reflects a small time lag in capturing it. This has no effect in the situations where the duration of the source application is predetermined since this provides a priori knowledge about the time of the source removal. Then the calculation of the source strength retrieval can be terminated when the source removal time is reached.

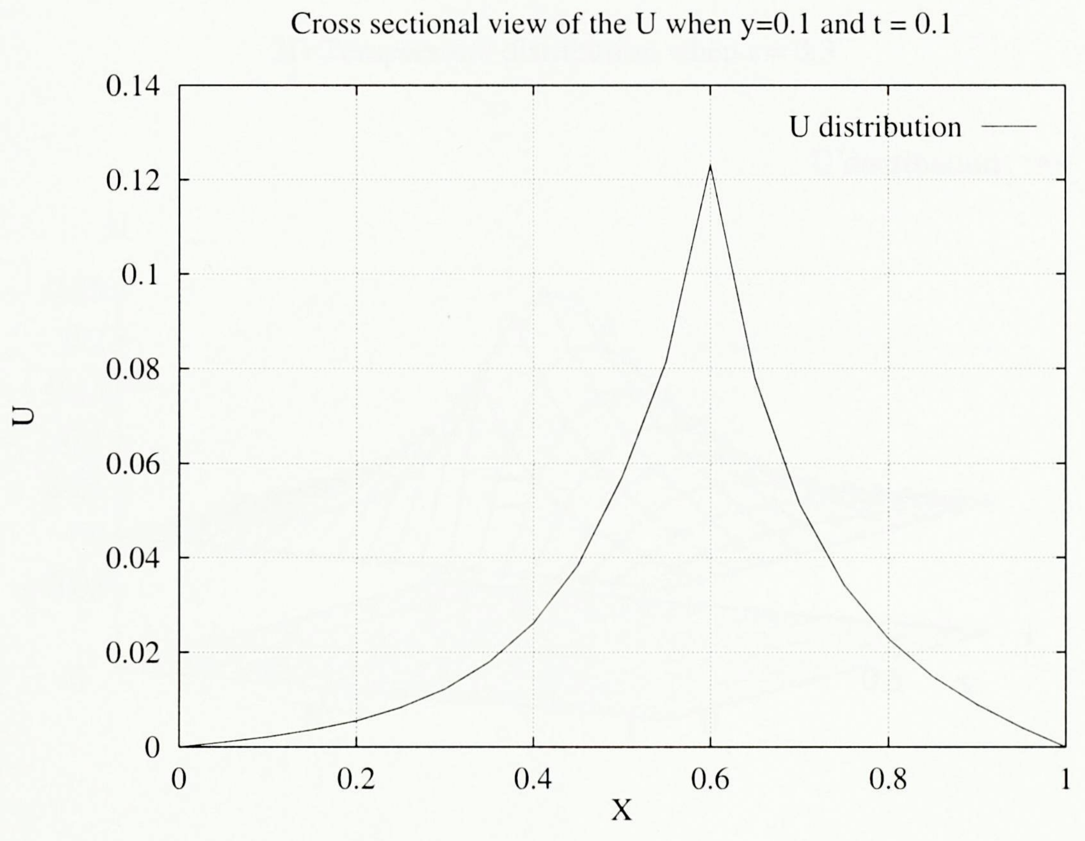
3.7 Closure

The assumptions made regarding idealised cutting are reasonable for many practical situations. However, there are situations where the assumptions will have to be reconsidered. For example, in the laser cutting of metals, the latent heat of the metal as well as the variation of liquid fraction will have to be included in the mathematical model to take into account the *phase change*. The homogeneous material properties assumption is valid, since problem partitioning (domain decomposition) can be carried out in such a way so that each subdomain will have such properties. As illustrated in developing the algorithms, problem partitioning also facilitates the use of existing numerical methods (in subdomains).

The use of alternative numerical algorithms to retrieve the unknown source term at the cutter is presented. These algorithms are developed by applying domain decomposition to the problem domain, in order to calculate the temperature field. Each algorithm is shown to retrieve the temperature field and source term accurately. The three algorithms perform the computation to the same effective accuracy. Implicit algorithms, Algorithm 2 and Algorithm 3, give better sequential execution times than Algorithm 1 (contains explicit schemes). Algorithm 3 is slightly more efficient since this only uses one system matrix.

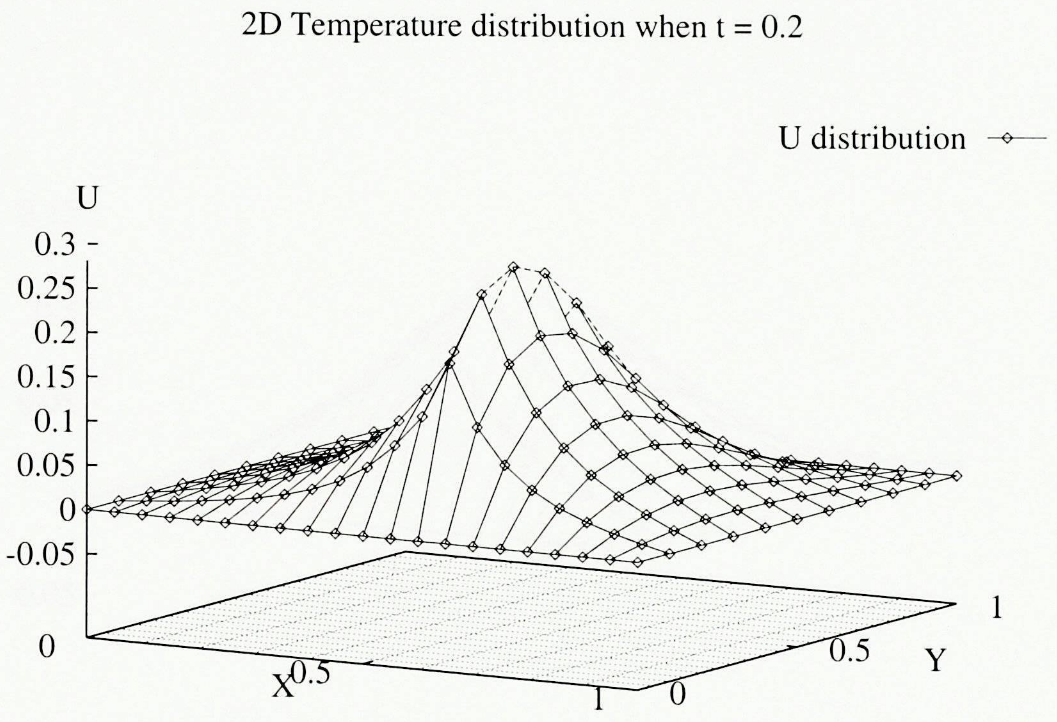


(a)

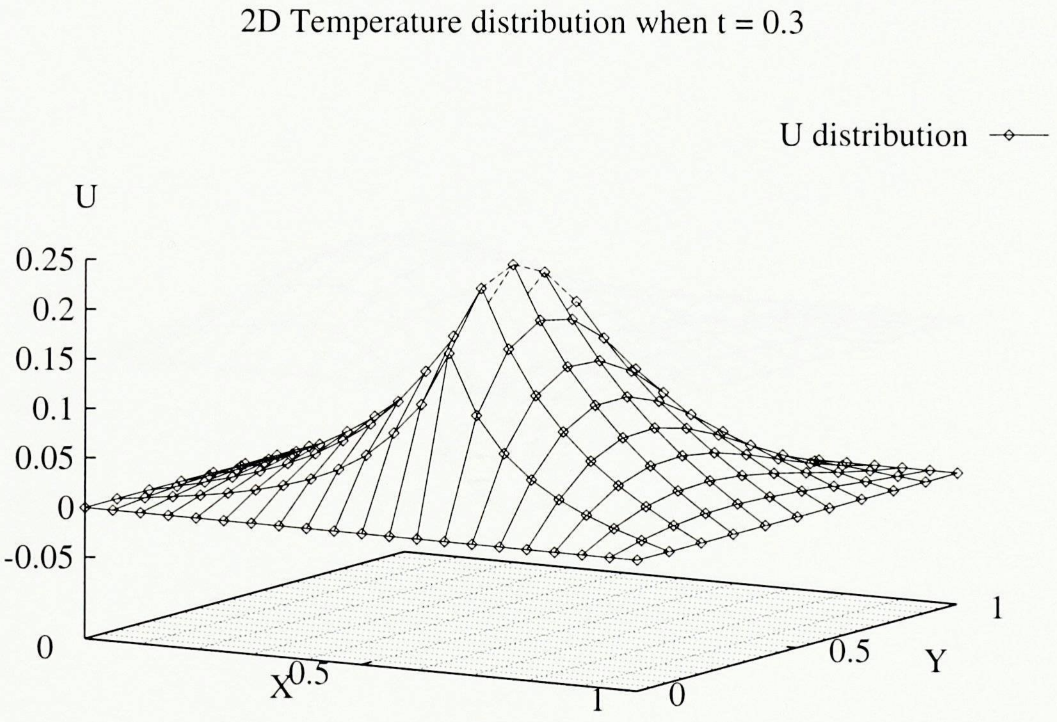


(b)

Figure 3.2: (a) Temperature distribution for $t = 0.1$, (b) a cross sectional view at $y = 0.1$ and $t = 0.1$.

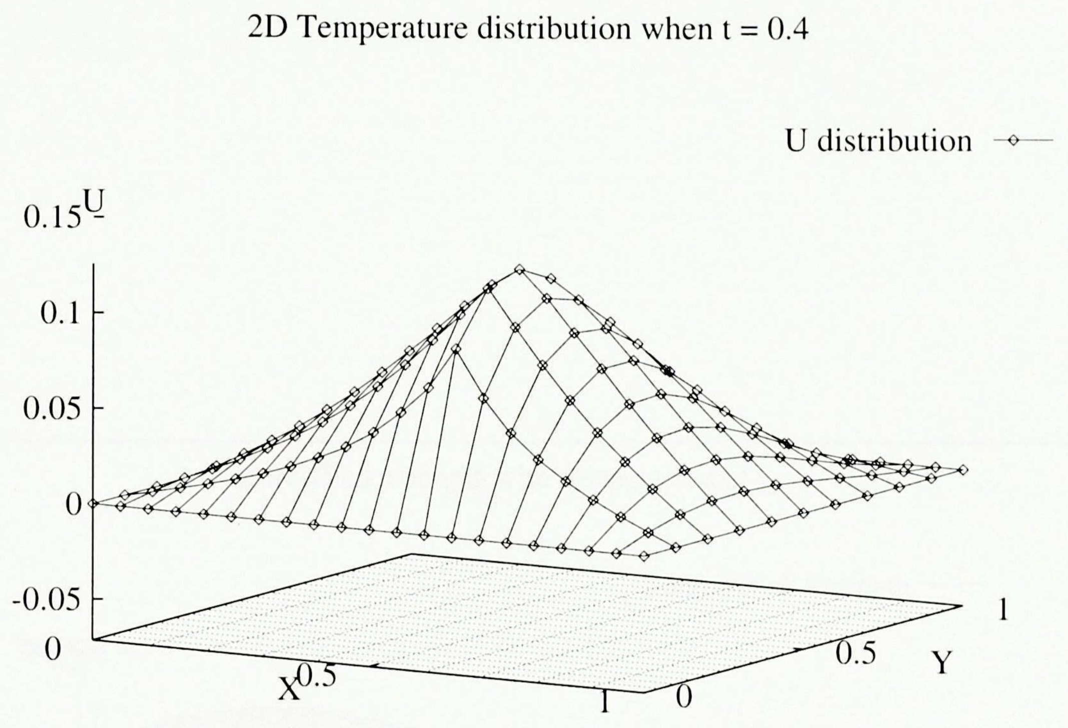


(a)

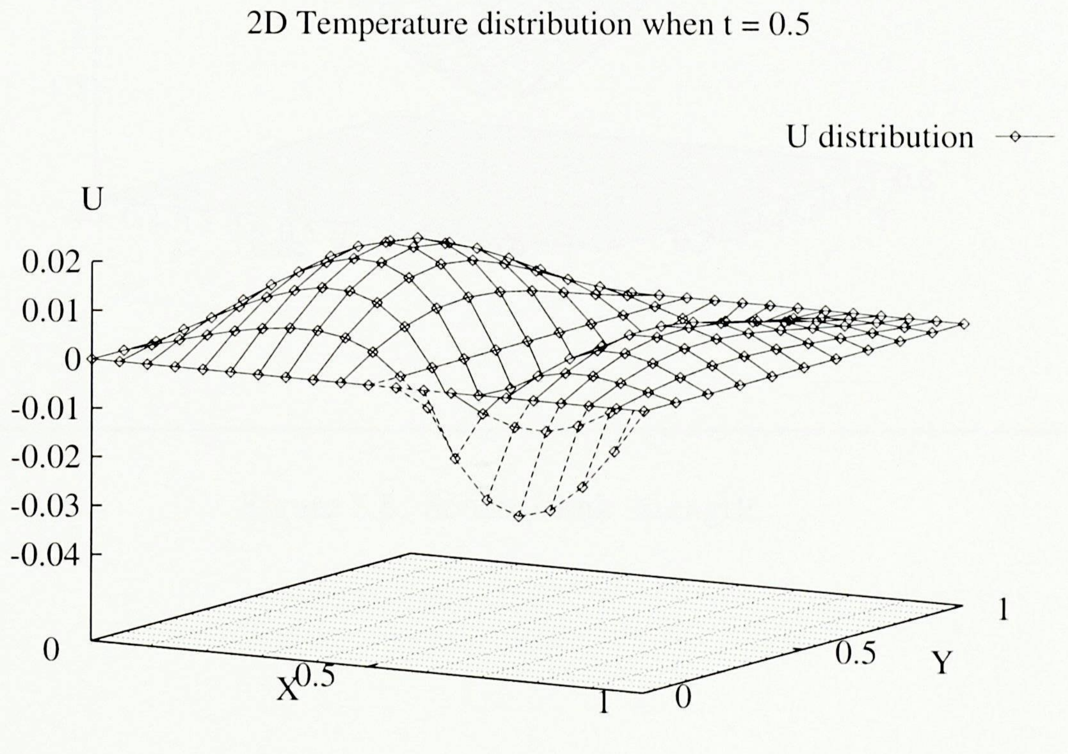


(b)

Figure 3.3: Temperature distributions for (a) $t = 0.2$ and (b) $t = 0.3$.



(a)



(b)

Figure 3.4: Temperature distributions for (a) $t = 0.4$ and (b) $t = 0.5$.

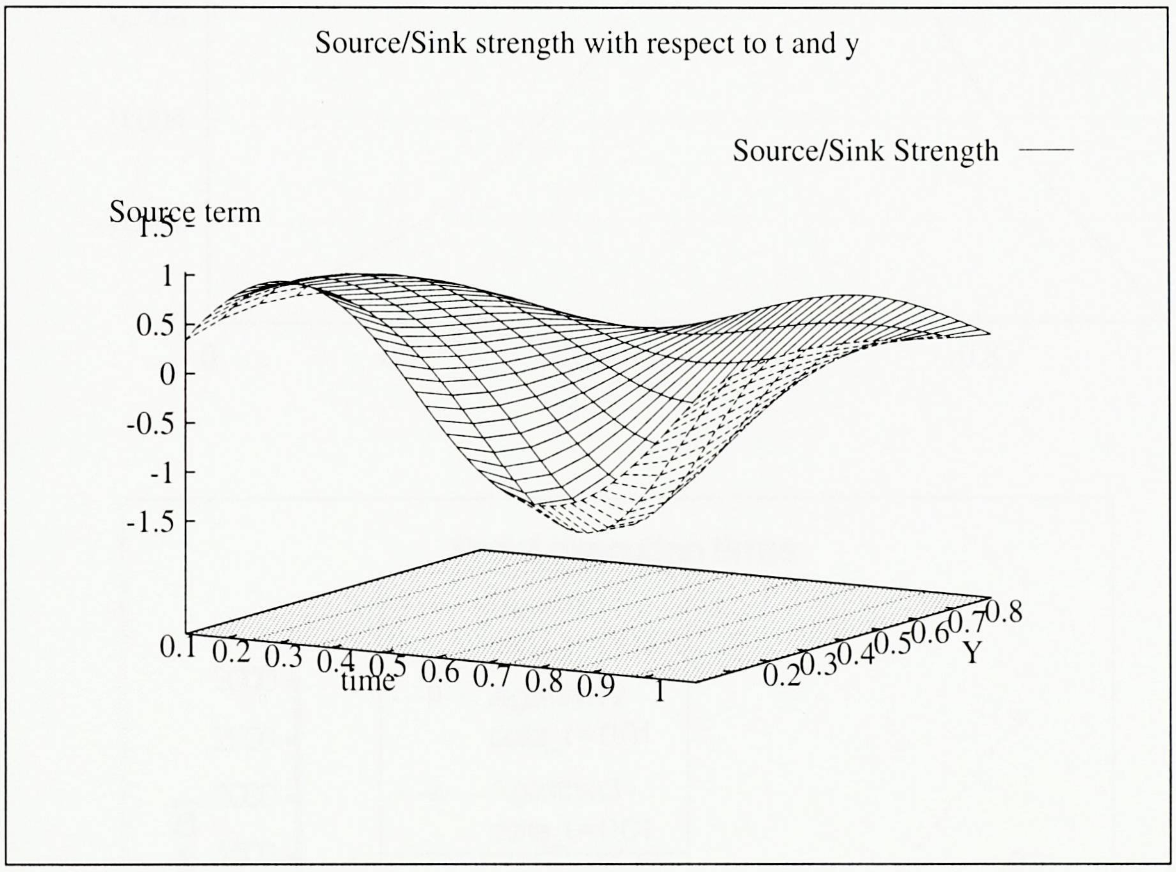
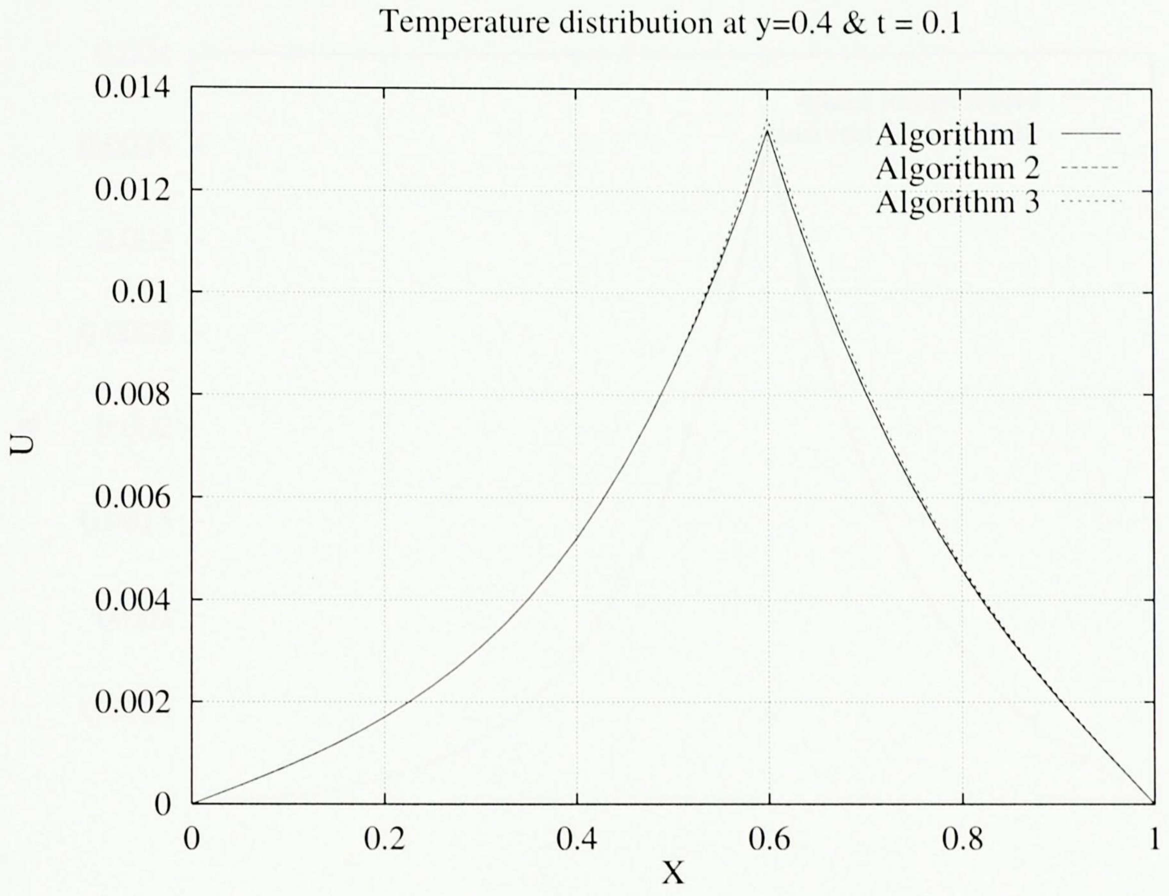
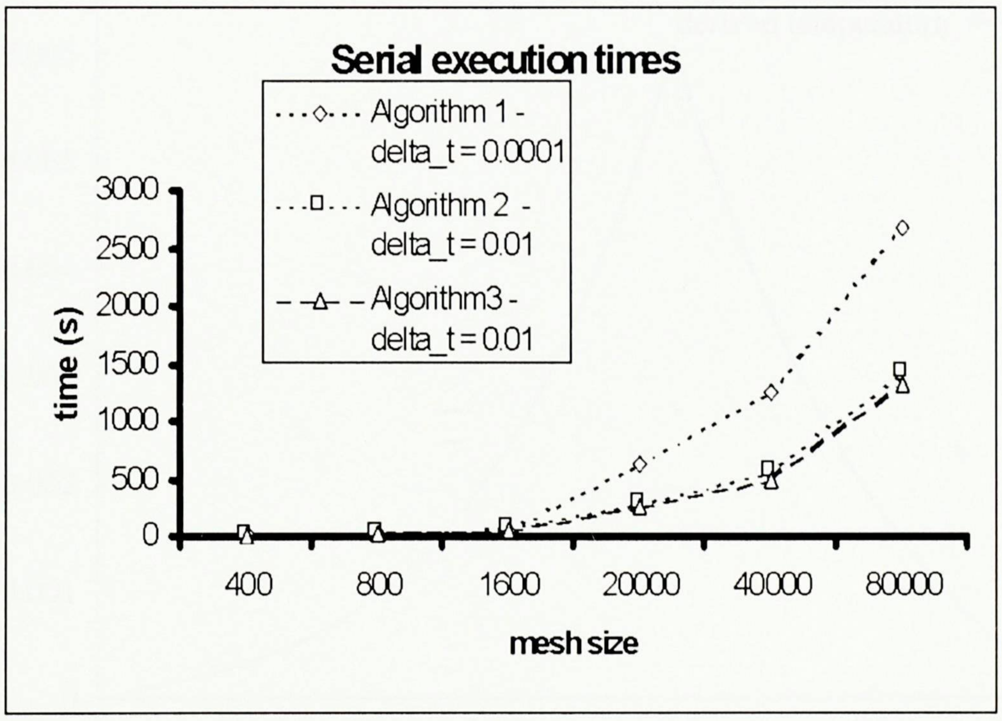


Figure 3.5: Source/Sink strength.



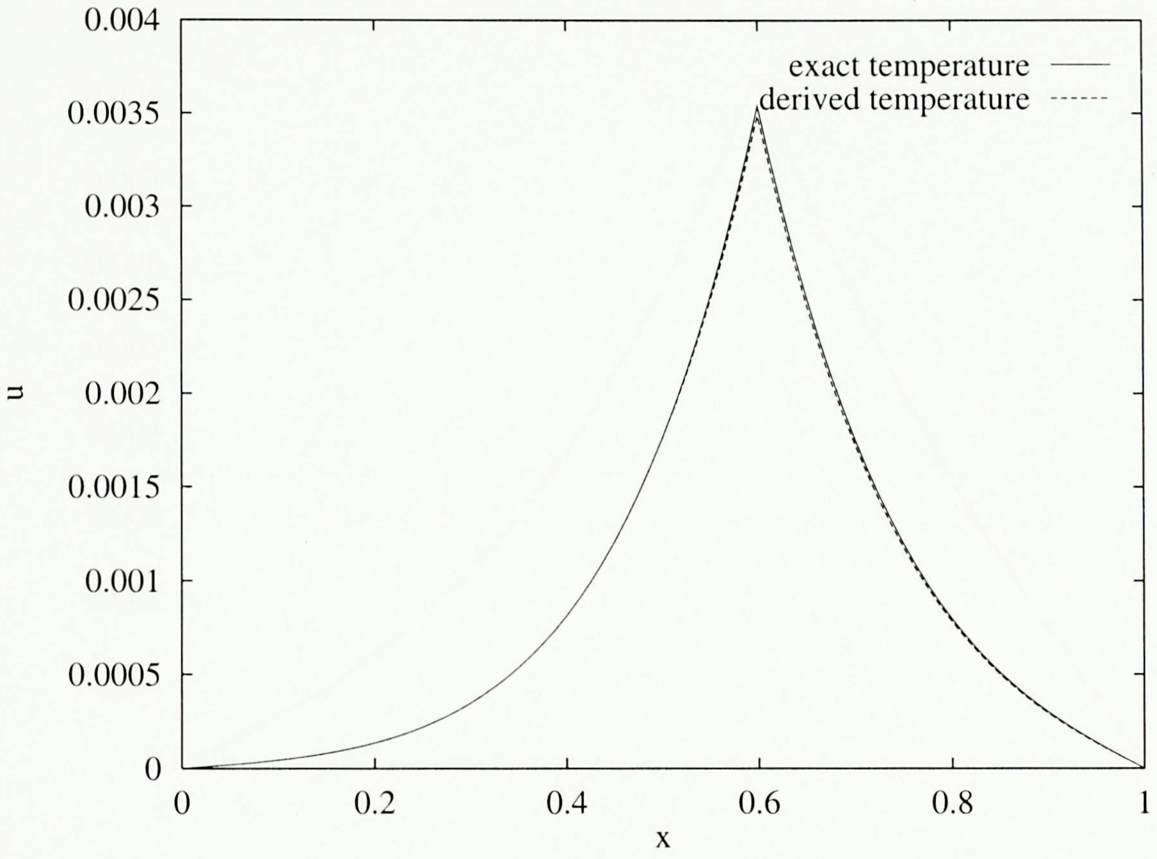
(a)



(b)

Figure 3.6: Horizontal solution profile and serial execution times for all three algorithms.

(a)



(b)

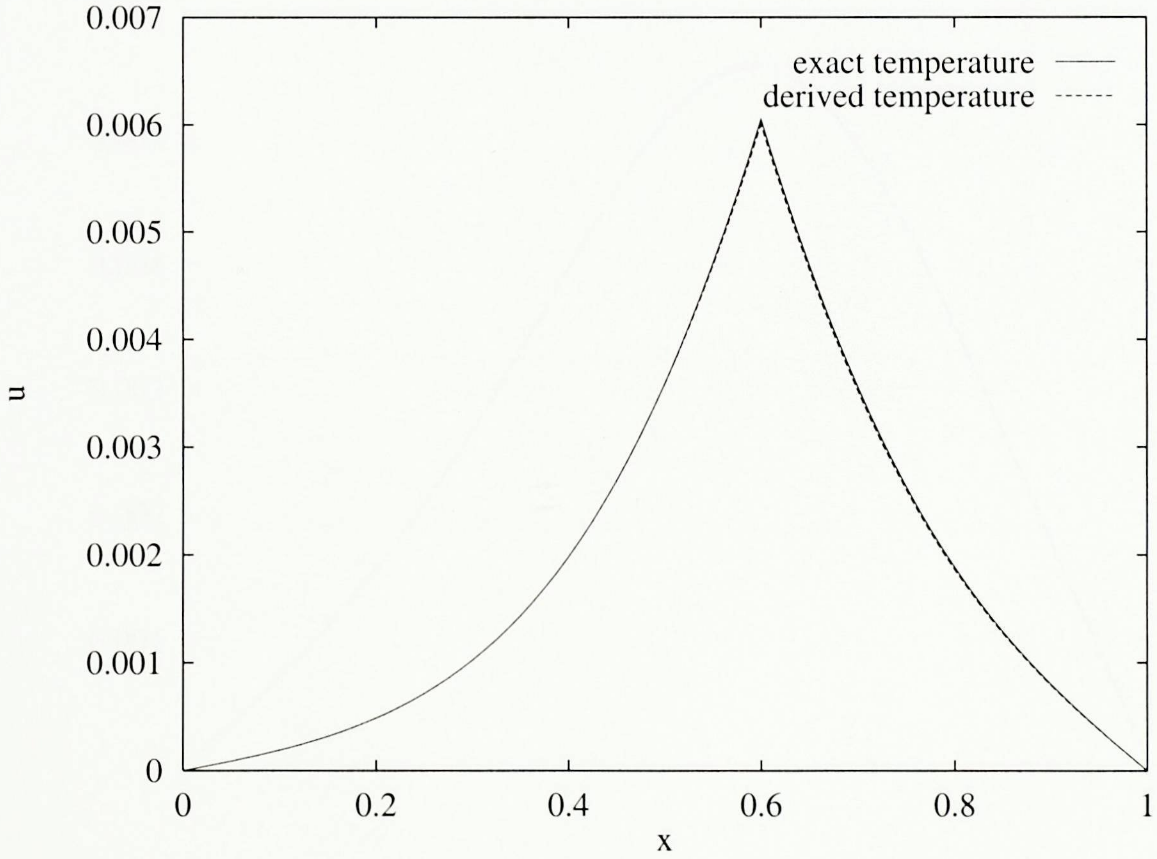
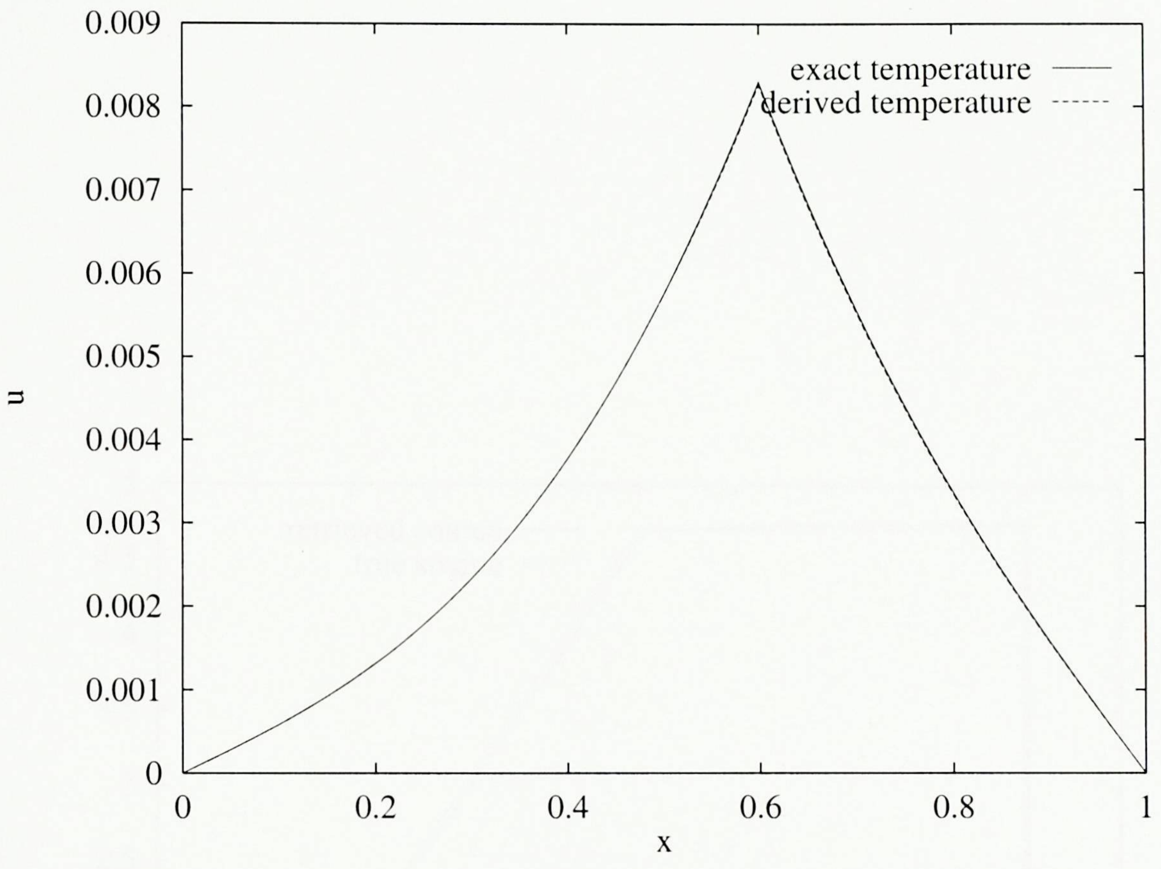


Figure 3.7: Accuracy of the retrieved temperature field, using Algorithm 1 at (a) t=0.04 and (b) t=0.06.

(a)


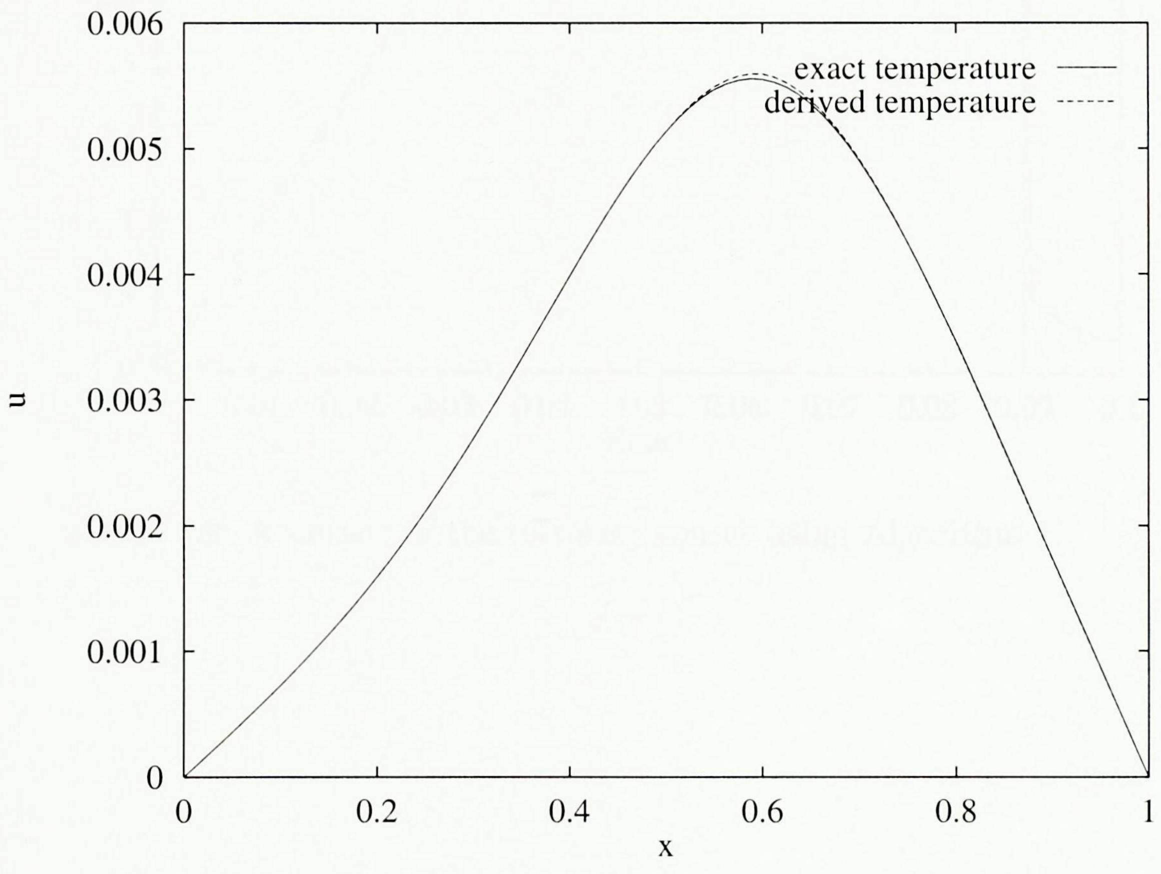
(b)


Figure 3.8: Accuracy of the retrieved temperature field, using Algorithm 1 at (a) t=0.09 and (b) t=0.1.

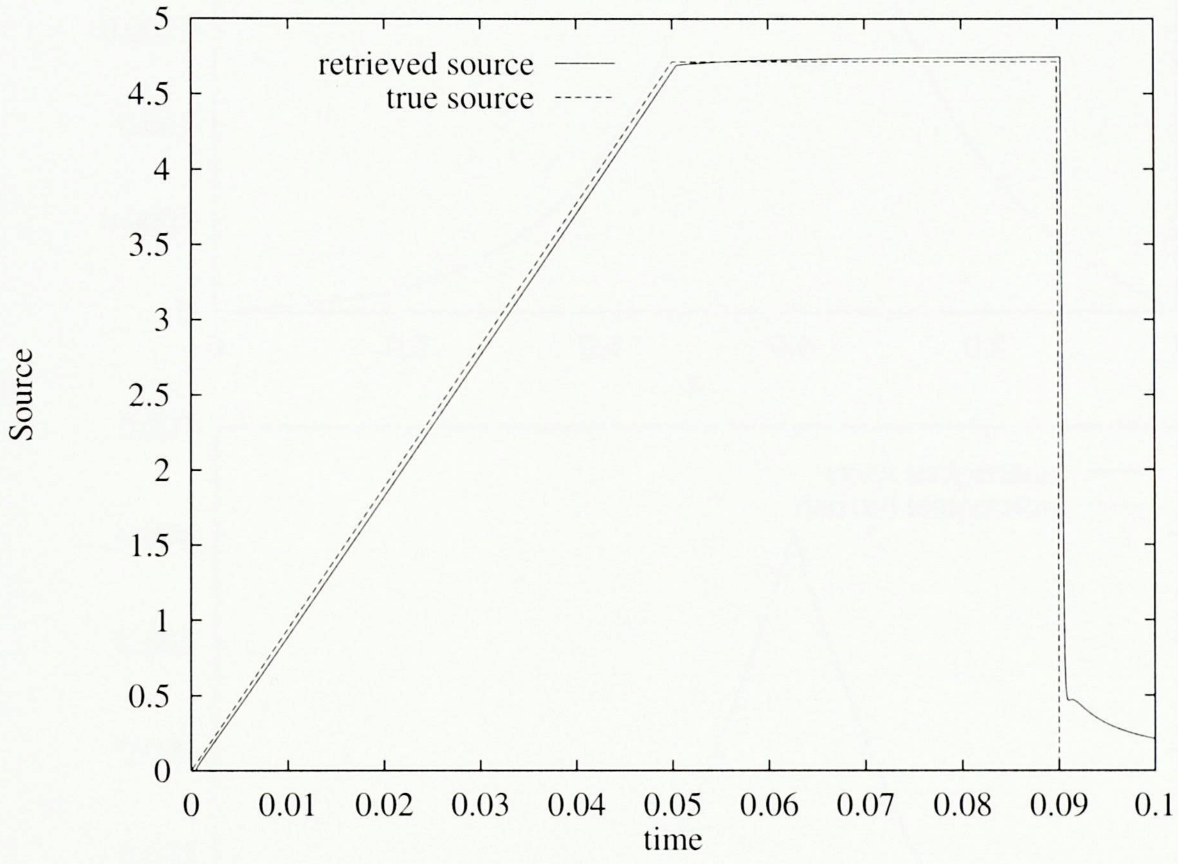
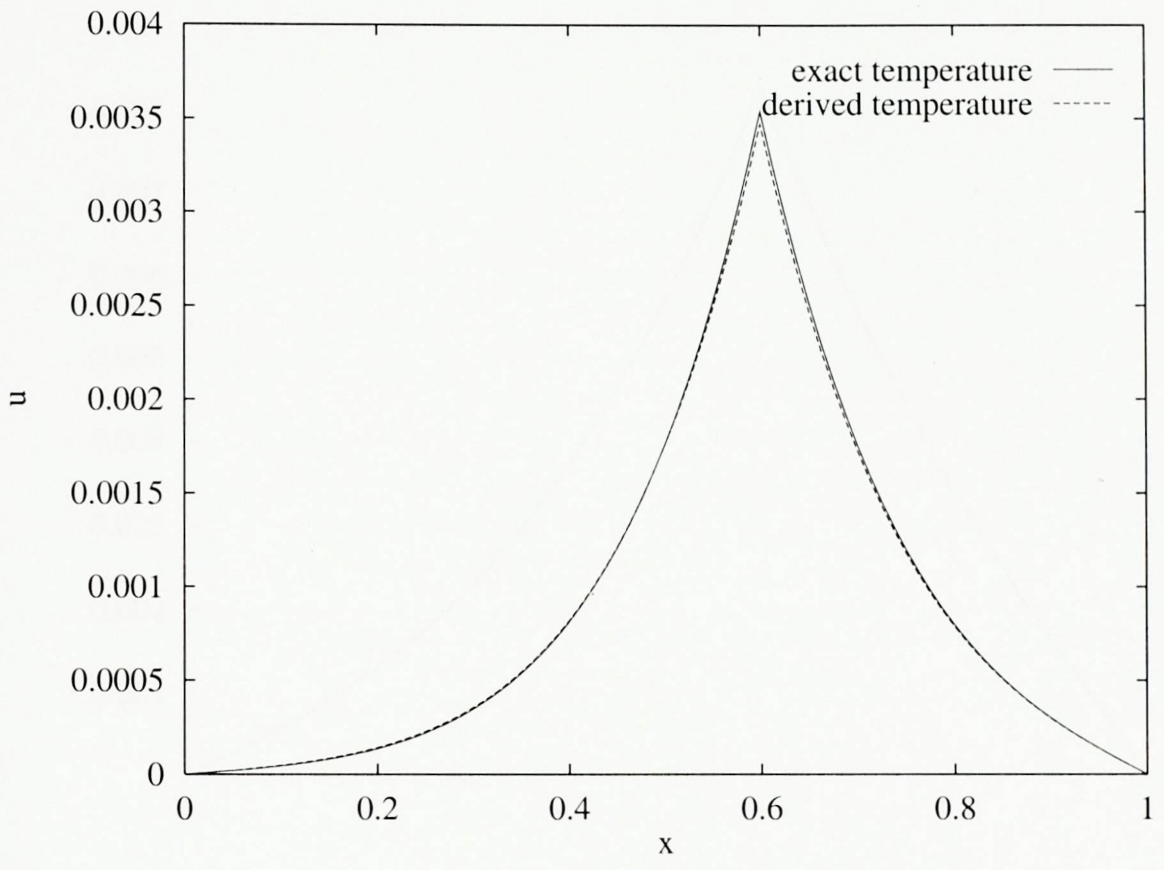
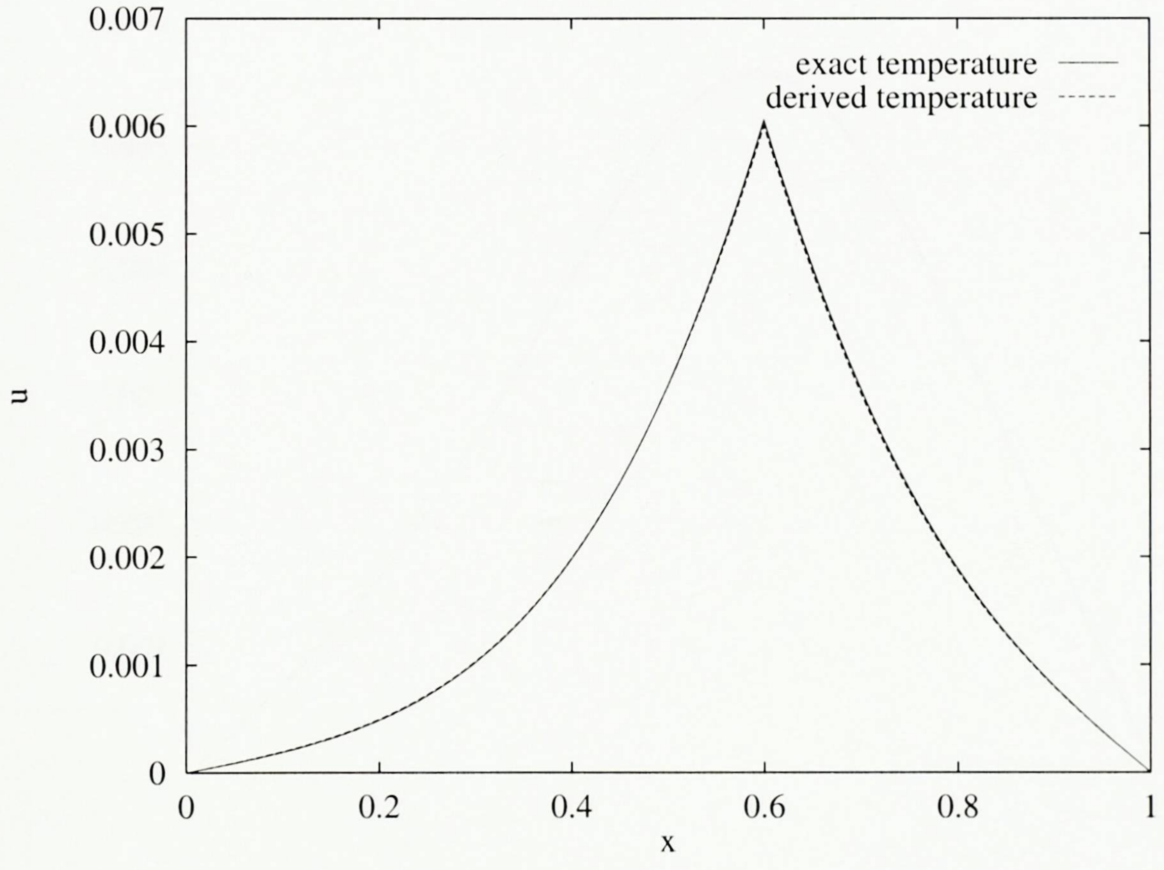


Figure 3.9: Accuracy of the retrieved source using Algorithm 1.

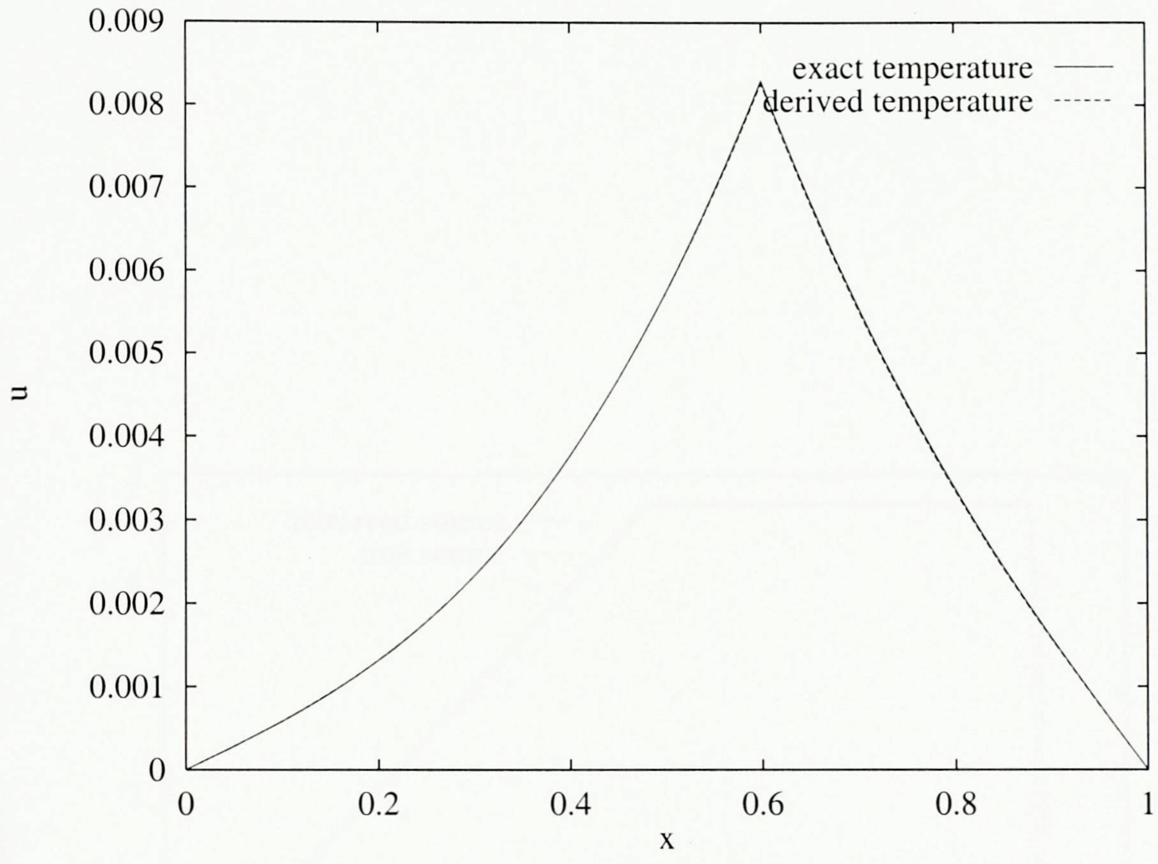


(a)

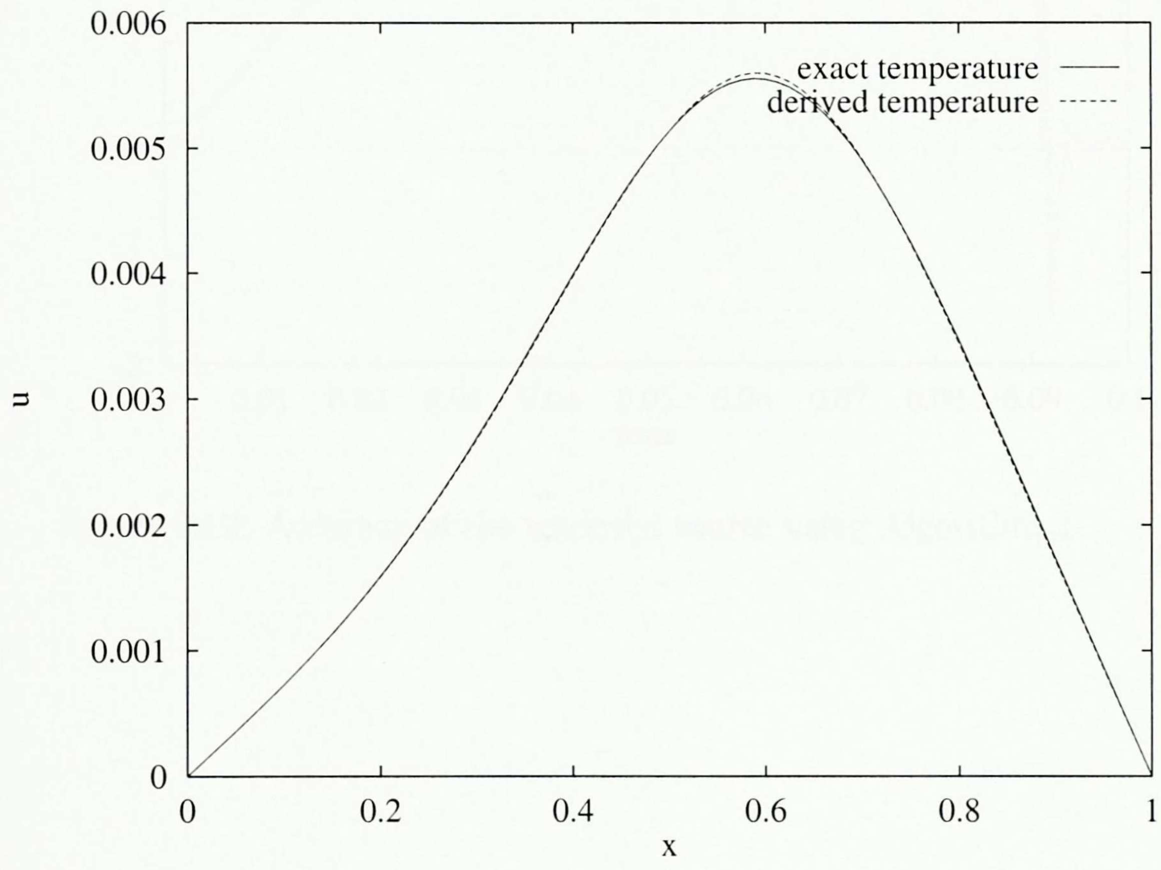


(b)

Figure 3.10: Accuracy of the retrieved temperature field, using Algorithm 2, at (a) t=0.04 and (b) t=0.06.



(a)



(b)

Figure 3.11: Accuracy of the retrieved temperature field, using Algorithm 2, at (a) t=0.09 and (b) t=0.1.

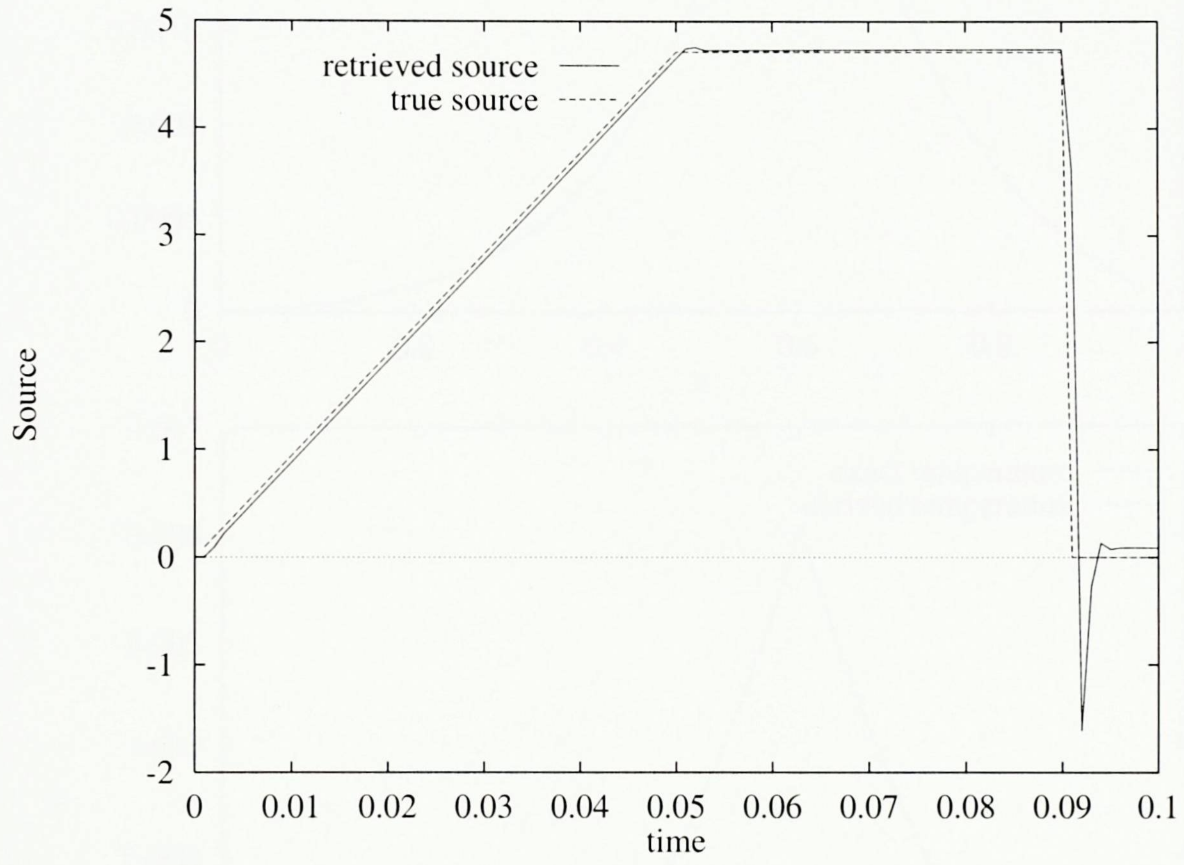
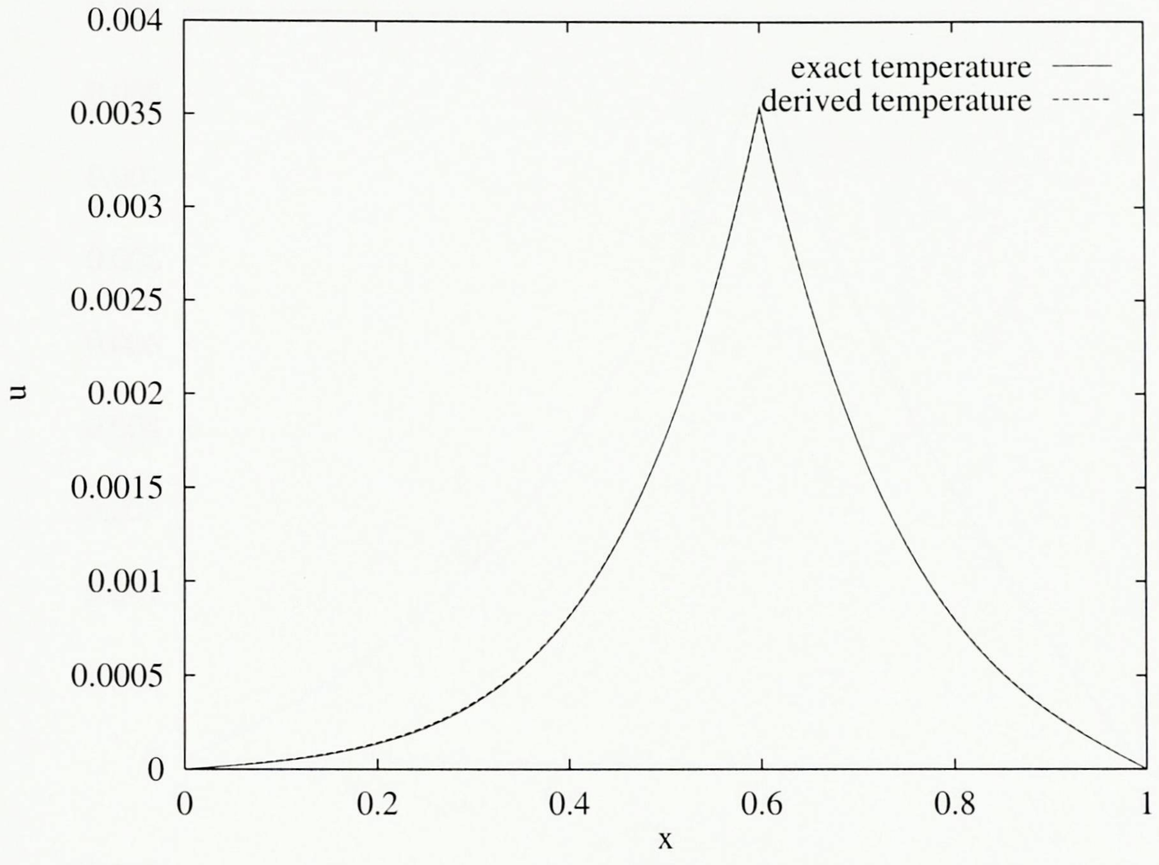
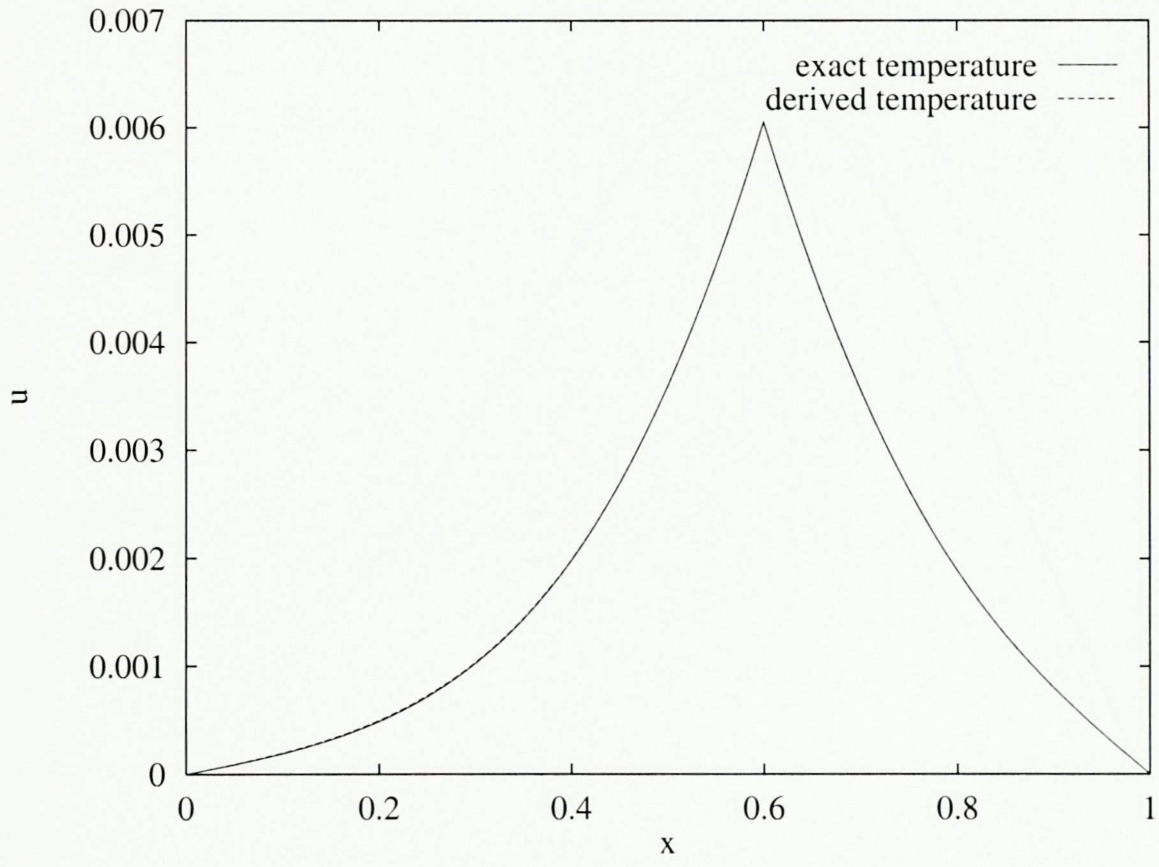


Figure 3.12: Accuracy of the retrieved source using Algorithm 2.

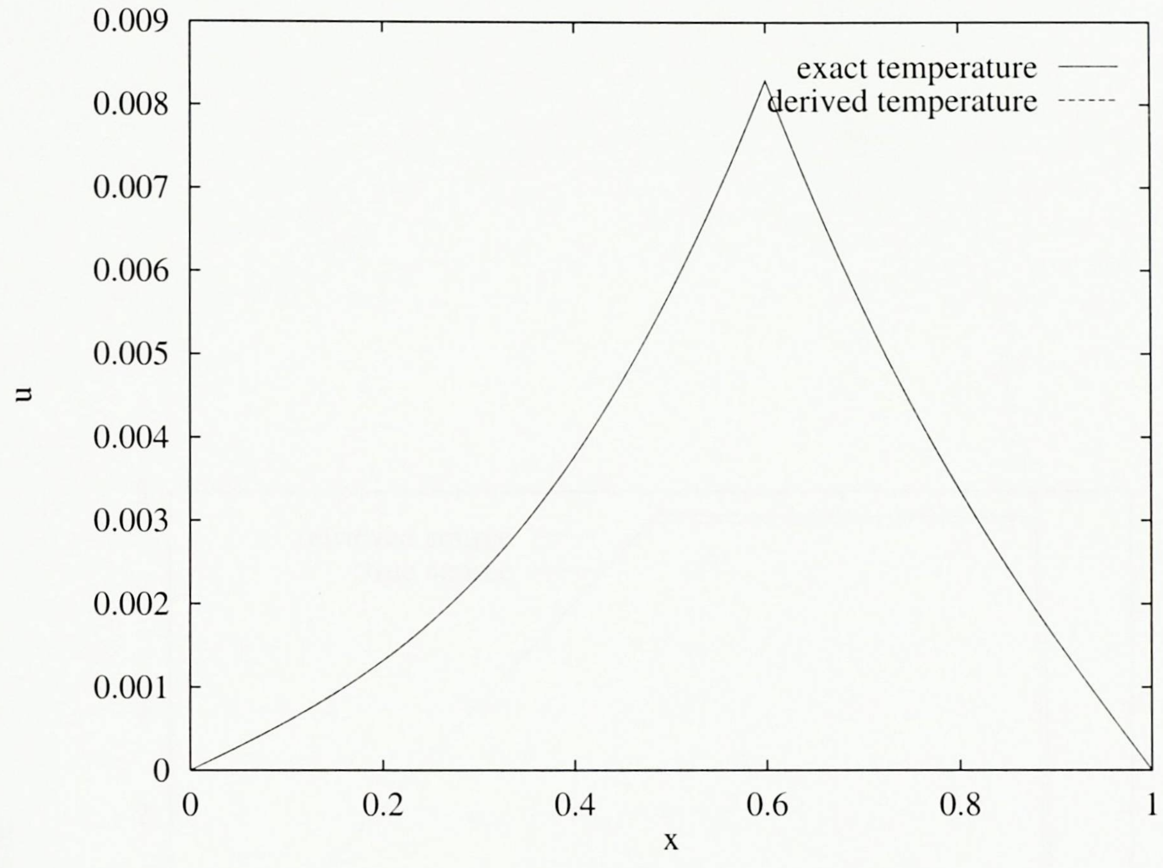


(a)

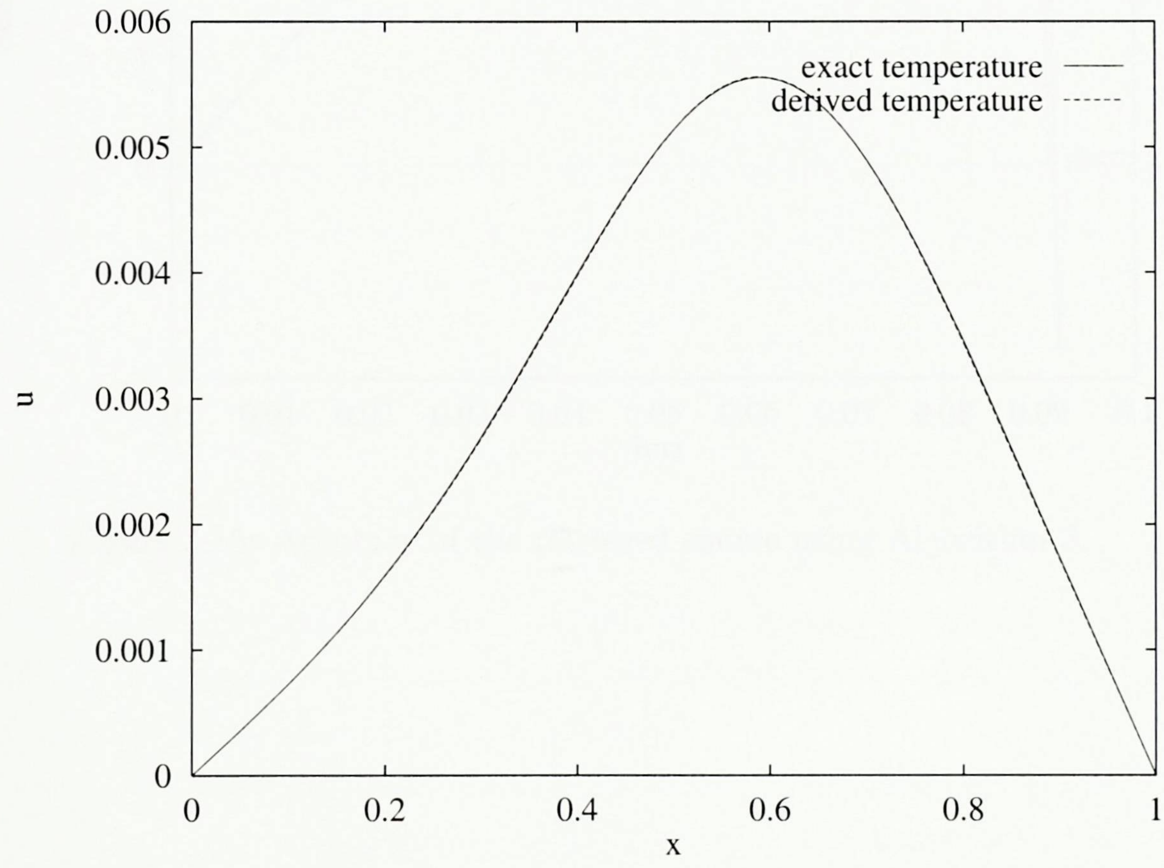


(b)

Figure 3.13: Accuracy of the retrieved temperature field, using Algorithm 3, at (a) t=0.04 and (b) t=0.06.



(a)



(b)

Figure 3.14: Accuracy of the retrieved temperature field, using Algorithm 3, at (a) t=0.09 and (b) t=0.1.

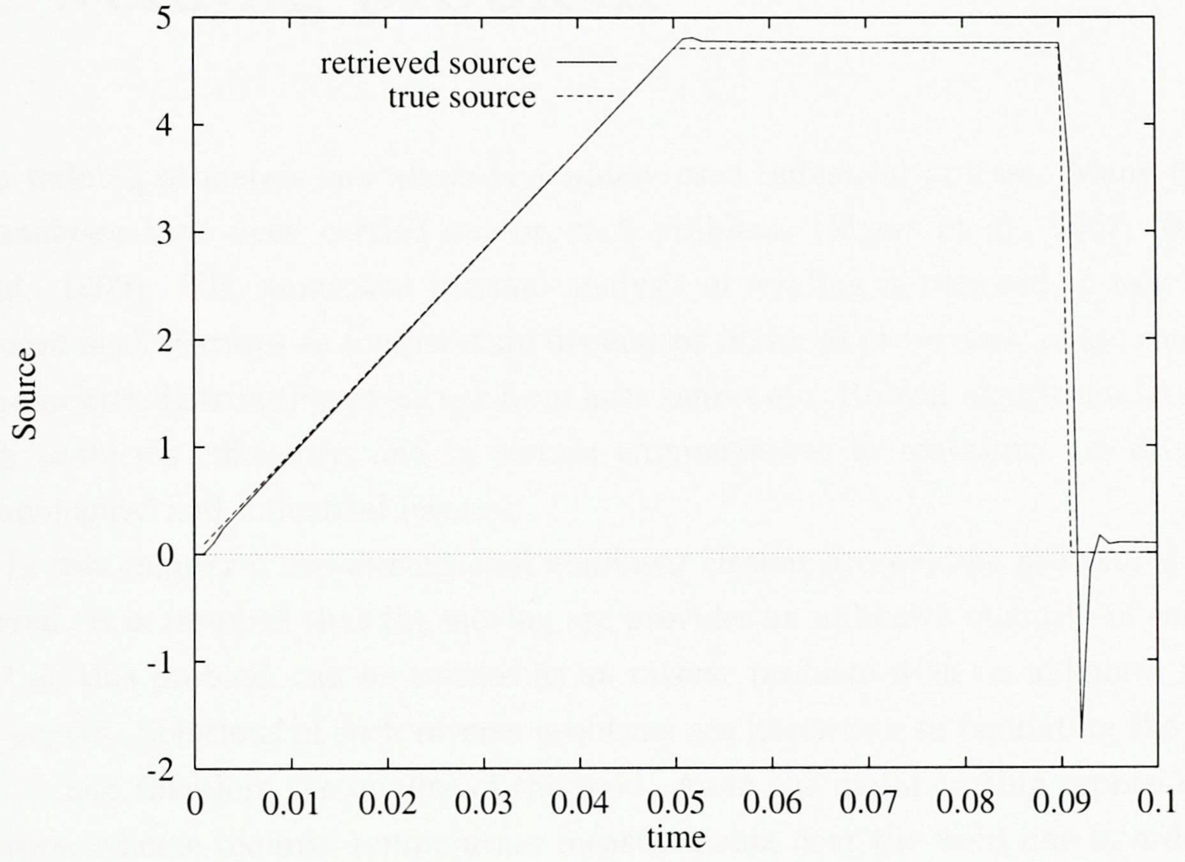


Figure 3.15: Accuracy of the retrieved source using Algorithm 3.

Chapter 4

A welding problem

The welding of metals and alloys is a widely used industrial process. Many types of analyses have been carried out on such problems (Myers et al., 1967; Taylor et al., 1999). The numerical thermal analysis of welding is required to take into account such features as temperature dependent material properties, phase change, non-uniform distribution of energy from heat source etc. Robust algorithms to solve such problems efficiently, and in certain circumstances in real-time, are of great technological and industrial interest.

In this chapter a two-dimensional nonlinear electric arc-welding problem is considered. It is assumed that the moving arc provides an unknown quantity of energy, that is, this problem can be treated as an inverse problem with an unknown moving source. Solutions of such inverse problems are important in regulating the heat source and therefore the quality of the weld. As in the metal cutting problem, the solution scheme requires temperature measurements near the weld line in order to calculate the temperature field and to retrieve the unknown source. One of the algorithms developed in Chapter 3 is used in the present calculations. As explained in that chapter, Algorithms 2 and 3 are suitable for single-processor (sequential) computing. Algorithm 3 is chosen for the present calculations since this is slightly more efficient (it only uses one system matrix).

The welding problem considered has been computed as a direct problem with a known heat source (Argyris et al., 1985; Demirdžić and Martinović, 1993). The numerical results from the direct problem together with the experimental thermocouple measurements (Argyris et al., 1985) are used to validate the results obtained by means of the inverse problem.

This chapter will further emphasize the effectiveness of using problem partitioning, described in the previous chapter, in generating standard, well-defined and continuous subproblems. It also further validates the source retrieval method and Algorithm 3 developed in previous chapter. It should also be noted that the application considered in this chapter, the welding problem, is more complex than the metal cutting problem (3). The main complexities are the moving source and the phase change due to welding, hence the numerical schemes described in the previous chapter have to be modified to take into account these complexities.

4.1 The 2d nonlinear welding problem

The welding problem considered here is the butt welding of two thin steel plates. Figure 4.1 illustrates the welding process. As in the previous chapter, the material properties, apart from thermal conductivity, are assumed to be homogeneous across the domain of interest. It is also assumed that the application of a welding tool at the weld line is equivalent to the application of a heat source at these points. The

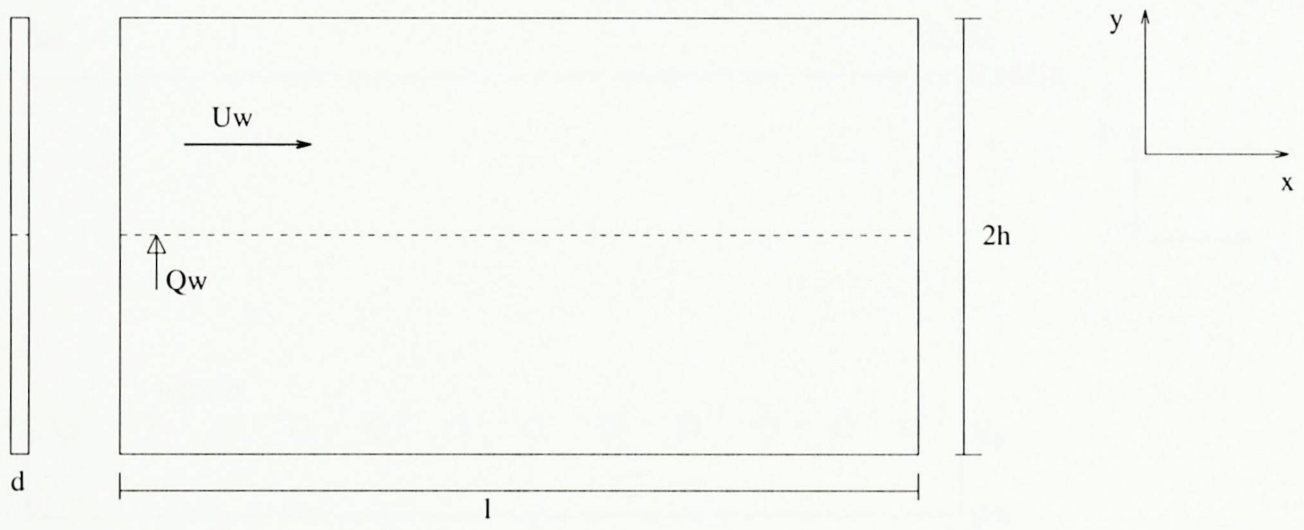


Figure 4.1: Geometry of the welded work-piece ($l = 0.5m$, $2h = 0.33m$, $d = 0.008m$, $U_w = 0.00333m/s$).

welding takes place along the dotted line, see Figure 4.1, and due to the symmetry of the problem only the upper half needs to be considered. Since the thickness of the plate, d, is very small, only two dimensional heat conduction is considered. Hence, the heat conduction of the plate is modelled using the following 2d nonlinear, unsteady, parabolic, heat conduction equation,

$$c^e \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}(k(T)\frac{\partial T}{\partial x}) + \frac{\partial}{\partial y}(k(T)\frac{\partial T}{\partial y}) - 2h_{eff}A(T - T_a) + Q_w \qquad (4.1)$$

subject to initial condition $T(x,y,0) = T_i(x,y)$, boundary conditions $B_0(T,y,t)|_{x=0}$, $B_1(T,y,t)|_{x=l}$, $C_0(T,x,t)|_{y=0}$ and $C_1(T,x,t)|_{y=h}$. Here $T(x,y,t)$ is the temperature distribution, $k(T)$ is the conductivity of the steel, t is the time, h_{eff} is the effective heat transfer, A is the surface area, T_a is the ambient temperature, $c^e = \rho c - L\frac{\partial f_l}{\partial T}$ is the effective specific heat, ρ is the density, c is the specific heat capacity, L is the latent heat, $\frac{\partial f_l}{\partial T}$ is the variation of liquid fraction, Q_w is the heat transfer rate from the moving arc. T_i, B_0, B_1, C_0 and C_1 are known functions. Since the source term, Q_w, in (4.1) is an unknown, the inverse problem here is to retrieve this unknown heat source. In order to deal with this additional unknown, temperature measurements near the weld line are required (see Figure 4.2). The sensors are attached at $y = y_s$, such that $0 < y_s < 0.165$. Let the temperature measured by means of the temperature sensors be $T(x, y_s, t) = T^*(x,t)$. The measured temperatures are used to retrieve temperature at the welding points.

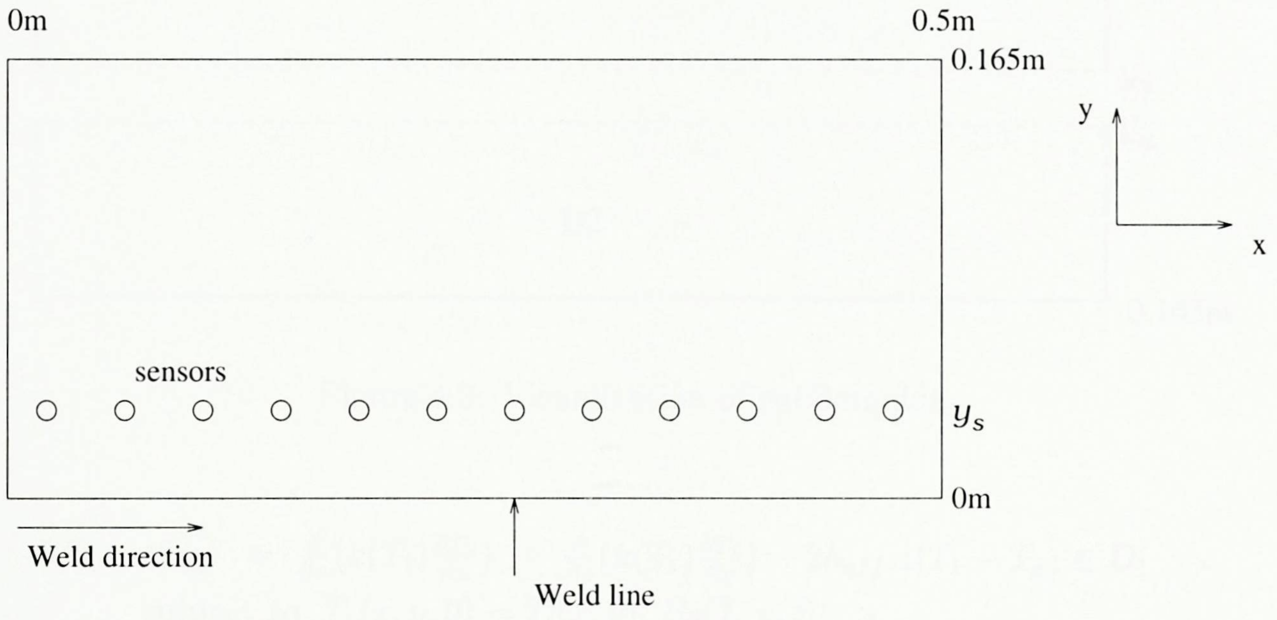


Figure 4.2: Sensors are located near the weld line.

4.2 Problem partitioning

Domain decomposition was applied to the original domain to generate two well defined, homogeneous, continuous and properly connected subdomains. The two subdomains are:

$$D_1 = \{(x,y) : 0 < x < 0.5 \text{ and } 0 < y < y_s\}$$
$$D_2 = \{(x,y) : 0 < x < 0.5 \text{ and } y_s < y < 0.165\}$$

The domain partitioning is carried out along the sensor points, Figure 4.3. As explained in the previous chapter, this level of partitioning is defined as problem partitioning. This problem partitioning removes the unknown source term Q_w. The two subproblems can be written as follows:

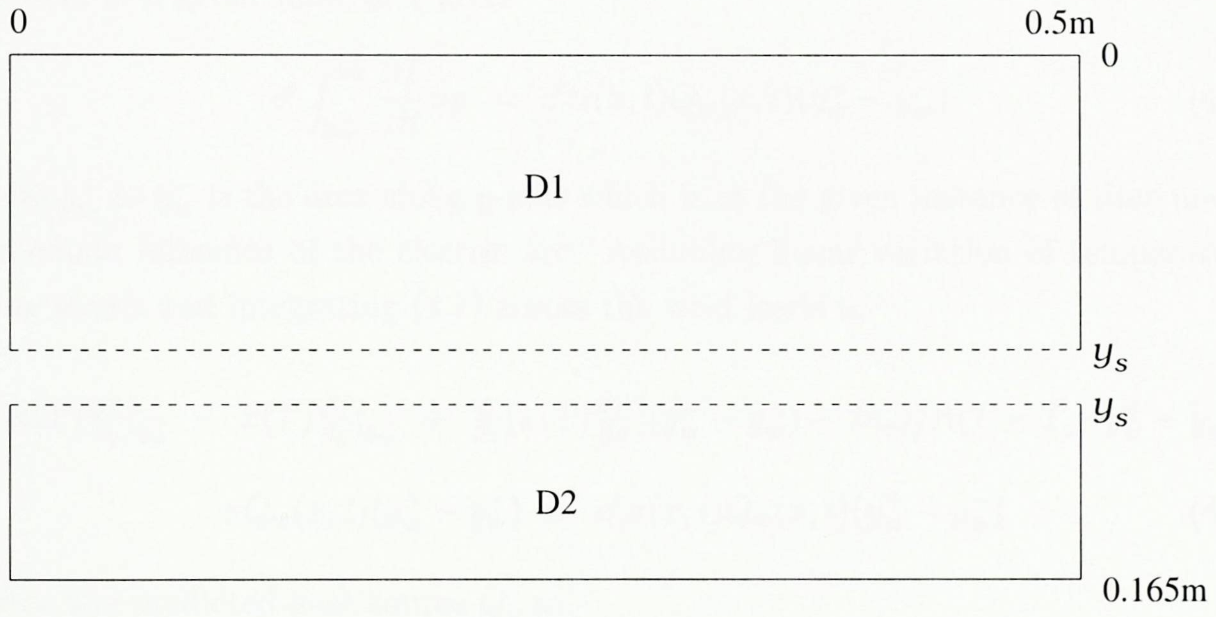


Figure 4.3: Visualization of subdomains.

$$SP_1:\; c^e\frac{\partial T_1}{\partial t} = \frac{\partial}{\partial x}(k(T_1)\frac{\partial T_1}{\partial x}) + \frac{\partial}{\partial y}(k(T_1)\frac{\partial T_1}{\partial y}) - 2h_{eff}A(T_1 - T_a) \in D_1$$
subject to $T_1(x,y,0) = T_i(x,y)$, $B_0(T,y,t)|_{x=0}$,

$B_1(T,y,t)|_{x=l}$, $T_1(x,y_s,t) = T^*(x,t)$, $C_1(T,x,t)|_{y=h}$.

$$SP_2:\; c^e\frac{\partial T_2}{\partial t} = \frac{\partial}{\partial x}(k(T_2)\frac{\partial T_2}{\partial x}) + \frac{\partial}{\partial y}(k(T_2)\frac{\partial T_2}{\partial y}) - 2h_{eff}A(T_2 - T_a) \in D_2$$
subject to $T_2(x,y,0) = T_i(x,y)$, $B_0(T,y,t)|_{x=0}$,

$B_1(T,y,t)|_{x=l}$, $T_2(x,y_s,t) = T^*(x,t)$,

$\frac{\partial T_2(x,y_s,t)}{\partial y} = \frac{\partial T_1(x,y_s,t)}{\partial y}$, $C_0(T,x,t)|_{y=0}$.

4.3 Source retrieval

As in Section 3.1, it is assumed that the rate of change of temperature on either side of the weld is directly proportional to the strength of the heat source. Hence in the neighbourhood of the weld,

$$c^e \frac{\partial T}{\partial t} = c^e \alpha(x,t) Q_w(x,t) \tag{4.2}$$

where $\alpha > 1$ is a time dependent function that also depends on x. The condition $\alpha > 1$ is to ensure an increase in temperature at the weld. Integrating (4.2) across the weld at a given value of x gives

$$c^e \int_{y_w^-}^{y_w^+} \frac{\partial T}{\partial t} dy = c^e \alpha(x,t) Q_w(x,t)(y_w^+ - y_w^-) \tag{4.3}$$

where y_w^+ to y_w^- is the area along y-axis which is at the given instance of time under immediate influence of the electric arc. Assuming linear variation of temperature along y-axis and integrating (4.1) across the weld leads to

$$\begin{aligned} k(T)\tfrac{\partial T}{\partial y}|_{y_w^+} - k(T)\tfrac{\partial T}{\partial y}|_{y_w^-} + \tfrac{\partial}{\partial x}(k(T)\tfrac{\partial T}{\partial x})(y_w^+ - y_w^-) - 2h_{eff}A(T-T_a)(y_w^+ - y_w^-) \\ +Q_w(x,t)(y_w^+ - y_w^-) = c^e \alpha(x,t) Q_w(x,t)(y_w^+ - y_w^-) \end{aligned} \tag{4.4}$$

Define the predicted heat source Q_p as,

$$\begin{aligned} Q_p(x,t) = k(T)\tfrac{\partial T}{\partial y}|_{y_w^+} - k(T)\tfrac{\partial T}{\partial y}|_{y_w^-} + \tfrac{\partial}{\partial x}(k(T)\tfrac{\partial T}{\partial x})(y_w^+ - y_w^-) \\ -2h_{eff}A(T-T_a)(y_w^+ - y_w^-) = \beta(x,t) Q_w(x,t) \end{aligned} \tag{4.5}$$

where $\beta = (y_w^+ - y_w^-)(c^e\alpha - 1)$. Then Q_p is used in (4.1) (instead of Q_w) which is solved as a direct problem with T_p being the corresponding temperature distribution. Then, the function β is calculated as follows,

$$\beta(x,t) = \frac{Q_p(x,t)}{Q_w(x,t)} = \frac{T_p(x,t)}{T(x,y_w,t)} \tag{4.6}$$

where $T(x,y_w,t)$ is the temperature at the weld line retrieved by solving the sub-problems. That is, to determine the proportional factor β, replace Q_w by Q_p in (4.1) and solve it as a direct problem in order to obtain the corresponding temperature distribution $T_p(x,t)$. Hence β can be computed. Then the true source Q_w may be retrieved from (4.5) after β is calculated from (4.6). Note that it is not necessary to compute $(y_w^+ - y_w^-)(c^e\alpha - 1)$.

4.4 Numerical example

The two subproblems are solved by using the Newton's method as described in Section 3.4.3 of the previous Chapter. For the welding problem, the function $F(T)$ and the corresponding Jacobian $J(T)$ are defined as follows,

$$F(T) = c^e\frac{\partial T}{\partial t} - \frac{\partial}{\partial x}(k(T)\frac{\partial T}{\partial x}) - \frac{\partial}{\partial y}(k(T)\frac{\partial T}{\partial y}) + 2h_{eff}A(T - T_a) \tag{4.7}$$

$$\begin{aligned} J(T) = {} & c^e\tfrac{\partial}{\partial t} - k'\tfrac{\partial^2 T}{\partial x^2} - k\tfrac{\partial^2}{\partial x^2} - k''(\tfrac{\partial T}{\partial x})^2 - 2k'\tfrac{\partial T}{\partial x}\tfrac{\partial}{\partial x} - k'\tfrac{\partial^2 T}{\partial y^2} \\ & -k\frac{\partial^2}{\partial y^2} - k''(\frac{\partial T}{\partial y})^2 - 2k'\frac{\partial T}{\partial y}\frac{\partial}{\partial y} + 2h_{eff}A \end{aligned} \tag{4.8}$$

$F(T)$ and $J(T)$ are obtained by a second order finite volume discretisation which leads to a set of a large sparse linear system and it is solved by using PETSc. The solution of SP_2 retrieves the temperature at the weld line. The temperature at, and around, the weld line is used to retrieve the unknown source term as described above.

4.5 Validation

The source term, Q_w, (as well as other coefficients) of equation (4.1) is known. Numerical results obtained from this direct problem can be used to validate the numerical results obtained from the inverse problem. A numerical example for the welding problem as given in (Argyris et al., 1985; Demirdžić and Martinović, 1993) is used in validation. The example provides the heat transfer rate from electric arc to the steel plate, that is a direct problem can be defined. To solve the welding problem (4.1) the following set of physical data was used in the sample:

$$Q_w = 1350\ W$$
$$T_a = 293\ K$$
$$h_{eff} = 60\ W/m^2K$$

$$\begin{aligned}
\rho &= 7850\ kg/m^3 \\
c &= 607\ J/kgK \\
L &= 272\ kJ/kg \\
T_s &= 1843\ K \\
T_l &= 1863\ K
\end{aligned}$$

where T_s is the solidus temperature and T_l is the liquidus temperature. The liquid fraction f_l is evaluated as,

$$f_l = \begin{cases} 0 & \text{if } T < T_s \\ \frac{(T-T_s)}{(T_l-T_s)} & \text{if } T_s \leq T \leq T_l \\ 1 & \text{if } T > T_l \end{cases}$$

The nonlinear conductivity is given by,

$$k(T) = \begin{cases} \frac{-27.2}{762}T + 64.9448 & \text{if } T \leq 1035K \\ \frac{8}{881}T + 18.6016 & \text{if } T > 1035K \end{cases}$$

The initial and boundary conditions are,

$$\begin{aligned}
&T_i = T_a \\
&B_0 = B_1 = k\tfrac{\partial T}{\partial x} + h_{eff}(T - T_a) = 0 \\
&C_0 = \tfrac{\partial T}{\partial y} = 0 \\
&C_1 = k\tfrac{\partial T}{\partial y} + h_{eff}(T - T_a) = 0
\end{aligned}$$

The direct problem is solved by using a second order finite volume discretisation. The source is applied only at cells which were at the given instant of time under immediate influence of the electric arc. A mesh size of 50×50 is used to obtain the following numerical results. Figure 4.4 shows the two dimensional temperature distribution at $t = 75s$. At this time step the arc is passing the midsection of the plate ($x = 0.25, y = 0$). Therefore, the temperature is at its highest at this section. Thermocouple temperature measurements are available for this problem from MPA, Stuttgart (Argyris et al., 1985). Figures 4.5 to 4.7 show the comparison of numerical results with the thermocouple measurements. Figure 4.5 compares the numerical results with measured results when the arc is passing the midsection of the plate. Figure 4.6 shows the comparison at 7.5s later, as expected cooling has begun (since the arc has moved downstream from the midsection). Figure 4.7 shows the temperature history at the midsection, it illustrates the rapid heating to the melting point when the arc approaches the midsection and the gradual cooling thereafter when the arc has passed the section.

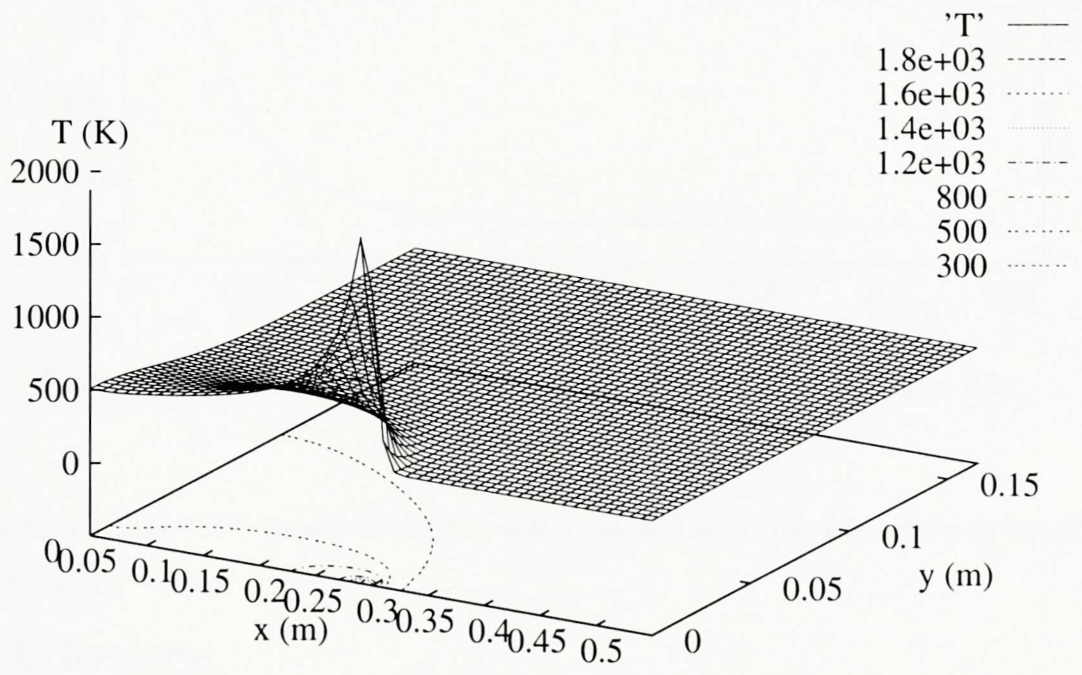


Figure 4.4: Temperature distribution at t = 75s.

To solve the inverse problem, sensors are placed at $y_s = 0.0033$, which is close enough for accurate measurement but not too close for extremely high temperature. The following results are obtained using the mesh configuration of 200×200. Figure 4.8 shows the accuracy of the retrieved temperature field at $x = 0.25$ and $t = 75s$. At this time step, the electric arc is passing over the point $x = 0.25$ and $y = 0$ (midsection), and as expected it generates high temperature values (and gradients) around this point. Figure 4.9 shows the accuracy of the retrieved source term at $x = 0.25$ and $y = 0$. The source retrieval is only activated when the electric arc is passing over this point. The run time to simulate 300 seconds of temperature

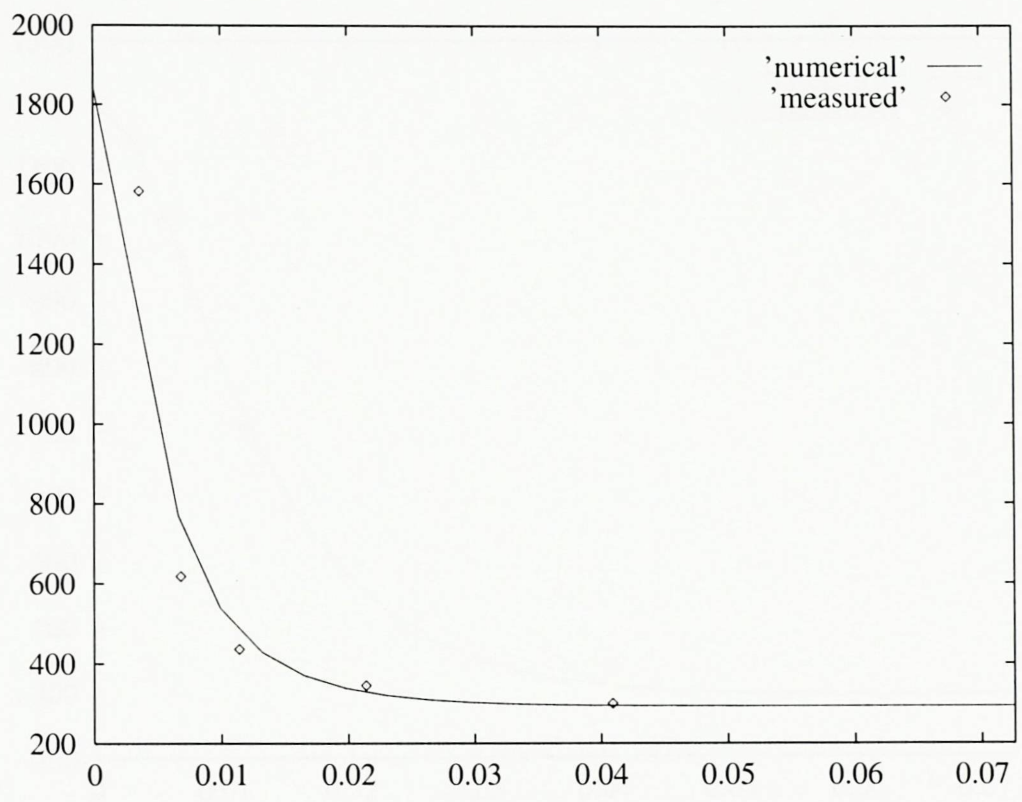


Figure 4.5: Temperature distribution at x=0.25m and t = 75s.

evaluation, for 200×200 mesh with $\Delta x = 0.1$, is 163 seconds on a Sun sparc-20 with 320Mb memory.

4.6 Closure

As illustrated in here (as well as in Chapter 3), the homogeneous and continuous subproblems generated due to problem partitioning (domain decomposition) facilitate the use of existing numerical methods and software (e.g., PETSc). This is an important advantage to industry as such numerical methods and software are very reliable, economical and well tested.

The direct problem is solved satisfactorily, considering that only the mean values of the material properties are used, apart from the temperature dependent heat conductivity. The numerical results from this direct problem are used as a set of validation for the comparison with the numerical results obtained from the inverse problem. The source retrieval method is equivalent to the one explained in Chapter 3. It is a simple and fast method. As shown in Section 4.5, the domain decomposition based algorithm retrieves the temperature field and the unknown source term accurately.

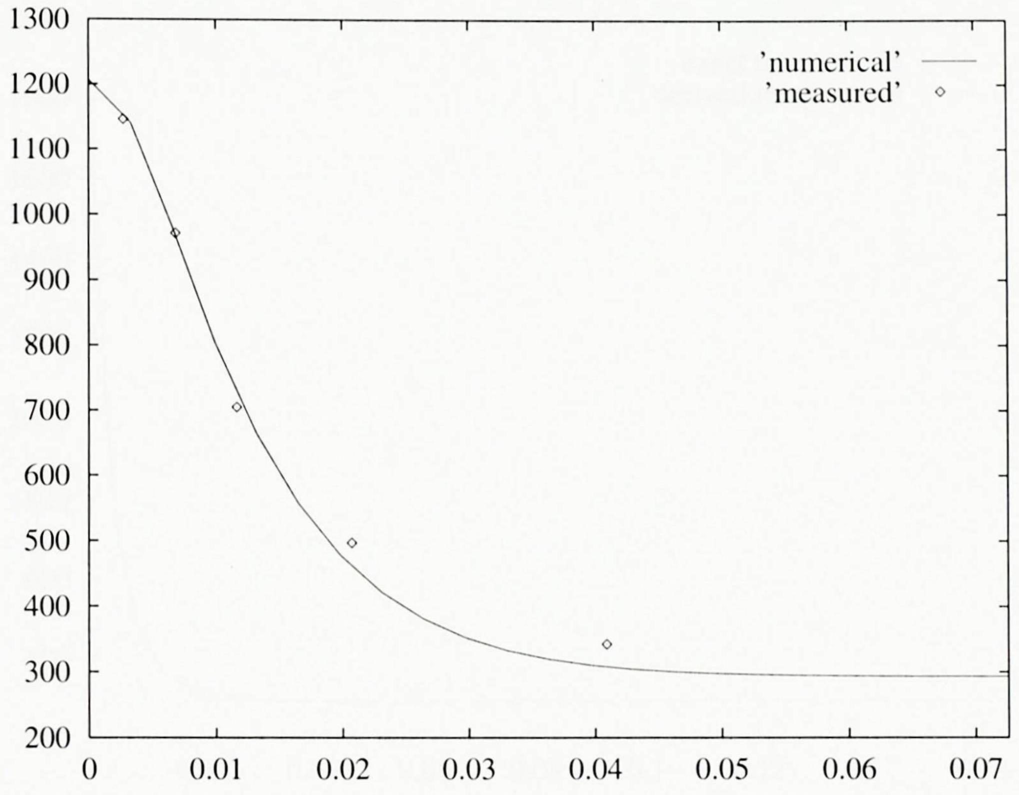


Figure 4.6: Temperature distribution at x=0.25m and t = 82.5s.

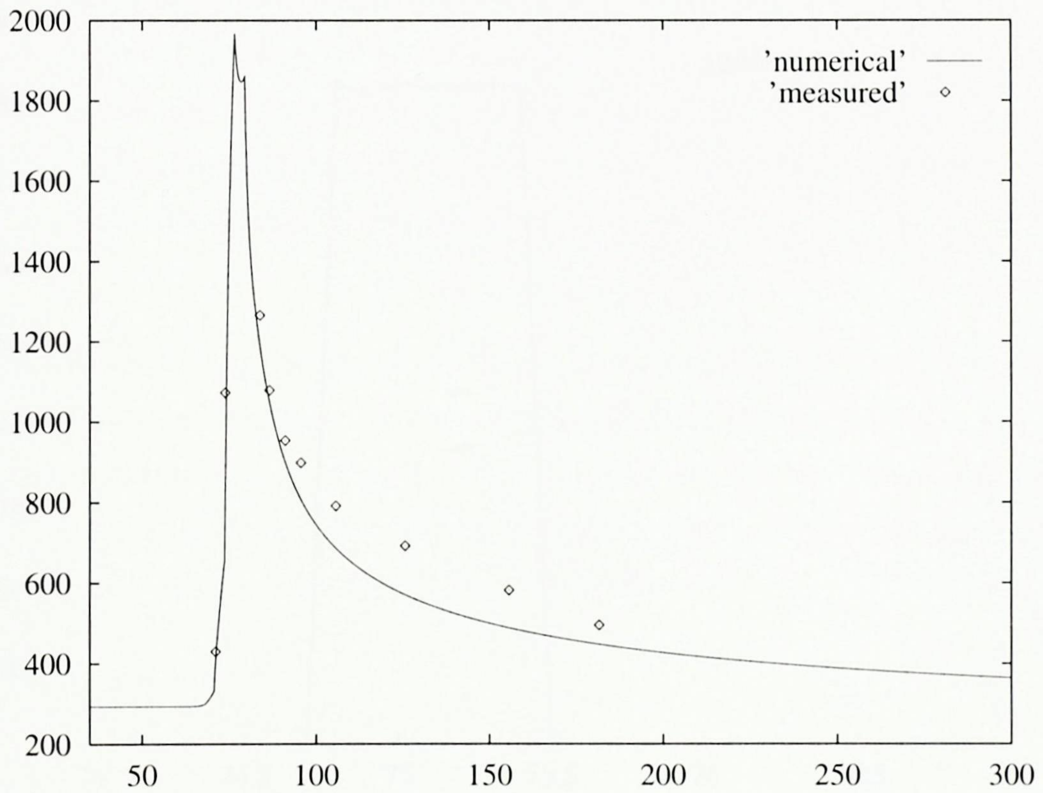


Figure 4.7: Temperature history at x=0.25m and y=0m.

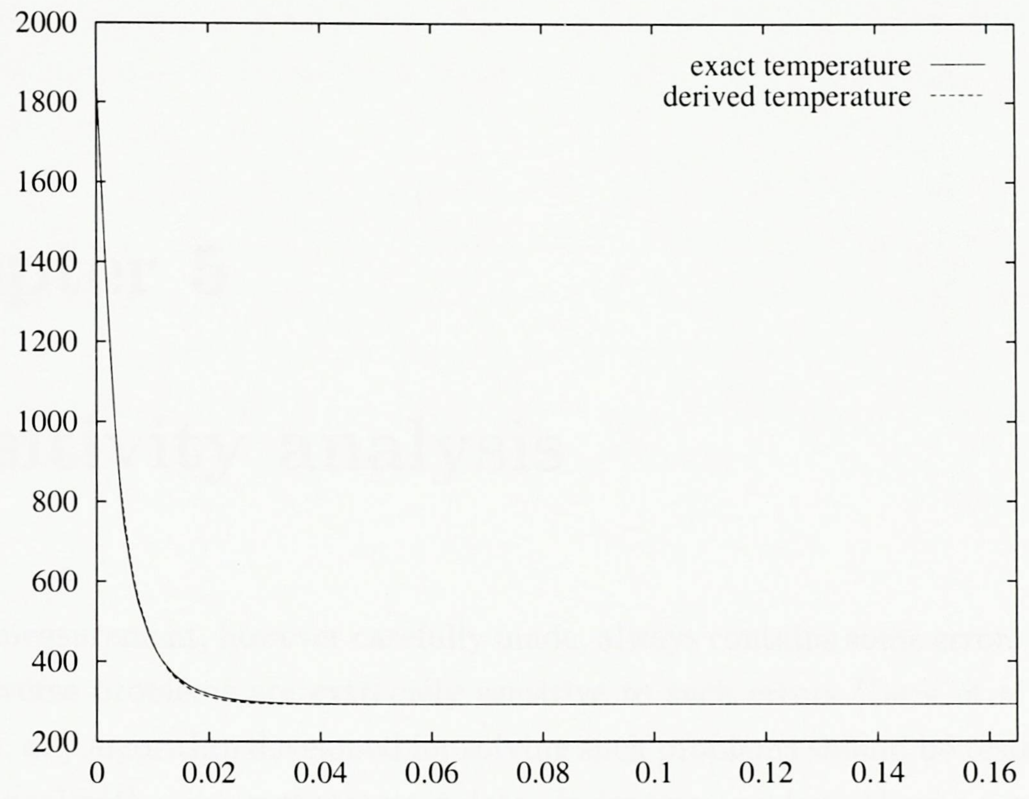


Figure 4.8: Accuracy of the temperature distribution at x=0.25m and t=75s.

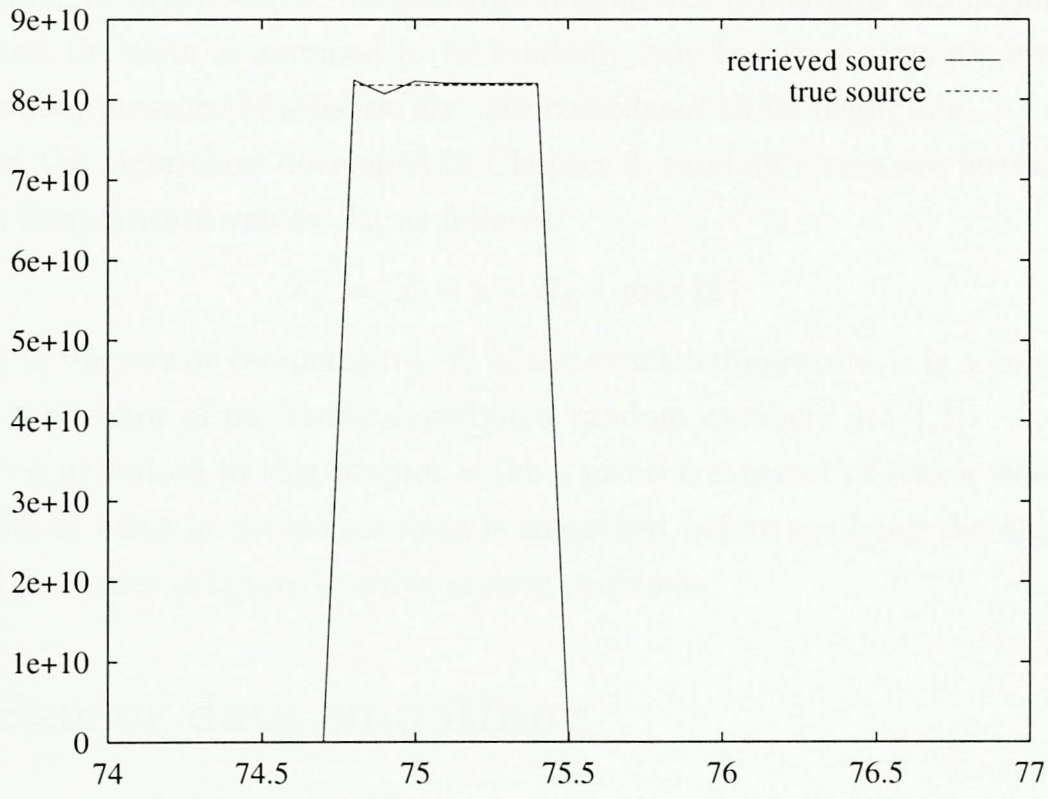


Figure 4.9: Accuracy of the source retrieval.

Chapter 5

Sensitivity analysis

Physical measurement, however carefully made, always contains some errors (Taylor, 1997). Inverse problems are extremely sensitive to such errors (Beck et al., 1985). Therefore, any algorithm developed for solving such problems should be tested for its ability to deal with noisy measurement data. In inverse problems there are a number of measurement quantities, such as material properties, sensor locations and time, in addition to temperature. Each is assumed to be accurately known apart from the temperature. In other words, temperature measurement contains the major source of error and the error is assumed to be random. Any known systematic errors due to calibration, presence of sensors, etc. are considered to be negligible.

To test the algorithms developed in Chapter 3, random errors are introduced to the exact temperature values, T_e, as follows,

$$T_s \;=\; T_e + \epsilon \times R_d \times \max|T| \tag{5.1}$$

where, T_s is the sensor temperature, T_e is the exact temperature, ϵ is a constant in (0,1], R_d is a vector of uniformly distributed random numbers in [-1,1].

Novel contribution in this chapter is the a priori treatment of sensor data. That is, the level of noise in the sensor data is smoothed before applying the algorithms described in earlier chapters to solve inverse problems.

5.1 Sensor data smoothing

One of the simplest noise smoothing techniques that can be applied is to minimise the problem,

$$\min ||T - T_s||^2_{L^2} \tag{5.2}$$

The minimisation amounts to a least squares polynomial fitting. As will be shown in the following section this provides a satisfactory level of noise treatment to retrieve the unknown source term.

There are many other smoothing techniques that can be applied to reduce noise (Tai et al., 1997; Lai et al., 1999). One of them is to solve the minimisation problem, $\min \|\nabla T\|^2_{L^2}$, subject to the constraint $\|T - T_s\|^2_{L^2} = \sigma^2$, where σ is the noise level. This method is shown to retrieve the discontinuities in source term better than the least squares method (Lai et al., 1999). However, the method distorts the source term considerably when sensor data contains 10% noise.

5.2 Examples

In the following examples three levels of error in sensor data are considered; $\epsilon = 0.001$, $\epsilon = 0.01$ and $\epsilon = 0.1$ corresponding to 0.1%, 1% and 10% respectively. Figures 5.1 to 5.3 show the results of retrieving the source term with noisy sensor data for the problem described in Section 3.6. It clearly shows that the noisy measurement data can not be directly used to retrieve the source term. Therefore, some kind of treatment must be applied prior to using such temperature measurements. Figure 5.4 shows the noisy (1% noise) and smoothed sensor data. This and all other smoothing are carried out using 7^{th} degree polynomial fits, this gave the best fit (correlation coefficient) to the noisy data sets generated. Lower order polynomials gave poor fits whereas higher order ones did not improve upon the 7^{th} degree polynomial fits.

In the following examples the noise present in the measurement data is smoothed using the above least squares method. Then the inverse algorithms are applied to retrieve the source term.

5.2.1 The metal cutting problem

The three algorithms, with the problem given in Section 3.6, are tested with smoothed data. Figures 5.5 to 5.13 illustrate the source retrieval using the mesh size of 40, for the three algorithms, with the three levels of noise mentioned above. The overall shape of the actual source is maintained for 0.1%, 1% and 10% of random noise. That is, using such a simple sensor data smoothing, the algorithms can retrieve the

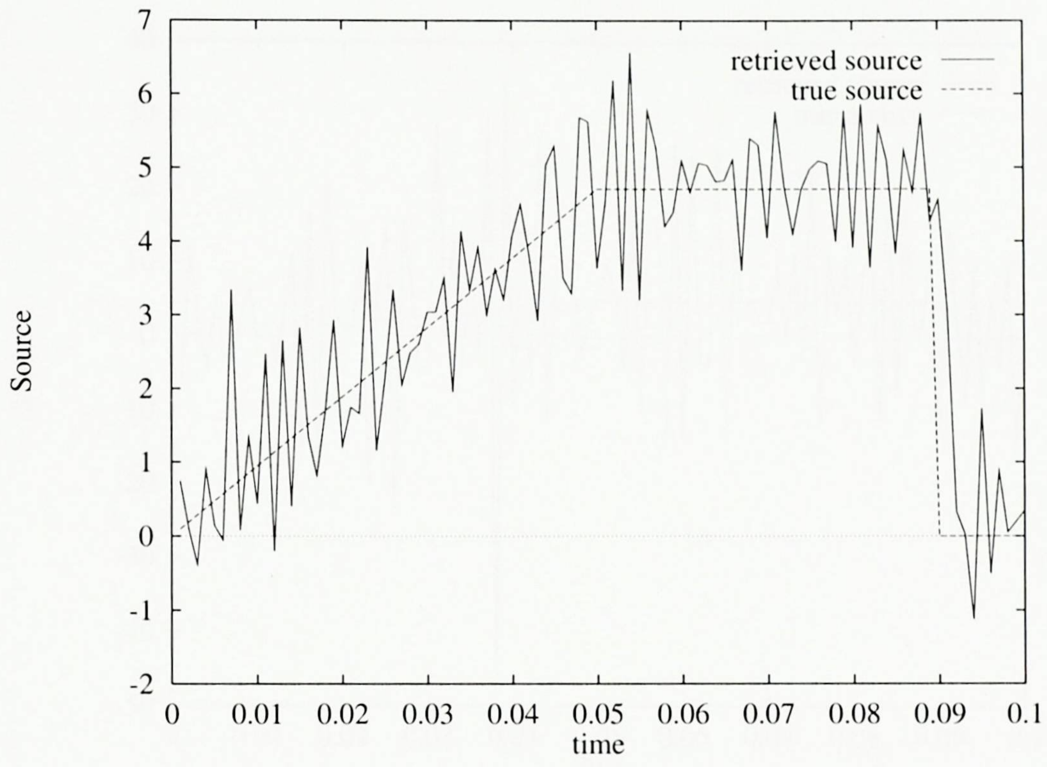


Figure 5.1: Retrieved source with 0.1% error in sensor data.

unknown source term satisfactorily even when there is 10% noise presence in the measurement data.

5.2.2 The welding problem

The welding problem described in Chapter 4 is also tested for the sensitivity of the algorithm to noisy sensor data. Figures 5.14 to 5.16 show the source retrieved for the three levels of noise with the above smoothing applied to the sensor data. The source is retrieved accurately for 0.1% and 1% error levels in sensor data. The retrieved source is less accurate when there is 10% noise in the sensor data.

5.3 Closure

The heat source retrieval relies on the use of accurate sensors for temperature measurements. However, since there will always be some level of noise presence in measurements, smoothing (noise treatment) is necessary before retrieving the source. Noise treatment using least squares polynomial fitting is adequate to retrieve the unknown source term when noisy sensor data are used.

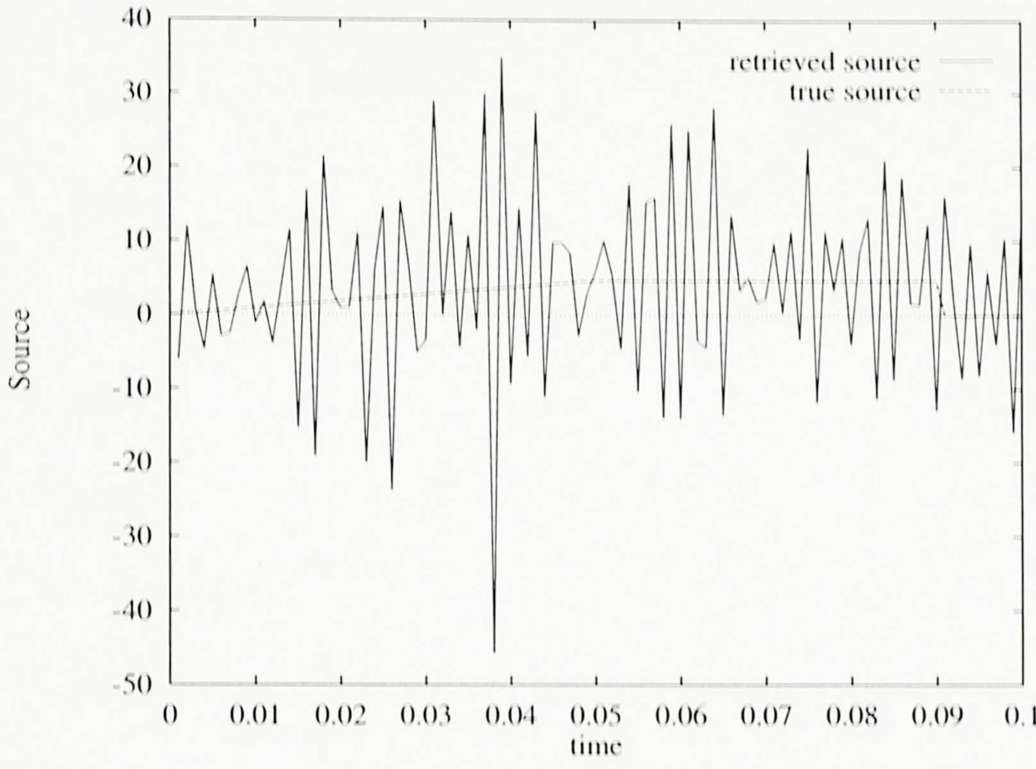


Figure 5.2: Retrieved source with 1% error in sensor data.

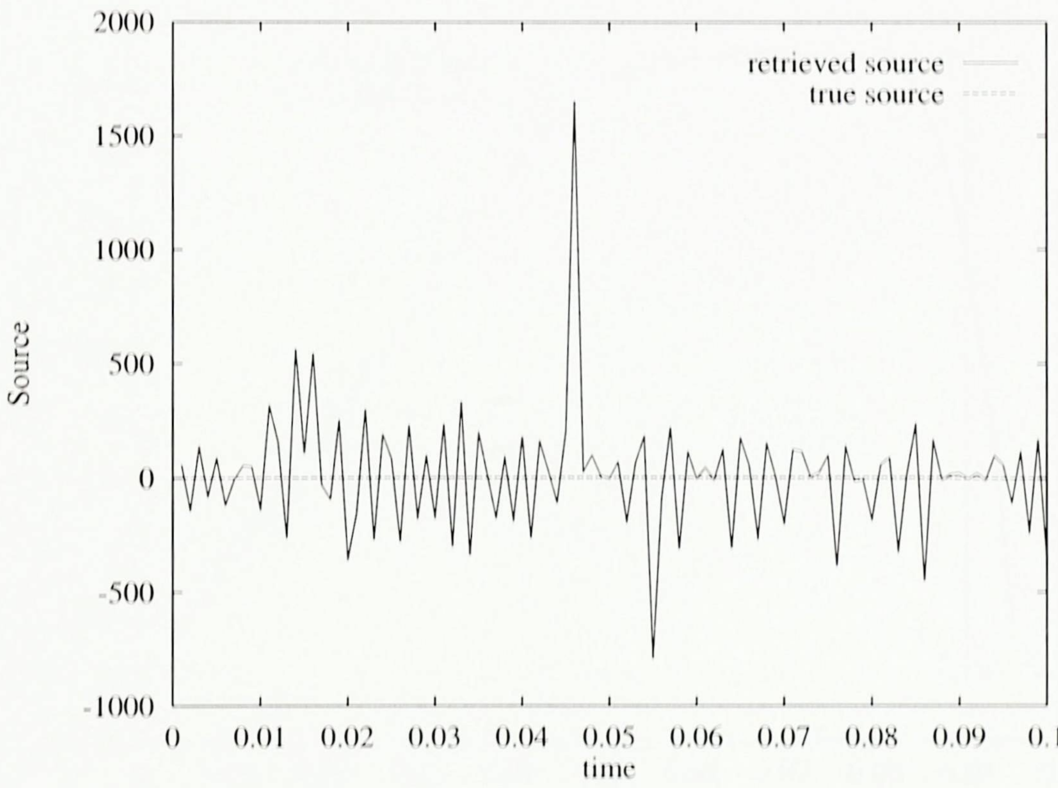


Figure 5.3: Retrieved source 10% error in sensor data.

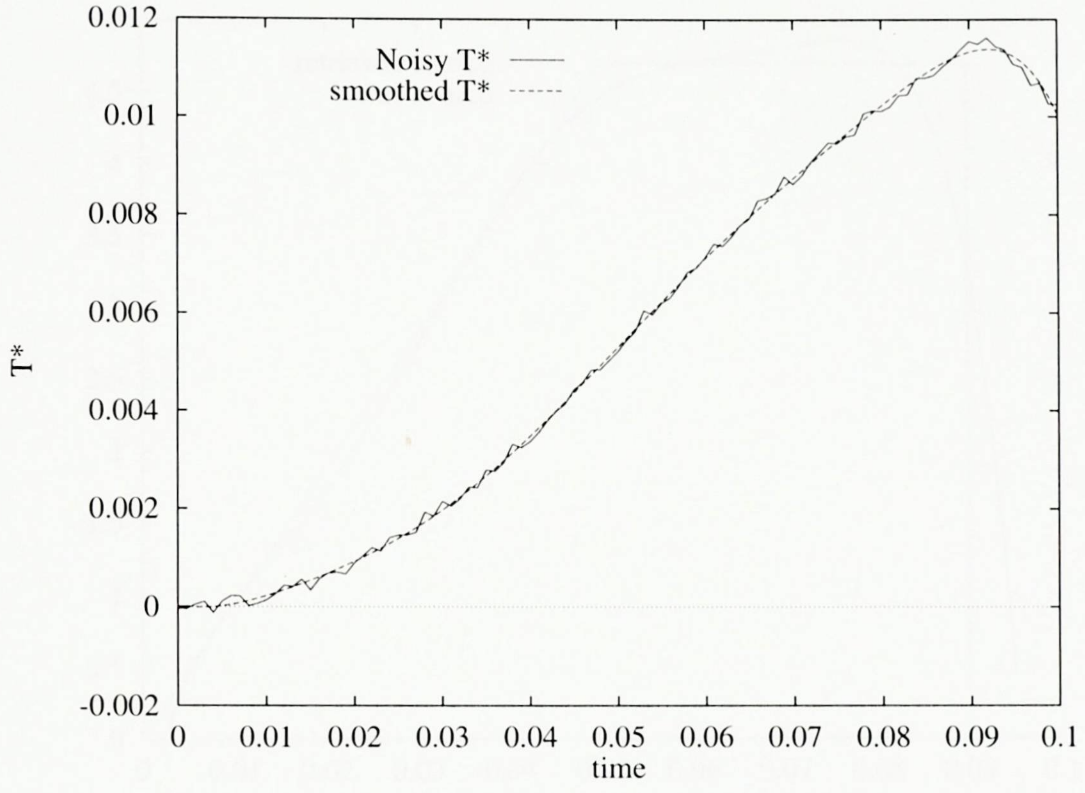


Figure 5.4: Noisy (1%) and smoothed sensor data.

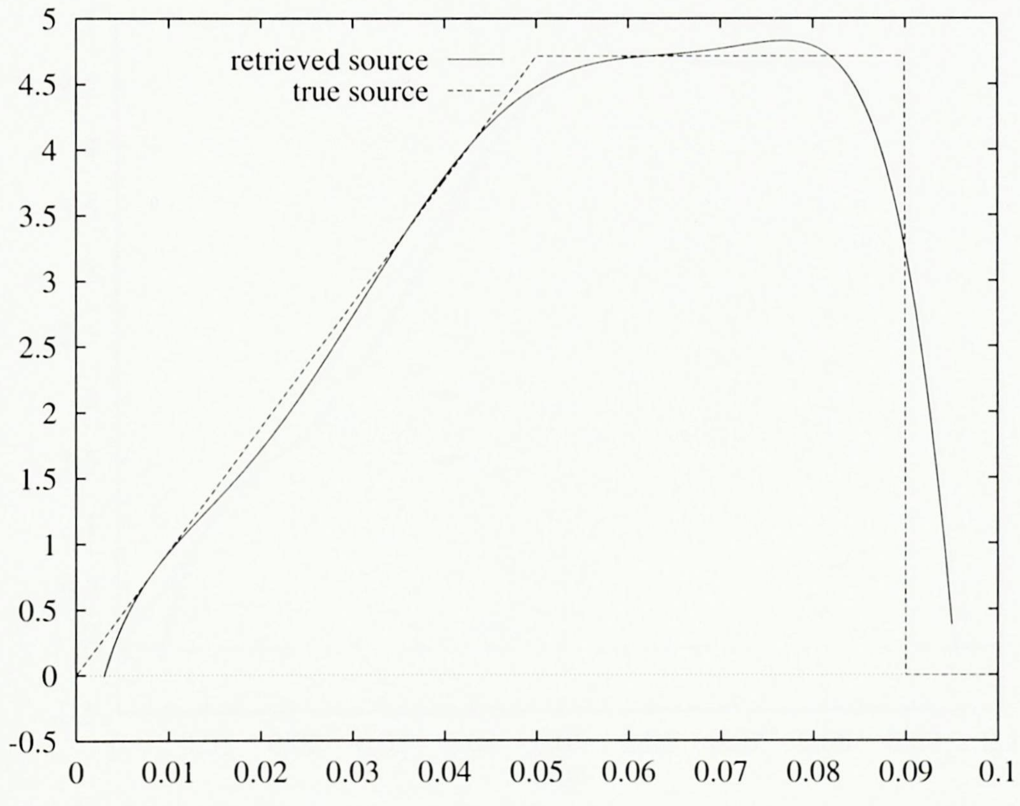


Figure 5.5: Retrieved source using Algorithm 1 with 0.1% error in sensor data.

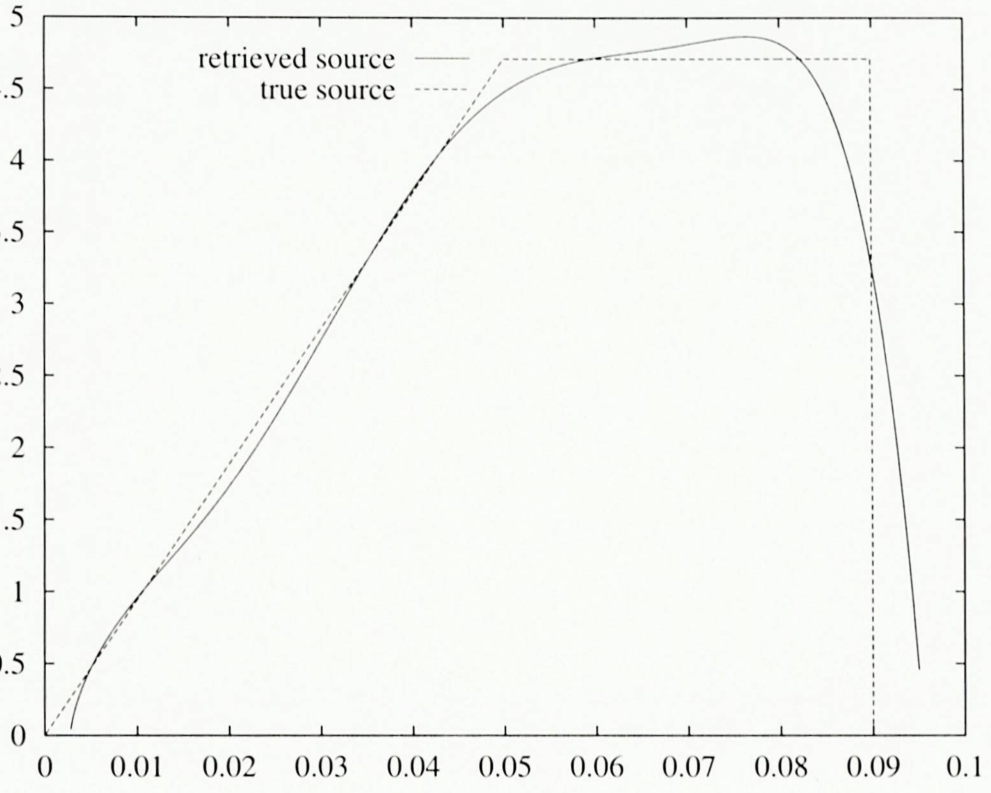


Figure 5.6: Retrieved source using Algorithm 1 with 1% in sensor data.

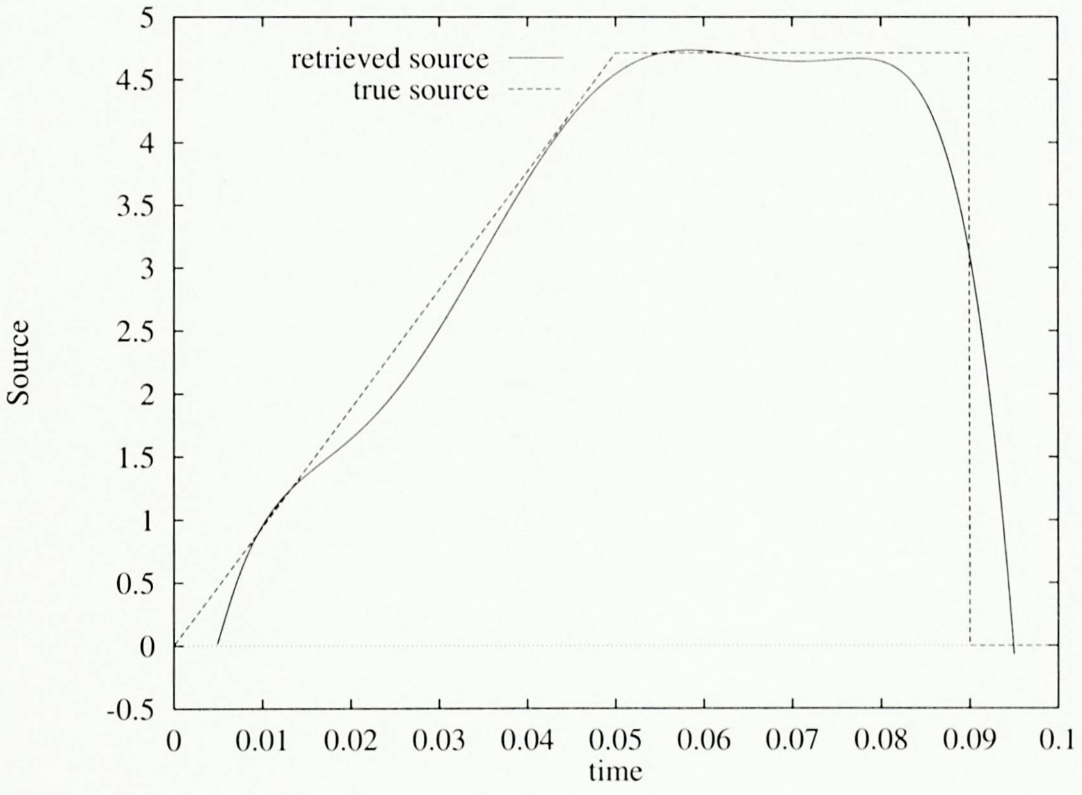


Figure 5.7: Retrieved source using Algorithm 1 with 10% error in sensor data.

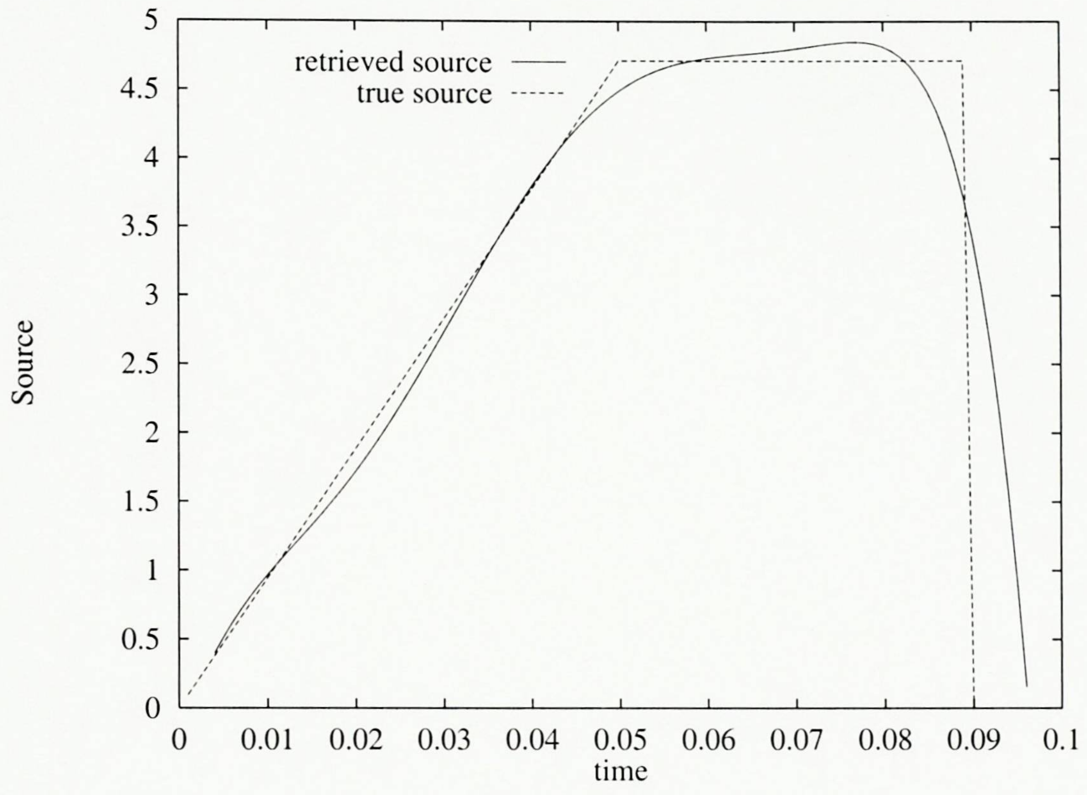


Figure 5.8: Retrieved source using Algorithm 2 with 0.1% error in sensor data.

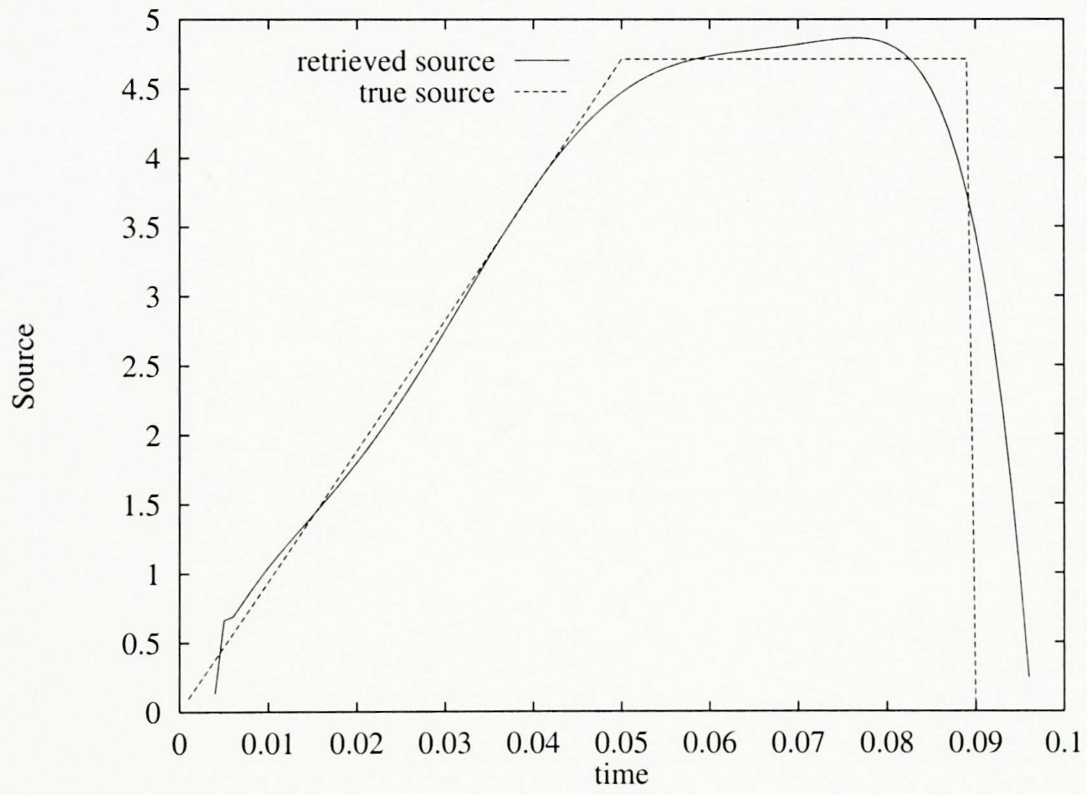


Figure 5.9: Retrieved source using Algorithm 2 with 1% error in sensor data.

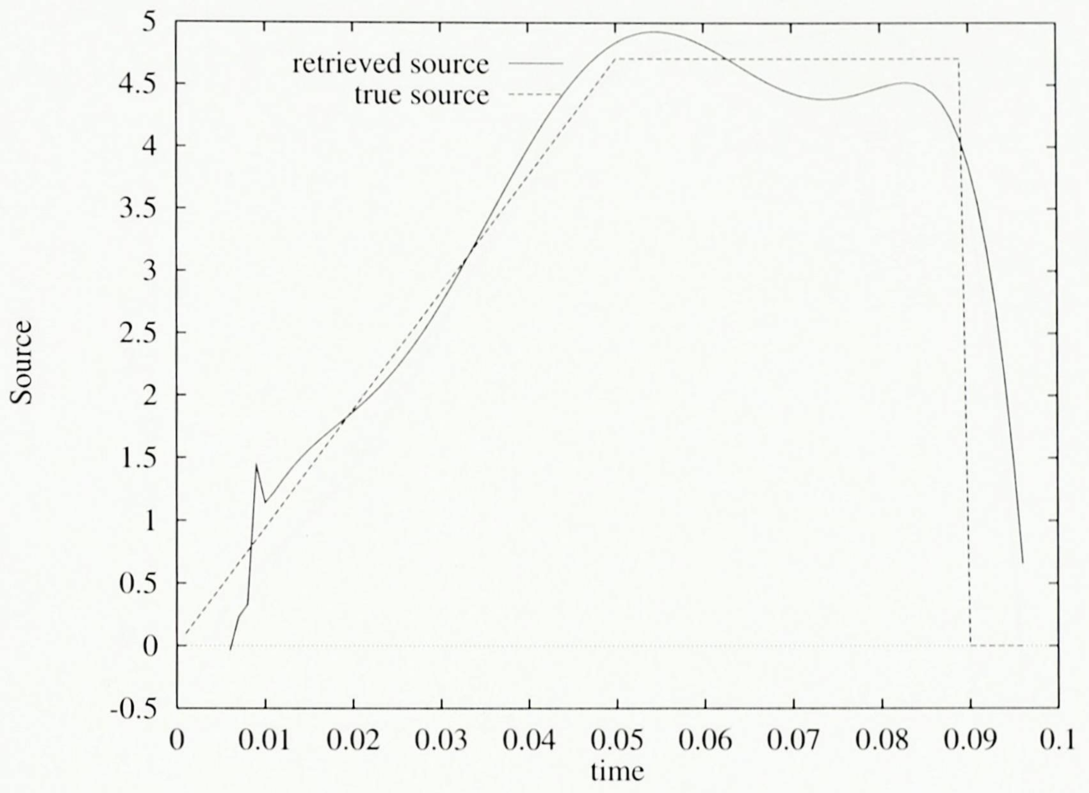


Figure 5.10: Retrieved source using Algorithm 2 with 10% error in sensor data.

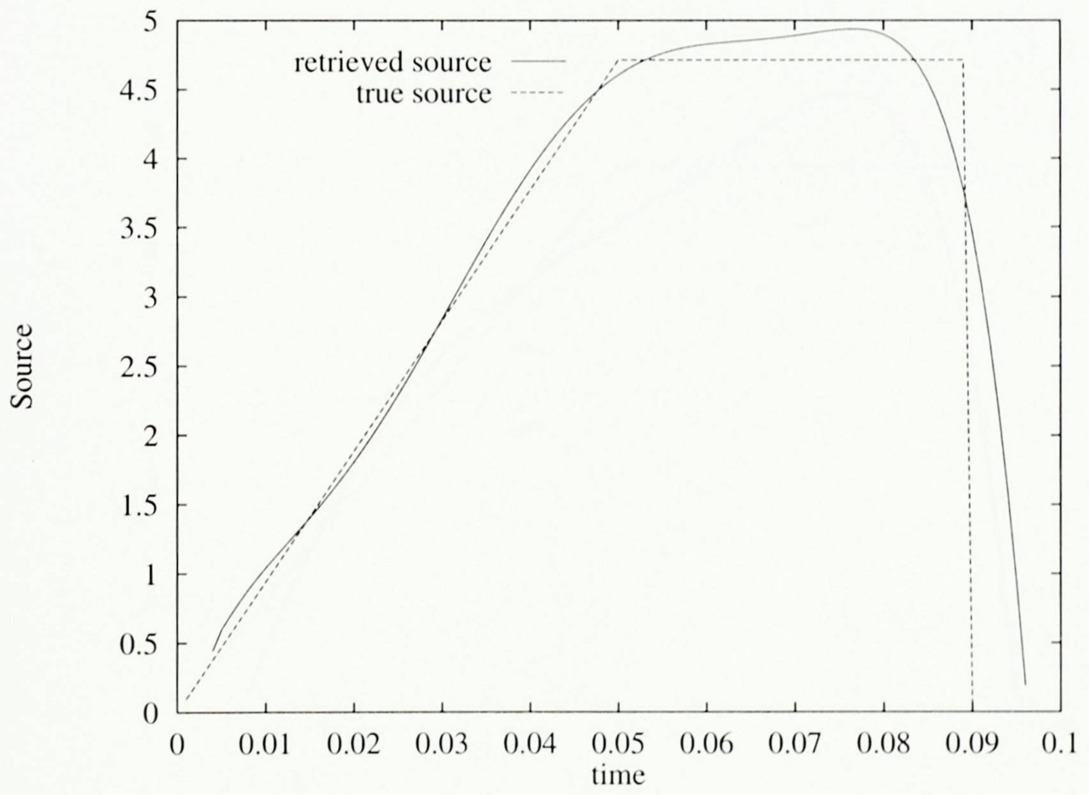


Figure 5.11: Retrieved source using Algorithm 3 with 0.1% error in sensor data.

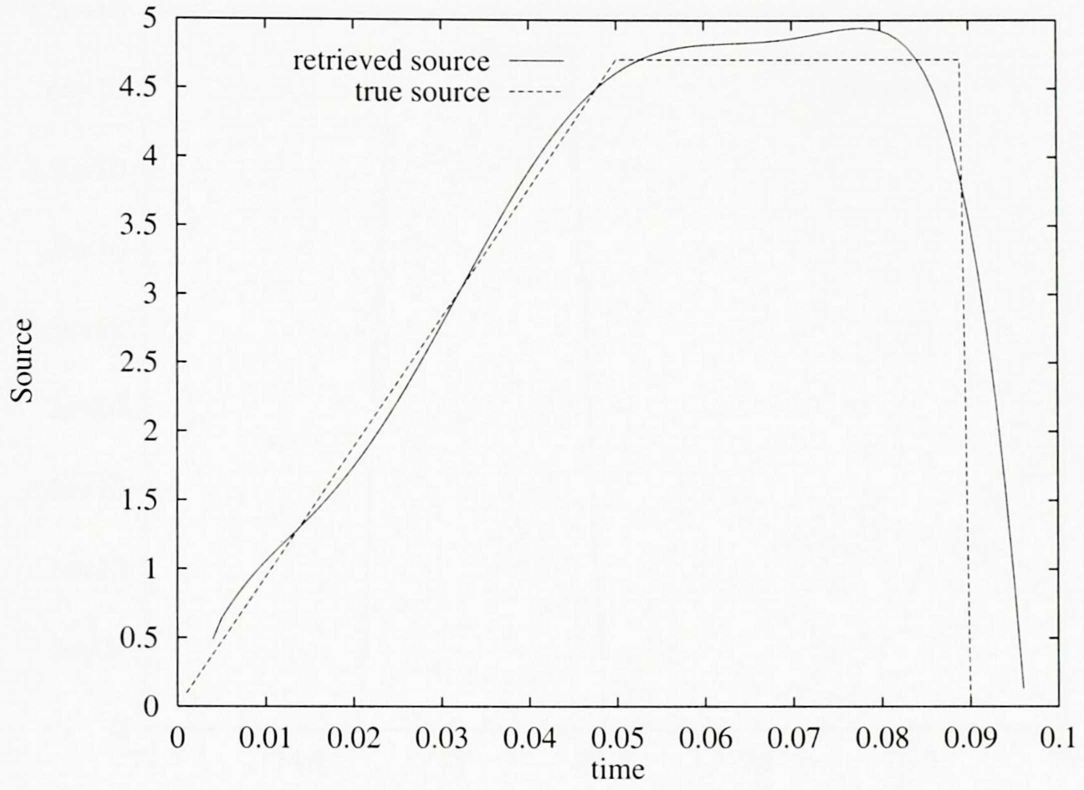


Figure 5.12: Retrieved source using Algorithm 3 with 1% error in sensor data.

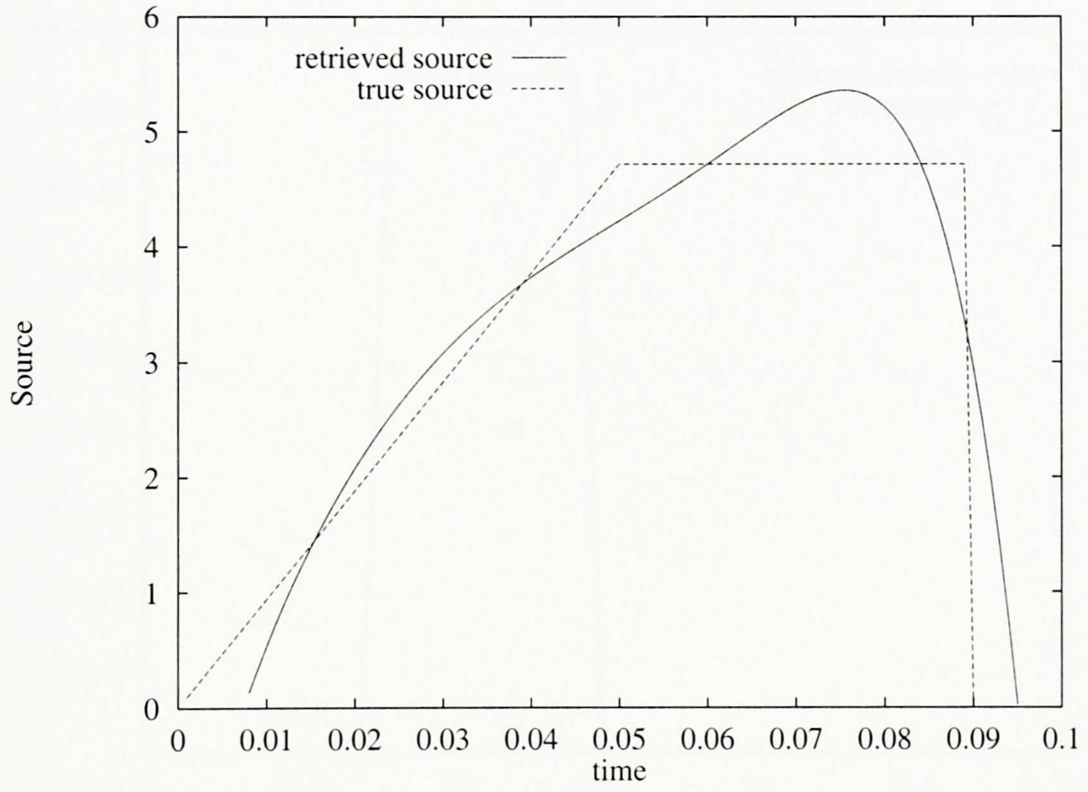


Figure 5.13: Retrieved source using Algorithm 3 with 10% error in sensor data.

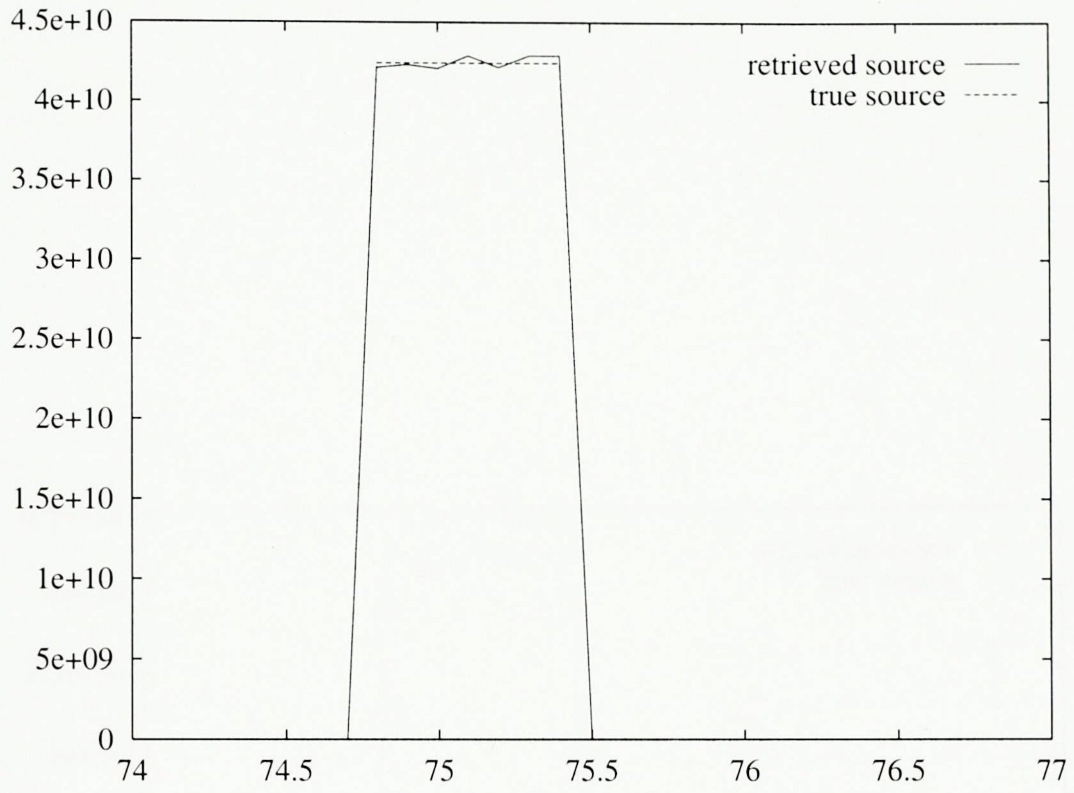


Figure 5.14: Retrieved source for the welding problem with 0.1% error in sensor data.

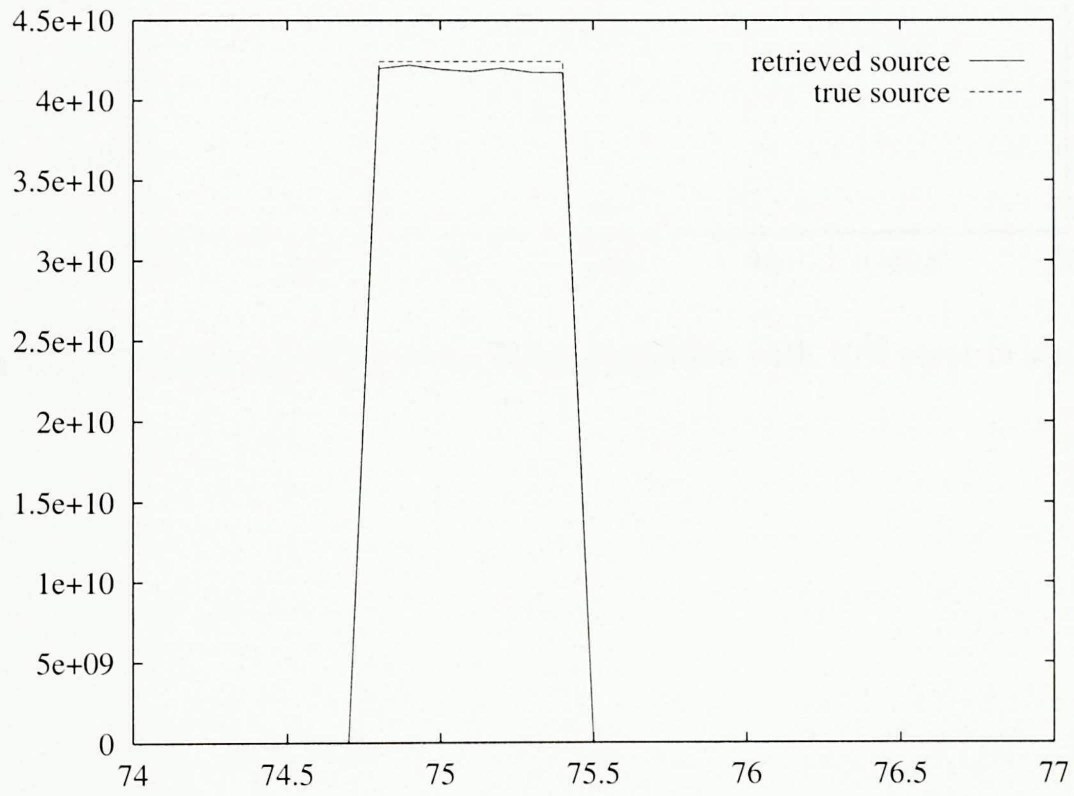


Figure 5.15: Retrieved source for the welding problem with 1% error in sensor data.

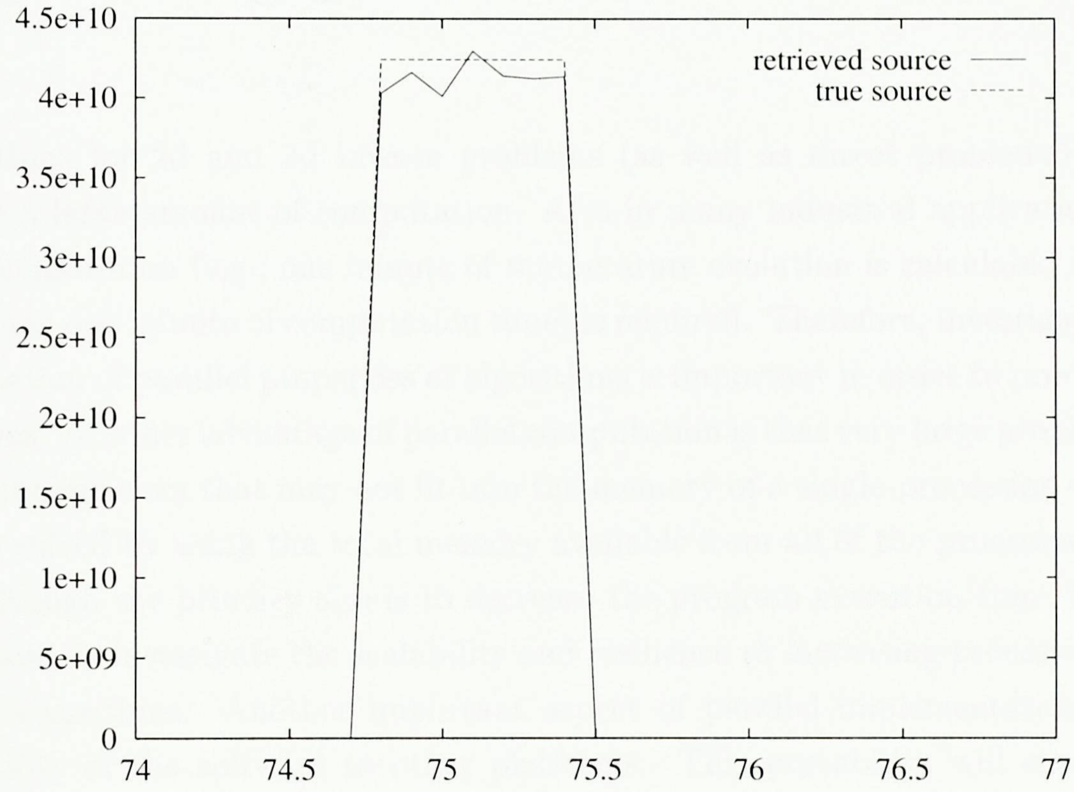


Figure 5.16: Retrieved source for the welding problem with 10% error in sensor data.

Chapter 6

Exploiting parallelism

Algorithms for 2d and 3d inverse problems (as well as direct problems) usually involve a large amount of computation. Also in many industrial applications real-time computation (e.g., one minute of temperature evolution is calculated using no more than one minute of computation time) is required. Therefore, investigation and exploitation of parallel properties of algorithms is important in order to produce fast solutions. Another advantage of parallel computation is that very large problem sizes (i.e., problem sizes that may not fit into the memory of a single-processing element) can be solved by using the total memory available from all of the processors.

Although the primary aim is to decrease the program execution time, it is also important to investigate the scalability and resilience to increasing processor count, of the algorithms. Another important aspect of parallel implementations is the portability of the software to other platforms. This portability will ensure that the software can be run on much larger (in terms of number of processors) parallel machines.

The parallel implementations are mainly tested on a networked workstation cluster (a distributed computing system). This type of parallel computing platform is becoming increasingly popular, due to low hardware and software costs, increased reliability of the networked computers and increased bandwidth of the communication network (Foster, 1995).

A theoretical performance model is also developed in this chapter to study one of the implementations. The theoretical model is validated against experimental results. The study shows the degree of usefulness of such a model for predicting performance of parallel implementation on a network of workstations.

The parallel properties of the three algorithms are examined with respect to the metal cutting problem.

Novel contributions in this chapter are the development of parallelisation strategies for implementing the algorithms developed in Chapter 3 in parallel computing environments. The exploitation of "domain parallelism" and "domain-data parallelism", explained in the following section, will illustrate the simplicity of parallelising the problem partitioning based algorithms. This chapter also introduces a method of grouping inter-processor communication in the context of domain decomposition algorithms. The "inter-domain" and "intra-domain" communications, defined in Section 6.3.2, group inter-processor communications in terms of the algorithms and aid developing good parallelisation strategies.

6.1 Parallelisation strategies

Good parallelisation strategies generate a good load balance amongst processors and reduce the volume of inter-processor communications. Therefore, the following parallelisation strategies for the algorithms primarily try to satisfy these two conditions.

An advantage of using the domain decomposition approach to solve inverse problems is that the technique naturally provides a coarse-grained parallel algorithm. In other words, each subdomain generated due to the problem partitioning can be mapped directly to a processor and these subproblems may be solved concurrently. This *primary* level of parallelism is referred to as "domain parallelism" (Ierotheou et al., 1998). However, domain parallelism has an obvious restriction in that it is limited by the number of subdomains and therefore it does not scale with an increasing number of processors. Since each subdomain consists of a homogeneous subproblem, data partitioning may then be carried out within each subdomain. This *secondary* level of parallelism is referred to as "domain-data parallelism"(Ierotheou et al., 1998). In partitioning the data, the number of grid-points associated with each of the subproblems is divided amongst the processors as evenly as possible. If N denotes the number of grid points and P denotes the number of processors and if $\frac{N}{P}$ is not an integer, then some processors will have more grid points than others. As a result, the remaining data are distributed as evenly as possible so as to minimise the load imbalance amongst the processors.

Considering Algorithm 1, the computational work performed in subdomain D_2 is much smaller (due to sensor and cutter points being very close to each other) than D_1 and D_3. Also, calculations in D_1 can be carried out independently and gradients from D_2 and D_3 at $x = x_c$ are used to retrieve the source term. Hence, to minimise the amount of communications and to give a better load balance amongst the processors, the parallel implementation of Algorithm 1 is carried out as follows. D_1 is assigned to one group of processors and D_2 and D_3 are assigned to another group of processors, see Figure 6.1 (dotted lines represent data partitioning). Each processor in a group exploits the data-parallelism within the subdomain or subdomains it owns.

For Algorithm 2, three groups of processors are created, one subdomain for each group, see Figure 6.1. In order to re-use the in-house software for initial value problem, D_2 is isolated. Again, each processor within a group exploits the data-parallelism in the subdomain it owns, see Figure 6.2. In Algorithms 1 and 2, there is a homogeneous data dependency within each subdomain. Therefore, data partition along x- and y- axes can be carried out arbitrarily within each subdomain.

In Algorithm 3, all three subdomains are solved using one global linearised system. Therefore, data parallelism is used to partition the global linearised system into blocks. However, data partitioning along the x-axis may produce blocks with different sparsity structures, depending upon how D_2 is allocated between the blocks, and therefore with potentially different load balances and interprocessor message patterns. Hence, data partitioning is only carried out along the y-axis.

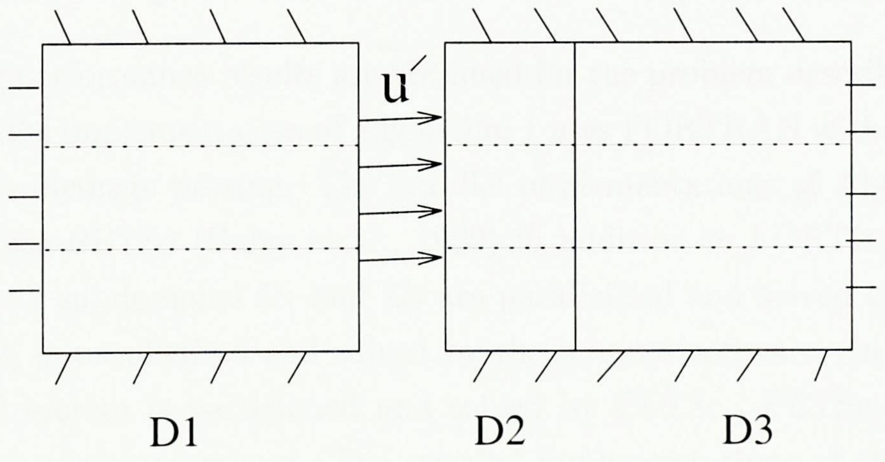


Figure 6.1: An example of partition for Algorithm 1

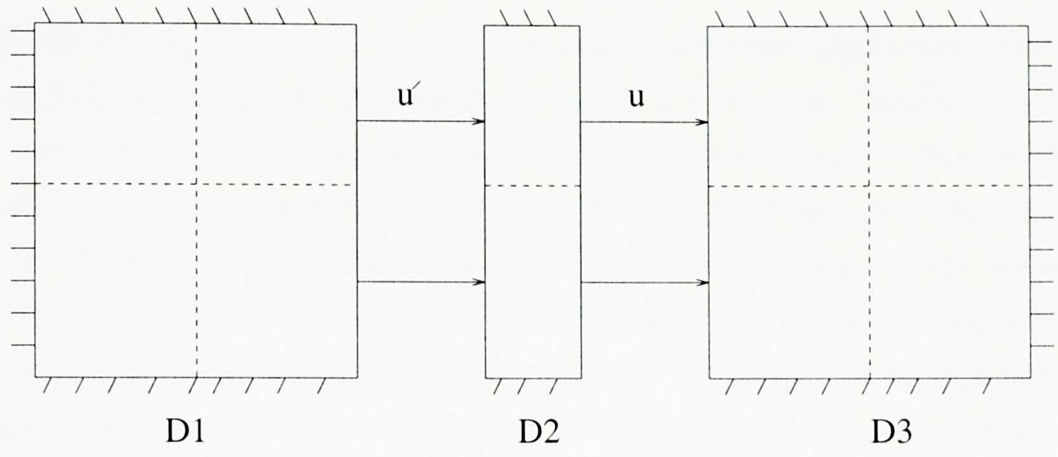


Figure 6.2: An example of partition for Algorithm 2

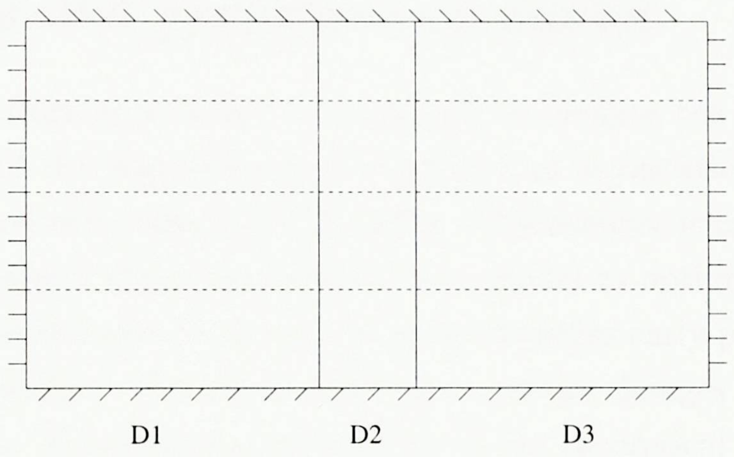


Figure 6.3: An example of partition for Algorithm 3

6.2 Parallel performance results

The following performance results are obtained for the problem described in Section 3.5. The parallel implementation of Algorithm 1 uses FORTRAN with MPI (Forum, 1995) calls for message passing. The parallel implementations of Algorithm 2 and Algorithm 3 use PETSc (Balay et al., 1999) in addition to FORTRAN and MPI. In Algorithms 2 subdomains D_1 and D_3 are parallelised and solved by PETSc and subdomain D_2 is parallelised and solved by the inhouse software. In Algorithm 3, the linearised system is partitioned and solved by PETSc. PETSc uses MPI for inter-processor communications. The parallel implementations of the Algorithms were tested on a network of Sun Sparc 5 workstations connected together by an Ethernet network. The parallel versions are portable to any Unix based platforms since FORTRAN, MPI and PETSc are available for these platforms. The execution times and the speedups are given in Figures 6.4, 6.5 and 6.6. Algorithm 1 shows very good scalability with increasing number of processors. This is due to the explicit methods in the Algorithm which avoid global synchronisations. Only halo (ghost) points need to be communicated between processors. Halo points define a set of points that is required by a processor, to complete its computation, which resides in the memories of its neighbouring processors. The other two Algorithms require additional communications in each step of the iterative scheme that lead to more moderate speedups.

6.3 A parallel performance model

A theoretical performance model is developed to analyse the performance of the parallel version of the Algorithm 1 on a network of workstations. The theoretical model is compared with experimental results. This exercise is carried out to see the practicality of using a theoretical performance model to optimise a parallel implementation. It is well known that such a model identifies early potential bottlenecks and other inefficiencies. Therefore, if necessary, design changes can be made before developing codes. However, the complexity of the theoretical performance model will determine how practical it is for design optimization.

The reason for choosing a network of workstations is that this type of distributed computing platform is becoming more popular and may form the basis of future parallel computing architectures (McColl, 1995; Burgess and Giles, 1995). Hence, the performance evaluation of algorithms on such parallel computing platforms is of interest. Performance evaluation of parallel algorithms is a complex issue due to many factors that may affect the algorithmic performance on a specific architecture (Dongarra and Dunigan, 1997). Such factors are due to software and hardware choices made during an implementation. Software factors include design decisions made such as the decomposition method, the compiler used and, in the case of an explicit message passing model, the communication library used (e.g., MPICH (B. Gropp and Doss, 1994), PVM (Geist et al., 1994) etc.). Some of the hardware factors include the type of processors used, levels of memory hierarchies in the platform (Burgess and Giles, 1995; Faringer, 1997; Keyes et al., 1997), processor to memory bandwidth (Burgess and Giles, 1995), type of communication network etc. A detailed discussion of hardware choices and their effect on the performance of message passing programs can be found (Dongarra and Dunigan, 1997). The MPICH communication library, which is used to implement the parallel version of the algorithm, is an implementation of MPI (Forum, 1995). A study carried out by Nupairoj (Nupairoj and Ni, 1994) considers the performance characteristics of different MPI implementations on a network of workstations. As shown in this study the performance of an MPI program may vary depending on the implementation used.

Modelling all factors would be too complex and may not be practical. For example, if software overheads such as protocols of a communication library are to be modelled, then the model would have to be adjusted every time a new communication library is used. Similarly, hardware features such as processor topologies vary with different platforms. To keep the performance model relatively simple, explicit modelling of some hardware and software factors are avoided. Instead, many of the above factors may be encapsulated by empirical data. For example, a communication test between two processors could be used to encapsulate communication protocol complexities, effects of network bandwidth etc. Such tests can be easily performed on different hardware and software platforms. However, it is important to be aware of the limits of the empirical data. For example, a communication model derived from a set of empirical data would be valid within the limits of message lengths being communicated in the test, and it may not necessarily be valid out of the limits. Also, the communication test described above will not include the effects of network contention due to larger number of processors trying to communicate simultaneously. Such factors become significant particularly when smaller problem sizes are used with a larger number of processors. This will be illustrated in the numerical tests given in Section 6.3.4.

Another factor that is not included in the following performance model, but may be important when solving very large problem sizes (e.g, $>$ 1million nodes), is the memory hierarchy (Burgess and Giles, 1995; Faringer, 1997; Keyes et al., 1997). For the range of problem sizes used in our experiments this factor does not have any significant influence on the predicted performance. The following performance model is built in order to study the performance results of the parallel execution. The theoretical performance model may be validated by using the parallel execution time and it may be used to show the limitations of the model. Computation and communication times for each sub-domain are modelled, in order to describe the program run-time for a time-step.

6.3.1 Computation time

The sub-domains D_1 and D_3 are computed using the five-point explicit scheme (3.8). Define a single precision floating point operation as the average time to execute an addition, subtraction, multiplication or division operation. Let N_1 and N_3 be the number of grid points along x-direction in D_1 and D_3, respectively. Also let N_y be the number of grid points along y-direction. Then from (3.8), with non-linear $k(u)$ and $u^*(y,t)$ as given in Section 3.5, the number of single precision floating point operations per time step in D_1 is $N_y(45N_1 + 22)$. In D_3, $u^*(y,t)$ is absent, hence it performs $N_y(45N_3 + 2)$ single precision floating point operations. Therefore, if P processors are used and T_1, T_3 are the computational times per time-step per processor for D_1 and D_3 respectively, then

$$T_1 = t_r \lceil \frac{N_y}{P} \rceil (45N_1 + 22) \tag{6.1}$$

$$T_3 = t_r \lceil \frac{N_y}{P} \rceil (45N_3 + 2) \tag{6.2}$$

where t_r is the computation time for a single precision floating point operation. Sub-domain D_2 performs the Euler P-C method (3.15). If the number of grid-points in the x-direction is N_2 and the same $k(u)$ as in D_1, D_3 is used, then the computation time for this sub-domain can be written as,

$$T_2 = t_r \lceil \frac{N_y}{P} \rceil (96N_2 + 28) \tag{6.3}$$

Numerical experiments performed on Sun Sparc 5 workstations showed that an integer operation takes about the same time as a single-precision floating point operation. Therefore, the number of integer operations for each sub-domain must also be considered in-order to establish the theoretical computing time. Note, integer operations are required in the implementation to manipulate vectors and therefore all such operations are not apparent from the numerical schemes. The computation times for each sub-domain, including integer operations, are given by,

$$T_1 = \lceil \frac{N_y}{P} \rceil (t_r(45N_1 + 22) + t_i(15N_1 + 17)) \tag{6.4}$$

$$T_2 = \lceil \frac{N_y}{P} \rceil (t_r(96N_2 + 28) + t_i(29N_2 + 2)) \tag{6.5}$$

$$T_3 = \lceil\frac{N_y}{P}\rceil(t_r(45N_3 + 2) + t_i(15N_3 + 32)) \tag{6.6}$$

where t_i is the computation time for an integer operation. The real operations and integer operations are kept separately because this provides the flexibility to apply the model to other distributed computing environments where the two operations may not take the same amount of time.

6.3.2 Communication time

For the type of problem partitioning strategy considered in this thesis, there are two types of communications being carried out. Firstly, inter-domain communications which define the communication **between** sub-domains or groups of sub-domains. Secondly, intra-domain communications which define the communication **within** the sub-domains (Figure 6.7). In the implementation, calculations in D_1 are carried out in a group of processors and calculations in D_2 and D_3 are carried out in another group of processors. The inter-domain communication is between the above two processor-groups. The group that performs D_1 computation sends N_y single precision floating point numbers to the other group. Let $T_C(l)$ be the communication cost of transferring l single precision numbers. Then, the communication cost for the inter-domain communication, T_{inter}, is

$$T_{inter} = T_C(\lceil\frac{N_y}{P}\rceil) \tag{6.7}$$

(Note, P is the number of processors in a processor-group.) For intra-domain communication all processors within a group perform this communication simultaneously. The sub-domains D_1 and D_3 communicate N_1 and N_3 lengths of data, respectively, per intra-domain communication, and there are 4 such communications. These are the two "send" and two "receive" operations required for the data exchange between the neighbour processors (i.e., the two-way intra-domain communication in figure 6.7). In D_2, the length of data being communicated is 1. This is the temperature at the surface of the partition for each grid point along the x-direction. There are 8 such communications, 4 for the predictor step and 4 for the corrector step. Since there are N_2 grid points along the x-direction in D_2, the total number of inter-domain communications for this sub-domain is $8N_2$. Let the communication costs for D_1, D_2 and D_3 be T_{C_1}, T_{C_2} and T_{C_3}, respectively, then

$$T_{C_1} = 4T_C(N_1) \tag{6.8}$$

$$T_{C_2} = 8N_2T_C(1) \quad (6.9)$$

$$T_{C_3} = 4T_C(N_3) \quad (6.10)$$

6.3.3 Parallel execution time

Based on the discussion above, let the two processor groups be g_1 and g_2, then the run time for each group, T_{g_1} and T_{g_2}, can be calculated as:

$$T_{g_1} = T_1 + T_{C_1} + T_{inter} \quad (6.11)$$

$$T_{g_2} = T_2 + T_3 + T_{C_2} + T_{C_3} + T_{inter} \quad (6.12)$$

That is,

$$T_{g_1} = \lceil \frac{N_y}{P} \rceil (t_r(45N_1 + 22) + t_i(15N_1 + 17)) + 4T_C(N_1) + T_C(\lceil \frac{N_y}{P} \rceil) \quad (6.13)$$

$$T_{g_2} = \lceil \tfrac{N_y}{P} \rceil (t_r(96N_2 + 45N_3 + 30) + t_i(29N_2 + 15N_3 + 34))$$

$$+ 8N_2T_C(1) + 4T_C(N_3) + T_C(\lceil \frac{N_y}{P} \rceil) \quad (6.14)$$

Therefore the total run-time for the algorithm, T_r, is assumed to be the run-time of the slowest group, i.e.,

$$T_r = \max(T_{g_1}, T_{g_2}) \quad (6.15)$$

Note, although the message length in inter-domain communication decreases as P increases, T_{inter} may not necessarily decrease due to the increase in competition for the communication network. Also, in the domain-data parallel version of the algorithm there is no intra-domain communication when only two processors are used. Hence, the terms T_{C_1}, T_{C_2} and T_{C_3} are dropped in this case.

6.3.4 Experimental results

To collect observed run times for different problem sizes and number of processors, the domain-data parallel version of the numerical algorithm is implemented using FORTRAN 77 with message passing MPI (MPICH) calls. This implementation is tested on a set of Sun Sparc 5 workstations connected together by a 10 Mbits per second ethernet network which returns performance of up to 600kBytes per second (the network used is a general purpose network not a dedicated testbed). Each workstation has a 70 MHz processor and 32 Mbyte RAM.

Two of the parameters required for the theoretical model are $t_r = 6.0 \times 10^{-7}$s and $t_i = 6.0 \times 10^{-7}$s. Define l as message length, a communication test was carried out to obtain communication times, $T_C(l)$, for messages of increasing lengths. For each length, the send and receive operations are repeated a number of times in order to get good average times for such operations. The range of the message lengths is between 1 to 200 since this is the range that is used in the testing of the algorithm. It is important to note that the communication test only uses two workstations and may not reflect competition for the communication network that is more likely to occur due to larger number of processors trying to communicate simultaneously. Statistical regression was applied to this data to obtain the following relationship between message length and communication time.

$$T_C(l) = 1.02 \times 10^{-3} + 5.19 \times 10^{-7} l - 1.36 \times 10^{-8} l^2 + 1.09 \times 10^{-10} l^3 \quad (6.16)$$

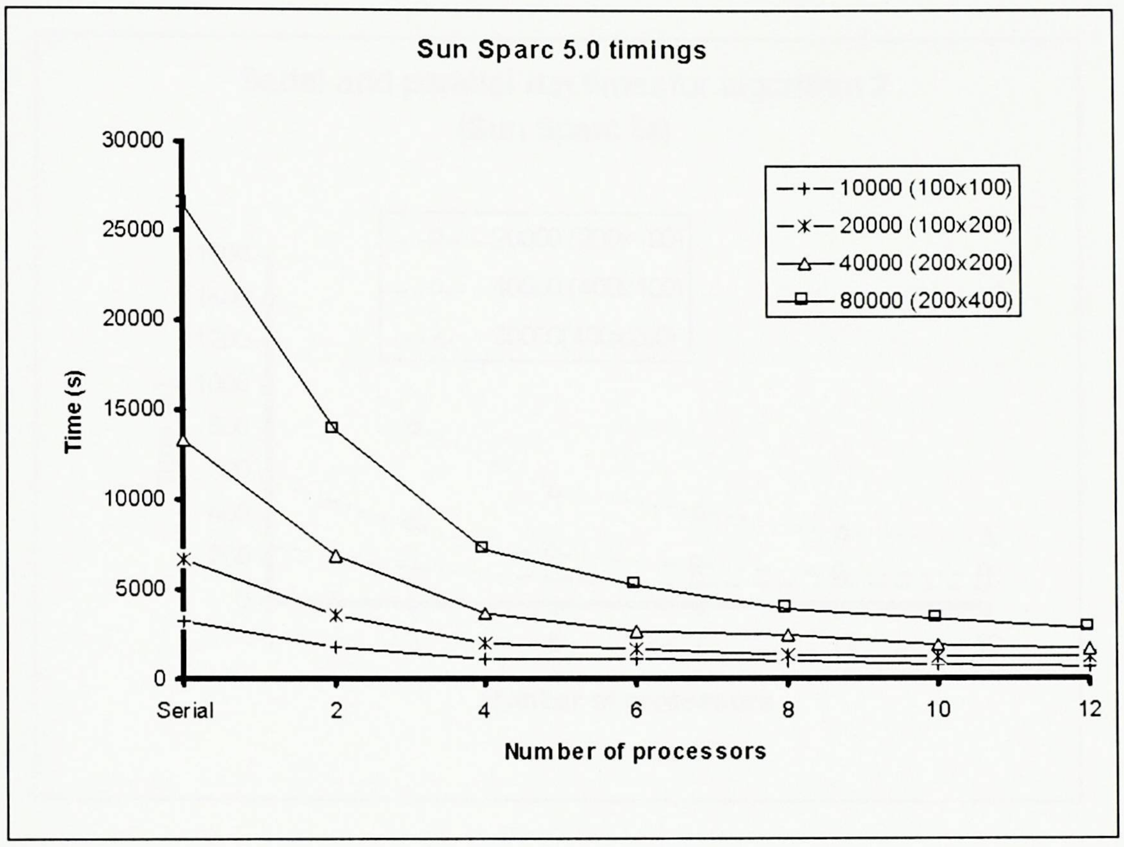
According to equation (6.16), the communication startup latency is 1.02×10^{-3} s. A cubic polynomial fit to the empirical data is chosen because this gives a better least-square fit to the data than the linear- or square- polynomial fit. Figures 6.8 to 6.13 show the theoretical and observed run times for different number of processors and different problem sizes, $Proc$ is the total number of processors used. Since there are only two groups of processors and the load is evenly balanced between them, each group gets half of the total number of processors. i.e., $P = \frac{Proc}{2}$. The problem sizes in Figures 6.8 to 6.13 are displayed as $N_y \times N_x$, where N_y is the total number of grid point along y-direction and N_x is the total number of grid points along x-direction. That is, the five problem sizes in Figures 6.8 to 6.13 should be interpreted as 50×100, 100×100, 200×100, 200×200 and 400×200. The theoretical and observed times are very close for smaller number of processors, e.g., $Proc = 2, 4, 6$. A significant difference starts to appear, in particular, for smaller problem sizes, when a relatively higher number of processors is used, e.g., $Proc = 8, 10, 12$. For smaller problem sizes the communication time, $T_C(l)$, plays a significant part in the overall run times. Prediction of $T_C(l)$, in an ethernet network becomes very complicated due to the fact that a larger number of processors is trying to transmit frames (packets) at the same time, this could potentially result in significant number of frame-collisions in the communication medium. Every time such a collision is detected, frame transmission is delayed (randomly) and predicting this delay is difficult for a general purpose ethernet network. For larger problem sizes the computation time dominates the communication time, hence the effects of

$T_C(l)$ is not significant. In general, the theoretical model gives better predictions for larger problem sizes, irrespective of the number of processors. For example, when a problem size of 400×200 is simulated, the theoretical run time is very close to the observed run time (Figures 6.8 to 6.13).

6.4 Closure

By exploiting the parallel properties of the three algorithms described in Chapter 3 faster versions of the algorithms were developed. As shown in section 6.2, the performance improvements are significant even on a distributed computing environment like the network of Sun workstations. It is clear that use of a domain decomposition method has simplified the parallelisation of the algorithms since subdomains provided us with coarse-grain level parallel algorithms. Two additional tasks were required in order to carry out an effective parallelisation. One of them is to consider whether to combine two or more subdomains into a group of processors or to solve each subdomain in a group of processors so that a good load balance is obtained amongst processors and reduced communication costs can be achieved. The other task is the exploitation of further parallelism within each subdomain in the form of data partitioning. This adheres to the hierarchical domain decomposition discussed in Chapter 3.

The performance model developed for Algorithm 1 shows that all the implementation details have to be considered in order to derive an accurate model. Some factors, such as communication protocol complexities that were required, were derived from empirical data. This was necessary in order to develop a useful and practical model. The performance model gives accurate predictions for large problem sizes.



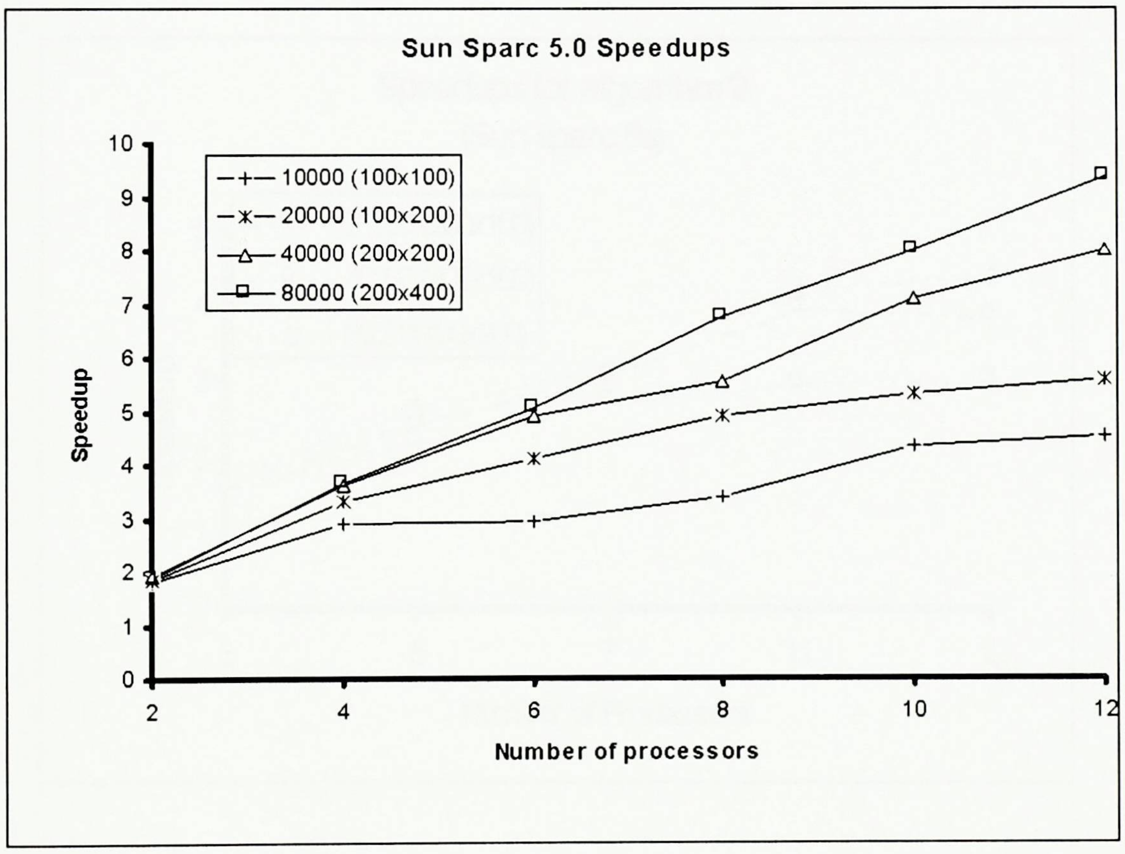
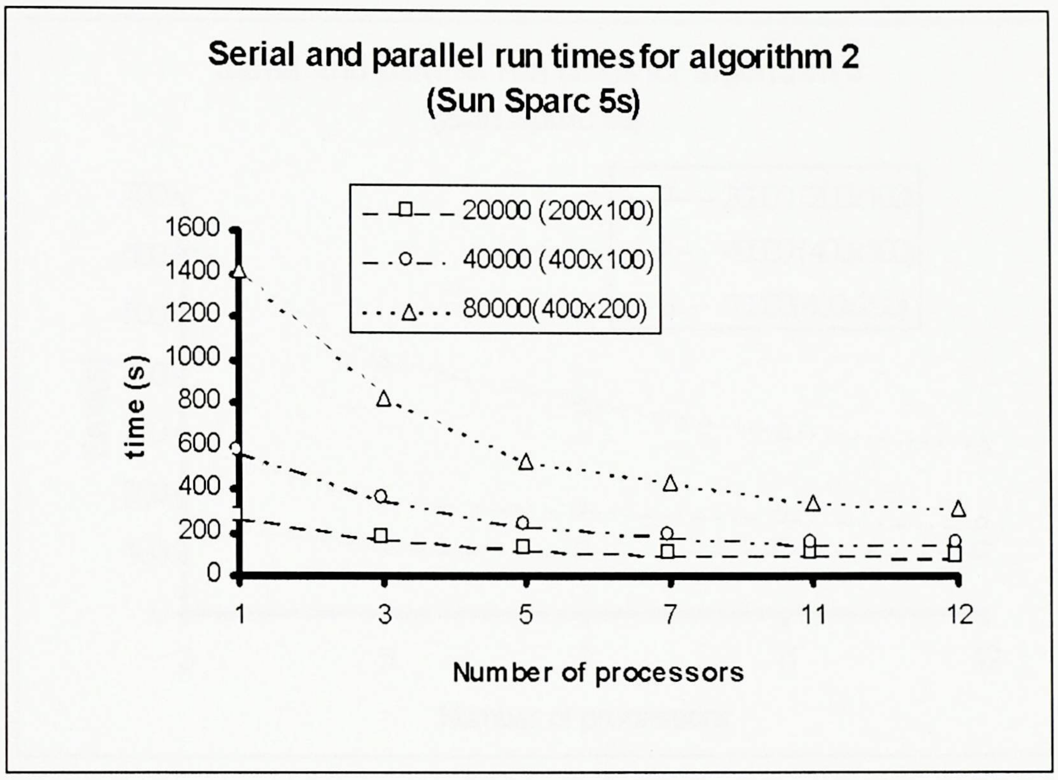


Figure 6.4: Parallel performance results for Algorithm 1.



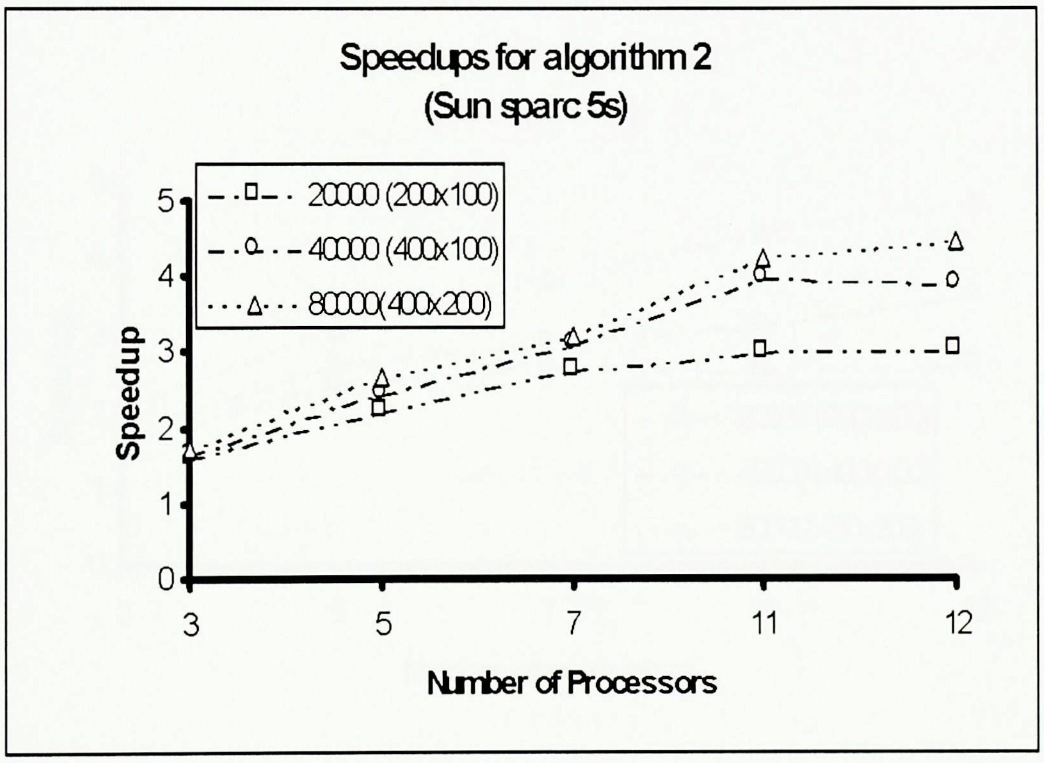
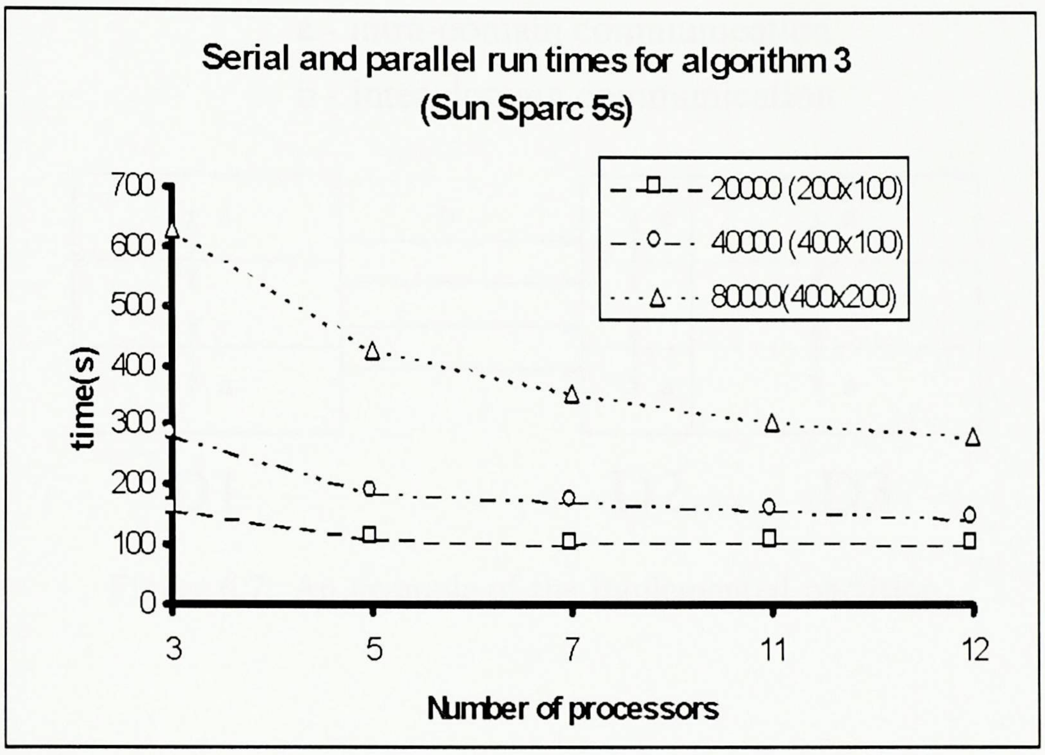


Figure 6.5: Parallel performance results for Algorithm 2.



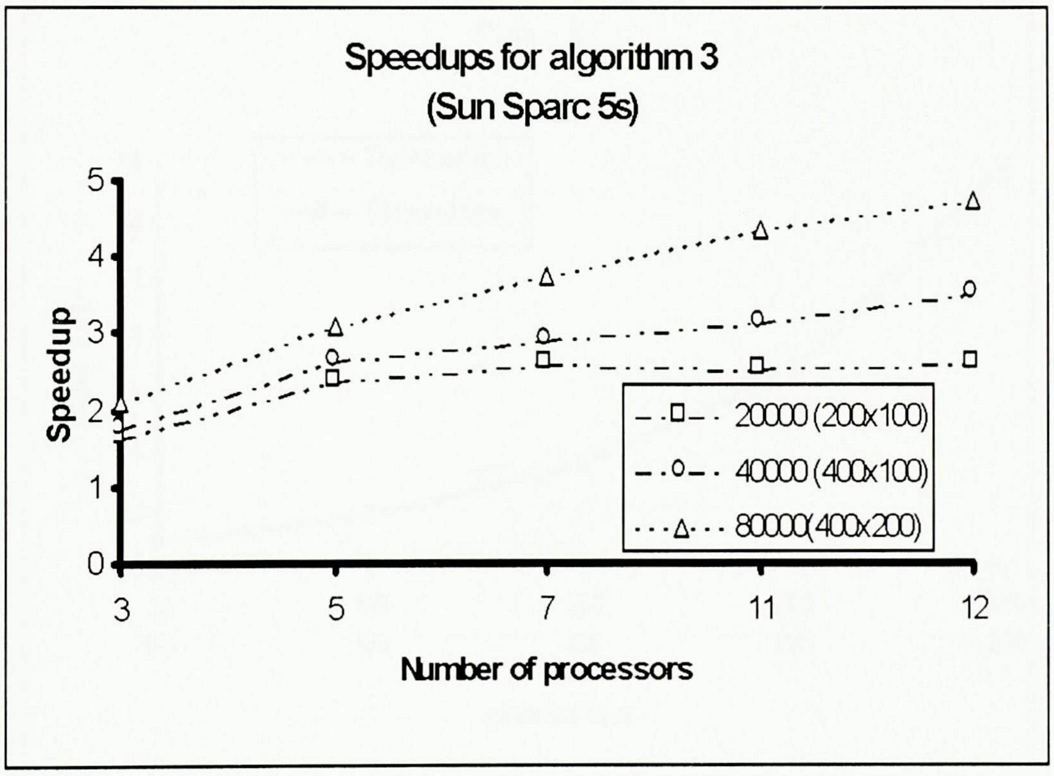


Figure 6.6: Parallel performance results for Algorithm 3.

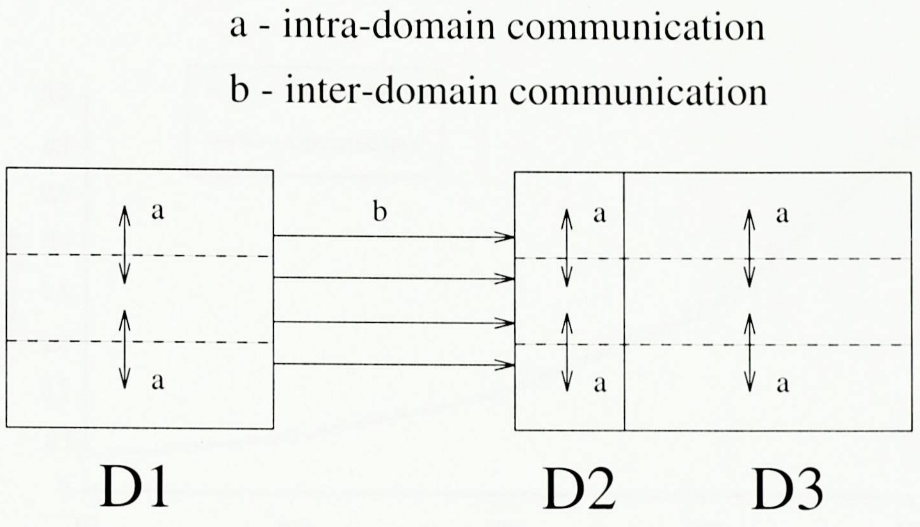

a - intra-domain communication
b - inter-domain communication
a
b
D1
D2
D3

Figure 6.7: An example of the implemented partition.

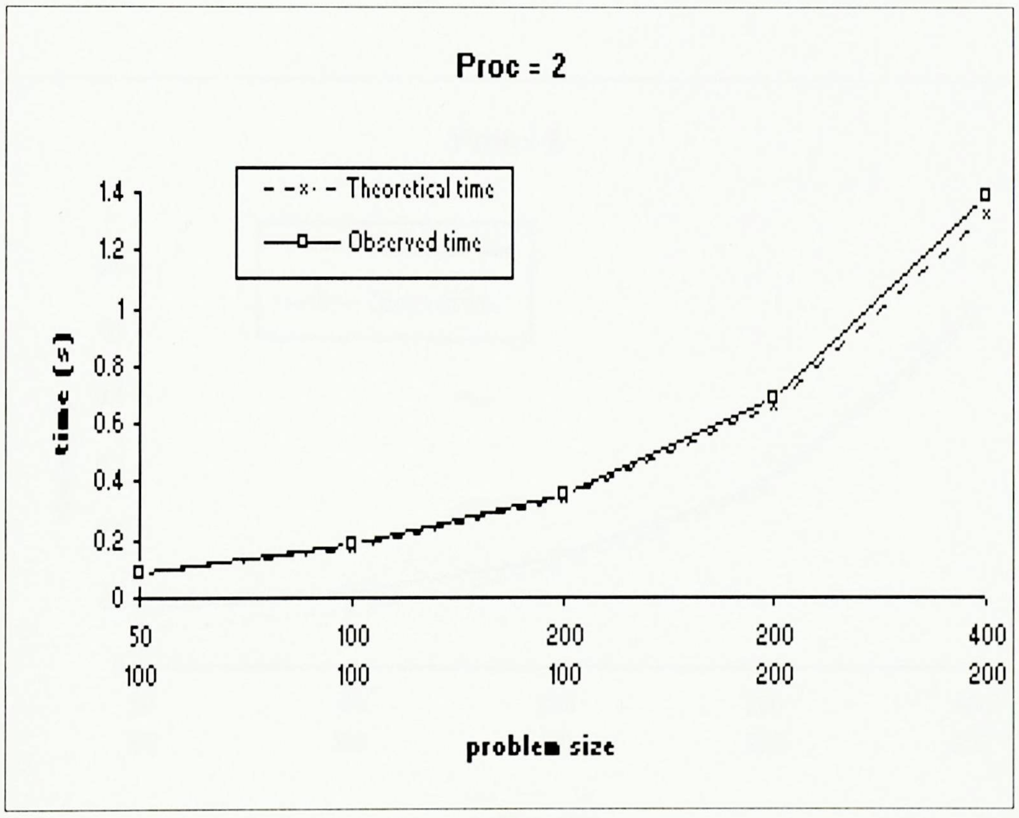


Figure 6.8: Theoretical and observed run times for Sun Sparc 5s when $Proc = 2$.

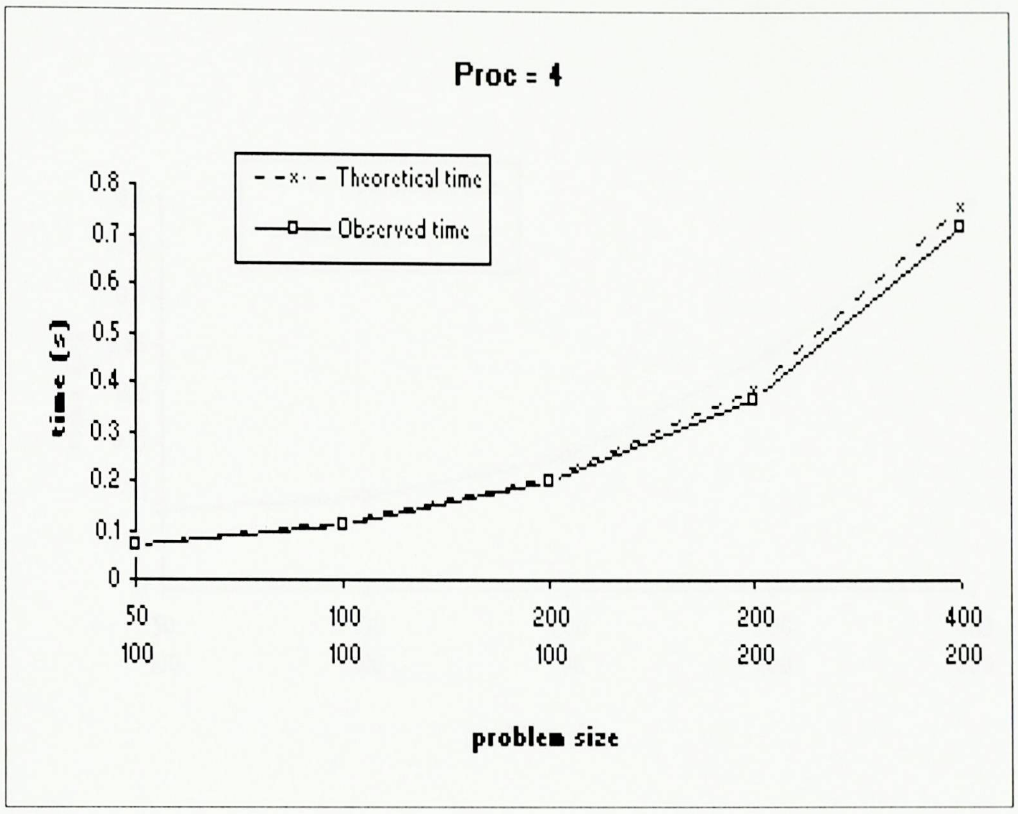


Figure 6.9: Theoretical and observed run times for Sun Sparc 5s when $Proc = 4$.

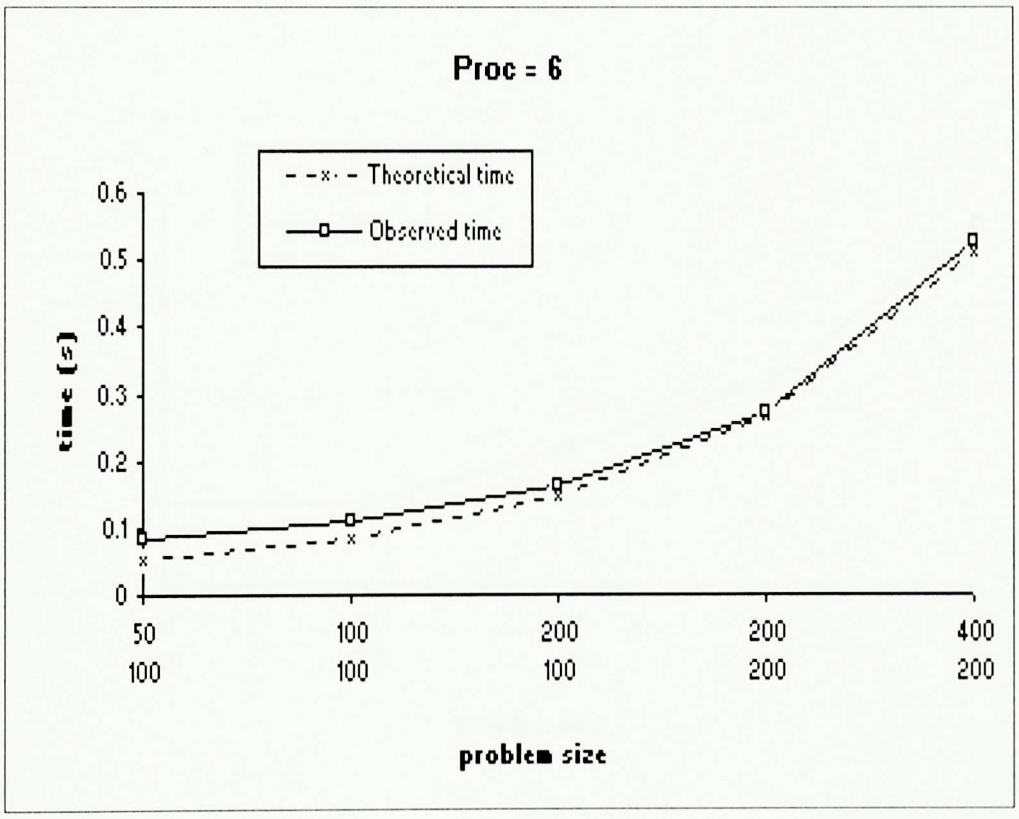


Figure 6.10: Theoretical and observed run times for Sun Sparc 5s when $Proc = 6$.

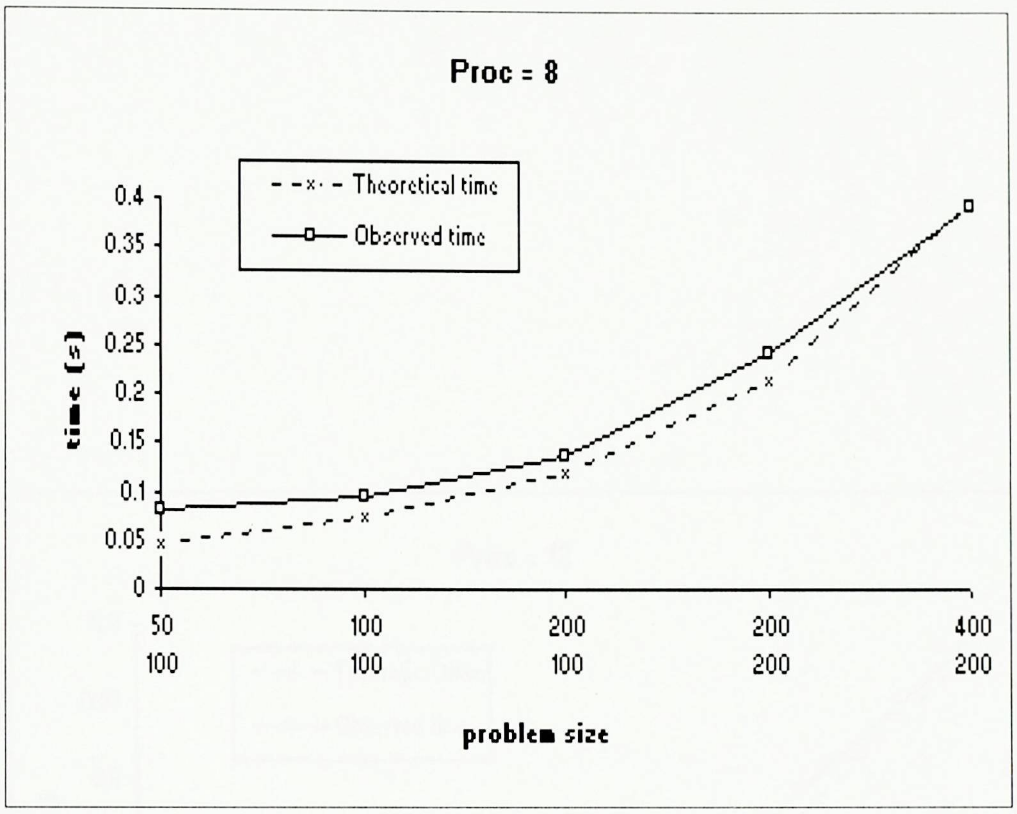


Figure 6.11: Theoretical and observed run times for Sun Sparc 5s when $Proc = 8$.

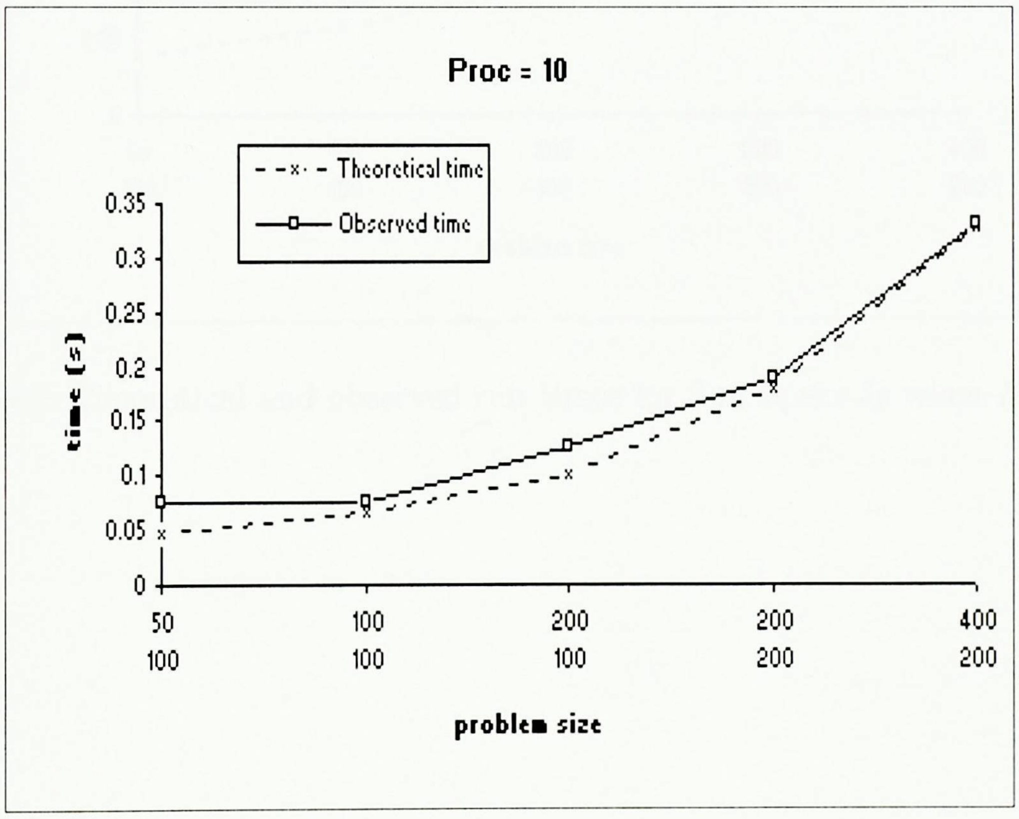


Figure 6.12: Theoretical and observed run times for Sun Sparc 5s when $Proc = 10$.

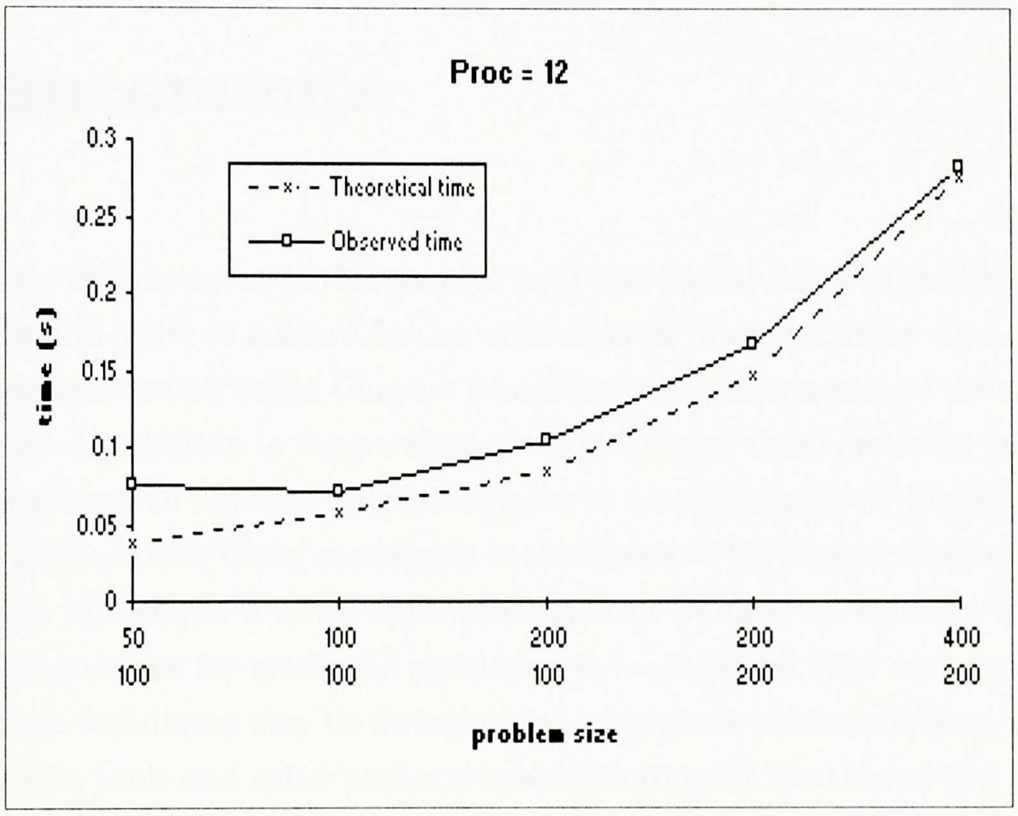


Figure 6.13: Theoretical and observed run times for Sun Sparc 5s when $Proc = 12$.

Chapter 7

Further performance enhancements

Performance enhancement in this chapter (and this thesis) refers to the fine tuning of an algorithm in order to achieve further reduction in total execution time. Parallelisation strategies considered in Chapter 6 improved the performance of the algorithms significantly. In addition to the parallelisation strategies, there are other factors that can be considered to improve the performance of an existing algorithm for nonlinear problems such as those being considered in this thesis. This chapter aims to highlight such factors and shows a novel approach that can be used to enhance the performance of algorithms for nonlinear problems. It is envisaged that such performance enhancement techniques may be incorporated into future automatic/semi-automatic parallelisation tools and other automatic/semi-automatic load balancing tools.

Performance enhancement factors can be separated into pre-implementation and implementation dependent factors (implementation in here and this thesis refers to software implementation). Pre-implementation factors are mainly due to mathematical properties of a given problem. Some of the pre-implementation factors include the choice of overlap region (in Schwarz type domain decomposition), preconditioner and exploitation of nonuniform nonlinearity. Some of the implementation dependent factors include the use of good data structures, efficient memory allocation (by using data locality) and compiler options. Some of the latter approaches are briefly discussed in previous chapters. The contribution of this chapter is to exploit a pre-implementation factor, namely the nonuniform nonlinearity, in order to further enhance the performance of algorithms for nonlinear problems.

The novel contribution of this chapter is the development of a method to exploit nonuniform nonlinearity in applications in order to improve performance of algorithms for such applications. A metric is proposed in order to implement the method.

7.1 Exploiting nonuniform nonlinearity

As mentioned in Chapter 2, the domain decomposition can be applied by considering the nonlinear properties of a problem. One factor that may attribute to the performance is the change of nonlinearity in a partial differential equation. As pointed out in (Keyes, 1998), when solving a problem with nonuniform nonlinearity, using an iterative method such as the Newton' s method, most of the work is being performed in the subregion where severe change of nonlinearity occurs. The regions with less severe change of nonlinearity require less work (number of iterations). If the problem is solved in the original domain, then unnecessary work is forced upon the regions with less severe change of nonlinearity. This can be avoided and the amount of computational work can be reduced by applying a domain decomposition method. If a subregion of a domain contains severe change of nonlinearity and the rest of the domain has moderate change of nonlinearity then the nonuniform nonlinearity can be exploited as follows. Decompose the domain so that the region containing the severe change of nonlinearity can be captured into one subdomain. Then solve the subdomain with the severe change of nonlinearity till convergence and then reset the corresponding region of the full domain with the converged result. Finally, solve the full domain to convergence. Performance improvement is expected since unnecessary computational work is avoided.

Since most of the work is done in the subdomain with the severe change of nonlinearity, it is expected that the number of iterations to solve the *full* domain, with the improved initial guess, will be reduced. Since the subdomain is small, effective number of iteration to solve the nonlinear problem is reduced. Therefore, total execution time is expected to be reduced. If the severe change of nonlinearity is confined to a very small region then the overhead is minimal and significant performance improvement is expected.

The subproblems in subdomains D_1 and D_2 in the metal cutting problem (Chapter 3) contain nonlinear heat conduction problems with high degree of nonlinearity.

Therefore, some preliminary studies are carried out in this chapter to see the effectiveness of exploiting such a property to enhance the performance of the algorithms. The problem considered is similar to the metal cutting problem. However, currently, only a steady state problem is considered in order to simplify the study.

7.2 Nonlinearity of heat conduction problems

To study the feasibility of exploiting the property of nonuniform nonlinearity in order to further improve the performance of the algorithm the following dimensionless 2d nonlinear, steady, heat conduction equation, defined in the domain $D = \{(x, y) : 0 < x < 1 \text{ and } 0 < y < 1\}$, is considered:

$$\frac{\partial}{\partial x}(k(u)\frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}(k(u)\frac{\partial u}{\partial y}) = 0 \in D, \tag{7.1}$$

where $u(x, y)$ is a dimensionless temperature distribution, $k(u)$ is the conductivity and the boundary condition is $u = g$ on ∂D. Here $k(u)$ is the main ingredient that introduces nonlinearity into the problem.

Domain decomposition should be applied such that much of the severe change of nonlinearity is captured into one subdomain. The main concern here is the location of the domain partitioning such that much of the severe change of nonlinearity is able to be captured into one subdomain. This can be achieved with the knowledge of the problem (intuition) as the example shows. However, the development of a metric to measure the extent of nonlinearity will aid the automatic placement of the domain partitioning. One possible metric is to choose

$$\max_{x \in D} \frac{\partial^2 u}{\partial x^2} \tag{7.2}$$

which has a direct relationship to the change of nonlinearity along spatial coordinates. Also, the experiments carried out with physical intuition show that the metric gives a good indication of where the domain partitioning should be applied.

7.3 Numerical tests

For the present study, the heat conduction equation (7.1) is solved with $k(u) = \frac{1}{1+cu^2}$ and $g = Aexp(-\sigma x^2)$. Figure 7.1 shows the steady state temperature distribution for this nonlinear problem. The reduced heat conductivity in the region of high temperature causes the steady state temperature distribution to develop high gradients and severe change of nonlinearity in the left region of the domain. In this example, the original domain is decomposed into two subdomains by placing a domain partitioning at a point in x-axis such that the left subdomain captures much of the severe change of nonlinearity. Then the nonuniform nonlinearity is exploited as follows. First, the left subdomain is solved until a certain convergence criterion has been satisfied and then the corresponding left region of the original (full) domain is reset with the converged result. Then the problem in the original domain is solved until the original problem is converged. To observe the effect on the performance of the algorithm, the domain partitioning is placed at various positions along x-axis, based on physical intuition.

Two methods are compared and are implemented using FORTRAN with PETSc. One method is to solve equation (7.1) in the original domain without exploiting the nonuniform nonlinearity. The other method is to carry out the domain decomposition and then exploit the nonuniform nonlinearity, as described above, in order to solve the equation (7.1). Two sets of experiments are carried out based on parameters A, σ and c. The first set uses $A = 2$, $\sigma = 40$ and $c = 30$ and the second set uses $A = 2$, $\sigma = 50$ and $c = 25$. Performance is measured in terms of total execution time and number of effective iterations. In the figures 7.2, 7.4 and 7.6 the "full-domain time" represents the total time it took to solve the problem in the original domain *without* exploiting the nonuniform nonlinearity. The "partial-full domain time" represents the total time it took to solve the problem with the domain partitioning and then exploiting nonuniform nonlinearity as explained earlier. The effective number of iterations is calculated as follows. The number of iterations that were needed to solve the left subdomain is multiplied by the ratio of left subdomain size to full domain size and the resulting number is then added to the number of iterations that took to solve the full domain with the reset left subregion.

Figure 7.2 shows the effect on the total execution time (in seconds) and the number of effective iterations as the domain partitioning point is moved along the x-axis. For this case the minimum execution time is reached at $x = 0.21875$ which is the optimum partitioning point, the minimum effective number of iterations is reached at $x = 0.2125$. Figure 7.3 shows the change of temperature gradient along x-axis $(\frac{\partial^2 u}{\partial x^2})$ and it shows the $\max_{x \in D} \frac{\partial^2 u}{\partial x^2} = 0.19375$ which is close to the optimum partition point. Figures 7.4 and 7.5 show performance results for a finer grid (320×80) with the same parameters and they show that the optimum partition point is at $x = 0.2125$. It can be seen from these results that the minimum number of effective iterations does not necessarily correspond to the minimum total execution time and that the metric places the partitioner closer to both of them. Figure 7.6 shows performance results for the second set of parameters. Here the minimum total execution time and effective number of iterations are reached at $x = 0.1875$. As can be seen from Figure 7.7 the maximum change of temperature gradient is reached at $x = 0.175$. It is clear from the results that the metric (7.2) does not give the optimum partitioning point. However, it gives a good indication of where the domain partitioning should be placed in order to gain a significant performance improvement. Other metrics such as $\max_{x \in D}(\frac{\partial^2 k}{\partial u^2})$ and $\min_{x \in D}(\frac{\partial u}{\partial x})$ are studied but metric (7.2) places the partitioning closer to the optimum point.

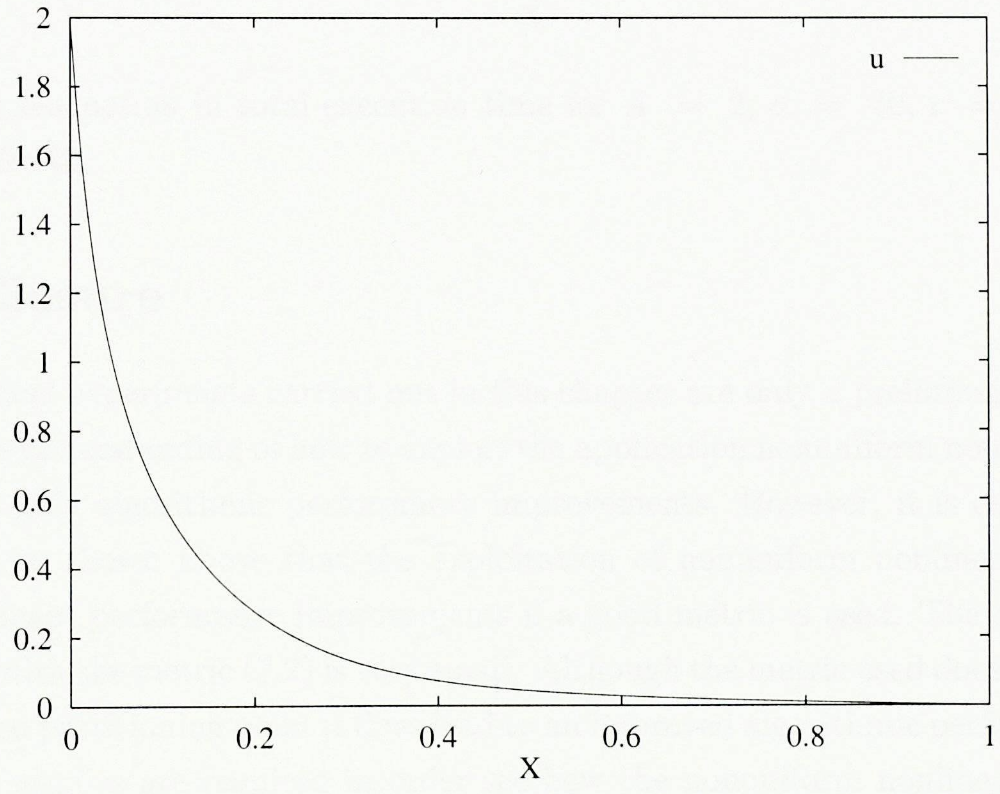


Figure 7.1: Temperature distribution at y=0.125 for $A = 2$, $\sigma = 40$, $c = 30$ and mesh = 160×40.

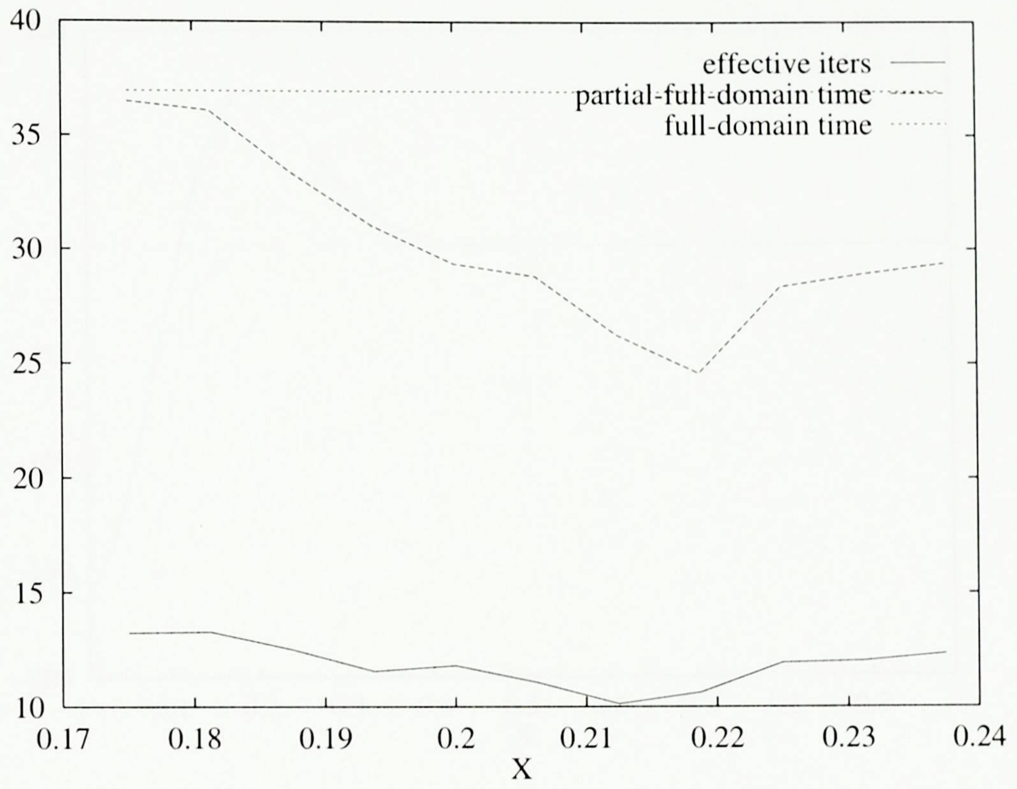


Figure 7.2: Reduction in total execution time for $A = 2$, $\sigma = 40$, $c = 30$ and mesh = 160×40.

7.4 Closure

The numerical experiments carried out in this chapter are only a preliminary study towards the understanding of how to exploit the application nonuniform nonlinearity in order to gain algorithmic performance improvements. However, it is clear from the results as shown above that the exploitation of nonuniform nonlinearity can yield significant performance improvements if a good metric is used. The overhead associated with the metric (7.2) is very small. Although the metric used does not give the optimum partitioning point it does lead to an improved algorithmic performance.

Further studies are required in order see how the nonuniform nonlinearity can be exploited in time dependent problems such as ones dealt in Chapters 3 and 4. In time dependent problems, the subdomain partitioning can be carried out at the beginning of each time step in addition to the beginning of each Newton and/or linear iteration.

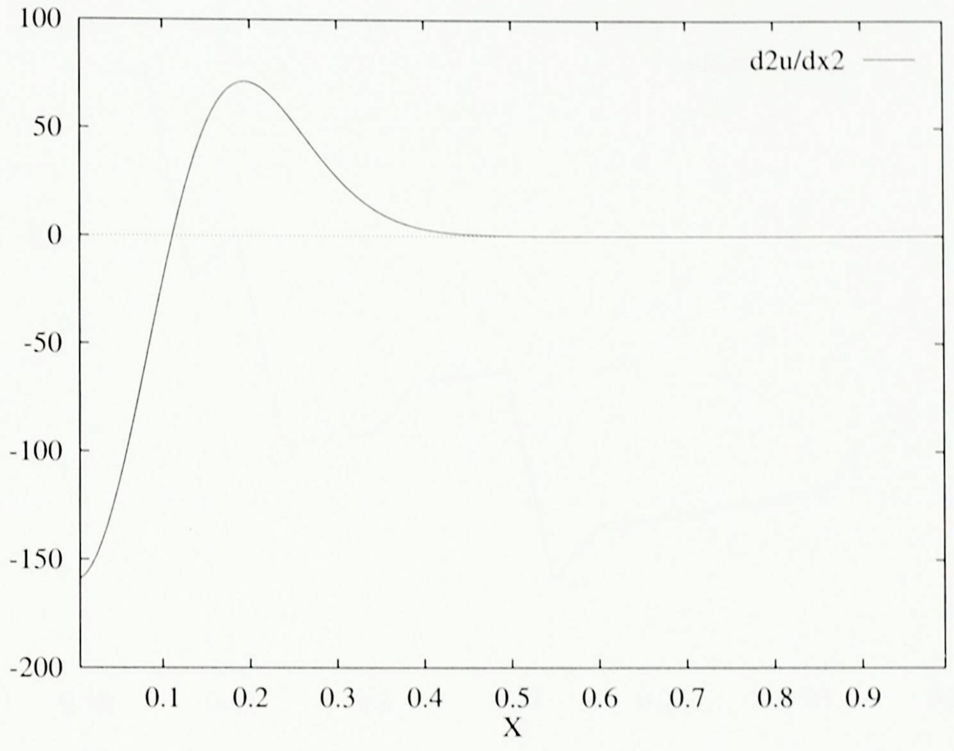


Figure 7.3: Change of temperature gradient for $A = 2$, $\sigma = 40$, $c = 30$ and mesh = 160×40.

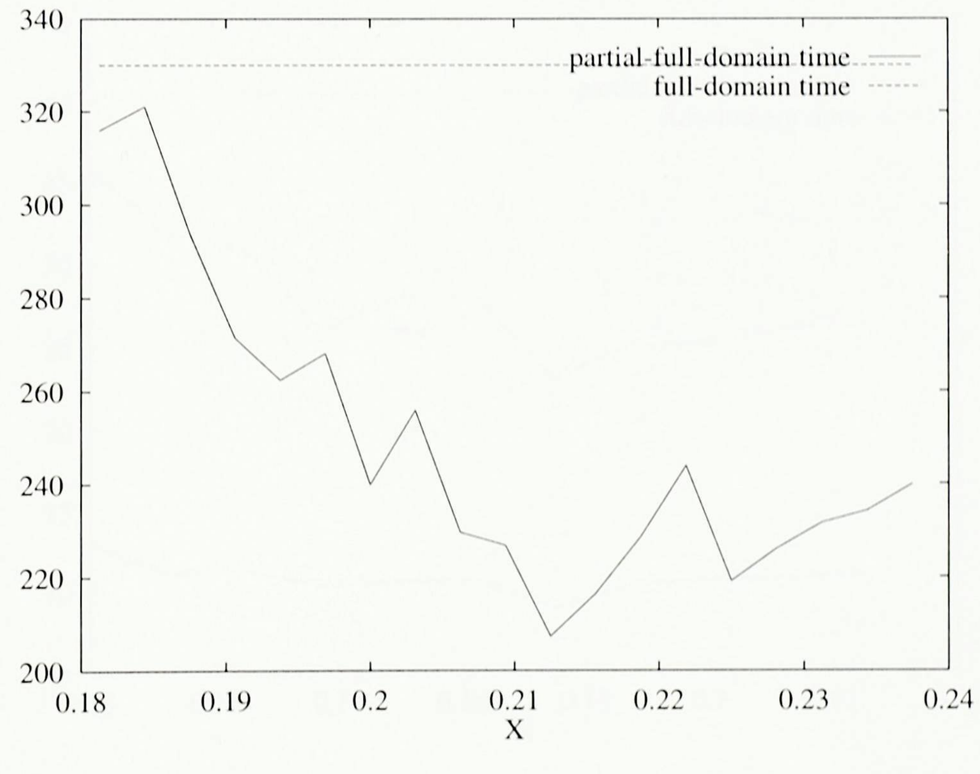


Figure 7.4: Reduction in total execution time for $A = 2$, $\sigma = 40$, $c = 30$ and mesh = 320×80.

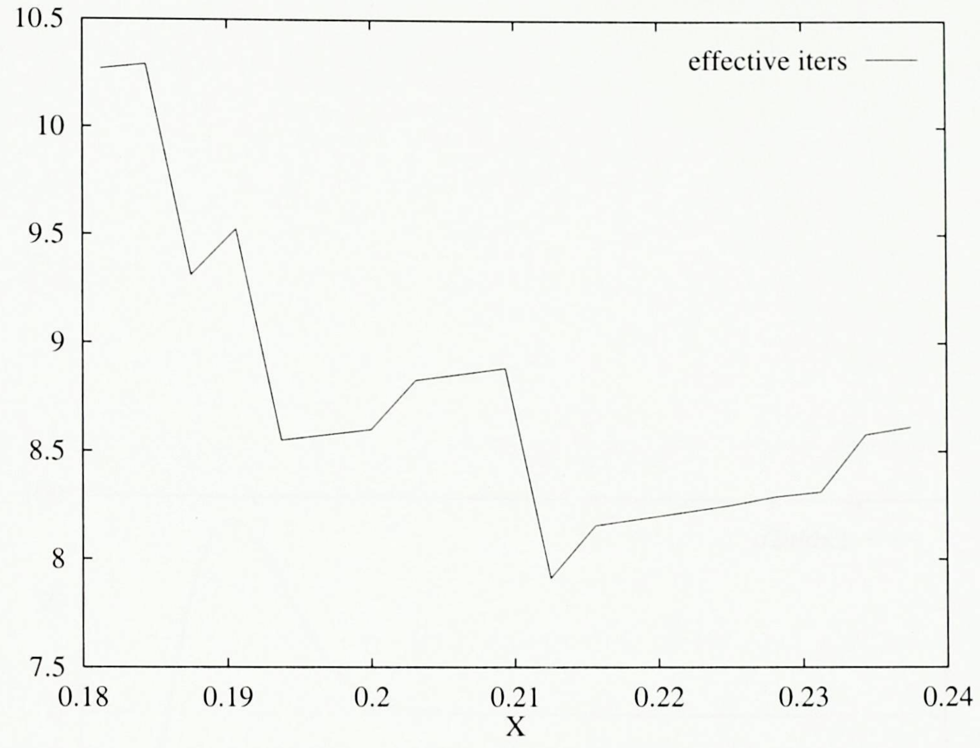


Figure 7.5: Reduction in effective number of iterations for $A = 2$, $\sigma = 40$, $c = 30$ and mesh = 320×80.

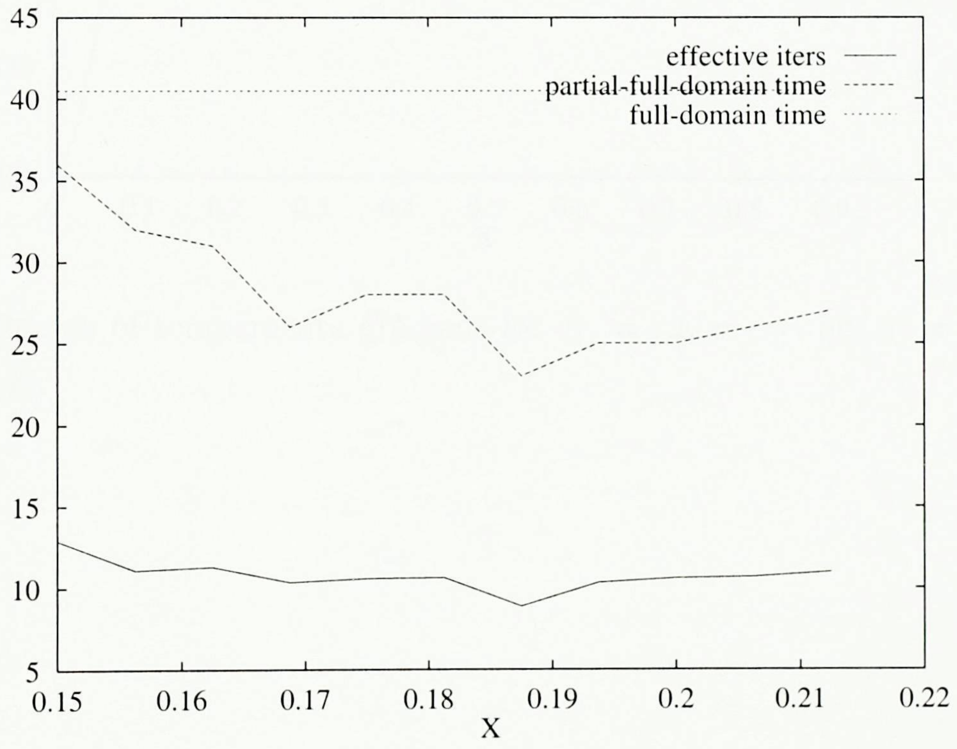


Figure 7.6: Reduction in total execution time for $A = 2$, $\sigma = 50$, $c = 25$ and mesh = 160×40.

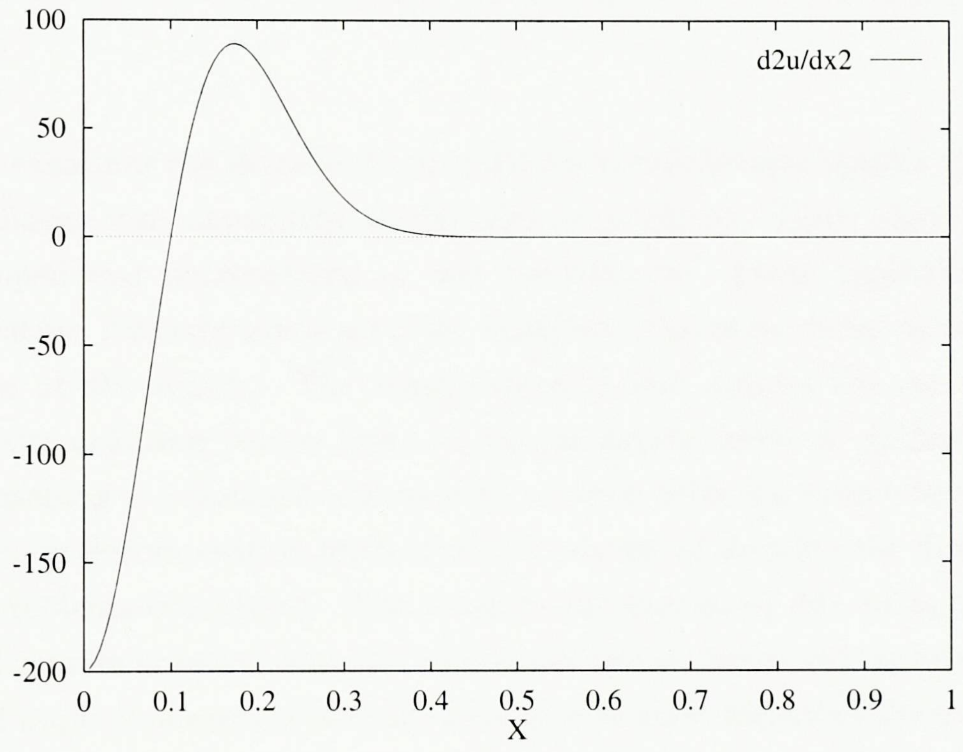


Figure 7.7: Change of temperature gradient for $A = 2$, $\sigma = 50$, $c = 25$ and mesh = 160×40.

Chapter 8

Conclusions and further work

This thesis examines the domain decomposition hierarchy and applies the concept to two nonlinear time dependent inverse source problems. Three algorithms have been developed and implemented to test the concept. These algorithms require the temperature measurements near the unknown source in order to retrieve the temperature at the source. The temperature at and around the source is used to retrieve the unknown source term. A unique source retrieval method based on physical reasoning is developed and used to retrieve unknown source terms.

The domain decomposition method used is shown to simplify the discontinuous nonlinear problems considered. The subdomains generated due to application of the method contain homogeneous and continuous subproblems and therefore existing explicit and implicit numerical schemes were used to solve the subproblems (Chapter 3). The use of existing numerical schemes also enable us to use existing software (e.g., PETSc) as an alternative to in-house software developed to solve the subproblems. The use of existing numerical methods and software provided flexible, efficient, fast and reliable algorithms. Algorithms developed are validated in the context of a metal cutting (Chapter 3) and a welding problem (Chapter 4). The algorithms are shown to retrieve temperature and source terms accurately.

It is shown, in Chapter 5, that a noise treatment procedure such as a least square polynomial fitting should be applied to the temperature measurements in order to provide a stable algorithm which retrieves sensible source terms. The least squares method provides an automatic noise treatment procedure (i.e., no parameter tuning is necessary) without a priori knowledge. The method isolates the noise treatment from the rest of the solution process and therefore it can be easily adapted for different situations.

Parallelism of the algorithms is exploited to develop a faster version of the algorithms. As shown in Chapter 6, the domain decomposition method provides the primary level parallelism (domain parallelism). Since the subdomains are homogeneous, data partition is carried out in each subdomain to provide the secondary level parallelism (domain-data parallelism). This enabled us to use more processors. Experiments carried out in a network of workstations showed that significantly faster algorithms can be developed using such a parallelisation strategy.

A preliminary study is carried out to see whether the change of nonlinearity can be exploited to further improve the performance of the algorithms. The study shows that a significant performance improvement can be achieved with the use of a good metric in order to make decisions on the domain partitioning. The metric used in this thesis gives moderate performance improvements.

As a by-product of this research a set of software tools is developed which can simulate inverse source problems accurately, even with considerably noisy measurement data, on sequential and parallel computing platforms.

Further work may begin by extending the algorithms for three dimensional computation. This should not provide significant complexities except increase in the amount of computation and more "sensors" may be required. The increased computation will provide better speedups in parallel computing platforms due to improved computation to communication ratio.

The extension of the algorithms for multiple unknown source terms is straightforward with extra sensors placed near each additional unknown source. The domain decomposition method generates two extra subdomains for each additional unknown source term. Future studies should also consider location of sensors and the accuracy of retrieval.

A significantly improved noise treatment process can further increase the accuracy of the source retrieval when working with noisy measurement data.

The parallel solution of welding problem can be carried along the same lines as the metal cutting problem. It should be noted that the amount of computation near the source is much greater than computation away from the source and that the source is moving. Therefore, a static domain partition may produce subdomains with different work loads as the solver is marching along time. This may have to be dealt with by dynamically allocating a different number of subdomains to a processor or dynamically repartitioning the subdomains so that each processor will have approximately the same amount of work. It may be possible to work out a reallocation or repartitioning schedule at pre-processor stage as the variation in computational load is predictable. This will decrease the overheads associated with working out when and where to carry out the reallocation or repartitioning.

The development of a good metric that exploits the change of nonlinearity of a differential equation defined in a given domain, as explained in Chapter 7, will provide significant performance improvements to the algorithms.

Finally this work lays the foundations for an efficient, self contained and reliable source recovery system which may be encapsulated in an automatic control system in many production situations.

References

Argyris, J. H., Szimmat, J., and William, K. J. (1985). Finite element analysis of arc-welding processes. In Lewis, R. W. and Morgan, K., editors, *Numerical Methods in Heat Transfer*, volume 3, pages 1–34. John Wiley and Sons.

B. Gropp, R. Lusk, T. S. and Doss, N. (1994). Portable MPI model implementation. *Argonne National Laboratory.*

Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F. (1997). Efficient management of parallelism in object oriented numerical software libraries. In Arge, E., Bruaset, A. M., and Langtangen, H. P., editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhauser Press.

Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F. (1999). PETSc 2.0 users manual. Technical Report ANL-95/11 - Revision 2.0.24, Argonne National Laboratory.

Beck, J. V., Blackwell, B., and St.Clair Jr, C. R. (1985). *Inverse Heat Conduction: Ill-Posed Problems.* Wiley-Interscience, London.

Bjørstad, P. E. and Karstad, T. (1995). Domain decomposition, parallel computing and petroleum engineering. In Keyes, D. E., Saad, Y., and Truhlar, D. G., editors, *Domain-Based parallelism and problem decomposition methods in computational science and engineering*, pages 39–56. siam.

Bryan, K. and Caudill, Jr., L. F. (1994). An inverse problem in thermal imaging. Technical Report 94-99, Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Cneter, Hampton, VA 23681-0001.

Burgess, D. A. and Giles, M. B. (1995). Renumbering unstructured grids to improve the performance of codes on hierarchical memory machines. *Report no. 95/06, Numerical Analysis Group, Oxford University Computing Laboratory.*

Burggraf, O. R. (1964). An exact solution of the inverse problem in heat conduction theory and applications. *J. Heat Transfer*, 86C:373–382.

Cannon, J. R., DuChateau, P., and Steube, K. (1990). Unknown ingredient inverse problems and trace-type functional differential equations. In Colton, D., Ewing, R., and Rundell, W., editors, *INVERSE PROBLEMS IN PARTIAL DIFFERENTIAL EQUATIONS*, volume 1, pages 187–202. SIAM.

Chan, T. F., Glowinski, R., Periaux, J., and Widlund, O. B. (1988). *Domain Decomposition Methods.* SIAM. ISBN 0-89871-233-5.

Chan, T. F. and Mathew, T. P. (1994). Domain decomposition algorithms. *Acta Numerica*, pages 61–143.

Chow, P. L., Ibragimov, I. A., and Khasminskii, R. Z. (1999). Statistical approach to some ill-posed problems for linear partial differential equations. *PROBABILITY THEORY AND RELATED FIELDS*, 113:421–441.

Colton, D., Ewing, R., and Rundell, W. (1990). *Inverse Problems in Partial Differential Equations.* SIAM. ISBN 0898712521.

Demirdžić, I. and Martinović, D. (1993). Finite volume method for thermo-elasto-plastic stress analysis. *Computer Methods in Apllied Mechanics and Engineering*, 109:331–349.

Dongarra, J. J. and Dunigan, T. (1997). Message-passing performance of various computers. Technical Report UT-CS-95-299, Oak Ridge National Laboratory. Postcript file available via http://www.netlib.org/utk/papers/commperf.ps.

D'Souza, N. (1975). Numerical solution of one-dimensional inverse transient heat conduction by finite difference method. *A.S.M.E.*, Paper No. 68-WA/HT-81.

Faringer, T. (1997). Estimating cache performance for sequential and data parallel programs. *Proc. of HPCN'97, Springer Lecture Notes in Computer Science, Vienna, Austria.*

Forum, M. P. I. (1995). *MPI: A Message-Passing Interface Standard.* available from http://www.mcs.anl.gov/mpi/.

Foster, I. (1995). *Designing and Building Parallel Programs.* Addison-Wesley, ISBN 0-201-57594-9 , Also available electronically, url is http://www-unix.mcs.anl.gov/dbpp/.

Frank, I. (1963). An application of least squares method to the solution of the inverse problem of heat conduction. *Heat Transfer*, 85C:378–379.

Frommer, A. and Maass, P. (1999). Fast cg-based methods for Tikhonov-Phillips regularization. *SIAM J. SCI. COMPUT.*, 20(5):1831–1850.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V. (1994). *PVM: A Users' Guide and Tutorial for Networked Parallel Computing.* MIT Press, Also available electronically, url is http://www.netlib.org/pvm3/book/pvm-book.html.

Ierotheou, C., Lai, C.-H., Palansuriya, C., and Pericleous, K. (1998). Simulation of 2-d metal cutting by means of a distributed algorithm. *The Computer Journal*, 41:57–63.

Imber, M. and Khan, J. (1972). Prediction of transient temperature distributions with embedded thermocouples. *AIAA J.*, 10:784–789.

Jones, M. T. and Plassmann, P. E. (1995). BlockSolve95 users manual: Scalable library software for the parallel solution of sparse linear systems. Technical Report ANL-95/48, Mathematics and Computer Science Division, ARGONNE NATIONAL LABORATORY, 9700 South Cass Avenue, Argonne, IL 60439.

Keyes, D. E. (1998). Trends in algorithms for nonuniform applications on hierarchical distributed architectures. In Salas, M. and Anderson, W., editors, *Workshop on Computational Aerosciences for the 21st Century.* Elsvier.

Keyes, D. E., Chan, T. F., Meurant, G., Scroggs, J. S., and Voigt, R. G. (1991). *Domain Decomposition Methods for Partial Differential Equations.* siam. ISBN 0-89871-288-2.

Keyes, D. E., Kaushik, D. K., and Smith, B. F. (1997). Prospects for CFD on petaflops systems. CFD Review, M. Hafez, et. al., eds., Wiley, available from http://www.cs.odu.edu/ keyes/papers.html.

Keyes, D. E., Saad, Y., and Truhlar, D. G. (1995). *Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering*. siam.

Kim, T. G. and Lee, Z. H. (1997). Time-varying heat transfer coefficients between tube-shape casting and metal mold. *Int. J. Heat Mass Transfer*, 49:3513–3525.

Kron, G. (1963). *Diakoptics: The Piecewise Solution of Large-Scale Systems*. MacDonald & Co, London.

Krutz, G. W., Schoenhals, R. J., and Hore, P. S. (1978). Application of finite element method to the inverse heat conduction problem. *Num. Heat Transfer*, 1:489–498.

Krzysztof, G., Cialkowski, M. C., and Kaminski, H. (1981). An inverse temperature field problem of the theory of thermal stresses. *Nucl. Eng. Des.*, 64:169–184.

Kunisch, K. and Tai, X.-C. (1996). Some non-overlapping domain decomposition methods for inverse problems. Ullensvang. To appear in the proc. of the 9th international conference on domain decomposition methods.

Lai, C.-H. (1994). Diakoptics, domain decomposition and parallel computing. *The Computer Journal*, 37:840–846.

Lai, C.-H., Cuffe, A., and Pericleous, K. (1998). A defect equation approach for the coupling of subdomains in domain decomposition methods. *Computers Math. Applic.*, 35:81–94.

Lai, C.-H., Ierotheou, C., Palansuriya, C., Pericleous, K., Espedal, M., and Tai, X.-C. (1999). Accuracy of a domain decomposition method for the recovering of discontinuous heat sources in metal sheet cutting. *Computing and Visualization in Science*, 2:149–152.

Langford, D. (1967). New analytic solutions of the one-dimensional heat equation for temperature and heat flow rate both prescribed at the same fixed boundary (with applications to the phase change problem). *Q. Appl. Math.*, 24:315–322.

Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Q. Appl. Math.*, 2:164–168.

McColl, W. F. (1995). Scalable computing. *LNCS, Springer-Verlag*, 1000.

Myers, P. S., Uyehara, O. A., and Borman, G. L. (1967). Fundamentals of heat flow in welding. *Welding Research Council Bulletin*, 123.

Neumaier, A. (1998). Solving ill-conditioned and singular linear systems: a tutorial on regularization. *SIAM REV.*, 40(3):636–666.

Nupairoj, N. and Ni, L. M. (1994). Performance evaluation of some MPI implementations on workstation clusters. *Tech. Rep. MSU-CPS-ACS-94, Department of Computer Science, Michigan State University.*

Palansuriya, C. J., Lai, C.-H., Ierotheou, C. S., and Pericleous, K. A. (1998). A domain decomposition based algorithm for non-linear 2d inverse heat condition problems. In Mandel, J., Farhat, C., and Cai, X.-C., editors, *Domain Decomposition Methods 10*, volume 218, pages 515–522. American Mathematical Society.

Palansuriya, C. J., Lai, C.-H., Ierotheou, C. S., Pericleous, K. A., and Keyes, D. (1999). Comparison of three algorithms for nonlinear metal cutting problems. In Lai, C.-H., Bjørstad, P. E., Cross, M., and Widlund, O. B., editors, *Domain Decomposition Methods in Sciences and Engineering*, pages 318–325. Domain Decomposition Press.

Patankar, S. (1980). *Numerical Heat Transfer and Fluid Flow.* Hemisphere.

Patel, P. M., Lau, S. K., and Almond, D. P. (1992). A review of image analysis techniques applied in transient thermographic nondestructive testing. *Nondestructive Testing and Evaluation*, 6:343–364.

Preziosi, L. (1993). An inverse source-sink problem for the nonlinear heat equation. *Math. Comput. Modelling*, 17:3–11.

Quarteroni, A. (1995). Domain decomposition methods for wave propagation problems. In Keyes, D. E., Saad, Y., and Truhlar, D. G., editors, *Domain-Based parallelism and problem decomposition methods in computational science and engineering*, pages 21–38. siam.

Saad, Y., Lo, G.-C., and Kuznetsov, S. (1998). *PSPARSLIB users manual: A Portable Library of Parallel Sparse Iterative Solvers*. Department of Computer Science, University of Minnesota, Minneapolis, MN.

Schwarz, H. A. (1869). Über einige abbildungsaufgaben. *Gesammelte Mathematische Abhandlungen*, 11:65–83.

Sirotkin, V. V. (1997). The solution of singularly perturbed parabolic problems by parallel algorithms combining Crank-Nicholson scheme and overlapping schwarz methods. *IMA Journal of Numerical Analysis*, 14:1–28.

Smith, B. F., Bjørstad, P. E., and Gropp, W. D. (1995). *Domain decomposition and multilevel methods for elliptic PDEs: Algorithms, Implementations and theory*. Cambridge University Press.

Stolz, Jr., G. (1960). Numerical solutions to an inverse problem of heat conduction for simple shapes. *Heat Transfer*, 82:20–26.

Tai, X.-C., Frøyen, J., Espedal, M., and Chan, T. (1997). Overlapping domain decomposition and multigrid methods for inverse problems. Technical Report 113, Department of Applied Mathematics, University Of Bergen. ISSN 0084-778x.

Taylor, G. A., Hughes, M., Strusevic, N., and Pericleous, K. A. (1999). Finite-volume methods applied to the computational modelling of welding phenomena. In Schwarz, M., Davidson, M., Easton, A., Witt, P., and Sawley, M., editors, *2nd Int. Conf. on CFD in the Minerals and Process Industries*, pages 405–410. ISBN 0643065598.

Taylor, J. R. (1997). *An introduction to error analysis*. University Science Books, Sausalito, California, USA, ISBN 0-935702-75-X.

Trujillo, D. M. (1978). Application of dynamic programming to the general inverse problem. *Int. j. numer. methods eng.*, 12:613–624.

Versteeg, H. K. and Malalasekera, W. (1995). *An introduction to Computational Fluid Dynamics The Finite Volume Method*. Longman Scientific & Technical.

Zwillinger, D. (1989). *Handbook of Differential Equations*. Academic Press Inc., San Diego.