

STABLE, GLOBALLY NON-ITERATIVE, NON-OVERLAPPING
DOMAIN DECOMPOSITION METHODS FOR THE EFFICIENT
SOLUTION OF PARABOLIC EVOLUTIONARY SYSTEMS

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agriculture and Mechanical College
in partial fulfillment of the
requirement for the degree of
Doctor of Philosophy

in

The Department of Computer Science

By

Yu Zhuang

B.S. Zhejiang University, July 1990

M.S. Louisiana State University, December 1995

M.S. Louisiana State University, May 1998

Ph.D. Louisiana State University, August 2000

September 2000

Acknowledgements

I would like to thank Professor Donald Kraft, Professor Aiichiro Nakano, Professor Frank Neubrandner, Professor Xian-He Sun, Professor John Tyler, and Professor John Wefel for serving on my PhD committee and for the knowledge and wisdom they imparted to me when I was taking their courses. I would especially like to thank Professor Xian-He Sun and Professor Donald Kraft for serving as my committee chairs. I am also indebted to Professor Frank Neubrandner for serving as my minor advisor for this Ph.D. degree in computer science.

Table of Contents

Acknowledgements	ii
Abstract	v
1 Introduction	1
1.1 The Problem	2
1.2 Considerations in Algorithm Design	5
1.2.1 Stability	6
1.2.2 Overlapping or Non-overlapping	8
1.2.3 Globally Iterative or Non-iterative	9
1.3 Existing Globally Non-iterative Algorithms	11
1.3.1 An Overlapping Algorithm	11
1.3.2 Non-overlapping Domain Decomposition Algorithms	12
1.3.3 Explicit-implicit Time Marching Algorithms	15
1.4 Outline of the Dissertation	18
2 The Domain Decomposition Algorithms	20
2.1 A Generic Parallel Domain Decomposition Algorithm	20
2.2 The Stabilized Forward-backward Euler Algorithm	24
2.3 The Stabilized Dawson-Du-Dupont Algorithm	36
2.4 An Algorithm with Factorized Subdomain Scheme	39
3 High Order Elliptic Solvers	41
3.1 Introduction	41
3.2 High Order Discretizations for the Laplace Operator	44
3.2.1 Existing High Order Discretization schemes	44
3.2.2 Directionally Decomposed High Order Discretization	46
3.3 The Picard-FFT Solution Methods	49
3.3.1 A Picard Process	49
3.3.2 High Order FFT Based Solvers	51
3.3.3 Another Picard Process	52
3.3.4 Numerical Experiments	53
3.4 The ADI Method for Separable Problems	56
3.4.1 A High Order ADI Solver	57
3.4.2 Numerical Experiments	60
3.5 Multilevel Acceleration of Iterative Methods	63
3.5.1 The Multilevel Acceleration Algorithm	63
3.5.2 Efficiency Analysis	65
3.5.3 Numerical Experiments	69
4 Numerical Experiments	73
4.1 Stability Testing	73
4.1.1 The Heat Equation	73

4.1.2 An Unstable Diffusion Problem	76
4.1.3 Convection-Diffusion Problems	78
4.1.4 An Unstable Convection-Diffusion Problem	81
4.2 Scalability Testing	84
4.3 High Order Spatial Discretization	87
4.4 Concluding Remarks	89
5 Summary and Future Work	90
References	95
Vita	99

Abstract

Parabolic systems are governed by time dependent partial differential equations. To obtain a high simulation quality that captures important features of a parabolic system requires solving the governing equation to an adequately high accuracy, which necessitates a large sampling size in both the spatial and temporal dimensions of the simulated system, and hence a large amount of processing data and high computing cost. Domain decomposition is an effective method of divide-and-conquer paradigm that divides the problem domain into several subdomains, reducing the original problem into several smaller interdependent problems which can be solved in parallel.

In this dissertation, we propose a class of stabilized explicit-implicit time marching (SEITM) domain decomposition algorithms for parabolic equations. Explicit-implicit time marching (EITM) algorithms are globally non-iterative nonoverlapping domain decomposition methods, which, when compared with Schwartz algorithm based parabolic solvers, are both computationally and communicationally efficient for each time step simulation but suffer from small time step size restrictions due to conditional stability. The proposed stabilization techniques in the SEITM algorithms retain the time-stepwise efficiency in computation and communication of the EITM algorithms but free the algorithms from small time step size restrictions, rendering SEITM algorithms excellent candidates for large scale parallel simulation problems. Three algorithms of the SEITM class are presented in this dissertation, which are mathematically analyzed and experimentally tested to show excellent numerical stability, computation and communication efficiencies,

and high parallel speedup and scalability.

1 Introduction

Computer simulation has become a powerful means of understanding the structure and behavior of complex systems, including climate and weather forecasting, air and water pollution management, molecular design for new material and new drugs, biochemistry and bioengineering, aerodynamics optimization of cars and planes, design of engines, etc. With the advent of cheaper but more powerful machines, computers can perform realistic “experiments” with results comparable to “physical experiments”. Such realistic computer “experiments” usually involve the solution of equation systems on a scale large in both time and space. Thus algorithms for solving large scale problems on advanced machine architecture is increasingly in demand.

Many physical, chemical and biological systems studied in science and engineering evolve with time. We call such a system an evolutionary system. Thus evolutionary systems are governed by time dependent differential equations. To numerically simulate an evolutionary system, we need to solve the differential equation or equations that govern the system. Since these governing equations have both time and space variables, the amount of simulation data is usually very large due to the large number of sampling points in a temporal-spatial grid. When the spatial domain of the system is irregular it is more challenging to achieve efficient simulations. Domain decomposition [13, 31] is an effective method to solve such problems by dividing the spatial problem domain into several less irregular subdomains, thus reducing the original problem into several smaller interdependent problems on less irregular sub-domains which can be solved in parallel.

1.1 The Problem

In this dissertation we are interested in domain decomposition based parallel algorithms for the simulation of evolutionary systems governed by the initial boundary value problem of the parabolic equation

$$\begin{cases} \frac{\partial u(t,x)}{\partial t} = \sum_{i=1}^k \frac{\partial}{\partial x_i} \left(a_i(x) \frac{\partial u}{\partial x_i} + 2b_i(x)u \right) + c(x)u + f(t,x), & x \in \Omega, t \geq 0 \\ Bu(t,x) = u_b(t,x), & x \in \partial\Omega, t \geq 0, \\ u(0,x) = u^0(x), & x \in \Omega, \end{cases} \quad (1.1)$$

where the problem domain Ω is a compact subset in \mathbf{R}^k with $k=2$ or 3 , and the spatial variable $x = (x_i)_{i=1}^k$. B is a boundary operator with $B = I$ representing the Dirichlet boundary condition and $B = \frac{\partial u}{\partial n}$ representing the Neumann boundary condition, here n is the normal vector of the boundary $\partial\Omega$. The functions a_i and b_i are continuously differentiable, and c and f are continuous. For notational simplicity, we denote the spatial operator by A , namely

$$Au = \sum_{i=1}^k \frac{\partial}{\partial x_i} \left(a_i(x) \frac{\partial u}{\partial x_i} + 2b_i(x)u \right) + c(x)u. \quad (1.2)$$

Then the evolutionary system (1.1) can be concisely written as

$$\begin{cases} \frac{\partial}{\partial t}u(t,x) = Au(t,x) + f(t,x), & x \in \Omega, t \geq 0 \\ Bu(t,x) = u_b(t,x), & x \in \partial\Omega, t \geq 0, \\ u(0,x) = u^0(x), & x \in \Omega, \end{cases} \quad (1.3)$$

The system covers many evolutionary processes including heat transfer, chemical convection and diffusion, microbial transport [54]. When the system (1.3) represents a heat transfer process, the function $c(x)$ corresponds to heat source at point x if $c(x) > 0$ or sink if $c(x) < 0$. The operator A could be indefinite if $c(x) > 0$

at some points x . Here indefinite refers to the situation that A does not have ellipticity and has both positive and negative eigenvalues. When A is indefinite, the system (1.3) is unstable and could have solutions with exponential growth for some initial conditions. Here an evolutionary system is called *stable* [12] if the spectrum (or counterpart of eigenvalues of a matrix) of the spatial operator A is non-positive.

The purposes of this dissertation are to propose a class of efficient and scalable parallel domain decomposition algorithms, and to examine their properties under both mathematical rigor and practical experimentation. In order to discuss the mathematics behind the algorithmical design and experimental observation, we put the problem in the Hilbert space $L^2(\Omega)$ with $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$ denoting the L^2 norm and inner product respectively. With the choice of this Hilbert space, we first list some properties of the problem (1.3) which are mathematical representation of important physical properties of time dependent systems. These properties, later in the dissertation, will be utilized to design and analyze our numerical simulation algorithms.

Much of the knowledge of human being is gained through experiments. But people learn from experiments only when the experiments are *repeatable*, that is, if one repeats an experiment with all the conditions being the same, the experiment should produce the same result; and when the experiment conditions have a small change, the experiment outcome should also has only a small change. This gives rise to the concept of well-posedness for an evolutionary system. An evolutionary system is called *well-posed* if

- (i) for each initial value in the domain of the densely defined spatial operator A ,

there exists a solution;

- (ii) for each solution $u(t, x; u^0)$ with initial value u^0 , $u(t_2 + t_1, x; u^0) = u(t_2, x; u^1)$, where $u^1 = u(t_1, x, y; u^0)$.
- (iii) $u(t, x, y; u^0)$ continuously depends on u^0 , i.e. there exists an increasing positive function $M(t)$ such that $\|u(t)\| \leq M(t)\|u^0\|$.

When the system (1.3) is well-posed, the solution for initial condition $u^0(x)$ can be expressed as [26, 49] $u(t, x) = e^{tA}u^0(x)$, where e^{tA} is a bounded linear operator for each $t \geq 0$ and satisfies $e^{(t_2+t_1)A} = e^{t_2A}e^{t_1A}$ and $\|e^{tA}\| \leq M(t)$. It is known [46] that systems satisfying the *quasi-dissipative* condition are well-posed. A quasi-dissipative evolutionary system is such a system that its spatial operator A satisfies $\langle Av, v \rangle + \langle v, Av \rangle \leq 2\omega\|v\|^2$ for all functions $v \in D(A)$ with $v(x) = 0$ at boundary points $x \in \partial\Omega$ for some constant ω , where $D(A)$ denotes the domain of the operator A . When the quasi-dissipativity constant $\omega = 0$, the system is called *dissipative*. For example, when A is the Laplace operator $\sum_i \frac{\partial^2}{\partial x_i^2}$, A is negative definite and thus dissipative. Using the rule of integrating by parts, it is verifiable that the general parabolic evolutionary system (1.3) is quasi-dissipative, and the quasi-dissipativity constant ω is bounded by $\max\{c(x) + \sum_{i=1}^k \frac{\partial b_i(x)}{\partial x_i} : x \in \Omega\}$. Quasi-dissipative systems include a large class of evolutionary processes, e.g. diffusion, convection and wave propagation processes.

To numerically simulate the system, it is necessary to discretize the equation spatially on a discrete sampling grid Ω_h with spatial mesh size $h = (h_{x_i})_{i=1}^k$, result-

ing in a spatially discrete evolutionary system

$$\begin{cases} \frac{d}{dt}u_h(t, x) &= A_h u_h(t, x) + f(t, x), & x \in \Omega_h, t \geq 0, \\ B_h u_h(t, x) &= u_b, & x \in \partial\Omega_h, t \geq 0, \\ u_h(0, x) &= u^0(x), & x \in \Omega_h, \end{cases} \quad (1.4)$$

The definition of well-posedness and quasi-dissipativity are also valid for the discrete system (1.4). With standard second order finite difference discretization schemes for the elliptic operator A (see [3]), the discrete evolutionary system (1.4) retains the quasi-dissipativity condition

$$\langle A_h v, v \rangle + \langle v, A_h v \rangle < 2\omega \|v\|^2 \text{ for all } v \in L^2(\Omega_h) \quad (1.5)$$

for all spatial mesh size h , where $\omega = \max\{c(x) + \sum_{i=1}^k \frac{\partial b_i}{\partial x_i} : x \in \Omega\}$. Here the L^2 space on the discrete domain Ω_h is the finite dimensional space \mathbf{R}^N with N being the number of grid points in Ω_h , and the inner product $\langle \cdot, \cdot \rangle$ in $L^2(\Omega_h)$ is defined as $\langle u, v \rangle = u^t v$ for vectors $u, v \in \mathbf{R}^N$.

1.2 Considerations in Algorithm Design

The final purpose of all numerical simulations is to arrive at an approximate numerical solution within a given error tolerance with the lowest possible computing time. For a parallel simulation algorithm running on multi-processor, distributed memory machines with each processor having its own local memory, the total computing time comes from two types of operations: computation operations — arithmetic operations of floating point numbers, and communication operations that cause the transfer of data among the local memory of different processors. In this section,

we list some algorithmic features that affect the computation and communication cost of domain decomposition based parallel algorithms.

1.2.1 Stability

In addition to spatial discretization, equation (1.4) also needs to be temporally discretized. To discretize (1.4) in time, we can use explicit schemes like the first order forward Euler scheme

$$u_h^{n+1} = (I + \Delta t A_h)u_h^n + O((\Delta t)^2)$$

to compute an approximate solution u_h^{n+1} at time step $n + 1$ from the already computed approximate solution u_h^n at time step n . The forward Euler scheme requires only a multiplication of the matrix $(I + \Delta t A_h)$ and the vector u_h^n . Thus the forward Euler scheme is computationally efficient for each time step calculation, and it is also efficiently parallelizable due to the sparsity of the matrix A_h . But it has a serious drawback — it is not unconditionally stable and requires a prohibitively small time step size of $\Delta t = \frac{h^2}{6}$ [15, 40] for the scheme to be stable for the simulation of the 3-D heat equation. This obviously necessitates a large number of simulation time steps. Like the forward Euler scheme, existing explicit schemes all are not unconditionally stable to the best of our knowledge, and have severe time step size restriction. Thus high temporal order explicit scheme have no advantage over the first order forward Euler scheme in the reduction of simulation time steps due to this time step size restriction.

To have freedom from time step size restrictions, implicit schemes like the first

order backward Euler scheme

$$(I - \Delta t A_h)u_h^{n+1} = u_h + O((\Delta t)^2) \quad (1.6)$$

or second order Crank-Nicolson scheme

$$(I - \frac{\Delta t}{2}A_h)u_h^{n+1} = (I + \frac{\Delta t}{2}A_h)u_h^n + O((\Delta t)^3) \quad (1.7)$$

can be used to approximate the spatially discretized problem (1.4) temporally. Both the back Euler scheme and the Crank-Nicolson scheme require solving an elliptic equation of the form

$$(I - \tau A_h)u_h = r. \quad (1.8)$$

It is much more computationally expensive to solve the above equation than to perform the matrix-vector multiplication in the forward Euler scheme. And the parallelization of an solver for equation (1.8) is also much more communicationally costly than to parallelize a matrix-vector product.

The Schwartz alternating algorithms [8, 9, 10, 13, 22, 31, 32, 56, 57] are globally iterative domain decomposition procedures for solving elliptic problems with domain decomposition based matrix splittings to enhance parallelism and localized treatment of irregular geometries. Here the term “globally” refers to the part of a solution process that is carried over the entire problem domain as opposed to solution processes for subdomain problems which could be either iterative or direct. Combined with multigrid technique using two grid levels, the Schwartz algorithms can achieve a good global convergence rate independent of spatial mesh size [8]. For parabolic evolutionary systems, after implicit temporal discretization,

a sequence of elliptic equations of the form (1.8) need to be solved, so the Schwartz algorithms can be used in the parallel simulation of parabolic systems by solving the elliptic equation (1.8) in parallel. Cai [6, 7] used the Schwartz in this manner to solve parabolic problems. Since the Schwartz algorithms are solvers for the elliptic equation (1.8) resulted from temporal discretization, it has an advantage in preserving the unconditional stability of implicit temporal discretizations as long as the Schwartz solver iterates until the solution error becomes small enough to have no influence on the stability of the temporal schemes.

1.2.2 Overlapping or Non-overlapping

The efficiency of a parallel algorithms depends on its computation and communication efficiencies. The computation efficiency is determined by the floating points operations the algorithm executes for a problem of given size, and the communication efficiency, in addition to machine dependent factors like data transferring rate and transmission startup overhead, depends on the number of data transmission operations and the amount of data transferred in each transmission operation. But since communication is much more time consuming per byte than computation, it is important for parallel algorithms to keep a calculated balance between computation and communication.

Schwartz algorithms can be classified into overlapping and non-overlapping depending on whether the subdomains overlap. If different subdomains are assigned to different processors, obviously overlapping algorithms have much larger communication cost for each iteration since the data on the overlap area need to be trans-

mitted between the processors assigned to the subdomains that cover the overlap area. However, if an overlapping algorithm has much “faster” convergence and requires many fewer iterations for solving equation (1.8) when compared with a non-overlapping algorithm, the high per-iteration communication cost may be compensated by the fast convergence. But how many iterations an overlapping algorithm needs to reduce in order to make itself a competitive one against non-overlapping algorithms is not very easy to determine. It depends on the size of overlap area, the nature of the problem that affects the convergence rate, and the machine on which the code of the algorithm is running. For evolutionary systems, only considering the convergence rate of an iterative solver for the equation (1.8) is not enough since there is another convergence that affects the overall computation and communication cost considerably — the convergence with respect to the temporal approximation. This makes the decision of choosing overlapping or non-overlapping, or the optimal (or near-optimal) degree of overlapping a difficult and programmably complicated process.

1.2.3 Globally Iterative or Non-iterative

One solution to this decision problem of choosing an overlapping size is to fix the number of iterations at each time step and then determine an appropriate size of overlapping. An example of this approach is Kuznetsov’s one-iteration overlapping Schwartz algorithm [34] for the solution of the elliptic equation (1.8) obtained from an implicit temporal discretization of parabolic problems. Kuznetsov’s elliptic solver has a good stability condition, but requires an overlap size of $O(\sqrt{\Delta t} \log \epsilon)$

for an local error tolerance of $O(\epsilon)$. With an $O(h^2)$ spatial discretization, the Crank-Nicolson scheme (1.7) has a local truncation error of $O(\Delta t h^2 + \Delta t^3)$. Thus, Kuznetsov's algorithm requires an overlap size of $O(\sqrt{h} \log(h^3))$ to reach a local accuracy of $O(h^3)$ with $\Delta t = h$.

Similar to the decision of choosing a degree of overlapping, the choice between global iterative and non-iterative algorithms also requires careful weighing of their respective advantages and disadvantages. And these advantages and disadvantages must be considered together with other concerns like overlapping and stability. For general parabolic evolutionary equations, the temporally discretized equation (1.8) is usually not separable and efficient non-iterative elliptic solvers like the FFT based direct method [29] are not applicable. For non-separable elliptic equations, iterative methods usually are more efficient than direct methods to reach an accuracy of the same order as the truncation error. However, global iterative algorithms incur repeated data transmission among processors.

For parabolic evolution problems, the solutions of the elliptic equation (1.8) are temporally related by the temporally non-discretized equation (1.4). Thus good initial guess of the solution of (1.8) can be constructed using efficient predicting procedures like explicit schemes. If we can reduce some, if not all, of the instability causing factors in the explicit schemes with one or two iterations, we might obtain an algorithm that is time-stepwisely more efficient than the implicit schemes but numerically more stable than the explicit schemes. Such considerations are the starting points of our algorithm design strategy.

1.3 Existing Globally Non-iterative Algorithms

In the section, we briefly describe some existing globally non-iterative (or one-iteration) domain decomposition algorithms.

1.3.1 An Overlapping Algorithm

In 1998, Mathew, Polyakov, Russo and Wang [47] gave an overlapping algorithm without the large overlap size requirement as in Kuznetsov's algorithm [34]. This algorithm of Mathew et al. uses the decomposed domain to construct a partition of unity which consists of non-negative smooth functions with each subdomain being associated with a member function. Each member function of the partition of unity is supported on and vanishes outside the subdomain it is associated with. Then a splitting of the spatial operator A into

$$A_h = A_1 + A_2 + \cdots + A_p \tag{1.9}$$

is constructed using the member functions that make up the partition of unity. After splitting of the operator, they temporally discretize the equation using Douglas' multi-dimensional ADI method [19]

$$u^{n+1} = (I - \frac{\Delta t}{2} A_1)^{-1} \cdots (I - \frac{\Delta t}{2} A_p)^{-1} (I + \frac{\Delta t}{2} A_p) \cdots (I + \frac{\Delta t}{2} A_1) u^n, \tag{1.10}$$

with the directional components of the operator A in Douglas' original ADI method replaced by the components in the operator splitting (1.9). When compared with Kuznetsov's algorithm, the algorithm of Mathew et al. is not subject to a large overlap size requirement because they used the partition of unit to form an exact splitting of A while Kuznetsov's algorithm uses an operator splitting which is only

approximate in the sense that

$$A_h \approx A_1 + A_2 + \cdots + A_p.$$

Kuznetsov utilized the property that the entries in the matrix $(I - tA_h)^{-1}$ decay rapidly when the positions of these entries are far away from the diagonal, so for entries far away from the diagonal are in fact dropped from $(I - tA_h)^{-1}$ with Kuznetsov's splitting.

Mathew, Polyakov, Russo and Wang's construction of the operator splitting through smooth, compactly supported member functions of the partition of unity has another advantage, that is the elimination of the requirement of predicting interface boundary conditions for the inverting operations in the ADI solving (1.10). This is due to the property that each member function of the partition of unity vanishes at the boundary of the subdomain with which it is associated. Since ADI has a second order temporal accuracy, this domain decomposition algorithm has an $O((\Delta t)^3)$ local temporal accuracy. However, just like the original multi-dimensional ADI method, it loses unconditional stability when the operator is split into more than two non-commutative components. Thus the stability suffers when the number of subdomains is large.

1.3.2 Non-overlapping Domain Decomposition Algorithms

Since non-overlapping algorithms have low communication cost for each time step, they are appealing for large problems on massively parallel machines. In the following, we list some non-overlapping domain decomposition algorithms that are not of the type of explicit-implicit time marching (EITM). The EITM algorithms

will be briefly described in the section that follows.

In 1991, Dryja [21] proposed a one-iteration finite element solver for the equation

$$(I - \frac{\Delta t}{2}A_h)u_h^{n+1} = (I + \frac{\Delta t}{2}A_h)u_h^n \quad (1.11)$$

resulted from the Crank-Nicolson temporal discretization. In this algorithm, Dryja uses a red-black ordering of the subdomains and divide the subdomains into two groups Ω_1 and Ω_2 . The equation (1.11) is then solved alternately on the two groups of the subdomains as in the following.

$$\begin{cases} \frac{u_h^{n+1/2} - u_h^n}{\Delta t} = A_h \frac{u_h^{n+1/2} + u_h^n}{2}, & \text{on } \Omega_1, \\ u_h^{n+1/2} = u_h^n, & \text{on } \Omega_2, \end{cases} \quad (1.12)$$

$$\begin{cases} \frac{u_h^{n+1} - u_h^{n+1/2}}{\Delta t} = A_h \frac{u_h^{n+1} + u_h^{n+1/2}}{2}, & \text{on } \Omega_2, \\ u_h^{n+1} = u_h^{n+1/2}, & \text{on } \Omega_1. \end{cases} \quad (1.13)$$

Thus from a temporal discretization point of view, Dryja's solver is a fractional step method. When viewed as an elliptic solver, it is a one-iteration multiplicative Schwartz algorithm. From (1.12) and (1.13) it can be easily derived that the algorithm is representable as

$$u_h^{n+1} = (I - \frac{\Delta t}{2}A_2)^{-1}(I + \frac{\Delta t}{2}A_2)(I - \frac{\Delta t}{2}A_1)^{-1}(I + \frac{\Delta t}{2}A_1)u_h^n,$$

where A_1 and A_2 are the restrictions of the matrix A_h on Ω_1 and Ω_2 respectively.

With the above representation, it is immediately seen that this algorithm is unconditionally stable since for each $i = 1, 2$,

$$\|(I - \frac{\Delta t}{2}A_i)^{-1}(I + \frac{\Delta t}{2}A_i)\| \leq e^{\omega \Delta t}$$

for some norm $\|\cdot\|$. However, as Dryja himself pointed out, this algorithm has a large global error of order $O(\sqrt{\Delta t} + h)$ when Δt is proportional to h .

An algorithm with better global error was proposed by Laevsky [35] for the entire domain decomposed into two subdomains Ω_1 and Ω_2 . His algorithm uses the Galerkin method for the elliptic ADI-solver of the Crank-Nicolson scheme

$$\begin{cases} \frac{u_h^{n+1/2} - u_h^n}{\Delta t/2} &= A_1 u_h^{n+1/2} + A_2 u_h^n, \\ \frac{u_h^{n+1} - u_h^{n+1/2}}{\Delta t/2} &= A_1 u_h^{n+1/2} + A_2 u_h^{n+1}, \end{cases} \quad (1.14)$$

where A_1 and A_2 are the restrictions of the matrix A_h on Ω_1 and Ω_2 respectively. The global error of this algorithm is of order $O(h + (\Delta t)^2 + \rho(1 + \rho)^{1/2})$ with $\rho = (\Delta t)^2/h$. However the term $\rho(1 + \rho)^{1/2}$ has placed a restriction on the ratio of the Δt and h .

The ADI scheme can be shown to be conditionally stable. The unconditional stability of ADI scheme does not hold in general (see [67]). However the ADI scheme with two splitting components is proven to satisfy the unconditional Von Neumann stability condition and also proven to be unconditionally convergent [67]. However when the operator is split into more than two non-commutative components, the ADI scheme lose unconditional Von Neumann stability (also mentioned in [37] for the unstability) and requires the time step size to be of $\Delta t = O(h^2)$ for the multi-component ADI scheme to converge. Thus when the entire domain is decomposed into more than two subregions, Laevsky's algorithm (1.14) not only shall lose its good stability condition, the term $\rho(1 + \rho)^{1/2}$ will enlarge its global error to order $O(1)$ as well.

Another algorithm of Laevsky [38] that is unconditionally convergent is a mod-

ification of the aforementioned Dryja's algorithm with a boundary treatment. This algorithm has a global error of $O(\Delta t + h)$.

1.3.3 Explicit-implicit Time Marching Algorithms

One reason for the low accuracy of the non-overlapping algorithms mentioned in the previous section is due to the lack of an adequately accurate boundary condition at the subdomain boundary points which are inside the entire problem domain. For instance, in Dryja's algorithm (1.12)-(1.13) and in Laevsky's algorithm (1.14), both the intermediate solution $u_h^{n+1/2}$ and the solution u_h^{n+1} have no available information for the interior boundary condition. An obvious solution is to use u_h^n to provide the interior boundary condition for $u_h^{n+1/2}$ and $u_h^{n+1/2}$ for u_h^{n+1} . This obviously introduces errors into the simulation process.

Explicit-implicit algorithms have solved the problem of the availability of interior boundary conditions. In 1988, Kuznetsov [34] proposed an explicit-implicit scheme using a nonoverlapping domain decomposition. The boundary value of u_h^{n+1} on the interior boundary of the subdomains is first predicted using an explicit method. Then stable implicit temporal discretization scheme can be applied to the equation on the subdomains and the resulted elliptic equation on each subdomain can be solved independently using the predicted interface boundary conditions together with the exterior boundary conditions. When the forward Euler scheme is used as the interface boundary condition predictor and the backward Euler scheme is used for the temporal discretization of the equation (1.3) on the subdomains, the

explicit-implicit method can be represented as

$$u_h^{n+1} = (I - \Delta t A_2)^{-1} (I + \Delta t A_1) u_h^n, \quad (1.15)$$

where A_1 denotes the restriction of A_h on the interface boundary and A_2 the restriction of A_h on the complement of interface boundary in the entire domain. However, the explicit predictor of the interface boundary condition causes numerical instability unless the time step size is restricted to $\Delta t = O(h^2)$.

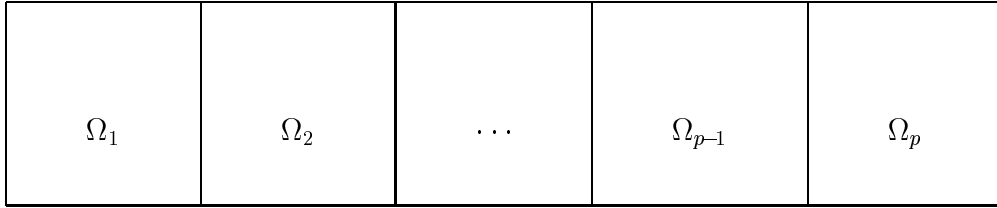


Figure 1

Dawson, Du and Dupont [18] proposed an alternative explicit-implicit algorithm in which the interface boundary condition predictor $(I + \Delta t A_1)$ in (1.15) is factorized into

$$(I + t A_1) = (I - t A_1^y)^{-1} (I + t A_1^x)$$

for a domain decomposed as in Figure 1, where A_1^x denote the x-directional differentiation component of the operator A_1 , and A_1^y the y-directional differentiation component. After the prediction of interface boundary conditions, the implicit backward Euler temporal scheme $(I - \Delta t A_h) u_h^{n+1} = u_h^n$ is applied to the system (1.3) on the subdomains, which can be solved independently on each subdomain.

Thus, Dawson, DU and Dupont's algorithm can be expressed as

$$u_h^{n+1} = (I - \Delta t A_2)^{-1} (I - \Delta t A_1^y)^{-1} (I + \Delta t A_1^x) u_h^n \quad (1.16)$$

Using a wider spatial mesh $H > h$ in the x-direction for the predictor, the time step size is restricted to $\Delta t = O(H^2)$ instead of $\Delta t = O(h^2)$. Subject to the restriction $\Delta t \leq H^2/2$ for the 2-D heat equation, the global error of their algorithm is of $O(h^2 + \Delta t + \Delta t H + H^3)$. Thus taking $H = \frac{h^{2/3}}{2}$, the time step restriction is only $\Delta t = \frac{h^{4/3}}{8}$ for the algorithm (1.16) but with a good global error of $O(h^2 + \Delta t + h^{2/3} \Delta t)$.

A penalized explicit-implicit algorithm proposed by Black [1] has remedied the stability related time step size restriction of the EITM algorithms, and achieved numerically verified unconditional stability. In her algorithm, Black [1] employed a Du Fort-Frankel scheme as the explicit predictor. The Du Fort-Frankel scheme contains a penalty term, which, in a fashion similar to what Funaro did in [24], penalizes the unsmoothness of the solution across the interface boundary. This penalized Du Fort-Frankel scheme achieves good numerical unconditional stability, however it introduces an error term of $O(\frac{\Delta t}{h})^2$ for the heat equation, making the algorithm inconsistent unless $\Delta t/h \rightarrow 0$. Thus consistency comes only after paying a price of restricting time step size to an order of $O(h^{3/2})$ to achieve a first order temporal accuracy, a restriction quantitatively similar to, though qualitatively different from, the restriction on the algorithm of Dawson, Du and Dupont. Though not explicitly given in her paper, Black's algorithm is supposed to have a global error of $O\left(h^2 + \Delta t + \left(\frac{\Delta t}{h}\right)^2\right)$.

From all publications known to the author, Dawson, Du and Dupont's algorithm and Black's algorithm have the highest accuracy among existing globally non-iterative, non-overlapping domain decomposition algorithms for parabolic problems. There are other papers by Laevsky and his colleagues [36, 37] on explicit-

implicit algorithms, but unconditional stability is not attained and the global errors are $O(h + \Delta t)$ or even larger.

1.4 Outline of the Dissertation

To eliminate the stability or consistency related time step size restriction of the EITM algorithms, we propose a class of stabilized explicit-implicit time marching (SEITM) domain decomposition algorithms. Together with a generic parallel SEITM algorithm with choices of the predictor, subdomain scheme and the stabilizer open to the algorithm implementors and users, in Chapter 2 we present three specific algorithms of this SEITM class as examples. One (denoted SEITM1) is the stabilization of the algorithm (1.15) with the backward Euler scheme as the stabilizer. Another (SEITM2) is the stabilization of Dawson, Du and Dupont's algorithm (1.16) by a stabilizer designed for the predictor. The third (SEITM3) differs from SEITM1 in that the subdomain temporal scheme is an approximate directional factorization of the backward Euler, which retains the same temporal accuracy but has reduced the computation complexity of the subdomain solver to $O(N)$ at each time step for a spatial domain of N grid points. This linear complexity of the SEITM3 algorithm also holds for non-selfadjoint problems whose spatially discretized matrix A_h is nonsymmetric. In Chapter 2 also have shown that the stabilization technique proposed in this dissertation not only unconditionally stabilizes the EITM algorithms and frees the EITM methods from time step size restrictions, but also does not affect the accuracy of the spatial discretization. More importantly for parallel computing, the stabilization has *zero* communica-

tion cost and very low computation cost, yielding excellent parallel speedup and scalability confirmed by testings (in Chapter 4) on SGI Origin 2000 computers.

Two of the algorithms (SEITM1 and SEITM2) require solving elliptic equations of the form $(I - \tau A_h)u = r$ on the subdomains. Since the proposed stabilization does not have an adverse effect on the spatial discretization accuracy, in order to be able to experimentally verify the preservation of the spatial accuracy of the SEITM algorithm, in Chapter 3 we develop several high order solvers for the generalized Helmholtz equation $-\Delta u + f(x, y)u = r$, which arises when the evolutionary system (1.1) has a heat source/sink term $c(x)$.

In Chapter 4 we carry out numerical experiments of the three SEITM algorithms on several testing problems, including several convection-diffusion problems and an unstable convection-diffusion problems with an indefinite spatial operator. We also have combined a high order elliptic solver presented in Chapter 3 with the SEITM1 algorithm, and applied the spatially high order SEITM1 algorithm to a testing problem. Experimental data are used to examine the stability, parallel speedup and efficiency of the SEITM algorithms, as well as the effectiveness of the spatial accuracy preservation. Summarizing remarks of the dissertation are given in Chapter 4.

2 The Domain Decomposition Algorithms

2.1 A Generic Parallel Domain Decomposition Algorithm

The entire domain Ω is divided into p subdomains $\Omega_1, \Omega_2, \dots, \Omega_p$ (e.g. as in figure below) with interface boundaries denoted by Γ . The complement of the interface boundary is the subdomains whose union is denoted by Γ^c , namely, $\Gamma^c = \Omega_1 \cup \Omega_2 \cup \dots \cup \Omega_p$, and then $\Omega = \Gamma \cup \Gamma^c$. With this decomposition of the original non-discrete

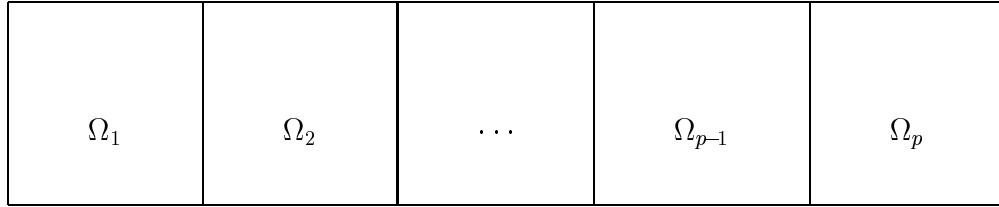


Figure 2

domain Ω , we define the partitioning of the discrete domain Ω_h simply by inheriting the partitioning of the original domain:

$$\begin{cases} \Omega_{h,i} = \Omega_h \cap \Omega_i & \text{for } i = 1, 2, \dots, p, \\ \Gamma_h = \Omega_h \cap \Gamma, \\ \Gamma_h^c = \Omega_h \cap \Gamma^c. \end{cases}$$

We denote the interface boundary between subdomains Ω_i and Ω_j by $\Gamma_{i,j}$ for $i < j$ ($\Gamma_{i,j}$ could be an empty set), and denote the i -th processor by p_i . Now a generic parallel domain decomposition algorithm (SEITM) for computing the solution u_h^{n+1} at the $(n+1)$ -th time step from the current n -th time step is given below.

The SEITM Algorithm

0. Assign subdomain Ω_i and interface boundary $\Gamma_{i,j}$ to p_i .

1. Compute u_h^{n+1} at Γ_h using an explicit scheme. Then pass from p_i to p_j the newly predicted u_h^{n+1} on $\Gamma_{i,j}$.

These computed data provide the interface boundary conditions.

2. Compute u_h^{n+1} on the subdomains Γ_h^c using any unconditionally stable scheme with the interface boundary conditions computed at step 1 as boundary conditions. Then pass part of the just computed data of u^{n+1} on the subdomain from p_j to p_i for the stabilization operation at the next step.

Using any unconditionally stable temporal scheme results in an elliptic equation to be solved. The solution of the elliptic equation can be carried out mutually independently on the subdomains and thus in parallel.

3. Throw away the interface boundary condition computed at step 1, and bring back u^n on Γ_h . Then implicitly re-compute u_h^{n+1} on Γ_h , using solution data u_h^{n+1} on Γ_h^c nearby as boundary conditions.

Go back to step 1 for the next time step iteration.

In step 1 of the algorithm, we do not pass any part of u^n from p_j to p_i before the explicit computation of the interface boundary condition (IBC) u^{n+1} . The computation of the IBC does need those data. However since processor p_i already received them from p_j at step 2 of the previous time step and no update has been performed on the data on the subdomains, no data transfer is necessary.

On the other hand, for any explicit-implicit domain decomposition method the two data transfer operations in the first two steps are necessary for predicting the interface boundary condition and for implicitly computing the solution on

the neighboring subdomain assigned to another processor. Thus compared with explicit-implicit domain decomposition algorithms, the SEITM algorithm has no extra communication cost as the following theorem states.

Theorem 2.1 *The stabilization (step 3) employed in SEITM algorithm to stabilize the explicitly predicted interface boundary condition is communicationally overhead free.*

Furthermore, using any method (including the explicit forward Euler temporal discretization which requires only one matrix-vector multiplication for each time step), it needs solution data from nearby to compute the solution on the interface boundary. This incurs one data transferring operation. And to compute the solution near the interface boundary also incurs one data transferring operation. Thus time-stepwisely, the minimal number of data transferring operations is two for *any* parallel algorithm using *any* temporal discretization scheme. Therefore, we have arrived at the following conclusion concerning the communication cost of the parallel SEITM methods.

Theorem 2.2 *The parallel SEITM algorithm is optimal in terms of number of data transferring operations for each time step.*

When each of the two data transferring operations are carried out by $p-1$ processors simultaneously with almost equal load,¹ the total communication time

¹This can be easily achieved when the domain is decomposed as in Figures 2 or 3, where the interface boundaries do not cross into each other. We leave the interface boundary treatment for more general domain partitioning strategies to a future project [68] and concentrate on the discussion of the proposed stabilization technique for EITM algorithms in this dissertation.

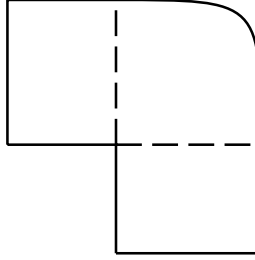


Figure 3

of each time step satisfies

$$T_{\text{comm}} \leq 2\alpha \frac{N_{\Gamma}}{p-1} + \beta, \quad (2.1)$$

where α is some system dependent data transfer rate, β is the communication startup overhead, and N_{Γ} denotes the number of grid points on the interface boundaries.

The computation cost of the stabilization process of the SEITM algorithm is proportional to the number of grid points N_{Γ} on the interface boundary Γ_h . But since N_{Γ} is much smaller than the total number of grid points (denoted by N), the computation overhead of the stabilization is very small and negligible. Assume the function of computation cost bound for the predictor, the subdomain solver and the stabilizer are the same and denoted by ϕ . Then the total computation cost at each time step is the sum of the prediction time $\phi\left(\frac{N_{\Gamma}}{p-1}\right)$, the subdomain solver time $\phi\left(\frac{N-N_{\Gamma}}{p}\right)$ and the stabilization time $\phi\left(\frac{N_{\Gamma}}{p-1}\right)$, yielding

$$T_{\text{comp}} = 2\phi\left(\frac{N_{\Gamma}}{p-1}\right) + \phi\left(\frac{N-N_{\Gamma}}{p}\right). \quad (2.2)$$

Cost function usually increases faster than linear or at least linearly, which means

$$\begin{cases} \phi(n) \geq p\phi\left(\frac{n}{p}\right), \\ \phi(n) \geq \phi(n-n') + f(n'), \end{cases} \quad (2.3)$$

for $n' \leq n$. Then the parallel speedup, defined as single processor execution time T_1 over parallel execution time T_p , can be estimated from (2.1), (2.2) and (2.3) as follows.

$$\begin{aligned}
S_p &= T_1/T_p = T_1/(T_{\text{comp}} + T_{\text{comm}}) \\
&= \frac{\phi(N)}{2\phi(\frac{N_\Gamma}{p-1}) + \phi(\frac{N-N_\Gamma}{p}) + 2\alpha\frac{N_\Gamma}{p-1} + \beta} \\
&\geq \frac{\phi(N)}{\frac{2\phi(N_\Gamma)}{p-1} + \frac{\phi(N)-\phi(N_\Gamma)}{p} + 2\alpha\frac{N_\Gamma}{p-1} + \beta} \\
&\approx \frac{p\phi(N)}{\phi(N) + \frac{p}{p-1}[2\alpha N_\Gamma + \beta + \phi(N_\Gamma)]}.
\end{aligned}$$

And the corresponding efficiency, defined as speedup over number of processors, is

$$E_p = \frac{\phi(N)}{\phi(N) + \frac{p}{p-1}[2\alpha N_\Gamma + \beta + \phi(N_\Gamma)]}.$$

As analyzed above, the SEITM methods are computationally and communicationally efficient for *each time step*. If they further has good stability to free themselves from excessively small time step size restriction, the time-stepwisely efficient SEITM will possess great potential for large simulation problems on distributed memory architecture machines. In next three sections, we shall present three specific SEITM algorithms where the stabilizers stabilize the respective interface boundary condition predictors.

2.2 The Stabilized Forward-backward Euler Algorithm

The SEITM algorithm given in Section 2.1 is generic and allows many choices for the explicit predictor in step 1, the subdomain scheme in step 2, and the stabilizer in step 3. Starting from this section, we shall present three SEITM algorithms

with different choices for the predictor, the subdomain temporal scheme and the stabilizer.

Before we present the algorithms, we would like to make precise the meaning of the notations to be used in the algorithms. Let $B_h = \Omega_h \cap \partial\Omega$ to denote the set of exterior boundary points. Let the solution u_h^n at time step n be a vector in $L^2(\Omega_h)$ which vanishes on B_h , where $L^2(\Omega_h)$ is defined at the end of Section 1.1. We denote by u_b^n and f^n the exterior boundary condition and the term $f(t, x)$ in (1.3) at time step n , with u_b^n being a vector in $L^2(\Omega_h)$ supported on B_h and $f^n \in L^2(\Omega_h)$. The square matrix A_h is the discrete spatial operator on $L^2(\Omega_h)$ which keeps the entries corresponding to boundary grid points invariant, that is $\chi_{B_h} A_h u_b^n = \chi_{B_h} u_b^n$, where χ_S is a diagonal matrix with 1 on the positions corresponding to the grid points in the subset $S \subset \Omega_h$ and 0 elsewhere. We use I denotes the identity matrix on $L^2(\Omega_h)$. With these notations we are ready to present, in this section, the simplest SEITM algorithm together with rigorous stability, accuracy and convergence analysis.

We choose the forward Euler scheme for the first step of the SEITM algorithm. Then the interface boundary condition predictor is mathematically representable as

$$\begin{cases} \chi_{\Gamma_h} u_h^{n+1/3} &= \chi_{\Gamma_h} [(I + \Delta t A_h)(u_h^n + u_b^n) + \Delta t f^n], \\ \chi_{\Gamma_h^c} u_h^{n+1/3} &= \chi_{\Gamma_h^c} u_h^n. \end{cases} \quad (2.4)$$

We choose the backward Euler scheme to discretize the equation (1.3) on the

subdomains, obtaining

$$\begin{cases} \chi_{\Gamma_h^c} (I - \Delta t A_h) (\chi_{\Gamma_h^c} u_h^{n+2/3} + \chi_{\Gamma_h} u_h^{n+1/3} + u_b^{n+1}) &= \chi_{\Gamma_h^c} [u_h^n + \Delta t f^{n+1}], \\ \chi_{\Gamma_h} u_h^{n+2/3} &= \chi_{\Gamma_h} u_h^{n+1/3}, \end{cases} \quad (2.5)$$

where the terms $\chi_{\Gamma_h} u_h^{n+1/3}$ and u_b^{n+1} on the left hand side serve as boundary conditions for the equation on the subdomains Γ_h^c .

In the stabilization process the predicted interface boundary condition $\chi_{\Gamma_h} u_h^{n+1/3}$ is thrown away and $\chi_{\Gamma_h} u_h^n$ is brought back, and then the backward Euler scheme is used for the stabilization, so it has the representation

$$\begin{cases} \chi_{\Gamma_h} (I - \Delta t A_h) (\chi_{\Gamma_h} u_h^{n+1} + \chi_{\Gamma_h^c} u_h^{n+2/3} + u_b^{n+1}) &= \chi_{\Gamma_h} [u_h^n + \Delta t f^{n+1}], \\ \chi_{\Gamma_h^c} u_h^{n+1} &= \chi_{\Gamma_h^c} u_h^{n+2/3}. \end{cases} \quad (2.6)$$

We call the algorithm given by (2.4)-(2.5)-(2.6) the SEITM1 algorithm.

Now we derive the temporal error of the SEITM1 algorithm. Let $u_h(t)$ denote the true solution of the spatially discretized problem (1.4). For $n \in \mathbf{N}$, let $e_h^n = u_h(n\Delta t) - u_h^n$. Then e_h^n is the temporal error of the computed solution u_h^n . We have the following result concerning the temporal error of solution computed using the SEITM1 algorithm.

Theorem 2.3 *The temporal error satisfies*

$$e_h^{n+1} = (I - \Delta t A_1)^{-1} \left[\chi_{\Gamma_h} + \chi_{\Gamma_h^c} (I - \Delta t A_2)^{-1} (I - \Delta t A_1) \right] e_h^n + O\left((\Delta t)^2\right), \quad (2.7)$$

where $A_1 = \chi_{\Gamma_h} A_h$ and $A_2 = \chi_{\Gamma_h^c} A_h$, and

$$O\left((\Delta t)^2\right) = \frac{(\Delta t)^2}{2} (I - \Delta t A_1)^{-1} \left[(I - \Delta t A_2)^{-1} \left(\chi_{\Gamma_h} \frac{d^2 u_h(\xi_1)}{dt^2} - \chi_{\Gamma_h^c} \frac{d^2 u_h(\xi_2)}{dt^2} \right) - \chi_{\Gamma_h} \frac{d^2 u_h(\xi_2)}{dt^2} \right]. \quad (2.8)$$

for some $\xi_1, \xi_2 \in [t_n, t_{n+1}]$.

Proof: For notational simplicity, let $n\Delta t = t_n$ for all $n \in \mathbb{N}$. We introduce two notations

$$\begin{cases} e_h^{n+1/3} &= \chi_{\Gamma_h} [u_h(t_{n+1}) + \chi_{\Gamma_h^c} u_h(t_n)] - u_h^{n+1/3}, \\ e_h^{n+2/3} &= u_h(t_{n+1}) - u_h^{n+2/3}. \end{cases}$$

We prove the theorem in three steps by proving the following three equations.

$$(i) \quad e_h^{n+1/3} = (I + \Delta t A_1) e_h + \frac{(\Delta t)^2}{2} \chi_{\Gamma_h} \frac{d^2 u_h(\xi_1)}{dt^2}.$$

$$(ii) \quad (I - \Delta t A_2) e_h^{n+2/3} = e_h^{n+1/3} - \frac{1}{2} (\Delta t)^2 \chi_{\Gamma_h^c} \frac{d^2 u_h(\xi_2)}{dt^2}.$$

$$(iii) \quad (I - \Delta t A_1) e_h^{n+1} = \chi_{\Gamma_h} e_h^n + \chi_{\Gamma_h^c} e_h^{n+2/3} - \frac{1}{2} (\Delta t)^2 \chi_{\Gamma_h} \frac{d^2 u_h(\xi_2)}{dt^2}.$$

(i). We start by establishing the relation between $e_h^{n+1/3}$ and e_h^n . Since $\chi_{\Gamma_h^c} u_h^{n+1/3} = \chi_{\Gamma_h^c} u_h^n$,

$$\begin{aligned} e_h^{n+1/3} &= u_h(t_{n+1}) - u_h^{n+1/3} \\ &= \chi_{\Gamma_h} [u_h(t_{n+1}) - u_h^{n+1/3}] + \chi_{\Gamma_h^c} [u_h(t_n) - u_h^n] \\ &= \chi_{\Gamma_h} [u_h(t_{n+1}) - u_h^{n+1/3}] + \chi_{\Gamma_h^c} e_h^n. \end{aligned} \tag{2.9}$$

Using Taylor's expansion, we have that

$$u_h(t_{n+1}) = (I + \Delta t A_h) [u_h(t_n) + u_h^n] + \Delta t f^n + \frac{(\Delta t)^2}{2} \frac{d^2 u_h(\xi_1)}{dt^2} \tag{2.10}$$

for some $\xi_1 \in [t_n, t_{n+1}]$. Then using (2.4) and (2.10), from (2.9) we obtain

$$\begin{aligned} e_h^{n+1/3} &= \chi_{\Gamma_h} (I + \Delta t A_h) [u_h(t_n) - u_h^n] + \chi_{\Gamma_h^c} e_h^n + \frac{1}{2} (\Delta t)^2 \chi_{\Gamma_h} \frac{d^2 u_h(\xi_1)}{dt^2} \\ &= \chi_{\Gamma_h} (I + \Delta t A_h) e_h^n + \chi_{\Gamma_h^c} e_h^n + \frac{1}{2} (\Delta t)^2 \chi_{\Gamma_h} \frac{d^2 u_h(\xi_1)}{dt^2}. \end{aligned}$$

Since $\chi_{\Gamma_h}^2 = \chi_{\Gamma_h}$ and $\chi_{\Gamma_h} \chi_{\Gamma_h^c} = 0$, from the above equation we obtain

$$\begin{aligned} e_h^{n+1/3} &= \chi_{\Gamma_h} (I + \Delta t \chi_{\Gamma_h} A_h) e_h + \chi_{\Gamma_h^c} (I + \Delta t \chi_{\Gamma_h} A_h) e_h + \frac{1}{2} (\Delta t)^2 \chi_{\Gamma_h} \frac{d^2 u_h(\xi_1)}{dt^2} \\ &= (I + \Delta t \chi_{\Gamma_h} A_h) e_h + \frac{1}{2} (\Delta t)^2 \chi_{\Gamma_h} \frac{d^2 u_h(\xi_1)}{dt^2}. \end{aligned}$$

Since $A_1 = \chi_{\Gamma_h} A_h$, the above equation becomes

$$e_h^{n+1/3} = (I + \Delta t A_1) e_h + \frac{(\Delta t)^2}{2} \chi_{\Gamma_h} \frac{d^2 u_h(\xi_1)}{dt^2}. \quad (2.11)$$

(i). Then we establish the equation relating $e_h^{n+2/3}$ and $e_h^{n+1/3}$. Since $\chi_{\Gamma_h^c} u^{n+1/3} = \chi_{\Gamma_h^c} u^n$ and $\chi_{\Gamma_h} u_h^{n+2/3} = \chi_{\Gamma_h} u_h^{n+1/3}$, the subdomain scheme (2.5) implies that

$$\chi_{\Gamma_h^c} (I - \Delta t A_h) [u_h^{n+2/3} + u_b^{n+1}] = \chi_{\Gamma_h^c} (u_h^{n+1/3} + \Delta t f^{n+1}). \quad (2.12)$$

Using Taylor expansion we obtain

$$(I - \Delta t A_h) [u_h(t_{n+1}) + u_b^{n+1}] = u_h(t_n) + \Delta t f^{n+1} - \frac{(\Delta t)^2}{2} \frac{d^2 u_h(\xi_2)}{dt^2}. \quad (2.13)$$

for some $\xi_2 \in [t_n, t_{n+1}]$. Thus from (2.12) and (2.13) it follows that

$$\begin{aligned} \chi_{\Gamma_h^c} (I - \Delta t A_h) [u_h(t_{n+1}) - u_h^{n+2/3}] &= \chi_{\Gamma_h^c} [u_h(t_n) - u_h^{n+1/3} - \frac{1}{2}(\Delta t)^2 \frac{d^2 u_h(\xi_2)}{dt^2}] \\ &= \chi_{\Gamma_h^c} [e^{n+1/3} - \frac{1}{2}(\Delta t)^2 \frac{d^2(\xi_2)}{dt^2}], \end{aligned}$$

which means that

$$\chi_{\Gamma_h^c} (I - \Delta t A_2) e_h^{n+2/3} = \chi_{\Gamma_h^c} [e^{n+1/3} - \frac{1}{2}(\Delta t)^2 \frac{d^2 u_h(\xi_2)}{dt^2}]. \quad (2.14)$$

On the other hand, since $\chi_{\Gamma_h} u_h^{n+2/3} = \chi_{\Gamma_h} u_h^{n+1/3}$ we have that

$$\begin{aligned} \chi_{\Gamma_h} (I - \Delta t A_2) e_h^{n+2/3} &= \chi_{\Gamma_h} (I - \Delta t A_2) [u_h(t_{n+1}) - u_h^{n+2/3}] \\ &= \chi_{\Gamma_h} (I - 0) [u_h(t_{n+1}) - u_h^{n+1/3}] \\ &= \chi_{\Gamma_h} e_h^{n+1/3}. \end{aligned}$$

Combining with (2.14), the above equation produces

$$(I - \Delta t A_2) e_h^{n+2/3} = e^{n+1/3} - \frac{1}{2}(\Delta t)^2 \chi_{\Gamma_h^c} \frac{d^2 u_h(\xi_2)}{dt^2}. \quad (2.15)$$

(iii). Now we establish the equation relating e_h^{n+1} to $e_h^{n+2/3}$ and e_h^n . From (2.6) we obtain $\chi_{\Gamma_h} (I - \Delta t A_h)[u_h^{n+1} + u_b^{n+1}] = \chi_{\Gamma_h} [u_h^n + \Delta t f^{n+1}]$, which, together with the Taylor expansion (2.13), implies

$$\chi_{\Gamma_h} (I - \Delta t A_h)[u_h(t_{n+1}) - u_h^{n+1}] = \chi_{\Gamma_h} [u_h(t_n) - u_h^n - \frac{1}{2}(\Delta t)^2 \frac{d^2 u_h(\xi_2)}{dt^2}],$$

which is equivalent to

$$\chi_{\Gamma_h} (I - \Delta t A_1)e_h^{n+1} = \chi_{\Gamma_h} [e_h^n - \frac{1}{2}(\Delta t)^2 \frac{d^2 u_h(\xi_2)}{dt^2}]. \quad (2.16)$$

On the other hand,

$$\begin{aligned} \chi_{\Gamma_h^c} (I - \Delta t A_1)e_h^{n+1} &= \chi_{\Gamma_h^c} e_h^{n+1} = \chi_{\Gamma_h^c} [u_h(t_{n+1}) - u_h^{n+1}] \\ &= \chi_{\Gamma_h^c} [u_h(t_{n+1}) - u_h^{n+2/3}], \end{aligned} \quad (2.17)$$

where the last equality is due to the equation $\chi_{\Gamma_h^c} u_h^{n+1} = \chi_{\Gamma_h^c} u_h^{n+2/3}$ in (2.6).

Combining (2.16) and (2.17) we obtain

$$(I - \Delta t A_1)e_h^{n+1} = \chi_{\Gamma_h} e_h^n + \chi_{\Gamma_h^c} e_h^{n+2/3} - \frac{1}{2}(\Delta t)^2 \chi_{\Gamma_h} \frac{d^2 u_h(\xi_2)}{dt^2}. \quad (2.18)$$

Now from (2.11), (2.15) and (2.18), the conclusion of this theorem follows. \square

The SEITM1 algorithm is proven in [67] to be unconditionally Von Neumann stable and unconditionally convergent when the spatially discrete system (1.4) is dissipative. Now we extend the result to the general case without dissipativity assumption.

Theorem 2.4 *Let $G(\Delta t, h)$ denote the error amplification matrix of the SEITM1 algorithm in (2.7), namely,*

$$G(\Delta t, h) = (I - \Delta t A_1)^{-1} \left[\chi_{\Gamma_h} + \chi_{\Gamma_h^c} (I - \Delta t A_2)^{-1} (I - \Delta t A_1) \right].$$

Let $\omega = \max\{c(x) + \sum_{i=1}^k \frac{\partial b_i(x)}{\partial x_i} : x \in \Omega\}$, where $b_i(x)$ and $c(x)$ are coefficient functions in the evolutionary system (1.1). Suppose that the spatially discretized system (1.4) satisfies the quasi-dissipativity condition (1.5). Then the algorithm is unconditionally Von Neumann stable in the sense that the spectral radius of $G(\Delta t, h)$ satisfies $\rho(G(\Delta t, h)) \leq e^{7\omega\Delta t}$ for $\Delta t \in [0, \frac{1}{8\omega}]$ for all $h > 0$.

To prove this theorem, we need Lemma 4.1, Lemma 4.2 and Lemma 4.3 in [67].

The three lemmas are given below.

Lemma 2.1 *Let E be a Hilbert space with inner product $\langle \cdot, \cdot \rangle$ and norm $\|\cdot\|$. Let A be a C_0 -semigroup generator on E satisfying the quasi-dissipative condition*

$$\langle Av, v \rangle + \langle v, Av \rangle \leq 2\omega \|v\|^2 \quad (2.19)$$

for all $v \in D(A)$ for some constant $\omega \geq 0$. Then

$$\left\| \left(I - \frac{\Delta t}{2} A \right)^{-1} \left(I + \frac{\Delta t}{2} A \right) \right\| \leq e^{2\omega\Delta t} \quad \text{for all } \Delta t \in [0, \frac{1}{\omega}].$$

Lemma 2.2 *If the matrix A_h is symmetric and satisfies the quasi-dissipative condition (1.5), then component matrices A_1 and A_2 defined in Theorem 2.3 (and hence A_h itself also) satisfy the quasi-dissipative condition*

$$\langle A_i v, v \rangle_h < \omega \langle v, v \rangle_h \quad \text{for all } h > 0$$

with respect to the inner product $\langle \cdot, \cdot \rangle_h$ given by

$$\langle f, g \rangle_h := 2\omega \langle f, g \rangle - \langle f, A_h g \rangle. \quad (2.20)$$

Lemma 2.3 *Suppose A_h satisfies the quasi-dissipative condition (1.5). Then for all $\Delta t \in [0, \frac{1}{8\omega}]$, $f \in L^2(\Omega_h)$ and $h > 0$,*

$$\| \chi_{\Gamma_h} (I - tA_1)^{-1} f \|_h^2 \leq e^{4\omega t} \| \chi_{\Gamma_h} f \|_h^2 + (e^{4\omega t} - 1) \| \chi_{\Gamma_h^c} f \|_h^2, \quad (2.21)$$

$$\| \chi_{\Gamma_h^c} (I - tA_2)^{-1} f \|_h^2 \leq e^{4\omega t} \| \chi_{\Gamma_h^c} f \|_h^2 + (e^{4\omega t} - 1) \| \chi_{\Gamma_h} f \|_h^2. \quad (2.22)$$

Proof of Theorem 2.4: Let $\widehat{G}(\Delta t, h) = (I - \Delta t A_1)G(\Delta t, h)(I - \Delta t A_1)^{-1}$. Since similar matrices have the same spectral radius, we have that $\rho(\widehat{G}) = \rho(G)$. But the $\| \cdot \|_h$ norm of \widehat{G} satisfies

$$\begin{aligned} \|\widehat{G}(\Delta t, h)\|_h^2 &= \|\chi_{\Gamma_h}(I - \Delta t A_1)^{-1} f\|_h^2 + \|\chi_{\Gamma_h^c}(I - \Delta t A_2)^{-1}(I + \Delta t A_1)(I - \Delta t A_1)^{-1} f\|_h^2 \\ &< e^{4\omega \Delta t} \left(\|\chi_{\Gamma_h} f\|_h^2 + \|\chi_{\Gamma_h^c}(I + \Delta t A_1)(I - \Delta t A_1)^{-1} f\|_h^2 \right) + \\ &\quad (e^{4\omega \Delta t} - 1) \left(\|\chi_{\Gamma_h^c} f\|_h^2 + \|\chi_{\Gamma_h}(I + \Delta t A_1)(I - \Delta t A_1)^{-1} f\|_h^2 \right), \end{aligned} \quad (2.23)$$

where the last inequality is due to Lemma 2.3. Since operator A_1 operates only on entries corresponding to grid points on Γ_h , it was shown [67] that

$$\chi_{\Gamma_h^c}(I + \Delta t A_1)(I - \Delta t A_1)^{-1} f = \chi_{\Gamma_h^c} f.$$

Thus from (2.23),

$$\begin{aligned} \|\widehat{G}(\Delta t, h)\|_h^2 &< e^{4\omega \Delta t} \left(\|\chi_{\Gamma_h} f\|_h^2 + \|\chi_{\Gamma_h^c} f\|_h^2 \right) + \\ &\quad (e^{4\omega \Delta t} - 1) \left(\|\chi_{\Gamma_h^c} f\|_h^2 + \|\chi_{\Gamma_h}(I + \Delta t A_1)(I - \Delta t A_1)^{-1} f\|_h^2 \right) \\ &= e^{4\omega \Delta t} \|f\|_h^2 + \\ &\quad (e^{4\omega \Delta t} - 1) \left(\|\chi_{\Gamma_h^c} f\|_h^2 + \|\chi_{\Gamma_h}(I + \Delta t A_1)(I - \Delta t A_1)^{-1} f\|_h^2 \right). \end{aligned} \quad (2.24)$$

By Lemma 2.2 and Lemma 2.1, we also have $\|(I + \Delta t A_1)(I - \Delta t A_1)^{-1}\|_h^2 < e^{8\omega \Delta t}$ for $\Delta t \in [0, \frac{1}{2\omega}]$. Then from inequality (2.24) we further obtain

$$\begin{aligned} \|\widehat{G}(\Delta t, h)\|_h^2 &\leq e^{4\omega \Delta t} \|f\|_h^2 + (e^{4\omega \Delta t} - 1) \left(\|\chi_{\Gamma_h^c} f\|_h^2 + e^{8\omega \Delta t} \|f\|_h^2 \right) \\ &\leq e^{4\omega \Delta t} \|f\|_h^2 + (e^{4\omega \Delta t} - 1) (\|f\|_h^2 + e^{8\omega \Delta t} \|f\|_h^2) \\ &\leq e^{14\omega \Delta t} \|f\|_h^2, \end{aligned} \quad (2.25)$$

which means that $\|\widehat{G}\|_h \leq e^{7\omega\Delta t}$. But $\rho(G) = \rho(\widehat{G}) \leq \|\widehat{G}\|_h$, so the conclusion of the theorem follows. \square

The next result concerns the global (including both temporal and spatial) error of the SEITM1 algorithm. It reveals that the SEITM1 algorithm does not reduce the order of the spatial discretization accuracy. This is good not only when compared with existing stable non-iterative domain decomposition, more importantly this property allows high order schemes be used to discretize the spatial operator A . Numerical experiments with high order spatial discretization are presented in Chapter 4.

Theorem 2.5 *Let $u(t)$ and $u_h(t)$ be the true solutions of the system (1.3) and the spatially discrete system (1.4) respectively whose discrete operator A_h is symmetric and satisfies the quasi-dissipativity condition (1.5). Suppose that both $u(t)$ and $u_h(t)$ are continuously twice differentiable in t and that the second derivative of $u_h(t)$ are uniformly bounded for all h on any time interval $[0, T]$. Let C^T be a bound of second derivative of $u(t)$ and $u_h(t)$ on $[0, T]$, namely,*

$$C_T = \max \left\{ \max_{0 \leq t \leq T} \left\| \frac{\partial^2 u(t)}{\partial t^2} \right\|_h, \max_{0 \leq t \leq T} \left\| \frac{d^2 u_h(t, x)}{dt^2} \right\|_h \right\}. \quad (2.26)$$

If A_h has a p -th order truncation error in approximating the operator A , namely $\|A_h P_h u(t) - P_h A u(t)\|_h \leq M_T h^p$ for all $t \in [0, T]$ for some constant M_T dependent on the solution u , where P_h is the projection operator from $L^2(\Omega)$ to $L^2(\Omega_h)$, then the approximate solution u_h^n satisfies

$$\max_{t_n \in [0, T]} \|u_h^n - u(t_n)\|_h = O(\Delta t) + O(h^p)$$

for all $h > 0$, where $t_n = n\Delta t$.

We first prove a lemma that concerns the spatial error of the SEITM1 algorithm.

Lemma 2.4 *Let $u(t)$ and $u_h(t)$ be as in Theorem 2.5. Then*

$$\max_{0 \leq t \leq T} \|u_h(t) - P_h u(t)\|_h \leq T M_T e^{\omega T} h^p.$$

Proof: Let $w(t) = u_h(t) - P_h u(t)$. Since $u_h(t)$ and $u(t)$ agree on the boundary condition and the inhomogeneous term $f(t)$, it is easily verifiable that

$$\begin{cases} \frac{dw(t)}{dt} = A_h w(t) + (A_h P_h - P_h A)u(t), & t \geq 0 \\ w(0) = 0. \end{cases}$$

Then by Theorem 2.1.3 on page 84 of [26],

$$w(t) = \int_0^t e^{(t-s)A_h} (A_h P_h - P_h A)u(s) ds.$$

Since A_h satisfies the quasi-dissipativity condition (1.5), by Lemma 2.2 A_h satisfies the following quasi-dissipative condition

$$\langle Av, v \rangle_h < \omega \langle v, v \rangle_h \quad \text{for all } h > 0$$

with respect to the inner product $\langle \cdot, \cdot \rangle_h$ given by (2.20). The Lumer-Phillips theorem [46] then implies that $\|e^{tA_h}\|_h \leq e^{\omega t}$. Thus,

$$\begin{aligned} \|w(t)\|_h &\leq \int_0^t \|e^{(t-s)A_h} (A_h P_h - P_h A)u(s)\|_h ds \\ &\leq M_T h^p \int_0^t e^{\omega(t-s)} ds \\ &\leq t e^{\omega t} M_T h^p \leq T M_T e^{\omega T} h^p. \end{aligned}$$

□

Proof of Theorem 2.5: We first establish the following inequalities

$$\|(I - \Delta t A_i)^{-1}\|_h \leq e^{2\omega \Delta t} \quad \text{for } \Delta t \in [0, \frac{1}{8\omega}]. \quad (2.27)$$

for $i = 1, 2$. By Lemma 2.2,

$$\langle A_i v, v \rangle_h < \omega \|v\|_h^2 \quad \text{for all } v \in L^2(\Omega_h).$$

But $\omega < \frac{e^{4\omega\Delta t} - 1}{2\Delta t e^{4\omega\Delta t}}$ for $\Delta t \in (0, \frac{1}{8\omega}]$, we have that

$$2\Delta t e^{4\omega\Delta t} \langle A_i v, v \rangle_h \leq (e^{4\omega\Delta t} - 1) \|v\|_h^2$$

for all $v \in L^2(\Omega_h)$. This is equivalent to

$$\|v\|_h^2 \leq e^{4\omega\Delta t} (\|v\|_h^2 - 2\Delta t \langle A_i v, v \rangle_h). \quad (2.28)$$

But obviously the right hand side of the above inequality satisfies

$$\begin{aligned} e^{4\omega\Delta t} (\|v\|_h^2 - 2\Delta t \langle A_i v, v \rangle_h) &\leq e^{4\omega\Delta t} (\|v\|_h^2 - 2\Delta t \langle A_i v, v \rangle_h + (\Delta t) 2 \|A_i v\|_h^2) \\ &= e^{4\omega\Delta t} \|(I - \Delta t A_i) v\|_h^2, \end{aligned}$$

which, together with (2.28), implies that

$$\|v\|_h \leq \|(I - \Delta t A_i) v\|_h \quad \text{for all } v \in L^2(\Omega_h)$$

for $\Delta t \in [0, \frac{1}{8\omega}]$. This immediately leads to the inequality (2.27).

We then show that

$$\max_{t_n \in [0, T]} \|u_h^n - u_h(t)\|_h \leq 6C_T T \Delta t. \quad (2.29)$$

We rewrite (2.7) in the form

$$e_h^n = G(\Delta t, h) e_h^{n-1} + \frac{(\Delta t)^2}{2} (I - \Delta t A_1)^{-1} Tr(n, h), \quad (2.30)$$

where $G(\Delta t, h)$ is the error amplification matrix, and $Tr(n, h)$ is given by

$$Tr(n, h) = (I - \Delta t A_2)^{-1} \left(\chi_{\Gamma_h} \frac{d^2 u_h(\zeta_n)}{dt^2} - \chi_{\Gamma_h^c} \frac{d^2 u_h(\xi_n)}{dt^2} \right) - \chi_{\Gamma_h} \frac{d^2 u_h(\xi_n)}{dt^2}.$$

for some $\zeta_n, \xi_n \in [t_n, t_{n+1}]$. Since $e_h^0 = 0$, from (2.30) we obtain by induction that

$$e_h^n = \frac{(\Delta t)^2}{2} \sum_{k=0}^{n-1} G(\Delta t, h)^k (I - \Delta t A_1)^{-1} \text{Tr}(n-k, h) \quad (2.31)$$

for $n \geq 0$. Let \widehat{G} be as in the proof of Theorem 2.4. Then (2.31) can be rewritten into

$$e_h^n = \frac{(\Delta t)^2}{2} (I - \Delta t A_1)^{-1} \sum_{k=0}^{n-1} \widehat{G}(\Delta t, h)^k \text{Tr}(n-k, h) \quad (2.32)$$

With inequality (2.27), we have that

$$\begin{aligned} \|\text{Tr}(n, h)\|_h &\leq \|(I - \Delta t A_2)^{-1} \left(\chi_{\Gamma_h} \frac{d^2 u_h(\zeta_n)}{dt^2} - \chi_{\Gamma_h^c} \frac{d^2 u_h(\xi_n)}{dt^2} \right)\|_h + \|\chi_{\Gamma_h} \frac{d^2 u_h(\xi_n)}{dt^2}\|_h \\ &\leq e^{2\omega \Delta t} \left(\|\chi_{\Gamma_h} \frac{d^2 u_h(\zeta_n)}{dt^2}\|_h + \|\chi_{\Gamma_h^c} \frac{d^2 u_h(\xi_n)}{dt^2}\|_h \right) + \|\chi_{\Gamma_h} \frac{d^2 u_h(\xi_n)}{dt^2}\|_h \\ &\leq C_T (2e^{2\omega \Delta t} + 1), \end{aligned} \quad (2.33)$$

where the last inequality is due to (2.26). With inequalities (2.27) and (2.33), from (2.32) we obtain

$$\begin{aligned} \|e_h^n\|_h &\leq \frac{(\Delta t)^2}{2} \|(I - \Delta t A_1)^{-1}\|_h \sum_{k=0}^{n-1} \|\widehat{G}(\Delta t, h)^k\|_h \cdot \|\text{Tr}(n-k, h)\|_h \\ &\leq \frac{(\Delta t)^2}{2} e^{2\omega \Delta t} \sum_{k=0}^{n-1} \|\widehat{G}(\Delta t, h)^k\|_h C_T (2e^{2\omega \Delta t} + 1) \\ &\leq \frac{(\Delta t)^2}{2} e^{2\omega \Delta t} \sum_{k=0}^{n-1} e^{7\omega \Delta t} C_T (2e^{2\omega \Delta t} + 1), \end{aligned}$$

where the last inequality is due to (2.25). Thus

$$\begin{aligned} \|e_h^n\|_h &\leq \frac{n(\Delta t)^2}{2} e^{2\omega \Delta t} e^{7\omega \Delta t} C_T (2e^{2\omega \Delta t} + 1) \\ &\leq 3C_T \frac{n(\Delta t)^2}{2} e^{2\omega \Delta t} e^{7\omega \Delta t} e^{2\omega \Delta t} \\ &= 3C_T \frac{n(\Delta t)^2}{2} e^{11\omega \Delta t} \\ &= \frac{3C_T t_n}{2} e^{11\omega \Delta t} \Delta t. \end{aligned}$$

Since $t_n \in [0, T]$ and $\Delta t \leq \min\{T, \frac{1}{8\omega}\}$, we have that

$$\|e_h^n\|_h \leq \frac{3C_T T}{2} e^{11/8} \Delta t \leq 6C_T T \Delta t.$$

which establishes (2.29).

With (2.29) and Lemma 2.4,

$$\begin{aligned} \|u_h^n - u(t)\|_h &\leq \|u_h^n - u_h(t)\|_h + \|u_h(t) - u(t)\|_h \\ &\leq 6T C_T \Delta t + T M_T e^{\omega T} h^p, \end{aligned}$$

which completes the proof of this theorem. \square

With Theorem 2.5, it is immediately obtainable that the SEITM1 algorithm is unconditionally convergent in the sense that

$$\lim_{\substack{\Delta t \rightarrow 0 \\ h \rightarrow 0}} \|u_h^n - u(n\Delta t)\|_h = 0$$

uniformly for $(h, n\Delta t) \in [0, 1] \times [0, T]$ for any $T > 0$.

2.3 The Stabilized Dawson-Du-Dupont Algorithm

In this section, we present a stabilizer for the explicit-implicit algorithm of Dawson, Du and Dupont. We call the stabilized algorithm the SEITM2 algorithm. The emphasis of this section is to illustrate how to design a stabilizer for a given interface boundary condition predictor.

Let A_h^x denote the x-directional difference component of the operator A_h , and A_h^y the y-directional difference component. Let $u_{b,x}$ be the exterior boundary condition at x-direction boundary and $u_{b,y}$ the exterior boundary condition at y-direction boundary. Then we represent the two steps of the Dawson-Du-Dupont Algorithm by

$$\begin{cases} \chi_{\Gamma_h} (I - \Delta t A_h^y)(u_h^{n+1/3} + u_{b,y}^{n+1}) = \chi_{\Gamma_h} [(I + \Delta t A_y)(u_h^n + u_{b,x}^n) + \Delta t f^n], \\ \chi_{\Gamma_h^c} u_h^{n+1/3} = \chi_{\Gamma_h^c} u_h^n, \end{cases} \quad (2.34)$$

and

$$\begin{cases} \chi_{\Gamma_h^c} (I - \Delta t A_h) (\chi_{\Gamma_h^c} u_h^{n+2/3} + \chi_{\Gamma_h} u_h^{n+1/3} + u_b^{n+1}) = \chi_{\Gamma_h^c} [u_h^n + \Delta t f^{n+1}], \\ \chi_{\Gamma_h} u_h^{n+2/3} = \chi_{\Gamma_h} u_h^{n+1/3}. \end{cases} \quad (2.35)$$

To stabilize the predictor (2.34), we use the scheme

$$\begin{cases} \chi_{\Gamma_h} (I - \Delta t A_h^y) (I - \Delta t A_h^x) (u_h^{n+1} + u_b^{n+1}) = \chi_{\Gamma_h} [u_h^n + \Delta t f^{n+1}], \\ \chi_{\Gamma_h^c} u_h^{n+1} = \chi_{\Gamma_h^c} u_h^{n+2/3}, \end{cases} \quad (2.36)$$

namely, the stabilizer is chosen to be $(I - \Delta t A_h^x)^{-1} (I - \Delta t A_h^y)^{-1}$. In this stabilizer we place the operator $(I - \Delta t A_h^x)^{-1}$ to the left of $(I - \Delta t A_h^y)^{-1}$. This ordering of the two factors in the stabilizer is important. The importance of this ordering can be seen from an analysis utilizing the following result obtainable from an analysis similar to that for Theorem 2.3 for the SEITM1 algorithm.

Theorem 2.6 *The temporal error of the SEITM2 algorithm satisfies*

$$\begin{aligned} e_h^{n+1} &= (I - \Delta t A_1^x)^{-1} (I - \Delta t A_1^y)^{-1} [\chi_{\Gamma_h} + \chi_{\Gamma_h^c} (I - \Delta t A_2)^{-1} (I - \Delta t A_1^y)^{-1} (I - \Delta t A_1^x)] e_h^n \\ &\quad + O((\Delta t)^2), \end{aligned}$$

where $A_1^x = \chi_{\Gamma_h} A_h^x$ and $A_1^y = \chi_{\Gamma_h} A_h^y$.

The theorem above states that the error amplification matrix of the SEITM2 algorithm is

$$G(\Delta t, h) = S(\Delta t, h) \left[\chi_{\Gamma_h} + \chi_{\Gamma_h^c} (I - \Delta t A_2)^{-1} P(\Delta t, h) \right],$$

where $S(\Delta t, h)$ is the stabilizer given by

$$S(\Delta t, h) = (I - \Delta t A_1^x)^{-1} (I - \Delta t A_1^y)^{-1}, \quad (2.37)$$

and $P(\Delta t, h)$ is the interface boundary condition predictor

$$P(\Delta t, h) = (I - \Delta t A_1^y)^{-1} (I - \Delta t A_1^x).$$

In the matrix $G(\Delta t, h)$, the instability causing operator is $P(\Delta t, h)$. The stabilizer stabilizes the EITM algorithm by killing the instability of the predictor $P(\Delta t, h)$ of the next time step. This can be seen from

$$\begin{aligned} G(\Delta t, h)^n &= S(\Delta t, h) \left[\chi_{\Gamma_h} S(\Delta t, h) + \chi_{\Gamma_h^c} (I - \Delta t A_2)^{-1} P(\Delta t, h) S(\Delta t, h) \right]^n S(\Delta t, h)^{-1} \\ &= S(\Delta t, h) \tilde{G}(\Delta t, h)^n S(\Delta t, h)^{-1}, \end{aligned}$$

where $\tilde{G}(\Delta t, h) = S(\Delta t, h)^{-1} G(\Delta t, h) S(\Delta t, h)$, namely,

$$\tilde{G}(\Delta t, h) = \left[\chi_{\Gamma_h} S(\Delta t, h) + \chi_{\Gamma_h^c} (I - \Delta t A_2)^{-1} P(\Delta t, h) S(\Delta t, h) \right].$$

From the equation above it is immediately seen that $P(\Delta t, h) S(\Delta t, h)$ needs to be stable in order for the algorithm to be stable. With the ordering of the two factors in $S(\Delta t, h)$,

$$\begin{aligned} P(\Delta t, h) S(\Delta t, h) &= [(I - \Delta t A_1^y)^{-1} (I - \Delta t A_1^x)] \cdot [(I - \Delta t A_1^x)^{-1} (I - \Delta t A_1^y)^{-1}] \\ &= (I - \Delta t A_1^y)^{-1} [(I - \Delta t A_1^x) (I - \Delta t A_1^x)^{-1}] (I - \Delta t A_1^y)^{-1}. \end{aligned}$$

The operator $(I - \Delta t A_1^y)^{-1}$ has good stability since it is an implicit scheme. The factor in the bracket also has a better stability than $(I - \Delta t A_1^x)$ since the unstable operator $(I - \Delta t A_h^x)$ from $P(\Delta t, h)$ is stabilized by the operator $(I - \Delta t A_h^x)^{-1}$ from $S(\Delta t, h)$.

However if the ordering of the two operators in $S(\Delta t, h)$ is reversed, namely,

$$S(\Delta t, h) = (I - \Delta t A_1^y)^{-1} (I - \Delta t A_1^x)^{-1},$$

then $P(\Delta t, h) S(\Delta t, h) = (I - \Delta t A_1^y)^{-1} (I - \Delta t A_1^x) (I - \Delta t A_h^y)^{-1} (I - \Delta t A_h^x)^{-1}$. But $(I - \Delta t A_h^y)^{-1}$ is not guaranteed to be able to stabilize $(I - \Delta t A_1^x)$, and the four factors in $P(\Delta t, h) S(\Delta t, h)$ also do not necessarily commute, so this time the stabilizer might not necessarily be able to stabilize the predictor.

2.4 An Algorithm with Factorized Subdomain Scheme

Both SEITM1 and SEITM2 algorithms require solving an elliptic equation of the form

$$(I - \Delta t A)u = r \quad (2.38)$$

on the subdomains. When the spatial operator A is not separable or even nonsymmetric, iterative solvers must be employed to solve the elliptic equation for each time step. In this section, we propose an algorithm which factorizes the left hand side of the equation (2.38) with a direction based splitting of the spatial operator A . This factorization reduces the computation cost to an linear order of $O(N)$ for a total of N grid points on the subdomains. With this factorization, the factorized SEITM algorithm becomes completely non-iterative, both globally and on each subdomains.

We choose to present the factorization for the SEITM1 algorithm. The factorization for the SEITM2 is exact the same. We assume that the spatial domain of the evolutionary system is two dimensional and the spatial operator A is splittable as $A = A^x + A^y$. For example, when A is the 2-D Laplace operator $A = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$, $A^x = \frac{\partial^2}{\partial x^2}$ and $A^y = \frac{\partial^2}{\partial y^2}$ form a directional splitting of the operator A . The SEITM algorithm with a directionally factorized subdomain temporal scheme is given below.

$$\begin{cases} \chi_{\Gamma_h} u_h^{n+1/3} &= \chi_{\Gamma_h} [(I + \Delta t A_h)(u_h^n + u_{b,x}^n) + \Delta t f^n], \\ \chi_{\Gamma_h^c} u_h^{n+1/3} &= \chi_{\Gamma_h^c} u_h^n, \end{cases} \quad (2.39)$$

$$\begin{cases} \chi_{\Gamma_h^c}(I-\Delta t A_h^x)(I-\Delta t A_h^y)(\chi_{\Gamma_h^c} u_h^{n+2/3} + \chi_{\Gamma_h} u_h^{n+1/3} + u_b^{n+1}) = \chi_{\Gamma_h^c} [u_h^n + \Delta t f^{n+1}], \\ \chi_{\Gamma_h} u_h^{n+2/3} = \chi_{\Gamma_h} u_h^{n+1/3}, \end{cases} \quad (2.40)$$

and

$$\begin{cases} \chi_{\Gamma_h}(I-\Delta t A_h^y)(I-\Delta t A_h^x)(u_h^{n+1} + u_b^{n+1}) = \chi_{\Gamma_h} [u_h^n + \Delta t f^{n+1}], \\ \chi_{\Gamma_h^c} u_h^{n+1} = \chi_{\Gamma_h^c} u_h^{n+2/3}. \end{cases} \quad (2.41)$$

We call the algorithm (2.39)-(2.40)-(2.41) the SEITM3 algorithm. The difference of this algorithm and SEITM1 is that in the second step (2.39), the SEITM3 algorithm has $(I - \Delta t A_h^x)(I - \Delta t A_h^y)$ on the left hand side while the SEITM1 algorithm has $(I - \Delta t A_h)$. For a two dimensional problem, the discretized directional components of A are usually tridiagonal matrices [3]. Thus $(I - \Delta t A_h^x)$ and $(I - \Delta t A_h^y)$ can be easily inverted with a computation cost of linear order. On the other end, the factorization introduces an $O((\Delta t)^2)$ error. Since $A = A^x + A^y$, we have that

$$(I - \Delta t A_h^x)(I - \Delta t A_h^y) = I - \Delta t A_h + (\Delta t)^2 A_h^x A_h^y.$$

Thus the error introduced by factorizing $(I - \Delta t A_h)$ into $(I - \Delta t A_h^x)(I - \Delta t A_h^y)$ is $(\Delta t)^2 A_h^x A_h^y$. But as indicated by Theorem 2.3, the SEITM1 algorithm has a $O((\Delta t)^2)$ temporal truncation error. Thus the error introduced by the directional factorization of the subdomain scheme does not decrease the order of temporal accuracy.

3 High Order Elliptic Solvers

Two of the algorithms (SEITM1) and SEITM2) require solving elliptic equations of the form

$$(I - \tau A_h)u = r \tag{3.1}$$

on the subdomains. For 3-D parabolic systems, the stabilization step of all three algorithms also involve solving elliptic equations of the form (3.1). Since the SEITM algorithms maintain the accuracy of the spatial discretization, in this chapter we propose several fourth order methods for the solution of the generalized Helmholtz equation

$$-\sum_{i=1}^k \frac{\partial^2}{\partial x_i^2} u + f(x)u = r$$

with $f(x) \geq 0$, which arises when the system (1.1) has a heat source-sink term. We present our solution methods for the 2-D problem

$$-\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u(x, y) + f(x, y)u(x, y) = r(x, y), \tag{3.2}$$

but the methods are applicable to 3-D problems as well.

3.1 Introduction

Execution time in general is approximately proportional to the number of floating point arithmetic operations. For a given error tolerance, a high order method allows much larger mesh sizes than a lower order method, resulting in significant reduction in the number of grid points and consequently execution time if the high order method has the same computation complexity as that of the lower order

method. We illustrate such time reduction by the computation count comparison of a fourth order method and a second order method of the same complexity.

We introduce the following notations: $E(Mthd)$ denotes the difference between the true solution and the numerical solution computed by method $Mthd$, $\varepsilon > 0$ is the error tolerance, i.e. the difference between the computed numerical solution and the true solution must be less than or equal to ε . With these notations, the error of a fourth-order method can be denoted by $E(order4)$, and the error of a second-order solver will be $E(order2)$. The solution error of the fourth order method in general satisfies

$$E(order4) = a \cdot h^4 \quad \text{for some problem dependent coefficient } a.$$

The error of a second order solution method in general satisfies

$$E(order2) = b \cdot h^2 \quad \text{for some problem dependent coefficient } b.$$

To meet the error tolerance, the fourth and second order methods need to take different mesh sizes and partition sizes, say dimensional partition size N and mesh size h for the fourth order solver, and partition size N' and mesh size h' for the second order method. Then

$$a \cdot h^4 \leq \varepsilon \quad \text{and} \quad b \cdot h'^2 \leq \varepsilon$$

Roughly we can equate them to yield

$$a \cdot h^4 \approx b \cdot h'^2. \tag{3.3}$$

Since $h = 1/N$ and $h' = 1/N'$, (3.3) is equivalent to

$$N' = \sqrt{\frac{b}{a}} N^2 = C N^2, \tag{3.4}$$

where

$$C = \sqrt{\frac{b}{a}}. \quad (3.5)$$

Thus, if our fourth order solver can satisfy the error tolerance by taking a partition of size N , then it requires the second order solver to take a partition size of CN^2 to achieve the same accuracy. Suppose that the problem domain is in k -dimensional space ($k = 2, 3$). Then the number of grid points of the grid for the fourth order solver is N^k while the number of grid points of the grid for the second solver is $(N')^k = (CN^2)^k$. Let T_4 and T_2 denote the time needed by the order 4 and order 2 methods respectively to solve a problem within a given error tolerance. Thus if the two methods have the same computation complexity $\phi(n)$ for a problem on a grid with n grid points, then (3.4) implies that

$$T_2 : T_4 = \frac{\phi(C^k N^{2k})}{\phi(N^k)}. \quad (3.6)$$

Since complexity function increases at least linearly as the input size increases, that is,

$$\phi(an) \geq a\phi(n) \quad \text{for } a \geq 1.$$

Then (3.6) implies that the time ratio of a second solver and a fourth solver of the same computation complexity is

$$T_2 : T_4 \geq \frac{C^k N^k \phi(N^k)}{\phi(N^k)} = C^k N^k.$$

The parameter C in general could vary largely from problem to problem. A fourth solver can attain fourth order only when the solution is at least five times differentiable. This can be easily seen from the Taylor expansion of a differentiable function. Thus problems which have only twice differentiable solutions, the fourth

order method has only second order accuracy, and the number C will be close very small, and a fourth order method probably has no gain in reducing execution time for a given error tolerance. But for problems with at least three times differentiable solutions, the fourth order can take advantage of the smoothness of the solutions and reduce the computation cost for a given error tolerance.

3.2 High Order Discretizations for The Laplace Operator

3.2.1 Existing High Order Discretization Schemes

High order discretization methods for the Laplace operator have been investigated for a long time. Collatz studied several finite difference methods for the 2-D Laplace operator in 1960 [11]. One of the fourth-order methods Collatz studied is the following square stencil 9-point scheme written in the stencil form

$$h^{-2} \begin{pmatrix} -\frac{1}{4} & -1 & -\frac{1}{4} \\ -1 & 5 & -1 \\ -\frac{1}{4} & -1 & -\frac{1}{4} \end{pmatrix} u^{i,j} = - \begin{pmatrix} \frac{1}{8} & & \\ & 1 & \\ & & \frac{1}{8} \end{pmatrix} \Delta u^{i,j} + O(h^4), \quad (3.7)$$

where $\Delta u = (\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})u$, which is a popular choice for the Dirichlet problem of Poisson equations. This high order discretization method together with others studied by Collatz were later generalized by Lynch and Rice to a method called HODIE [44] for calculating the coefficients of finite difference discretization of general elliptic equations for almost “any” numerical order. A discretization formula the HODIE method derives is an equation that equates a linear combination of the unknown solution $u^{i,j}$ at some discretization points to a linear combination of the

to-be-discretized differential operator at some evaluation points, for instance as in

$$\sum_{i,j \in S_1} a_{i,j} u^{i,j} = \sum_{i,j \in S_2} b_{i,j} \Delta u^{i,j}.$$

The coefficients $a_{i,j}$ and $b_{i,j}$ in the formula are determined by, first, expanding all terms on both sides of the equation in Taylor series, then setting up a linear system of equations for the to-be-determined coefficients $a_{i,j}$ and $b_{i,j}$ for each order from 0 up to the desired order, and finally solving the linear system for the coefficients.

For Neumann problems, when scheme (3.7) applied at the boundary grid point $(0,j)$, the right hand side needs the value of $\Delta u^{-1,j}$, which is outside the domain. So this method is not applicable to the Neumann boundary problem. A modification is hence necessary. In 1987 Boisvert successfully calculated fourth order discretization coefficients using the HODIE method for the Neumann problem of Helmholtz equations [5]. Zhuang and Sun [59, 70] modified the Collatz scheme (3.7) and obtained the following fourth-order formula

$$h^{-2} \begin{pmatrix} -\frac{1}{4} & -1 & -\frac{1}{4} \\ -1 & 5 & -1 \\ -\frac{1}{4} & -1 & -\frac{1}{4} \end{pmatrix} u^{i,j} = \frac{3}{2} \Delta u^{i,j} - \frac{h^2}{8} \Delta^2 u^{i,j} + O(h^4). \quad (3.8)$$

which is more general than both the Collatz formula (3.7) and Boisvert's formula given on page 199-200 of [52]. The formula (3.8) is applicable to Neumann boundary problem [59, 70].

However all the existing popular high order discretization schemes, like (3.7) and (3.8), have a square stencil and thus not directly utilizable by the efficient ADI

method. The cross stencil fourth order finite central difference

$$-\Delta u^{i,j} = h^{-2} \begin{pmatrix} & & & & \frac{1}{12} \\ & & & & \frac{4}{3} \\ & & & & \frac{4}{3} \\ \frac{1}{12} & -\frac{4}{3} & 5 & -\frac{4}{3} & \frac{1}{12} \\ & & & & \frac{4}{3} \\ & & & & \frac{1}{12} \end{pmatrix} u^{i,j}$$

is easily directionally splittable and can be combined directly with the ADI method. However it not only has a boundary problem (see Collatz [11]), but also its pentadiagonal directional components can not be as efficiently solved as tridiagonal systems. So in the next section, we present a method for deriving fourth order discretization formulas for the Laplace operator, which is of greater ease in coefficient calculating and flexible in obtaining formulas for different boundary conditions.

3.2.2 Directionally Decomposed High Order Discretization

In this section, we present a fourth order finite difference method for the discretization of the Laplace operator in k dimensional space for $k = 2, 3$. The method we introduce in this section is based on the one-dimensional Taylor expansion. It is much simpler in calculating the discretization coefficients when compared with existing high order discretization methods, especially with discretization meshes different in different directions. The method can also easily produce formulas for both Dirichlet and Neumann boundary problems, and applicable directly to directional operator splitting as in the ADI method.

For a sufficiently smooth function u defined on a 2-D domain $[x_0, x_m] \times [y_0, y_n]$, we use $u^{i,j}$ denote its value on a discrete grid point (x_i, y_j) with $x_i = x_0 + i * h_x$ and

$y_j = y_0 + j * h_y$. From the Taylor expansion for functions defined on 1-D domain, we have that

$$\frac{u^{i-1,j} - 2u^{i,j} + u^{i+1,j}}{h_x^2} = u_{xx}^{i,j} + \frac{h_x^2}{12} u_{xxxx}^{i,j} + O(h_x^4),$$

which is equivalent to

$$D_{xx}u^{i,j} = (I + \frac{h_x^2}{12} \frac{\partial^2}{\partial x^2})u_{xx}^{i,j} + O(h_x^4), \quad (3.9)$$

where D_{xx} denotes the finite difference operator given by

$$D_{xx}u^{i,j} = \frac{u^{i-1,j} - 2u^{i,j} + u^{i+1,j}}{h_x^2}.$$

Thus we obtain the following fourth order semi-discrete formula for the Laplace operator

$$[(I + \frac{h_x^2}{12} \frac{\partial^2}{\partial x^2})^{-1} D_{xx} + (I + \frac{h_y^2}{12} \frac{\partial^2}{\partial y^2})^{-1} D_{yy}]u^{i,j} = \Delta u^{i,j}. \quad (3.10)$$

The above formula is not fully discretized. It still contains differential terms. However it is these differential terms that provide the flexibility of this formula. To maintain a fourth order accuracy of the formula, finite difference approximation for the differential terms $\frac{\partial^2}{\partial x^2}$ and $\frac{\partial^2}{\partial y^2}$ needs to be only second order, and many second order finite difference schemes exist for the approximation of the second derivatives. For Dirichlet boundary problems, discretization needs to be applied to interior points only since the solution values on the boundary are known. Thus we can choose D_{xx} and D_{yy} respectively to approximate the differential terms $\frac{\partial^2}{\partial x^2}$ and $\frac{\partial^2}{\partial y^2}$ in the formula (3.10) for Dirichlet problems, and obtain a fully discretized formula

$$[(I + \frac{h_x^2}{12} D_{xx})^{-1} D_{xx} + (I + \frac{h_y^2}{12} D_{yy})^{-1} D_{yy}]u^{i,j} = \Delta u^{i,j}. \quad (3.11)$$

This formula has an obvious advantage that it is already directionally decomposed and can be easily incorporated into the ADI method.

For Neumann boundary problems, formula (3.10) is first rewritten into

$$[(I + \frac{h_y^2}{12} \frac{\partial^2}{\partial y^2})D_{xx} + (I + \frac{h_x^2}{12} \frac{\partial^2}{\partial x^2})D_{yy}]u^{0,j} = (I + \frac{h_y^2}{12} \frac{\partial^2}{\partial y^2})(I + \frac{h_x^2}{12} \frac{\partial^2}{\partial x^2})\Delta u^{i,j}. \quad (3.12)$$

Then we throw away terms of order $h_x^2 h_y^2$ and obtain

$$[(I + \frac{h_y^2}{12} \frac{\partial^2}{\partial y^2})D_{xx} + (I + \frac{h_x^2}{12} \frac{\partial^2}{\partial x^2})D_{yy}]u^{0,j} = (I + \frac{h_y^2}{12} \frac{\partial^2}{\partial y^2} + \frac{h_x^2}{12} \frac{\partial^2}{\partial x^2})\Delta u^{i,j}.$$

We approximate the differential term $\frac{\partial^2}{\partial x^2}$ and $\frac{\partial^2}{\partial y^2}$ on the left hand side by D_{xx} and D_{yy} respectively, but leave the differential terms on the right side un-discretized, resulting in the following partially discrete finite difference approximation of the Laplace operator

$$[D_{xx}(I + \frac{h_y^2}{12} D_{yy}) + (I + \frac{h_x^2}{12} D_{xx})D_{yy}]u^{i,j} = (I + \frac{h_x^2}{12} \frac{\partial^2}{\partial x^2} + \frac{h_y^2}{12} \frac{\partial^2}{\partial y^2})\Delta u^{i,j}. \quad (3.13)$$

It is obvious that this formula is a generalization of the modified Collatz scheme (3.8) in that it allows different mesh sizes in different directions. Formula (3.13) is not fully discretized. It still contains differential operators $\frac{\partial^2}{\partial x^2}$ and $\frac{\partial^2}{\partial y^2}$. This is the main difference between our discretization formula and most existing schemes which are usually fully discretized. But since a finite difference approximation of differential operators needs to be only second order for the truncation error to remain fourth order, it is easy to find many simple second order finite difference approximations for these differential terms (e.g. 1-D formulas in [43]). The main advantage of formula (3.13) is that it allows people to choose different approximation schemes for $\frac{\partial^2}{\partial x^2}$ and $\frac{\partial^2}{\partial y^2}$. This flexibility is especially useful for Neumann

problems since for interior grid points, the finite central difference

$$\frac{f_{i-1} - 2f_i + f_{i+1}}{h^2} = f_i''$$

can be used while for boundary points other approximation formulas can be used to avoid using values of Δu outside the domain.

3.3 The Picard-FFT Solution Methods

In this section we propose two Picard-type iterative solution methods for the equation (3.2) discretized by a fourth order scheme on a rectangular domain, with an FFT based solver used for each iteration.

3.3.1 A Picard Process

We solve the equation (3.2) using the following Picard-type process

$$[-\Delta + \alpha]u_{n+1} = [\alpha - f(x, y)]u_n + r, \quad (3.14)$$

where the constant α is chosen to be

$$\alpha = \frac{\min_{x,y} f(x, y) + \max_{x,y} f(x, y)}{2}.$$

The convergence rate of the Picard process is independent of the mesh size as is estimated below.

Since $\min_{x,y} f(x, y) \leq f(x, y) \leq \max_{x,y} f(x, y)$, it follows that

$$\begin{aligned} |\alpha - f(x, y)| &\leq \max\{|\alpha - \min_{x,y} f(x, y)|, |\alpha - \max_{x,y} f(x, y)|\} \\ &= \frac{\max_{x,y} f(x, y) - \min_{x,y} f(x, y)}{2} \\ &= \frac{f_M - f_m}{2}, \end{aligned}$$

where $f_m = \min_{x,y} f(x, y)$ and $f_M = \max_{x,y} f(x, y)$. Then the error amplification operator $G = (-\Delta + \alpha)^{-1}(\alpha - f(x, y))$ of the Picard process (3.14) is bounded in L^2 norm by

$$\begin{aligned} \|G\| &\leq \|(-\Delta + \frac{f_m + f_M}{2})^{-1}\| \frac{f_M - f_m}{2} \\ &\leq \|(-\Delta + f_m + \frac{f_M - f_m}{2})^{-1}\| \frac{f_M - f_m}{2}. \end{aligned} \quad (3.15)$$

Let λ_m denote the least eigenvalue of $-\Delta + f_m$, and let $d_M = (f_M - f_m)/2$. Then the L^2 norm of G is bounded by

$$\|G\| \leq (\lambda_m + d_M)^{-1} d_M = 1 - \frac{\lambda_m}{\lambda_m + d_M}. \quad (3.16)$$

Thus the spectral radius of the error amplification matrix G is bounded by $\frac{d_M}{d_M + \lambda_m}$, and the convergence rate r is therefore at least

$$r = -\log \frac{d_M}{d_M + \lambda_m} = \log \frac{d_M + \lambda_m}{d_M}, \quad (3.17)$$

which is independent of the mesh size. Thus if one needs to reduce the error e_0 of an initial guess to the Picard process (3.14) to a given iteration stopping error tolerance e_s , the number of iterations n of the Picard process satisfies

$$\left(\frac{d_M}{\lambda_m + d_M} \right)^n e_0 = e_s,$$

which implies that the iteration number satisfies

$$\text{Iterations} = \frac{\log(e_0/e_s)}{\log \frac{\lambda_m + d_M}{d_M}}. \quad (3.18)$$

With a fourth order spatial discretization, if we choose a fourth order error tolerance $O(h^4)$, then the number of iterations of the Picard process is

$$\text{Iterations} = \frac{4 \log h}{\log \frac{d_M}{\lambda_m + d_M}} = O(-\log h) \quad (3.19)$$

when assuming $e_0 = O(1)$.

3.3.2 High Order FFT Based Solvers

For each Picard iteration (3.14), a Helmholtz equation needs to be solved. We choose the fourth order scheme (3.13) to discretize (3.14), obtaining

$$[(I + \frac{h_x^2}{12}D_{xx} + \frac{h_y^2}{12}D_{yy})\alpha - D_{xx}(I + \frac{h_y^2}{12}D_{yy}) - (I + \frac{h_x^2}{12}D_{xx})D_{yy}]u_{n+1}^{i,j} = s_n^{i,j}, \quad (3.20)$$

where

$$s_n^{i,j} = (I + \frac{h_x^2}{12}\frac{\partial^2}{\partial x^2} + \frac{h_y^2}{12}\frac{\partial^2}{\partial y^2})[(\alpha - f^{i,j})u_n^{i,j} + r^{i,j}]. \quad (3.21)$$

The differential operators $\frac{\partial^2}{\partial x^2}$ and $\frac{\partial^2}{\partial y^2}$ need to be approximated only to a second order accuracy, and different approximation schemes are allowed to accommodate both Neumann and Dirichlet boundary conditions. Group the terms on the left hand side of (3.21) into two groups with one containing D_{yy} and the other without, and we obtain

$$[(I + \frac{h_x^2}{12}D_{xx})\alpha - D_{xx}]u_{n+1}^{i,j} - (I + \alpha\frac{h_y^2}{12} + \frac{h_x^2}{12}D_{xx} + \frac{h_y^2}{12}D_{xx})D_{yy}u_{n+1}^{i,j} = s_n^{i,j}. \quad (3.22)$$

When equipped with a Dirichlet boundary condition, the FFT based Fast Sine Transform (FST) can efficiently tridiagonalize equation (3.22) by diagonalize the submatrices resulted from the finite difference operator D_{yy} and the boundary condition. With a Neumann boundary condition, a Fast Cosine Transform (FCT) can efficiently tridiagonalize equation (3.22) by diagonalize the submatrices resulted from the finite difference operator D_{yy} and the boundary condition. After transform, (3.22) becomes a sequence of tridiagonal equations of the form

$$[(I + \frac{h_x^2}{12}D_{xx})\alpha - D_{xx}]u_{n+1}^{i,j} - \lambda(I + \alpha\frac{h_y^2}{12} + \frac{h_x^2}{12}D_{xx} + \frac{h_y^2}{12}D_{xx})u_{n+1}^{i,j} = Ps_n^{i,j},$$

where λ goes through all the eigenvalues of the submatrices resulted from D_{yy} and the boundary condition, and P is the FST or FCT transform matrix. The

tridiagonal equations can be solved efficiently by a tridiagonal solver and then the solution of the pre-transformed equation can be recovered by an inverse sine transform or an inverse cosine transform. The two FFT based transform operations have a computation complexity of $O(-h_x^{-1}h_y^{-1} \log h_y)$ and the tridiagonal solver has a complexity of $O(h_x^{-1}h_y^{-1})$. Thus each Picard iteration in (3.14) has an $O(-h_x^{-1}h_y^{-1} \log h_y)$ computation cost. Then by (3.19), the computation cost of the entire Picard-FFT solution method (3.14) for equation (3.2) discretized on a grid of mesh size $(h_x, h_y) = (C_1 h, h)$ to stop at error tolerance $e_s(h)$ with a starting error of $e_0(h)$ is bounded by

$$C_2 h^{-2} (\log h) \log \frac{e_s(h)}{e_0(h)} \quad (3.23)$$

for some constant $C_2 > 0$.

3.3.3 Another Picard Process

From the analysis given in Section 3.3.1 for the convergence rate (3.17) of the Picard process (3.14), the spectral radius of the error amplification matrix of (3.14) is bounded by

$$\|(-\Delta + \alpha)^{-1}[\alpha - f(x, y)]\| \leq \|(-\Delta + \alpha)^{-1}\| \cdot \|\alpha - f(x, y)\|.$$

Thus if we replace the constant α by a function $\alpha(x)$ so that the norm $\|\alpha(x) - f(x, y)\|$ can be reduced, then the convergence rate of the Picard process

$$[-\Delta + \alpha(x)]u_{n+1} = [\alpha(x) - f(x, y)]u_n + r, \quad (3.24)$$

might be higher than that of (3.14). In the Picard process (3.14) the equation to be solved for each Picard iteration can still be solved by an efficient FFT method.

With the choice of $\alpha(x) = \frac{\min_y f(x,y) + \max_y f(x,y)}{2}$ for the Picard process (3.24), an analysis similar to that for (3.18) shows that the convergence rate of (3.24) is a constant independent of mesh size $h = (h_x, h_y)$, and the iteration number required for the numerical solution to reach an accuracy of $O(h^4)$ is of $O(-\log h)$. Hence the computation cost of the Picard process (3.24) has an asymptotic computational complexity of $O(h^{-2}(\log h)^2)$, the same as the asymptotic complexity of the previous Picard process (3.14). But since the error amplification matrix $[-\Delta + \alpha(x)]^{-1}[\alpha(x) - f(x, y)]$ of the new Picard process (3.24) is bounded in L^2 norm by

$$\|[-\Delta + \alpha(x)]^{-1}[\alpha(x) - f(x, y)]\| \leq \|[-\Delta + f_m(x) + \frac{f_M(x) - f_m(x)}{2}]^{-1}[\frac{f_M(x) - f_m(x)}{2}]\|, \quad (3.25)$$

where $f_m(x) = \min_y f(x, y)$ and $f_M(x) = \max_y f(x, y)$. Compared with (3.15), the bound given in (3.25) could possibly be smaller. Thus reduced number of iterations is expected.

3.3.4 Numerical Experiments

To test the accuracy and efficiency of the high order Picard-FFT solver, we choose three testing problems with known solutions, and they are:

1. $-\Delta u + (\cos(x+y) + x^2y + 1)u = r$ with $u(x, y) = \sin(2x + 3y)$;
2. $-\Delta u + e^{x-y}u = r$ with $u = (x+y)^{3.5}[\cos(x) - 1]$; and
3. $-\Delta u + (\cos(x-y) + e^{x+y})u = r$ with $u(x, y) = \cos(x+y) + x^2y$.

The testing problem domain is chosen to be the square $[0, 3] \times [0, 3]$, and uniform mesh size $h_x = h_y = 3/N$ is chosen, where N is the number of grid points on each x- and y-dimension. We tested the two Picard-FFT methods on a SUN Ultra-Sparc 10 workstation running operating system SunOS 5.8 with 64-bit arithmetic operations used. The measured iteration number of the two Picard-FFT methods, the execution time in seconds, and the error defined as the maximal difference between the computed solution and the true solution are listed in Tables 3.1 to 3.3 for the three testing problems, with PiFFT1 denoting the solver based on the Picard process (3.14) and PiFFT2 the solver based on the Picard process (3.24).

Table 3.1: $-\Delta u + (\cos(x+y) + x^2y + 1)u = r$ on $[0, 3] \times [0, 3]$ with $u = \sin(2x + 3y)$

Method	N	16	32	64	128	256	512
PiFFT1	Iterations	15	23	31	39	48	56
	Time(sec.)	0.01	0.10	0.51	2.62	14.1	70.0
	Error	1.6e-04	1.1e-05	8.6e-07	6.7e-08	3.5e-09	2.5e-10
	Order		3.9	3.8	3.7	3.9	3.9
PiFFT2	Iterations	7	9	11	14	16	18
	Time(sec.)	0.01	0.06	0.32	1.62	7.96	38.8
	Error	2.2e-05	1.4e-06	1.4e-07	5.0e-09	5.1e-10	5.3e-11
	Order		4.0	3.6	4.0	3.8	3.7

In the tables, we use a metric *Order* [69] to indicate the numerical order of a solver, which is calculated as follows:

$$\text{Order}(m, n) = \frac{\log \frac{\text{Error}(m)}{\text{Error}(n)}}{-\log \frac{m}{n}}.$$

Table 3.2: $-\Delta u + e^{x-y}u = r$ on $[0, 3] \times [0, 3]$ with $u = (x + y)^{3.5}(\cos(x) - 1)$

Method	N	16	32	64	128	256	512
PiFFT1	Iterations	31	44	56	67	77	87
	Time(sec.)	0.03	0.18	0.92	4.55	22.4	105.6
	Error	9.6e-04	6.1e-05	3.8e-06	2.4e-07	1.5e-08	1.4e-09
	Order		4.0	4.0	4.0	4.0	3.9
PiFFT2	Iterations	17	20	22	25	28	31
	Time(sec.)	0.03	0.14	0.61	2.85	13.9	63.6
	Error	9.1e-04	5.7e-05	3.7e-06	2.3e-07	1.4e-08	9.8e-10
	Order		4.0	4.0	4.0	4.0	4.0

The definition of this metric is based on the observation that for a numerical method of order p , the error will decrease at a rate of $\left(\frac{1}{2}\right)^p$ when a uniformly spaced grid doubles its grid points in each direction. The log plot of error against grid size (or mesh size) is usually used to measure the order of a numerical method. The metric *Order* used here gives the value of the slope of the log plot of the error vs. grid size between the smallest testing grid size and the indicated grid size. Since the slope of a curve is difficult to be exactly visually determined, the metric *Order* is a clearer quantitative indication of the order of a numerical method. The values of the order calculated using the measured data of error indicate that the high order Picard-FFT solvers is approximately of fourth order accuracy.

Table 3.3: $-\Delta u + (\cos(x-y) + e^{x+y})u = r$ on $[0, 3] \times [0, 3]$ with $u = \cos(x+y) + x^2y$

Method	N	16	32	64	128	256	512
PiFFT1	Iterations	77	128	181	232	282	330
	Time(sec.)	0.08	0.52	2.98	15.4	81.7	439.1
	Error	1.3e-03	1.0e-04	6.7e-06	4.5e-07	2.8e-08	1.8e-09
	Order		3.7	3.8	3.8	3.9	3.9
PiFFT2	Iterations	34	45	55	65	76	86
	Time(sec.)	0.05	0.31	1.53	7.48	37.3	177.4
	Error	3.9e-04	2.1e-05	1.4e-06	9.6e-08	4.9e-09	3.3e-10
	Order		4.2	4.1	4.0	4.1	4.0

3.4 The ADI Method for Separable Problems

The two Picard processes (3.14) and (3.24) for the generalized Helmholtz equation

$$-\Delta u + f(x, y)u = r(x, y)$$

are sensitive to the range of the values of the function $f(x, y)$. When the range is large, the two processes converge slowly as shown from testing data in Table 3.5 for the problem

$$-\Delta u + (e^{3x} + e^{3y})u = r$$

on the domain $[0, 3] \times [0, 3]$, which is solved using the Picard process (3.24). In this section, we propose an ADI method for the separable generalized Helmholtz equation

$$-\Delta u + [f_1(x) + f_2(y)]u = r(x, y).$$

3.4.1 A High Order ADI Solver

We apply the fourth order finite difference scheme (3.11) to the equation (3.24), obtaining

$$[f_1^i + f_2^j - (I + \frac{h_x^2}{12}D_{xx})^{-1}D_{xx} - (I + \frac{h_y^2}{12}D_{yy})^{-1}D_{yy}]u_{n+1}^{i,j} = r^{i,j},$$

Then we assemble the above equations at all grid points into a matrix in tensor product form ([45]) with boundary condition incorporated, and obtain

$$[(F_1 - D_{m-1}) \otimes I_{n-1}] U + [I_{m-1} \otimes (F_2 - D_{n-1})] U = R, \quad (3.26)$$

where D_n is an $n \times n$ matrix given by

$$D_n = \frac{1}{h^2} \begin{pmatrix} \frac{5}{6} & \frac{1}{12} & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{12} & \frac{5}{6} & \frac{1}{12} & \cdots & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & \frac{1}{12} & \frac{5}{6} & \frac{1}{12} \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{12} & \frac{5}{6} \end{pmatrix}^{-1} \begin{pmatrix} -2 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & -2 & 1 & \cdots & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -2 \end{pmatrix}, \quad (3.27)$$

I_n denotes the identity matrix in an n dimensional space \mathbf{R}^n , \otimes is the tensor product notation [45], F_1 and F_2 are diagonal matrices corresponding to the functions $f_1(x)$ and $f_2(y)$ respectively, U the solution vector, and R is the vector corresponding to the right had side of (3.24) with boundary conditions incorporated.

Birkhoff and Varga showed in [4] that when the directional components $(F_1 - D_{m-1}) \otimes I_{n-1}$ and $I_{m-1} \otimes (F_2 - D_{n-1})$ of the discretized equation (3.26) are symmetric, positive definite and commutative, the Peaceman-Rachford [50] ADI method has a convergence rate of $O((\log h)^{-1})$. Thus these positive definite and commutative conditions are essential for a high convergence rate. The commutativity condition

is obvious with the tensor product notation. The positive definite condition also holds as we shall see below.

Remark 3.1 *The matrices $(F_1 - D_{m-1}) \otimes I_{n-1}$ and $I_{m-1} \otimes (F_2 - D_{n-1})$ are symmetric and positive definite.*

Proof: We prove the above statement only for the matrix $(F_1 - D_{m-1}) \otimes I_{n-1}$. The proof for the matrix $I_{m-1} \otimes (F_2 - D_{n-1})$ is similar and thus omitted.

Since F_1 is diagonal and positive definite, it suffices to show that $-D_{m-1}$ is positive definite. Now let P_{m-1} denote the matrix

$$P_{m-1} = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,m-2} & p_{1,m-1} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,m-2} & p_{2,m-1} \\ p_{3,1} & p_{3,2} & \cdots & p_{3,m-2} & p_{3,m-1} \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ p_{m-1,1} & p_{m-1,2} & \cdots & p_{m-1,m-2} & p_{m-1,m-1} \end{pmatrix},$$

where $p_{i,j} = \sin(\frac{ij\pi}{m})$ for $i, j = 1, 2, \dots, m-1$. A straightforward calculation shows that $P_{m-1}^{-1} D_{m-1} P_{m-1} = \Lambda$, where Λ is a diagonal matrix given by

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & \lambda_{m-2} & 0 \\ 0 & 0 & 0 & \cdots & 0 & \lambda_{m-1} \end{pmatrix},$$

and $\lambda_i = \frac{12 \cos(\frac{i\pi}{m}) - 12}{5 + \cos(\frac{i\pi}{m})} h^{-2} < 0$, for $i = 1, 2, \dots, m-1$. For matrix P_{m-1} , we have that

$P_{m-1}^{-1} = \frac{2}{m} P_{m-1}^T$. Thus, $P_{m-1}^T D_{m-1} P_{m-1} = \frac{m}{2} \Lambda$, and $D_{m-1} = \frac{2}{m} P_{m-1} \Lambda P_{m-1}^T$. Therefore

$-D_{m-1}$ is symmetric and positive definite. □

Applying the ADI method of Peaceman and Rachford to the discrete equation (3.26) as in the following.

$$\begin{cases} (\rho_n + A_x)U^{n+\frac{1}{2}} &= (\rho_n - A_y)U^n + R, \\ (\rho_n + A_y)U^{n+1} &= (\rho_n - A_x)U^{n+\frac{1}{2}} + R, \end{cases} \quad (3.28)$$

where A_x and A_y respectively denote the two terms $(F_1 - D_{m-1}) \otimes I_{n-1}$ and $I_{m-1} \otimes (F_2 - D_{n-1})$ on the left hand side of equation (3.26). Since A_x and A_y are symmetric, positive definite and commutative, the ADI method (3.28) with Wachspress parameters [60, 61] has a convergence rate of $O(-(\log h)^{-1})$ for $h = \max\{h_x, h_y\}$. Thus, denoting the convergence rate on a grid with mesh size h by $r(h)$, we can assume that

$$r(h) \approx -C_0(\log h)^{-1} \quad (3.29)$$

for some constant $C_0 > 0$.

If the initial error of the guess solution is $e_o(h)$ and the iteration-stopping error tolerance is $e_s(h)$, then by the definition of the convergence rate [42], the number of iterations equals $\frac{\log(e_o(h)/e_s(h))}{r(h)}$. With estimate (3.29), we arrive at

$$\text{Iterations} \approx C_1(\log h) \log \frac{e_s(h)}{e_o(h)}, \quad (3.30)$$

where $C_1 = \frac{1}{C_0}$. Each ADI iteration consists of solving four tridiagonal systems, two for evaluating the right hand sides of (3.28) necessitated by using the fourth order finite difference (3.11), and two for inverting $(\rho_n + A_1)$ and $(\rho_n + A_2)$ on the left hand sides of (3.28). Solving the four tridiagonal systems is the major computation cost of each ADI iteration. Hence we can assume that the computation cost per iteration is bounded by $C'_2 h^{-2}$ for some constant $C'_2 > 0$. Then by (3.30), the

computation cost for the ADI method (3.28) to stop at tolerance $e_s(h)$ with a starting error of $e_o(h)$ is

$$C_2 h^{-2} (\log h) \log \frac{e_s(h)}{e_o(h)}, \quad (3.31)$$

where $C_2 = C_1 C_2'$. For instance, if the error tolerance is chosen to be $e_s(h) = h^4$ and the initial error is assumed to be $e_o(h) = 1$, then the ADI has a complexity of $O(h^{-2}(\log h)^2)$.

3.4.2 Numerical Experiments

Three equations with known solutions have been chosen to test the accuracy and efficiency of the High-order ADI (HADI). The three problems are

- (i) $-\Delta u + (x^2 + \sin(y) + y)u = r$ on $[0, 3] \times [0, 3]$ with $u(x, y) = \cos(2x + 3y)$;
- (ii) $-\Delta u + (e^{3x} + e^{3y})u = r$ on $[0, 3] \times [0, 3]$ with $u(x, y) = (x+y)^{3.5}(\cos(x)-1)$; and
- (iii) $-\Delta u + (x^3 + x \cos(x) + e^y)u = r$ on $[0, 3] \times [0, 3]$ with $u(x, y) = \sin(x+y)$.

The domain of the testing problems are chosen to be the square $[0, 3] \times [0, 3]$ with the same uniform mesh size h on each dimension. $N = 3/h$ is the number of grid points on each x- and y-dimension. In the tests, the iteration stopping criterion is chosen to be the difference between the approximate solution at two consecutive iterations, which is set to $h^4/10$ for the first and third problems and set to $0.5h^4$ for the second problem on a grid of mesh size h . We choose different error tolerance because the solution of the second problem is less smooth than the first and third and thus have a larger discretization error. We solved the three problems with the proposed HADI method on an SUN UltraSparc 10 workstation, and the testing

results are listed in Tables 3.4–3.6. For comparison, we also have solved the two problems using the Picard method (3.24) denoted as PiFFT with a fourth order FFT solver at each Picard iteration. The data from the PiFFT solver are listed together in the tables with the data from the HADI solver.

Table 3.4: $-\Delta u + (x^2 + \sin(y) + y)u = r$ on $[0, 3] \times [0, 3]$ with $u = \cos(2x + 3y)$

Method	N	16	32	64	128	256	512
HADI	Iterations	9	15	19	25	33	43
	Time(sec.)	0.00	0.03	0.16	0.89	5.32	33.8
	Error	2.6e-04	1.8e-05	1.3e-06	6.5e-08	3.9e-09	2.6e-10
	Order		3.9	3.8	4.0	4.0	4.0
PiFFT	Iterations	7	9	12	14	16	19
	Time(sec.)	0.01	0.05	0.28	1.40	6.85	34.0
	Error	2.8e-05	2.4e-06	6.7e-08	7.0e-09	7.8e-10	3.2e-11
	Order		3.5	4.4	4.0	3.8	3.9

Table 3.5: $-\Delta u + (e^{3x} + e^{3y})u = r$ on $[0, 3] \times [0, 3]$ with $u = (x + y)^{3.5}[\cos(x) - 1]$

Method	N	16	32	64	128	256	512
HADI	Iterations	13	19	25	29	36	41
	Time(sec.)	0.01	0.05	0.22	1.02	6.92	32.1
	Error	1.5e-02	8.2e-04	3.6e-05	6.0e-06	1.6e-07	4.1e-08
	Order		4.2	4.4	3.8	4.1	4.0
PiFFT	Iteration	683	1000*	1000	1000	1000	1000
	Time(sec.)	0.92	—	—	—	—	—
	Error	1.9e-02	—	—	—	—	—
	Order		—	—	—	—	—

* The PiFFT method fails to reach the given error tolerance within 1000 Picard iterations.

Table 3.6: $-\Delta u + (x^3 + x \cos(x) + e^y)u = r$ on $[0, 3] \times [0, 3]$ with $u = \sin(x+y)$

Method	N	16	32	64	128	256	512
HADI	Iterations	8	14	18	25	32	41
	Time(sec.)	0.01	0.03	0.15	0.89	5.14	32.2
	Error	7.5e-05	4.8e-07	2.9e-07	8.1e-09	1.6e-09	9.8e-11
	Order		7.3	4.0	4.4	3.9	3.9
PiFFT	Iterations	15	20	24	29	34	38
	Time(sec.)	0.02	0.12	0.58	2.88	14.5	67.7
	Error	1.3e-04	6.4e-06	5.5e-07	2.6e-08	1.3e-09	1.1e-10
	Order		4.3	3.9	4.1	4.2	4.0

In the tables, the *Iteration* rows respectively list the numbers of ADI iterations of the HADI method and the numbers of Picard iterations for the Picard-FFT method for the problems solved on the grid of the indicated sizes. The *Time* rows give the CPU time taken to solve the problems, and the *Error* rows show the maximal errors of the numerical solutions for the three problems solved on the grid of the indicated size. The running results supports that the high order ADI method is both accurate and efficient.

The Picard process fails to reach the prescribed error tolerance within 1,000 iterations for the second problem, because the values of $f_1(x) + f_2(y)$ (which is $e^{3x} + e^{3y}$ for problem 2 on square $[0, 3] \times [0, 3]$) cover a very large range from 2 to above 16, 206. The Picard process (3.14) converges slowly when $f_1(x) + f_2(y)$ has a large range of values. In the case of the first or third problem, $f_1(x) + f_2(y)$ has a much smaller range of values, and Picard method converges to adequate accuracy within a reasonable number of iterations for each grid size tested.

3.5 Multilevel Acceleration of Iterative Methods

When the elliptic equation (3.2) is resulted from temporal discretization of a parabolic problem, we can obtain a good initial guess to the solution of the elliptic equation. If (3.2) is a stand-alone elliptic equation, an iterative solution method can be accelerated by a multilevel process described below.

3.5.1 The Multilevel Acceleration Algorithm

Suppose that we have a iterative method

$$u_{n+1} = Gu_n \tag{3.32}$$

for the linear system

$$A_h u_h = R_h \tag{3.33}$$

obtained from the discretization of the elliptic equation

$$Lu = r. \tag{3.34}$$

Suppose that the iterative method has a convergence rate $r(h)$ on a grid of mesh size h , and that the computation cost for each iteration of (3.32) is $\phi(h)$. Then by the definition of convergence rate [42], the number of iterations needed for the numerical solution to reach an accuracy of $e_s(h)$ from an initial guess of error $e_0(h)$ satisfies

$$\text{Iterations} = \frac{\log \frac{e_0(h)}{e_s(h)}}{r(h)}. \tag{3.35}$$

Then the computation cost of the iterative method to reduce the error from $e_0(h)$ to $e_s(h)$ is

$$\frac{\phi(h)}{r(h)} \log\left(\frac{e_0(h)}{e_s(h)}\right). \tag{3.36}$$

When $e_0(h) = 1$ and the iteration stopping tolerance $e_s(h)$ is chosen to be of the same order as the discretization error, say $O(h^p)$, the total computation cost is of

$$O\left(-\frac{\phi(h)}{r(h)} \log h\right). \quad (3.37)$$

Our multilevel acceleration algorithm requires the following data structure in addition to the existing data structure used for the original iterative method (3.32). The original grid the iterative method uses is designated as the finest grid. We choose every other grid point in each direction to form the next coarser level grid, a subset of the original grid with double mesh sizes, and no extra data structure is needed for this grid level except an integer to indicate the grid level. By the same procedure, we designate a sequence of grids, each a subset of the immediate finer level grid with double mesh sizes. All of them need no extra data structure other than an index of one single integer number. The number of grid levels is chosen to be of $O(-\log_2 h)$, namely, the coarsest grid has a size close to 2×2 .

Our multilevel procedure starts from the coarsest grid level. On all the grids, applying the same discretization scheme to the elliptic equation (3.34) the iterative method (3.32) is solving, we obtain discrete equations with the same matrix form (as the iterative method (3.32) is applied to) but of different mesh sizes and different problem sizes. At each level, the iterative method (3.32) is then employed to solve the discrete equation, with the solution of the equation at the immediate coarser grid level interpolated to the current grid level as the initial guess. The interpolation method is chosen to have an interpolation accuracy of the same order as the discretization order, and the iterative process (3.32) stopping tolerance e_s

at a grid level is also chosen to be of the same order as that of the discretization at that level. This multilevel algorithm solves the equation from the coarsest grid level up to the finest, and the whole multilevel process ends after the iterative method (3.32) finishes its iterations on the finest level.

Unlike classic multigrid methods, the proposed solver is a one-way multilevel method, of both algorithmic and data-structural simplicity. Its implementation needs only one subroutine and one loop more than the single grid iterative method (3.32) — an interpolation subroutine and a loop that goes through all grid levels. Such simplicity provides great potential for its applicability in complex systems and in combination with parallel and/or domain decomposition methods.

3.5.2 Efficiency Analysis

While classic multilevel methods are mainly utilizing the smoothing effect [63] of the single grid solvers (or relaxation schemes in the language of multilevel methods) to achieve computation reduction, our method relies on the initial error reduction via interpolation from coarser grids to reduce the iteration numbers on finer grids. To analyze the computation complexity of this multilevel acceleration algorithm more closely, we denote the true solution of the original differential equation (3.34) by u , denote the exact numerical solution of the discrete equation (3.33) on a grid with mesh size h by u_h , denote the approximate numerical solution of the discrete equation at the same grid level by U_h , and denote the initial guess solution interpolated from the immediate coarser grid to the current grid by \hat{U}_h .

Suppose that the discretized equation (3.33) has an accuracy of order $O(h^p)$,

which means that there exists a constant $C_3 > 0$ such that

$$\|u_h - u\| \leq C_3 h^p.$$

The iteration stopping error tolerance $e_s(h)$ is chosen to be of the same order as that of the discretization. Thus, we have that

$$\|U_h - u_h\| \leq C_4 h^p$$

for some constant $C_4 > 0$. Therefore,

$$\|U_h - u\| \leq \|U_h - u_h\| + \|u_h - u\| \leq (C_3 + C_4)h^p.$$

This inequality holds for every grid level. Hence for the immediate coarser grid, we have that

$$\|U_{2h} - u\| \leq (C_3 + C_4)(2h)^p = 2^p(C_3 + C_4)h^p.$$

The solution at the immediate coarser grid is interpolated to the current grid as the initial guess solution in such a way that it maintains p -th order accuracy, i.e. this initial guess \hat{U}_h satisfies

$$\|\hat{U}_h - U_{2h}\| \leq C_5 h^p$$

for some constant $C_5 > 0$. By definition, the error of the initial guess is $\|\hat{U}_h - u_h\|$,

so

$$\begin{aligned} e_o(h) &= \|\hat{U}_h - u_h\| \\ &\leq \|\hat{U}_h - U_{2h}\| + \|U_{2h} - u\| + \|u - u_h\| \\ &\leq C_5 h^p + 2^p(C_3 + C_4)h^p + C_3 h^p \\ &= [(2^p + 1)C_3 + 2^p C_4 + C_5]h^p. \end{aligned}$$

Then the ratio of the initial error to the iteration stopping error tolerance satisfies

$$e_o(h) : e_s(h) \leq \frac{(2^p+1)C_3 + 2^p C_4 + C_5}{C_4}. \quad (3.38)$$

Then by (3.35), the number of iterations needed at grid level with mesh sizes h is

$$\text{Iterations} \leq \frac{\log \frac{(2^p+1)C_3 + 2^p C_4 + C_5}{C_4}}{r(h)} = \frac{C_6}{r(h)}, \quad (3.39)$$

where the positive constant $C_6 = \log \frac{(2^p+1)C_3 + 2^p C_4 + C_5}{C_4}$. Since the computation cost of each iteration in (3.32) is $\phi(h)$ on a grid of mesh size h , the computation cost of the multilevel iterative method at grid level with mesh size h is the number of iterations times the computation cost per iteration, which is $C_6 \frac{\phi(h)}{r(h)}$. The interpolation cost is proportional to the number of grid points, and thus can be assumed to be $C_7 h^{-2}$ for a 2-D problem or $C_7 h^{-3}$ for a 3-D problem. Then the total computation cost for a k dimensional problem on a grid of mesh size h is $C_7 h^{-k} + C_6 \frac{\phi(h)}{r(h)}$. Since for a k -dimensional problem, the per-iteration computation cost $\phi(h)$ is at least $O(h^{-k})$ and the convergence rate is at most $O(1)$, which implies that $C_7 h^{-k} \leq C_8 \cdot C_6 \frac{\phi(h)}{r(h)}$. Thus the computation cost of this multilevel algorithm at a grid level of mesh size h is bounded by

$$C_9 \frac{r(h)}{\phi(h)} \quad (3.40)$$

for some constant $C_9 > 0$.

The above estimates of the iteration number and computation cost on a grid of mesh h are valid for all levels except the coarsest grid level, since on the coarsest level, the initial guess is not obtained from interpolation. Thus the ratio of initial error to error tolerance may not necessarily satisfy (3.38), and hence the iterative solver (3.32) could possibly takes more iterations than (3.39) on the coarsest grid.

However, since the number of levels is chosen in such a way that the coarsest grid has very few grid points as described in the second paragraph of this section, the computation cost on the coarsest grid is of $O(1)$ and thus negligible.

Add up the computation costs at all grid levels, we obtain

$$C_9 \left[\frac{\phi(h)}{r(h)} + \frac{\phi(2h)}{r(2h)} + \frac{\phi(4h)}{r(4h)} + \dots \right] \leq \frac{C_9}{r(h)} [\phi(h) + \phi(2h) + \phi(4h) + \dots] \quad (3.41)$$

for the total computation cost. For a k -dimensional problem, the cost function $\phi(h)$ for each iteration increases at least in the order of $O(h^{-k})$ as h decreases, which means that

$$\phi(2h) \leq 2^{-k} \phi(h),$$

which, together with (3.41), implies that the total computation cost of the multi-level iterative algorithm is bounded by

$$\frac{C_9}{r(h)} \left[\phi(h) + 2^{-k} \phi(h) + 4^{-k} \phi(h) + \dots \right] \leq \frac{2^k}{2^k - 1} C_9 \frac{\phi(h)}{r(h)} \leq 1.4 C_9 \frac{\phi(h)}{r(h)}, \quad (3.42)$$

an improvement of $O(-\log h)$ when compared with the single grid iterative solver (3.37).

For the Picard process (3.14) with the FFT-based Helmholtz solver for each Picard iteration, the per-iteration cost is $\phi(h) = O(-h^{-2} \log h)$ and the convergence rate $r(h)$ is bounded by a constant independent of mesh size h . Thus when accelerated by the multilevel process, the multilevel Picard-FFT solver has a computation complexity of $O(-h^{-2} \log h)$ according to (3.42). For the ADI solver (3.28), the per-iteration cost is $\phi(h) = O(h^{-2})$ and the convergence rate is $r(h) = O(-\log h)$. Thus the multilevel ADI method has a complexity of $O(-h^{-2} \log h)$ by (3.42).

3.5.3 Numerical Testings

To test the multilevel acceleration algorithm, we apply the algorithm to the high order FFT-based Picard method (3.14) and the high order ADI method (3.28), obtaining two multilevel methods denoted by MPiFFT and MADI respectively. We have used the MPiFFT algorithm to solve the same problems the single grid PiFFT solved in Section 3.3.4, and they are

(i) $-\Delta u + (\cos(x+y) + x^2y + 1)u = r$ with $u(x, y) = \sin(2x + 3y)$;

(ii) $-\Delta u + e^{x-y}u = r$ with $u = (x+y)^{3.5}[\cos(x) - 1]$; and

(iii) $-\Delta u + (\cos(x-y) + e^{x+y})u = r$ with $u(x, y) = \cos(x + y) + x^2y$.

We also have used the MADI to solve the same problems the single grid high order ADI method have solved in Section 3.4.2 and they are

(iv) $-\Delta u + (x^2 + \sin(y) + y)u = r$ on $[0, 3] \times [0, 3]$ with $u(x, y) = \cos(2x + 3y)$;

(v) $-\Delta u + (e^{3x} + e^{3y})u = r$ on $[0, 3] \times [0, 3]$ with $u(x, y) = (x+y)^{3.5}(\cos(x) - 1)$; and

(vi) $-\Delta u + (x^3 + x \cos(x) + e^y)u = r$ on $[0, 3] \times [0, 3]$ with $u(x, y) = \sin(x + y)$.

Experimental tests have been conducted on a SUN UltraSparc 10 workstation, and the measured number of iterations, execution time, and numerical errors of the multilevel methods are listed in Tables 3.7–3.12 together with the numerical results obtained from the corresponding single grid solvers.

In the tests, the coarsest grid of the multilevel methods is chosen to be of size 8×8 , and the iteration stopping criterion for the two multilevel methods is chosen to be the difference between the approximate solution at two consecutive iterations,

Table 3.7: $-\Delta u + (\cos(x+y) + x^2y + 1)u = r$ on $[0, 3] \times [0, 3]$ with $u = \sin(2x + 3y)$

Method	N	16	32	64	128	256	512
PiFFT1	Iterations	15	23	31	39	48	56
	Time(sec.)	0.01	0.10	0.51	2.62	14.1	70.0
	Error	1.6e-04	1.1e-05	8.6e-07	6.7e-08	3.5e-09	2.5e-10
MPiFFT	Iterations	6	7	8	8	8	9
	Time(sec.)	0.01	0.03	0.17	0.68	3.00	14.4
	Error	1.3e-04	1.0e-05	6.0e-07	4.4e-08	3.2e-09	1.6e-10

Table 3.8: $-\Delta u + e^{x-y}u = r$ on $[0, 3] \times [0, 3]$ with $u = (x+y)^{3.5}(\cos(x) - 1)$

Method	N	16	32	64	128	256	512
PiFFT1	Iterations	31	44	56	67	77	87
	Time(sec.)	0.03	0.18	0.92	4.55	22.4	105.6
	Error	9.6e-04	6.1e-05	3.8e-06	2.4e-07	1.5e-08	1.4e-09
MPiFFT	Iterations	8	9	10	10	10	11
	Time(sec.)	0.01	0.05	0.21	0.86	3.74	17.7
	Error	9.4e-04	5.9e-05	3.6e-06	2.3e-07	1.4e-08	1.3e-09

which is set to $0.1h^4$ on a grid of mesh size h . The testing is conducted on the square domain $[0, 3] \times [0, 3]$ with the same uniform mesh size h on each dimension. $N = 3/h$ is the number of grid points on each x- and y-dimension.

Tables 3.7 to 3.9 contain the experimental results of the multilevel Picard-FFT solver (MPiFFT) as compared with the single grid Picard-FFT solver (PiFFT1). For the MPiFFT solver, the *Iteration* row lists the Picard iterations on the finest grid level when the indicated grid is the finest grid. Tables 3.10 to 3.12 present the

Table 3.9: $-\Delta u + (\cos(x-y) + e^{x+y})u = r$ on $[0, 3] \times [0, 3]$ with $u = \cos(x+y) + x^2y$

Method	N	16	32	64	128	256	512
PiFFT1	Iterations	77	128	181	232	282	330
	Time(sec.)	0.08	0.52	2.98	15.4	81.7	439.1
	Error	1.3e-03	1.0e-04	6.7e-06	4.5e-07	2.8e-08	1.8e-09
MPiFFT	Iterations	26	35	40	42	44	45
	Time(sec.)	0.03	0.16	0.82	3.47	16.0	72.1
	Error	1.3e-03	9.7e-05	6.7e-06	4.5e-07	2.9e-08	1.8e-09

Table 3.10: $-\Delta u + (x^2 + \sin(y) + y)u = r$ on $[0, 3] \times [0, 3]$ with $u = \cos(2x+3y)$

Method	N	16	32	64	128	256	512
HADI	Iterations	9	15	19	25	33	43
	Time(sec.)	0.00	0.03	0.16	0.89	5.32	33.8
	Error	2.6e-04	1.8e-05	1.3e-06	6.5e-08	3.9e-09	2.6e-10
MADI	Iterations	5	6	7	8	9	9
	Time(sec.)	0.01	0.02	0.08	0.37	1.92	9.52
	Error	3.2e-04	2.0e-05	1.2e-06	7.4e-08	4.5e-09	3.5e-10

experimental results of the multilevel ADI (MADI) method for the three testing problems as compared with the single grid high order ADI solver (HADI). For the MADI method, the row *Iteration* lists the number of ADI iterations on the indicated grid when it is the finest grid level. The row starting with *Time* give the CPU time taken to solve the problems, and the *Error* rows show the maximal difference between the computed numerical solution and the true solution on the grid of the indicated size. The testing results show that the iteration numbers of the multilevel

Table 3.11: $-\Delta u + (e^{3x} + e^{3y})u = r$ on $[0, 3] \times [0, 3]$ with $u = (x+y)^{3.5}[\cos(x) - 1]$

Method	N	16	32	64	128	256	512
HADI	Iterations	13	19	25	29	36	49
	Time(sec.)	0.01	0.05	0.22	1.02	6.92	38.4
	Error	1.5e-02	8.2e-04	3.6e-05	6.0e-06	1.6e-07	3.6e-10
MADI	Iterations	4	7	6	8	8	9
	Time(sec.)	0.00	0.01	0.06	0.37	1.75	9.3
	Error	9.0e-04	9.4e-06	5.4e-07	3.0e-08	4.3e-09	3.0e-10

Table 3.12: $-\Delta u + (x^3 + x \cos(x) + e^y)u = r$ on $[0, 3] \times [0, 3]$ with $u = \sin(x+y)$

Method	N	16	32	64	128	256	512
HADI	Iterations	8	14	18	25	32	41
	Time(sec.)	0.01	0.03	0.15	0.89	5.14	32.2
	Error	7.5e-05	4.8e-07	2.9e-07	8.1e-09	1.6e-09	9.8e-11
MADI	Iterations	4	4	4	7	4	8
	Time(sec.)	0.00	0.01	0.04	0.30	1.02	7.50
	Error	4.3e-05	1.0e-06	5.0e-07	2.1e-09	2.1e-09	1.5e-11

methods increase much more slowly than those of the single grid methods, and the execution time the multilevel solvers have taken to solve the problems to an adequate accuracy are also much shorter, thus experimentally verifying the multilevel algorithm as an efficient acceleration process that maintains the accuracy of single grid solvers.

4 Experimental Results

In this chapter, we present some numerical results obtained by applying to parabolic problems the stabilized explicit-implicit domain decomposition algorithms proposed in Chapter 2. Four different types of problems have been chosen to test the proposed domain decomposition algorithms. They are a problem with a symmetric and negative definite spatial operator, a problem with a symmetric and indefinite spatial operator, several problems with nonsymmetric spatial operators, and an unstable problem with a non-symmetric and indefinite spatial operator. In the experiments, stability and scalability of the SEITM algorithms are carefully examined, together with the accuracy of a spatial high order discretization.

4.1 Stability Testing

4.1.1 The Heat Equation

In this section, we apply the SEITM algorithms to the two dimensional heat equation

$$\begin{cases} \frac{\partial u(t,x,y)}{\partial t} = (\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})u, & x \in \Omega, t \geq 0 \\ u(t, x, y) = u_b(t, x, y), & x \in \partial\Omega, t \geq 0, \\ u(0, x, y) = u^0(x, y), & x \in \Omega \end{cases} \quad (4.1)$$

on the rectangular domain $\Omega = [0, 3] \times [0, 1]$. The heat equation has a “good” spatial operator — the Laplacian which is symmetric, and negative definite (and hence dissipative). In the numerical experiment, we partition the domain into $m+1$ intervals of length h in the x-dimension, and partition the y-dimension into $n+1$ intervals of the same mesh size h as in x-dimension. On this regularly structured

grid of size $m \times n$, we use the second order central finite difference

$$\frac{f_{i-1} - 2f_i + f_{i+1}}{h^2} = f_i'' \quad (4.2)$$

for the discretization of the Laplace operator, resulting in the following discrete Laplace operator of size $(mn) \times (mn)$

$$\Delta_h = h^{-2} \begin{pmatrix} D_n & 2I_n & 0 & \cdot & 0 & 0 & 0 \\ 2I_n & D_n & 2I_n & \cdot & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & 2I_n & D_n & 2I_n \\ 0 & 0 & 0 & \cdot & 0 & 2I_n & D_n \end{pmatrix}, \quad (4.3)$$

where I_n is the identity matrix in the n -dimensional space \mathbf{R}^n , and D_n is a $n \times n$ matrix given by

$$D_n = \begin{pmatrix} -4 & 1 & 0 & \cdot & 0 & 0 & 0 \\ 1 & -4 & 1 & \cdot & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & 1 & -4 & 1 \\ 0 & 0 & 0 & \cdot & 0 & 1 & -4 \end{pmatrix}.$$

Then the spatially discretized heat equation has the form $\frac{d}{dt}u_h = \Delta_h u_h + b_h(t)$, where $b_h(t)$ is resulted from the boundary conditions.

We solved the spatially discretized heat equation by the SEITM1 and SEITM2 algorithms presented in Chapter 2, with the domain divided into three equal size squares as in Figure 4. The time intervals chosen for simulation are the unit interval $[0, 1]$. We choose a spatial mesh size of $h = 1/64$, and have used several different time discretization sizes Δt . The experiments were carried out on an NCSA Origin 2000 machine with a maximum of 256 nodes, each of 250 MHz, running IRIX 6.5.9

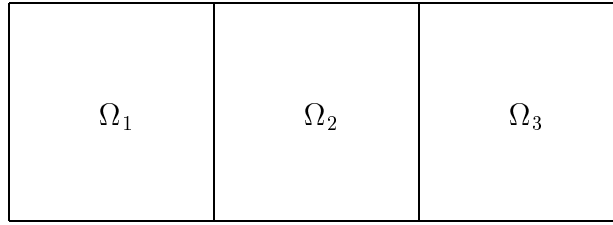


Figure 4

operating system with 64-bit arithmetic operation chosen. The measured errors of numerical solutions at time $t = 1$ are listed in Table 4.1 for the indicated temporal discretization sizes.

Table 4.1: $u_t = \Delta u$ with $u(t, x, y) = e^{-2t} \cos(x+y)$

Δt	1/50	1/100	1/200	1/400	1/800
SEITM1	4.2e-03	8.9e-04	1.5e-04	5.3e-05	3.1e-05
SEITM2	3.8e-04	2.2e-04	1.2e-04	6.8e-05	3.6e-05
SEITM3	4.8e-03	1.1e-03	2.2e-04	3.7e-05	1.4e-05
BEuler	6.1e-04	3.0e-04	1.5e-04	7.6e-05	3.8e-05
EITM1	∞	∞	∞	∞	∞
EITM2	3.9e+54	2.4e+98	∞	∞	∞
EITM3	∞	∞	∞	∞	∞

The table lists the maximal error of the solution at $t = 1$.

The symbol ∞ denotes an error larger than $1.0e + 100$.

The spatial domain is $[0, 3] \times [0, 1]$ with a mesh size $h = 1/64$.

Since the SEITM algorithms are stabilized EITM algorithms, for stability comparison we also solved the heat equation problem using the EITM algorithms — the SEITM algorithms without the stabilization. On the other hand, since the backward Euler (listed as BEuler in the tables) method is the most stable method due to the Widder’s theorem [2], it can be considered as the benchmark for stability. We

solved the heat equation using the backward Euler method on the non-partitioned entire domain by one processor, and the measured errors of the solutions computed by the BEuler method are also listed in Table 4.1. It is well known that for an unconditionally stable method, the simulation error remains small even when the time step size Δt is large relative to the spatial mesh size [41, 42]. As indicated by the experimental results in Table 4.1, the errors of the SEITM algorithms remain relatively small when the time step size Δt is large, and they are almost as small as those of the backward Euler method, experimentally supporting the stability analysis given in Chapter 2.

4.1.2 An Unstable Diffusion Problem

In this section, we test the SEITM algorithms on the problem

$$\begin{cases} \frac{\partial u(t,x,y)}{\partial t} = \Delta u + 3u, & x \in \Omega, t \geq 0 \\ u(t, x, y) = 0, & x \in \partial\Omega, t \geq 0, \\ u(0, x, y) = u^0(x, y), & x \in \Omega \end{cases} \quad (4.4)$$

on the rectangular domain $\Omega = [0, 2\pi] \times [0, \pi]$. On this spatial domain, the eigenvalues of the Laplace operator are $-\frac{j^2}{4} - k^2$ for $j, k = 1, 2, 3, \dots$. Thus the spatial operator $A = -\Delta + 3$ has positive eigenvalues and the homogeneous problem (4.4) could have solutions of exponential growth, e.g. $u(t, x, y) = e^t \sin(x) \sin(y)$ is a solution for the initial condition $u(0, x, y) = \sin(x) \sin(y)$. An evolutionary (or dynamical) system is stable if and only all the eigenvalues of the spatial operator are non-positive. Hence the testing problem (4.4) is unstable.

In the numerical experiment, we choose a spatial grid with the x-dimension divided into $m+1 = 256$ subintervals of equal length $h = \pi/128$ and the y-dimension

into $n + 1 = 128$ intervals of the same mesh size $h = \pi/128$. On this regularly structured grid of size $m \times n$, we use the second order central finite difference (4.2) for the discretization of the Laplace operator, and obtain the following spatially discretized problem

$$\frac{d}{dt}u_h = (\Delta_h + 3)u_h \quad (4.5)$$

where Δ_h is the discrete Laplace operator given by (4.3).

We solved the spatially discretized heat equation by the SEITM algorithms with the domain divided into two equal size square subdomains as in Figure 5. The time

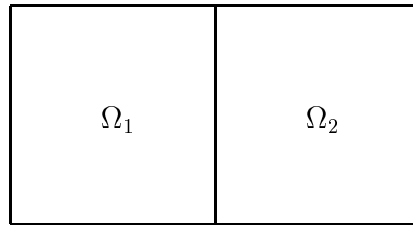


Figure 5

intervals chosen for simulation are the unit interval $[0, 1]$. We used several different time discretization sizes Δt . The measured errors of numerical solutions at time $t = 1$ are listed in Table 4.2 for the indicated temporal discretization sizes.

To examine the stabilization effectiveness of SEITM algorithms, we also have solved the unstable problem using the EITM algorithms and listed the solution errors in the Table 4.2. For stability comparison, we solved the discrete problem (4.5) using the backward Euler method on the non-partitioned entire domain by one processor. Measured errors of the solutions computed by the BEuler method are also listed in Table 4.2. As indicated by the experimental results in Table 4.2, the errors of the SEITM1 and SEITM2 algorithms remain relatively small when

Table 4.2: $u_t = \Delta u + 3u$ with $u(t, x, y) = e^t \sin(x) \sin(y)$

Δt	1/25	1/50	1/100	1/200	1/400	1/800
SEITM1	5.7e-02	2.8e-02	1.4e-02	7.1e-03	3.7e-03	2.0e-03
SEITM2	5.7e-02	2.8e-02	1.4e-02	7.1e-03	3.7e-03	2.0e-03
SEITM3	3.0e-01	1.4e-01	7.0e-02	3.5e-02	1.7e-02	8.8e-03
BEuler	5.7e-02	2.8e-02	1.4e-02	7.1e-03	3.7e-03	2.0e-03
EITM1	3.5e+45	4.4e+94	∞	∞	∞	∞
EITM2	3.6e+12	4.3e+33	4.3e+65	∞	∞	∞
EITM3	2.6e+42	5.9e+88	∞	∞	∞	∞

The spatial domain is $[0, 2\pi] \times [0, \pi]$ with mesh size $h = \pi/128$.

the time step size Δt is large, and they are the same as the errors of the backward Euler method. The errors of SEITM3 algorithm are about 3 to 5 times larger than those of the backward Euler, but is much smaller than the EITM algorithms.

4.1.3 Convection-diffusion Problems

In this section, we test the SEITM algorithms on the problem

$$\begin{cases} \frac{\partial u(t,x,y)}{\partial t} = \Delta u + 9.9 \sin(x) \frac{\partial}{\partial x} u - 9.9 \cos(x) u, & x \in \Omega, t \geq 0, \\ u(t, x, y) = 0, & x \in \partial\Omega, t \geq 0, \\ u(0, x, y) = u^0(x, y), & x \in \Omega \end{cases} \quad (4.6)$$

on the rectangular domain $\Omega = [0, 2\pi] \times [0, \pi]$. We partition the x-dimension of the domain into $m+1 = 256$ intervals of length $h = \pi/128$, and partition the y-dimension into $n+1 = 128$ intervals of the same mesh size $h = \pi/128$. On this grid of size $m \times n$, we use the second order finite difference (4.2) for the discretization

of the Laplace operator, and discretize $\frac{\partial}{\partial x}u$ by the central finite difference

$$\frac{f_{i+1} - f_{i-1}}{h} = f'_i. \quad (4.7)$$

We solved the spatially discretized equation by the SEITM algorithms with the domain divided into two equal size squares as in Figure 5. The time interval chosen for simulation is the unit interval $[0, 1]$. We used several different time discretization sizes Δt . The measured errors of numerical solutions at time $t = 1$ are listed in Table 4.3 for the indicated temporal discretization sizes. We also have solved the problem using the EITM algorithms and the backward Euler method (BEuler). The solution errors computed by the EITM and BEuler algorithms are listed in the Table 4.3.

Table 4.3: $u_t = \Delta u + 9.9 \sin(x)u_x - 9.9 \cos(x)u$ with $u = e^{-2t} \sin(x) \sin(y)$

Δt	1/25	1/50	1/100	1/200	1/400	1/800
SEITM1	1.1e-02	5.5e-03	2.8e-03	1.5e-03	8.0e-04	4.6e-04
SEITM2	1.1e-02	5.5e-03	2.8e-03	1.5e-03	8.0e-04	4.6e-04
SEITM3	1.9e+00	3.7e-01	1.3e-01	5.4e-02	2.5e-02	1.2e-02
BEuler	1.1e-02	5.5e-03	2.8e-03	1.5e-03	8.0e-04	4.6e-04
EITM1	8.0e+42	1.8e+88	∞	∞	∞	∞
EITM2	4.0e+13	1.9e+34	2.0e+66	∞	∞	∞
EITM3	1.7e+42	4.4e+88	∞	∞	∞	∞

The spatial domain is $[0, 2\pi] \times [0, \pi]$ with mesh size $h = \pi/128$.

As indicated by the experimental results in Table 4.3, the errors of the SEITM1 and SEITM2 algorithms remain small even when the time step size Δt is large, and the errors are the same as those of the backward Euler method. The errors of the SEITM3 algorithm are about 30 to 100 times larger than the backward Euler,

but still much smaller than the errors computed by the EITM algorithms. The proof of the unconditional stability of the SEITM algorithms given in Chapter 2 is built upon the self-adjointness (or symmetry) of the spatial operator. Though rigorously it is still open if the SEITM algorithms remain unconditionally stable for problems with non-symmetric operators, the numerical experiment data show that the proposed SEITM1 and SEITM2 algorithms are robust and retain the unconditional stability for non-selfadjoint problems.

To see how different degrees of nonsymmetry affects the numerical error or stability of the SEITM3 algorithm, we test the SEITM3 algorithm on the problem

$$\begin{cases} \frac{\partial u(t,x,y)}{\partial t} = \Delta u + \alpha \sin(x) \frac{\partial}{\partial x} u - \alpha \cos(x) u, & x \in \Omega, t \geq 0, \\ u(t, x, y) = 0, & x \in \partial\Omega, t \geq 0, \\ u(0, x, y) = u^0(x, y), & x \in \Omega \end{cases}$$

with varying α . The testing results are listed in Table 4.4.

Table 4.4: $u_t = \Delta u + \alpha \sin(x)u_x - \alpha \cos(x)u$ with $u = e^t \sin(x) \sin(y)$

α	Δt	1/25	1/50	1/100	1/200	1/400	1/800
0.9	SEITM3	6.4e-03	3.2e-03	1.6e-03	8.2e-04	4.2e-04	2.2e-04
3.9	SEITM3	4.5e-02	2.1e-02	1.0e-02	5.2e-03	2.6e-03	1.3e-03
6.9	SEITM3	2.8e-01	1.0e-01	4.4e-02	2.1e-02	1.0e-02	5.0e-03
9.9	SEITM3	1.9e+00	3.7e-01	1.3e-01	5.4e-02	2.5e-02	1.2e-02
0.9	BEuler	1.1e-02	5.4e-03	2.7e-03	1.4e-03	6.9e-04	3.5e-04
3.9	BEuler	1.1e-02	5.4e-03	2.7e-03	1.4e-03	7.2e-04	3.8e-04
6.9	BEuler	1.1e-02	5.5e-03	2.8e-03	1.4e-03	7.6e-04	4.2e-04
9.9	BEuler	1.1e-02	5.5e-03	2.8e-03	1.5e-03	8.0e-04	4.6e-04

The spatial domain is $[0, 2\pi] \times [0, \pi]$ with mesh size $h = \pi/128$.

We also solved the same problem with varying coefficients α by the backward

Euler method. The maximal errors computed by the BEuler method are listed in Table 4.4. The measured numerical errors indicate that when α is small, SEITM3 algorithm exhibits quite small errors. As α increases, the numerical errors also increase.

4.1.4 An Unstable Convection-diffusion Problem

In this section, we test the SEITM algorithms on the problem

$$\begin{cases} \frac{\partial u(t,x,y)}{\partial t} = \Delta u + \sin(x) \frac{\partial}{\partial x} (\sin(x)u) + (3 - \sin(2x))u, & x \in \Omega, t \geq 0 \\ u(t, x, y) = 0, & x \in \partial\Omega, t \geq 0, \\ u(0, x, y) = u^0(x, y), & x \in \Omega \end{cases} \quad (4.8)$$

on the rectangular domain $\Omega = [0, 2\pi] \times [0, \pi]$. On $L_0^2(\Omega)$, the spatial operator

$$Au = \Delta u + \sin(x) \frac{\partial}{\partial x} (\sin(x)u) + (3 - \sin(2x))u$$

is indefinite. This can be seen from the analysis given below. We rewrite the spatial operator into $A = A_1 + A_2 + A_3$, where

$$\begin{cases} A_1 u = (\Delta + 2)u, \\ A_2 u = \sin(x) \frac{\partial}{\partial x} (\sin(x)u), \\ A_3 u = \sin(2x) + 1. \end{cases}$$

The eigenvalues of the Laplace operator are $-\frac{j^2}{4} - k^2$ for $j, k = 1, 2, 3, \dots$, so the eigenvalues of A_1 are $2 - \frac{j^2}{4} - k^2$ for $j, k = 1, 2, 3, \dots$. Thus A_1 is indefinite and has a positive eigenvalue equal to 0.75. Obviously the multiplication operator $A_3 = \sin(2x) + 1$ is symmetric and positive semi-definite, which means that

$$\langle A_3 v, v \rangle \geq 0 \quad \text{for all } v \in L_0^2(\Omega). \quad (4.9)$$

The operator A_2 is both positive semi-definite and negative semi-definite in the sense that

$$\langle A_2 v, v \rangle + \langle v, A_2 v \rangle = 0 \quad \text{for all } v \in L_0^2(\Omega). \quad (4.10)$$

The above equality holds because for $u, v \in L_0^2(\Omega)$,

$$\begin{aligned} \langle A_2 u, v \rangle &= \int_{\Omega} \sin(x) \frac{\partial}{\partial x} (\sin(x) u(x, y)) \overline{v(x, y)} dx dy \\ &= \int_0^{\pi} \left[\overline{v(x, y)} \sin(x)^2 u(x, y) \Big|_{x=0}^{x=2\pi} \right] dy - \int_{\Omega} \sin(x) u(x, y) \frac{\partial \sin(x) \overline{v(x, y)}}{\partial x} dx dy \\ &= 0 - \int_{\Omega} \overline{\sin(x) \frac{\partial}{\partial x} (\sin(x) v(x, y))} u(x, y) dx dy \\ &= - \langle u, A_2 v \rangle. \end{aligned}$$

Now we take $v(x, y) = \sin(0.5x) \sin(y)$, an eigenvector of A_1 with respect to the eigenvalue 0.75, and have that

$$\begin{aligned} \langle Av, v \rangle + \langle v, Av \rangle &= 2 \langle A_1 v, v \rangle + (\langle A_2 v, v \rangle + \langle v, A_2 v \rangle) + 2 \langle A_3 v, v \rangle \\ &= 1.5 \|v\|^2 + (\langle A_2 v, v \rangle + \langle v, A_2 v \rangle) + 2 \langle A_3 v, v \rangle \\ &\geq 1.5 \|v\|^2, \end{aligned} \quad (4.11)$$

where the last inequality is due to (4.9) and (4.10). Thus A is indefinite by (4.11). Since the spatial operator is indefinite, the homogeneous problem (4.8) could have solutions of exponential growth, e.g. $u(t, x, y) = e^t \sin(x) \sin(y)$ is a solution for the initial condition $u(0, x, y) = \sin(x) \sin(y)$. An evolutionary (or dynamical) system is stable if and only all the eigenvalues of the spatial operator are non-positive. Hence the testing problem (4.8) is unstable.

To solve the problem numerically, we partition the x-dimension of the domain into $m+1 = 256$ intervals of length $h = \pi/128$, and partition the y-dimension into $n+1 = 128$ intervals of the same mesh size $h = \pi/128$. On this grid of size $m \times n$, we use the second order finite difference (4.2) for the discretization of the Laplace

operator, and discretize $\frac{\partial}{\partial x}u$ by the central finite difference (4.7). We solved the spatially discretized equation by the SEITM algorithms with the domain divided into two equal size squares as in Figure 5. The time interval chosen for simulation is the unit interval $[0, 1]$. We used several different time discretization sizes Δt . The measured errors of numerical solutions at time $t = 1$ are listed in Table 4.5 for the indicated temporal discretization sizes. We also have solved the problem using the EITM algorithms and the backward Euler method (BEuler). The solution errors computed by the EITM and BEuler algorithms are also listed in the Table 4.5.

Table 4.5: $u_t = \Delta u + \sin(x)^2 u_x + [3 - \sin(x) \cos(x)]u$ with $u = e^t \sin(x) \sin(y)$

Δt	1/25	1/50	1/100	1/200	1/400	1/800
SEITM1	5.7e-02	2.8e-02	1.4e-02	7.1e-03	3.7e-03	2.0e-03
SEITM2	5.7e-02	2.8e-02	1.4e-02	7.2e-03	3.7e-03	2.0e-03
SEITM3	3.9e-01	1.9e-01	9.1e-02	4.5e-02	2.3e-02	1.1e-02
BEuler	5.7e-02	2.8e-02	1.4e-02	7.2e-03	3.7e-03	2.0e-03
EITM1	4.4e+45	5.6e+94	∞	∞	∞	∞
EITM2	8.6e+21	2.2e+42	1.6e+74	∞	∞	∞
EITM3	2.6e+42	6.1e+88	∞	∞	∞	∞

The spatial domain is $[0, 2\pi] \times [0, \pi]$ with mesh size $h = \pi/128$.

As indicated by the experimental results in Table 4.5, the errors of the SEITM1 and SEITM2 algorithms remain small even when the time step size Δt is large, and the errors are almost the same as those of the backward Euler method. The errors of the SEITM3 algorithm are about 5 times larger than the backward Euler, but still much smaller than the errors computed by the EITM algorithms. The numerical experiment data show that the SEITM1 and SEITM2 algorithms are

robust and retain the unconditional for unstable, non-selfadjoint problems.

4.2 Scalability Testing

To examine the scalability of the proposed domain decomposition algorithms, we apply the SEITM1 and SEITM3 algorithms to the convection-diffusion equation

$$\begin{cases} \frac{\partial u(t,x,y)}{\partial t} = \Delta u + \sin(x)u_x - \cos(x)u, & x \in \Omega, t \geq 0 \\ u(t, x, y) = 0, & x \in \partial\Omega, t \geq 0, \\ u(0, x, y) = u^0(x, y), & x \in \Omega \end{cases}$$

on spatial domains $[0, p\pi] \times [0, \pi]$, where p is the number of subdomains the entire domain is to be partitioned into. The domain partitioning is along the x-direction as shown in Figure 6, with each subdomain being a square and assigned to a processor. Uniform grid is applied to the domain with mesh size $\frac{\pi}{256}$ in both the

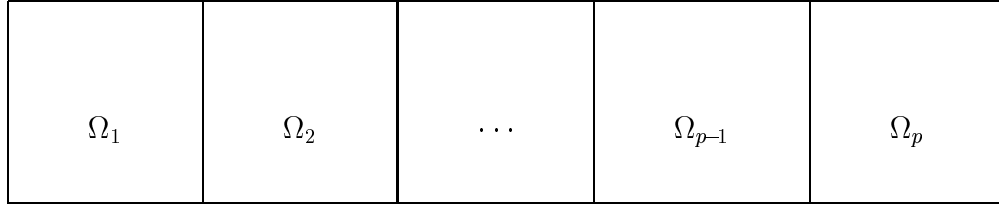


Figure 6

x- and y-directions. On this grid we use the second order finite difference (4.2) for the discretization of the Laplace operator, and discretize $\frac{\partial}{\partial x}u$ by the central finite difference (4.7). The simulation time interval is $[0, 1]$, and the time step size is $\Delta t = 5.0e-03$.

We solved the problem by the SEITM1 and SEITM3 algorithms on a dedicated queue of an NCSA Origin 2000 machine with a maximum of 256 nodes each of

250 MHz, running IRIX 6.5.9 operating system. 64-bit arithmetic operation was chosen. In the experiments, we measured the computation time (T_{comp}), the communication time (T_{comm}), the total execution time (T_{total}), and the maximal errors of the numerical solutions at time $t = 1$. These measured data are listed in Table 4.6 and Table 4.7 together with parallel speedup and efficiency calculated using the total execution time T_{total} . In the tables, the unit of T_{comp} , T_{comm} and T_{total} is second.

Table 4.6: Solving $u_t = \Delta u + \sin(x)u_x - \cos(x)u$ by SEITM1

Processors	T_{total}	T_{comp}	T_{comm}	Speedup	Efficiency	Max-Err
1	2.08e+01	2.08e+01	0.0e-02	1	100%	1.4e-03
2	2.12e+01	2.10e+01	1.1e-01	1.96	98.1%	1.4e-03
3	2.12e+01	2.10e+01	1.9e-01	2.94	98.1%	1.4e-03
4	2.12e+01	2.10e+01	1.5e-01	3.92	98.1%	1.4e-03
5	2.12e+01	2.10e+01	1.7e-01	4.91	98.1%	1.4e-03
6	2.13e+01	2.10e+01	2.5e-01	5.86	97.7%	1.4e-03
7	2.13e+01	2.11e+01	2.9e-01	6.84	97.7%	1.4e-03
8	2.13e+01	2.11e+01	2.2e-01	7.81	97.7%	1.4e-03
10	2.14e+01	2.11e+01	3.1e-01	9.72	97.2%	1.4e-03
12	2.14e+01	2.10e+01	4.1e-01	11.7	97.2%	1.4e-03
14	2.14e+01	2.10e+01	3.7e-01	13.6	97.2%	1.4e-03
16	2.14e+01	2.10e+01	3.8e-01	15.6	97.2%	1.4e-03
20	2.15e+01	2.10e+01	4.8e-01	19.3	96.7%	1.4e-03
24	2.15e+01	2.11e+01	4.3e-01	23.2	96.7%	1.4e-03
28	2.16e+01	2.11e+01	5.0e-01	27.0	96.3%	1.4e-03
32	2.16e+01	2.10e+01	5.6e-01	30.8	96.3%	1.4e-03

Speedup and efficiency are computed using T_{total} .

The spatial domain is $[0, p\pi] \times [0, \pi]$ with $h = \pi/256$, where p is the number of processors.

The testing time interval is $[0, 1]$ with $\Delta t = 5.0e-03$.

Table 4.7: Solving $u_t = \Delta u + \sin(x)u_x - \cos(x)u$ by SEITM3

Processors	T_total	T_comp	T_comm	Speedup	Efficiency	Max-Err
1	9.89e+00	9.89e+00	0.0e-02	1	100%	8.4e-04
2	1.04e+01	1.02e+01	1.9e-01	1.90	95.1%	8.4e-04
3	1.04e+01	1.02e+01	2.3e-01	2.85	95.1%	8.4e-04
4	1.05e+01	1.02e+01	2.9e-01	3.77	94.2%	8.4e-04
5	1.05e+01	1.02e+01	3.4e-01	4.71	94.2%	8.4e-04
6	1.06e+01	1.02e+01	3.5e-01	5.60	93.3%	8.4e-04
7	1.05e+01	1.02e+01	3.1e-01	6.59	94.2%	8.4e-04
8	1.05e+01	1.02e+01	2.8e-01	7.54	94.2%	8.4e-04
10	1.05e+01	1.02e+01	2.7e-01	9.92	94.2%	8.4e-04
12	1.05e+01	1.02e+01	3.0e-01	11.3	94.2%	8.4e-04
14	1.07e+01	1.01e+01	5.5e-01	12.9	92.4%	8.4e-04
16	1.06e+01	1.02e+01	4.1e-01	14.9	93.3%	8.4e-04
20	1.05e+01	1.01e+01	4.1e-01	18.8	94.2%	8.4e-04
24	1.05e+01	1.01e+01	4.0e-01	22.6	94.2%	8.4e-04
28	1.06e+01	1.02e+01	3.5e-01	26.1	93.3%	8.4e-04
32	1.07e+01	1.01e+01	5.5e-01	29.6	92.4%	8.4e-04

Speedup and efficiency are computed using T_{total} .

The spatial domain is $[0, p\pi] \times [0, \pi]$ with $h = \pi/256$, where p is the number of processors.

The testing time interval is $[0, 1]$ with $\Delta t = 5.0e-03$.

The experimental data show that the computation time almost remains the same as the problem size increases with the machine ensemble size such that the memory usage on each processor remains the same. That is the problem size is scaled up following the memory-bounded constraint [58]. This phenomenon is well under expectation since the stabilization process has a very low computation overhead. The communication time increases slowly as the number of processors increases. As the number of processors increases from one to thirty two, the ef-

efficiency has decreased less than 4 percentage points for the SEITM1 algorithm, and decreased less than 8 percentage points for the SEITM2 algorithm. This rate of performance decrease is very slow, matching well with the analysis given in Section 2.1.

The difference in the decreased percentage between the SEITM1 and SEITM2 algorithm is due to the difference of computation costs of the two algorithms. The computation time used by the SEITM3 algorithm is less than half of that used by the SEITM1 algorithm while the communication time consumed by the two algorithms are almost the same. This difference in computation time is well under expectation since, as analyzed in Chapter 2, the SEITM3 has a linear order computation cost while SEITM1 algorithm using an FFT based subdomain solver has an $O(N \log N)$ computation cost on a grid with N grid points.

4.3 High Order Spatial Discretization

In this section, we apply the SEITM1 algorithm with a fourth order discretization scheme for the spatial operator of the following homogeneous problem of the diffusion problem

$$\begin{cases} \frac{\partial u(t,x,y)}{\partial t} = \Delta u + 3u, & x \in \Omega, t \geq 0 \\ u(t, x, y) = 0, & x \in \partial\Omega, t \geq 0, \\ u(0, x, y) = u^0(x, y), & x \in \Omega \end{cases} \quad (4.12)$$

on the rectangular domain $[0, 2\pi] \times [0, \pi]$. In the numerical experiment, we partition the domain into $2n$ intervals of length h in the x-dimension, and partition the y-dimension into n intervals of the same mesh size h as in x-dimension, resulting in

a grid of size $(2n-1) \times (n-1)$. In the SEITM1 algorithm, we partition the domain into two square subdomains of equal size, and the interface boundary of the grid consists of grid points

$$\{(i, j) : i = n \text{ and } j = 1, 2, \dots, n-1\}.$$

We discretize the Laplace operator by

$$(I + \frac{h_x^2}{12}D_{xx} + \frac{h_y^2}{12}D_{yy})\Delta u_{i,j} = [D_{xx}(I + \frac{h_y^2}{12}D_{yy}) + (I + \frac{h_x^2}{12}D_{xx})D_{yy}]u^{i,j}, \quad (4.13)$$

which is the formula (3.13) with $\frac{\partial^2}{\partial x^2}$ and $\frac{\partial^2}{\partial y^2}$ replaced by D_{xx} and D_{yy} , where $D_{xx}u_{i,j} = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2}$ and $D_{yy}u_{i,j} = \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h^2}$. With formula (4.13), both the subdomain scheme and the stabilization of the SEITM1 algorithm require solving an equation of the form

$$[I - \Delta t(I + \frac{h^2}{12}D_{xx})^{-1}D_{xx} - \Delta t(I + \frac{h^2}{12}D_{yy})^{-1}D_{yy}]u = r.$$

Since the stabilization in the SEITM1 algorithm is applied on the interface boundaries, the equation to be solved on the interface boundaries are tridiagonal systems and can be easily solved. The equation on the subdomains can be solved by the FFT-based direct solver described in Chapter 3. With these fourth order spatial discretization schemes for the Laplace operator, we solve the diffusion equation (4.12) from time $t = 0$ to $t = 1$ with several different spatial discretization sizes h and different time discretization sizes Δt . The measured errors of numerical solutions at time $t = 1$ are listed in Table 4.8 for the indicated spatial mesh sizes and temporal discretization sizes.

In the testings, the time step sizes were chosen to be quite small as compared to the spatial mesh sizes. Since the error of a numerical solution of time dependent

Table 4.8: $u_t = \Delta u + 3u$ on $[0, 2\pi] \times [0, \pi]$ with $u = e^t \sin(x) \sin(y)$

$(\pi/h, 1/\Delta t)$	(8, 500)	(8, 1000)	(8, 2000)	(16,2000)	(16,4000)	(16,8000)
Order 2	7.3e-02	7.2e-02	7.1e-02	1.8e-02	1.8e-02	1.8e-02
Order 4	2.4e-03	1.0e-03	3.2e-04	6.6e-04	3.2e-04	1.5e-04

partial differential equations comes from both the temporal and spatial discretization, to examine the accuracy of the spatial discretization we must choose spatial mesh sizes large enough so that spatial discretization errors dominate the temporal error. The experimental results show that the fourth order method has a much higher accuracy than the second order method, and when the spatial mesh size is halved, the error of the fourth order method decreases markedly while the error of the second order method almost remains the same.

4.4 Concluding Remarks

The experiments carried out have been supporting the assertion proven in Chapter 2 that SEITM algorithms are unconditionally stable for evolutionary problems with symmetric operators, and also showing the robustness of the SEITM1 and SEITM2 algorithms in maintaining good stability results for unsymmetric problems. While achieving excellent stability results, the SEITM algorithms also have low communication cost and high parallel speedup and efficiency, proving themselves ideal solvers for large scale parallel simulation problems on distributed memory machines.

5 Summary and Future Work

We have developed the stabilized explicit-implicit time marching (SEITM) domain decomposition methods by adding a stabilization step to the explicit-implicit time marching (EITM) methods. The EITM methods are globally non-iterative, non-overlapping domain decomposition methods, which are computationally and communicationally efficient for each time step calculation when compared with Schwartz method based parabolic solvers, and preserve the accuracy of temporal and spatial discretization when compared with non-EITM based globally non-iterative, non-overlapping domain decomposition methods. However the EITM methods suffer from either stability or consistency related time step size restrictions, while Schwartz methods could maintain the good stability condition of implicit temporal discretization schemes. The proposed SEITM methods have inherited the advantages of EITM methods in time-stepwise efficiency and discretization accuracy preserving but are free from stability or consistency related time step size restrictions. Thus the SEITM methods have the advantages of both the EITM methods and the Schwartz method based parabolic solvers. But what's more important for parallel computing, especially for large scale simulation problems, is that the SEITM methods achieve stability by adding zero communication cost and negligible computation cost to the EITM methods, yielding excellent parallel speedup and scalability confirmed by testings on SGI Origin 2000 super-computers. Thus, the SEITM methods have great potential for large scale simulation problems on distributed memory machines.

Among the three methods (SEITM1, SEITM2 and SEITM3) proposed, the

SEITM1 and SEITM2 methods have the same computation and communication cost, while the SEITM3 method has same communication cost as the other two but lower computation cost — an $O(N)$ time-stepwise computation cost on a grid of size N for all problems including the computationally challenging convection-diffusion problems. However numerical experiments show that the SEITM3 method becomes less stable for some convection-diffusion problems with “larger” convection terms. Since the four convection-diffusion problems chosen for testing in Section 4.1.3 also have sink-source terms which assume positive values at some points, it is unknown if these positive values of the sink-source terms have made these testing problems unstable. Thus it is not fully understood even from an experimental point of view if the measured weaker stability results of the SEITM3 methods are due to the non-symmetry of the spatial operators of the testing problems only or due to the combination of the nonsymmetry or (possible) instability of the testing problems. Rigorously, the stability is still not understood for all the three SEITM methods, so one mathematical issue the newly proposed SEITM methods open up is how operator splitting (especially domain decomposition based) of nonsymmetric operators will affect the stability of the numerical method.

This mathematical issue has another important implication in the applicability of SEITM methods to hyperbolic problems. Hyperbolic problems like the wave equations are dissipative but nonsymmetric. In Chapter 2, the parallel speedup analysis does not require any property of parabolic systems. Thus when applied to hyperbolic systems, the high computation and communication efficiency of the SEITM methods should not change too much. In the proofs of stability

and convergence, the properties of the parabolic system we have used are the quasi-dissipativity and symmetry. The wave equations are dissipative (even better than quasi-dissipative) but nonsymmetric. Numerical testings of the SEITM1 and SEITM2 methods on nonsymmetric parabolic problems have shown excellent stability. It is therefore reasonable to expect good stability results when SEITM1 and SEITM2 methods are applied to hyperbolic problems. Hence to understand the stability condition of the SEITM methods for nonsymmetric problems has significance for both convection-diffusion problems and hyperbolic problems.

Another important issue in understanding the SEITM methods is to find the smallest size the subdomains can be chosen without affecting the stability of the methods. If there is no restriction on the size of the subdomain, then linear order (hence optimal) computation complexity can be achieved for each time step calculation. The reason for this linear order complexity is that the computation costs of the SEITM methods, as analyzed in Chapter 2, depend linearly on the number of subdomains but possibly super-linearly (except SEITM3) on the number of grid points. Thus by partitioning the entire domain into subdomains of size $O(1)$, linear order computation cost can also be achieved for the SEITM1 and SEITM2 methods. Since SEITM1 and SEITM2 methods exhibit better stability results than SEITM3 in numerical experiments, it is important to investigate the restrictions, if any, on the size of subdomains.

The two aforementioned issues, as well as more flexible domain partitioning strategies mentioned in Chapter 2 for allowing the crossing of interface boundaries, are a few example research issues the SEITM methods have opened up. There

are many other interesting directions created by the newly proposed SEITM methods that further investigation can proceed. For example, many systems studied in sciences and engineering are governed by time dependent PDEs which are computationally expensive to simulate, and hence the potential contribution of SEITM methods to other disciplines is obvious. We conclude this dissertation by the following physiology problem the SEITM methods can help to provide insight into.

The electric potential of a passive neuron is governed by an inhomogeneous diffusion equation with a sink-source term [33, 51, 53]. The governing equation is simple. However, the enormous complexity of neuron dendritic geometry presents an obstacle to the solution and hence quantitative understanding of the electric characteristics of a neuron. One approach to understand the electric property on the dendritic tree is the equivalent cable equation method introduced by Rall, with which a branching tree is replaced by many unbranching cables in a certain way to retain the dendritic tree's essential electric characteristics. A recent technique for generating the equivalent cable equation is the Lanczos procedure [39] based method that tridiagonalizes the system of spatially discretized cable equations coupled at the branching nodes [48, 62], obtaining an equation equivalent to a one-dimensional spatially discretized linear parabolic equation in a sense of, among all senses, tridiagonal matrix structure for the discrete spatial operator. One obvious disadvantage of the Lanczos procedure, also as pointed by Whitehead and Rosenberg, is the possibility of break-down of the iterative process. However, the SEITM methods proposed in this dissertation particularly fit this problem for handling the enormous complexity of the dendritic spatial geometry. The non-

overlapping domain decomposition based operator splitting of the SEITM methods can easily decompose the dendritic trees into a bunch of non-branching cables with each cable as a subdomain. This domain decomposition process not only frees from any possible breakdown (a drawback of the Lanczos procedure not the equation), a more efficient numerical solution is also expected since the SEITM methods are globally non-iterative while Lanczos procedure is.

References

- [1] K. BLACK, *Polynomial collocation using a domain decomposition solution to parabolic PDE's via the penalty method and explicit/implicit time marching*, J. Sci. Comput., 7 (1992), no. 4, pp. 313–338.
- [2] B. BÄUMER AND F. NEUBRANDER, *Laplace transform methods for evolution equations*, Conferenze del Seminario di Matematica dell' Università Bari 259 (1994), 27-60.
- [3] G. BIRKHOFF AND R. LYNCH, *Numerical Solution of Elliptic Problems*, SIAM Philadelphia, 1984.
- [4] G. BIRKHOFF AND R. S. VARGA, *Implicit alternating direction methods*, Trans. AMS, 92 (1959), pp. 13–24.
- [5] R. F. BOISVERT, *fourth-order-accurate Fourier method for the Helmholtz equation in three dimensions*, ACM Trans. Math. Software, 13 (1987), pp. 221–234.
- [6] X.-C. CAI, *Additive Schwartz algorithms for parabolic convection-diffusion equations*, Numer. Math., 60 (1991), pp. 41–61.
- [7] ———, *Multiplicative Schwartz methods for parabolic problems*, SIAM J. Sci. Comput., 15 (1994), pp. 587–603.
- [8] X.-C. CAI, W. D. GROPP AND D. E. KEYES, *A comparison of some domain decomposition algorithms for nonsymmetric elliptic problems*, J. Numer. Lin. Alg. Appl., 1993.
- [9] X.-C. CAI AND O. B. WIDLUND, *Domain decomposition algorithms for indefinite elliptic problems*, SIAM J. Sci. Statist. Comput. **13** (1992), no. 1, pp. 243–258.
- [10] ———, *Multiplicative Schwartz algorithms for nonsymmetric and indefinite elliptic problems*, SIAM J. Numer. Anal., 30 (1993), pp. 936–952.
- [11] L. COLLATZ, *The Numerical Treatment of Differential Equations*, Springer, 1960.
- [12] M. H. CARPENTER, D. GOTTLIEB AND S. ABARBANEL, *Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes*, NASA ICASE Technical Report No. 93-9, NASA Langley Research Center Hampton, VA 23681-0001, 1993.
- [13] T. F. CHAN AND T. MATHEW, *Domain decomposition algorithms*, Acta Numerica, 1994, pp. 61–143.
- [14] P. CHERNOFF, *Note on product formulas for operator semigroups*, J. Functional Anal. 2 (1968), 238-242.
- [15] R. COURANT, K. FRIEDRICHS AND H. LEWY, *Über die partiellen Differenzialgleichungen der mathematischen Physik*, Math. Ann., Vol. 100, 1928, pp. 32–74.
- [16] J. CRANK AND P. NICOLSON, *A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type*. Proc. Cambridge Philos. Soc. 43, (1947). pp. 50–67.
- [17] C. DAWSON AND T. DUPONT, *Explicit/implicit, conservative domain decomposition procedures for parabolic problems based on block-centered finite difference*, SIAM J. Numer. Anal. 31 (1994), no. 4, pp. 1045–1061.

- [18] C. DAWSON, Q. DU, AND AND T. DUPONT, *A finite difference domain decomposition algorithm for numerical solution of the heat equation*, Math. Comp. 57 (1991), no. 195, pp. 63–71.
- [19] J. DOUGLAS, *Alternating direction implicit methods for three space variables*, Numer. Math., 4 (1962), pp. 41–63.
- [20] J. DOUGLAS AND J. GUNN, *A general formulation of alternating direction method: Part I. Parabolic and hyperbolic problems*, Numer. Math., 6 (1964), pp. 428–453.
- [21] M. DRYJA, *Substructuring methods for parabolic problems*. Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations (Moscow, 1990), pp. 264–271, SIAM, Philadelphia, PA, 1991.
- [22] M. DRYJA AND O. B. WIDLUND, *An additive variant of the Schwarz alternating method for the case of many subregions*, Tech. Rep. 339, Courant Inst., New York Univ., 1987.
- [23] E. DU FORT AND S. FRANKEL, *Stability conditions in the numerical treatment of parabolic differential equations*, Math. Tables and Other Aids to Computation, 7 (1953), PP. 135–152.
- [24] D. FUNARO, *A multidomain spectral approximation of elliptic equations*, Numer. Methods Partial Differential Equations, 2 (1986), no. 3, pp. 187–205
- [25] D. FUNARO AND D. GOTTLIEB, *A new method of imposing boundary conditions in pseudospectral approximations of hyperbolic equations*, Math. Comp., 51 (1988), pp. 599–613.
- [26] J. A. GOLDSTEIN, *Semigroups of Linear Operators and Applications*. Oxford University Press 1985.
- [27] W. D. GROPP AND D. E. KEYES, *Domain decomposition on parallel computers*, Impact of Computing in Science and Engineering, 1 (1989), pp. 421–439.
- [28] R. HIRSH, *Higher order accurate difference solutions of fluid mechanics problems by a compact differencing technique*, J. Comput. Phys., 19 (1975), pp. 90–109.
- [29] R. HOCKNEY, *A fast direct solution of Poisson's equation using Fourier analysis*, J. ACM, 12 (1965), pp. 95–113.
- [30] H. KREISS AND J. OLIGER, *Methods for the approximate solution of time dependent problems*. GARP Report No 10, 1973.
- [31] D. E. KEYES, *Domain decomposition: a bridge between nature and parallel computers*, NASA ICASE Technical Report No. 92-44, NASA Langley Research Center Hampton, VA 23681-0001, 1992.
- [32] D. E. KEYES AND W. D. GROPP, *A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation*, SIAM J. Sci. Statis. Comput., 8 (1987), pp. 166–202.
- [33] C. KOCH, *Biophysics of Computation: Information Processing in Single Neurons*, Oxford University Press, New York, New York, 1999.
- [34] Y. A. KUZNETSOV, *New algorithms for approximate reralization of implicit difference schemes*, Sov. J. Numer. Ana.Math. Modell. 3 (1988), pp. 99–114.

- [35] YU. M. LAEVSKY, *A domain decomposition algorithm without overlapping subdomains for the solution of parabolic equations*. (Russian) Zh. Vychisl. Mat. i Mat. Fiz. 32 (1992), no. 11, pp. 1744–1755; translation in Comput. Math. Math. Phys. 32 (1992), no. 11, pp. 1569–1580 (1993)
- [36] ———, *Explicit-implicit domain decomposition method for solving parabolic equations*. (Russian) Computing methods and technology for solving problems in mathematical physics (Russian), pp. 30–46, Ross. Akad. Nauk Sibirsk. Otdel., Vychisl. Tsentr, Novosibirsk, 1993.
- [37] Y. M. LAEVSKY AND S. V. GOLOLOBOV, *Explicit-implicit domain decomposition methods for the solution of parabolic equations*. (Russian) Sibirsk. Mat. Zh. 36 (1995), no. 3, pp. 590–601, ii; translation in Siberian Math. J. 36 (1995), no. 3, pp. 506–516
- [38] YU. M. LAEVSKY AND O. V. RUDENKO, *Splitting methods for parabolic problems in nonrectangular domains*. Appl. Math. Lett. 8 (1995), no. 6, pp. 9–14.
- [39] C. LANCZOS, *An iterative method for the solution of the problem of linear differential and integral operators*, J. Res. Natn. Bur. Stand., Vol. 45 (1950), pp. 255–282.
- [40] P. LAX AND B. WENDROFF, *On the stability of difference schemes*, Comm. Pure Appl. Math., Vol. 15 (1962), pp. 363–371.
- [41] P. LAX AND R. RICHTMYER, *Survey of the stability of linear finite difference*, Comm. Pure Appl. Math. 9 (1956), pp. 267–293.
- [42] Q.-Y. LEE, N.-C. WANG, AND D.-Y. YI, *Numerical Analysis*, Central China Institute of Technology Press, 3rd ed., 1986.
- [43] S. LELE, *Compact finite difference schemes with spectral-like resolution*, J. Comp. Phys., 103 (1992), pp. 16–42.
- [44] R. E. LYNCH AND J. R. RICE, *High accuracy finite difference approximation to solutions of elliptic partial differential equations*, Proc. Nat. Acad. Sci., 75 (1978), pp. 2541–2544.
- [45] R. E. LYNCH, J. R. RICE, AND D. H. THOMAS, *Tensor product analysis of partial differential equations*, Bull. American Math. Soc., 70 (1964), pp. 378–384.
- [46] G. LUMER AND R. S. PHILLIPS, *Dissipative operators in A Banach space*, Pac. J. Math. 11 (1961), pp 679-698.
- [47] T. MATHEW, P. POLYAKOV, G. RUSSO AND J. WANG, *Domain decomposition operator splittings for the solution of parabolic equations*, SIAM J. Sci. Comput. 19 (1998), no. 3, pp. 912–932.
- [48] J. M. OGDEN, J. R. ROSENBERG AND R. R. WHITEHEAD, *The Lanczos procedure for generating equivalent cables*, in Modeling in the Neurosciences (Poznanski ed.), Harwood Academic Publishers, 1999.
- [49] A. PAZY, *Semigroups of Linear Operators and Applications to Partial Differential Equations*. Springer-Verlag 1983.
- [50] D. W. PEACEMAN AND J. H. H. RACHFORD, *The numerical solution of parabolic and elliptic differential equations*, J. SIAM, 3 (1955), pp. 28–41.
- [51] R. R. POZNANSKI, *Modeling in the Neurosciences*, Harwood Academic Publishers, 1999.

- [52] J. R. RICE AND R. F. BOISVERT, *Solving Elliptic Problems Using ELLPACK*, Springer-Verlag, New York, 1985.
- [53] W. RALL, *Theory of physiological properties of dendrites*, Ann. NY. Acad. Sci., Vol. 96 (1962), pp. 1071–1092.
- [54] K. A. RUSCH, *Development of a marsh-based system to treat domestic wastewater from coastal dwellings*, J. Louisiana Section Amer. Soc. Civil Engineers, Vol. 6, No. 2, Mar. 98, pp. 4–21.
- [55] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company 1995.
- [56] H. A. SCHWARTZ, *Gesammelte Mathematische Abhandlungen*, Vol. 2, Springer, Berlin, 1890, pp. 133–143. First published in *Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich*, vol. 115, 1870, pp. 272–286.
- [57] B. F. SMITH, P. E. BJORSTAD AND W. D. GROPP, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 1996.
- [58] X.-H. SUN, AND L. NI, *Scalable Problems and Memory-Bounded Speedup*, Journal of Parallel and Distributed Computing, Vol. 19, pp. 27-37, Sept. 1993.
- [59] X.-H. SUN AND Y. ZHUANG, *A high-order direct solver for Helmholtz equations with Neumann boundary conditions*. NASA ICASE Technical Report No. 97-11, NASA Langley Research Center Hampton, VA 23681-0001, 1997.
- [60] E. L. WACHSPRESS, *CURE: a generalized two-space-dimension multigroup coding for the IBM 704*. Knolls Atomic Power Laboratory Report No. KAPL 1724, General Electric Co., Schenectady, New York, April 1957.
- [61] E. L. WACHSPRESS AND G. J. HABETLER, *An alternating direction implicit iteration technique*, J. SIAM 8, (1960), pp. 403–424.
- [62] R. R. WHITEHEAD AND J. R. ROSENBERG, *On trees as equivalent cables*, Proc. R. Soc. Lond. B., Vol 252 (1993), pp. 103–108.
- [63] J. XU, *An introduction to multilevel methods*, Wavelets, Multilevel Methods and Elliptic PDEs (M. Ainsworth, J. Levesley, W. A. Light and M. Marletta ed.), Oxford Science Publications, (1997), pp. 213–302.
- [64] Y. ZHUANG, *PhD dissertation proposal*, Department of Computer Science, Louisiana State University Baton Rouge, July 1999.
- [65] ———, *Toward understanding the Von Neumann stability condition*, preprint.
- [66] ———, *Stabilized approximation for evolution equations*, preprint.
- [67] ———, *PhD dissertation*, Department of Mathematics, Louisiana State University Baton Rouge, August 2000.
- [68] ———, *An efficient interface boundary condition treatment for stabilized explicit/implicit time marching domain decomposition methods*, in preparation.
- [69] Y. ZHUANG AND X.-H. SUN, *A high-order ADI solver for separable generalized Helmholtz equations*, to appear in *Advances in Engineering Software*.
- [70] ———, *A high-order fast direct solver for singular Poisson equations*, submitted.

Vita

The author was born on March 7, 1968, in Hanzhou, China. After obtaining his high school diploma in 1986, he entered Zhejiang University in September of the same year, where he started his career training as a mathematician. After writing a thesis on extrapolation based approximation methods, he graduated with a degree of Bachelor of Science in applied mathematics in July 1990. He was assigned to Yonggu Rubber Factory as an official employee in August 1990 and during the winter of 1990 and 1991, he took a leave from Yonggu and worked as an un-official employee for Shanghai Wu-Song Electric Equipement Company as a software engineer until the Spring of 1992 when he returned to Yonggu. After working as an information analyst in Yonggu for a few months, he left for the United States and entered the graduate program in the Department of Mathematics of Louisiana State University in the Fall of 1992. In December 1995, he received a master of science degree from the Department of Mathematics at Louisiana State University. During the summers of 1997 and 1998, he went to Los Alamos National Laboratory and worked in the Scientific Computing group (CIC-19) of CIC division. He received a master of science degree from the Department of Computer Science at Louisiana State University in May 1998, and received the degree of Doctor of Philosophy in mathematics at Louisiana State University in August 2000. In December 2000, the degree of of Doctor of Philosophy in computer science will be conferred.