



**SEQUENTIAL AND PARALLEL MODIFIED ACCELERATED
OVERRELAXATION ITERATIVE METHODS FOR SOLVING INTEGER
AND FRACTIONAL ORDER PARTIAL DIFFERENTIAL EQUATIONS
WITH MULTI-CORE ARCHITECTURE**

By

NIK AMIR SYAFIQ BIN NIK MAZLAN

**Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia,
in Fulfilment of the Requirements for the Degree of Doctor of Philosophy**

November 2024

IPM 2024 2



All material contained within the thesis, including without limitation text, logos, icons, photographs and all other artwork, is copyright material of Universiti Putra Malaysia unless otherwise stated. Use may be made of any material contained within the thesis for non-commercial purposes from the copyright holder. Commercial use of material may only be made with the express, prior, written permission of Universiti Putra Malaysia.

Copyright ©Universiti Putra Malaysia



DEDICATIONS

Dad: Nik Mazlan bin Mamat

Mum: Sriviowarti binti Noerdin

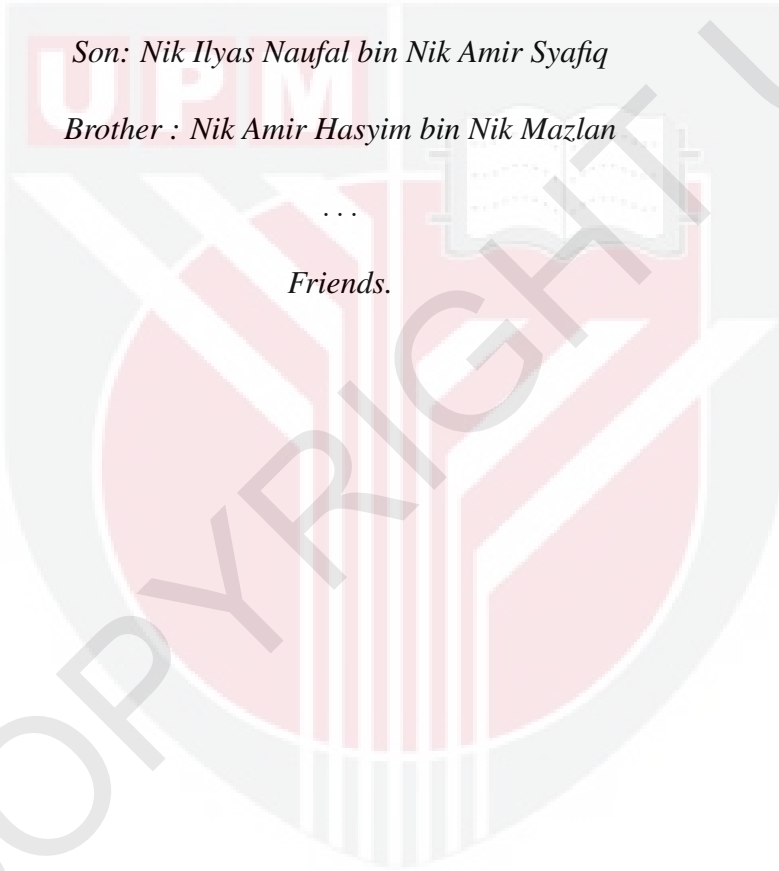
Wife: Nur Syazana binti Rashid

Son: Nik Ilyas Naufal bin Nik Amir Syafiq

Brother : Nik Amir Hasyim bin Nik Mazlan

...

Friends.



Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirement for the degree of Doctor of Philosophy

**SEQUENTIAL AND PARALLEL MODIFIED ACCELERATED
OVERRELAXATION ITERATIVE METHODS FOR SOLVING INTEGER
AND FRACTIONAL ORDER PARTIAL DIFFERENTIAL EQUATIONS
WITH MULTI-CORE ARCHITECTURE**

By

NIK AMIR SYAFIQ BIN NIK MAZLAN

November 2024

Chair : Associate Professor Norazak bin Senu, Ph.D.

Institute: Mathematical Research

This thesis deals with sequential and parallel Modified Accelerated Overrelaxation (MAOR) iterative methods for solving the Helmholtz equation and Fractional Poisson equation. The thesis examined different finite-difference approximations, in the form of point and group approaches, that are discretized on the second order, fourth order and fractional order iterative methods. The utilization of relaxation schemes along with the finite-difference approximation would increase the convergent rate of the iterative methods. This thesis would consider the MAOR scheme which was developed as an extension of the Accelerated Overrelaxation (AOR) scheme and is compared to some previously developed relaxation schemes. The OpenMP parallel program was implemented together with the iterative methods and relaxation schemes which would reduce the execution time of the iterative methods while maintaining the number of iterations. To assist this process, different ordering strategies are applied to the solution domain which ensured that the points in the iterative process are independent from each other. Numerical experiments showed that the MAOR scheme produced the least

number of iteration when compared to other relaxation schemes. The implementation of parallel computing on all iterative methods has reduced the execution time when the number of processors increases which also increased the parallel efficiency. However, the MAOR scheme utilizes more points during the iterative process which contributed to having a higher execution time compared to some relaxation schemes.

Keywords: Fractional Poisson equation, Modified accelerated overrelaxation, Riemann - Liouville fractional derivative, Shifted Grünwald estimate.

SDG: GOAL 9: Industry, Innovation and Infrastructure



Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Doktor Falsafah

**KAEDAH LELARAN RELAKSASI DIPECUTKAN YANG DIUBAH SUAI
BERJUJUKAN DAN SELARI BAGI MENYELESAIKAN PERSAMAAN
PEMBEZAAN SEPARA PERINGKAT INTEGER DAN PECAHAN DENGAN
SENI BINA BERBILANG TERAS**

Oleh

NIK AMIR SYAFIQ BIN NIK MAZLAN

November 2024

Pengerusi: Profesor Madya Norazak bin Senu, Ph.D.

Institut : Penyelidikan Matematik

Tesis ini membincangkan kaedah lelaran Relaksasi Dipecutkan yang Diubah Suai (RDDS) secara berjujukan dan selari untuk menyelesaikan persamaan Helmholtz dan persamaan Pecahan Poisson. Tesis ini telah mengkaji penghampiran perbezaan terhingga yang berbeza, di dalam bentuk pendekatan titik dan kumpulan, yang didiskretkan kepada kaedah lelaran peringkat kedua, peringkat keempat dan peringkat pecahan. Penggunaan skema relaksasi bersama-sama dengan penghampiran perbezaan terhingga akan meningkatkan kadar tumpuan kaedah lelaran. Tesis ini akan mempertimbangkan skema RDDS yang dikembangkan sebagai lanjutan daripada skema Relaksasi Dipecutkan (RD) dan ianya dibandingkan dengan beberapa skema relaksasi yang telah dibina sebelum ini. Program selari *OpenMP* dilaksanakan bersama-sama dengan kaedah lelaran dan skema relaksasi yang akan mengurangkan masa perlakuan proses lelaran sambil mengekalkan bilangan lelaran yang sama. Bagi membantu proses ini, strategi penertiban yang berbeza digunakan pada domain penyelesaian bagi memastikan bahawa titik dalam proses lelaran tidak bersandar antara satu sama lain. Eksperimen

berangka menunjukkan bahawa skema RDDS menghasilkan bilangan lelaran paling sedikit jika dibandingkan dengan skema relaksasi yang lain. Pelaksanaan pengkomputeran selari pada semua kaedah lelaran telah mengurangkan masa pelaksanaan apabila bilangan pemproses meningkat yang juga meningkatkan kecekapan selari. Walau bagaimanapun, skema RDDS menggunakan lebih banyak titik semasa proses lelaran yang menyumbangkan kepada masa pelaksanaan yang lebih tinggi berbanding dengan beberapa skema relaksasi lain.

Kata Kunci: Persamaan Pecahan Poisson, Relaksasi dipecutkan yang diubahsuai, Terbitan pecahan Riemann - Liouville, Anggaran anjakan Grünwald.

SDG: MATLAMAT 9: Industri, Inovasi dan Infrastruktur

ACKNOWLEDGMENTS

Praised be to the All Mighty Allah for of giving me health, strength, faith, endurance, guidance, to mentioned but nothing to be able to complete this thesis. May his peace and blessing be upon the Prophet Muhammad (SAW), who guides us to the right path we follow.

First and foremost, I would like to express my sincere and deepest gratitude to the chairman of the supervisory committee, Assoc. Prof. Dr. Norazak Senu for his continuous moral support, constructive comments, assistance and guidance throughout my studies. My sincere appreciation also goes to the member of the supervisory committee, Prof. Dr. Mohamed Othman and Assoc. Prof. Dr. Nor Asilah Wati Abdul Hamid for their valuable support, knowledge and courage. A special thanks to Prof. Dr. Mohamed Othman for continuing to oversee my studies even after he has retired. Thank you from the bottom of my heart.

My sincere appreciation also goes to my parents, Nik Mazlan Mamat and Sriviowarti Noerdin, my brother, Nik Amir Hasyim, and my wife, Nur Syazana Rashid, who have always been there for me when I had problems. Their support always gave me strength and motivation to complete my studies. Thank you so much for being so supportive and understanding.

I am thankful to all INSPEN's staff and my friends for sharing valuable knowledge, encouragements and all the meaningful moments. I am also indebted to the Ministry of Higher Education for the Fundamental Research Grant Scheme (FRGS) scholarship which was led by the principal researcher, Prof. Dr. Mohamed Othman and his research team, which enables me to persue this study in the first place.

This thesis was submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfilment of the requirement for the degree of Doctor of Philosophy. The members of the Supervisory Committee were as follows:

Norazak bin Senu, Ph.D.

Associate Professor
Faculty of Science
Universiti Putra Malaysia
(Chairperson)

Mohamed bin Othman, Ph.D.

Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

Nor Asilah Wati binti Abdul Hamid, Ph.D.

Associate Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

ZALILAH MOHD SHARIFF, Ph.D.

Professor and Dean
School of Graduate Studies
Universiti Putra Malaysia

Date: 13 March 2025

TABLE OF CONTENTS

	Page
ABSTRACT	i
ABSTRAK	iii
ACKNOWLEDGEMENTS	v
APPROVAL	vi
DECLARATION	viii
LIST OF TABLES	xiii
LIST OF FIGURES	xvii
LIST OF ABBREVIATIONS	xxi
 CHAPTER	
 1 INTRODUCTION	 1
1.1 Partial Differential Equation	1
1.1.1 Helmholtz Equation	3
1.1.2 Fractional Poisson Equation	3
1.2 Iterative Method	5
1.3 Relaxation Schemes	6
1.4 Parallel Computing	7
1.4.1 Parallel Algorithm	8
1.4.2 Parallel Architecture	8
1.4.3 Type of Parallel Computers	10
1.4.4 Performance for Parallel System	12
1.5 Motivation	13
1.6 Significant of Study	15
1.7 Research Questions	16
1.8 Problem Statement	16
1.9 Objectives	17
1.10 Research Scope and Limitations	18
1.11 Thesis Structure	18
 2 RESEARCH BACKGROUND AND LITERATURE REVIEW	 21
2.1 Introduction	21
2.2 Second Order Iterative Methods	21
2.3 Fourth Order Iterative Methods	27
2.4 Fractional Order Iterative Methods	33
2.5 Accelerated Overrelaxation (AOR) Scheme	36
2.5.1 Scheme Derivation	36
2.5.2 Modified AOR (MAOR) Scheme	36
2.6 Related Works	37
2.6.1 Comparative Analysis on Second Order Iterative Methods	37
2.6.2 Comparative Analysis on Fourth Order Iterative Methods	42
2.6.3 Comparative Analysis on Fractional Order Iterative Methods	44

2.6.4	Research Gap	47
3	SECOND ORDER MAOR ITERATIVE METHODS FOR SOLVING HELMHOLTZ EQUATION	49
3.1	Introduction	49
3.2	Implementation of Sequential Algorithms for Second Order MAOR Iterative Methods	49
3.2.1	Sequential Second Order Point Iterative Methods with MAOR Scheme	49
3.2.2	Sequential Second Order Group Iterative Methods with MAOR Scheme	56
3.3	Construction of Second Order Parallel Algorithm	65
3.3.1	Parallel Second Order Point Iterative Methods with MAOR Scheme	66
3.3.2	Parallel Second Order Group Iterative Methods with MAOR Scheme	70
3.4	Experimental Results and Discussions	77
4	FOURTH ORDER MAOR ITERATIVE METHOD FOR SOLVING HELMHOLTZ EQUATION	93
4.1	Introduction	93
4.2	Implementation of Sequential Algorithms for Fourth Order MAOR Iterative Methods	93
4.2.1	Sequential Fourth Order Point Iterative Methods with MAOR Scheme	93
4.2.2	Sequential Fourth Order Group Iterative Methods with MAOR Scheme	103
4.3	Construction of Fourth Order Parallel Algorithm	116
4.3.1	Parallel Fourth Order Point Iterative Methods with MAOR Scheme	117
4.3.2	Parallel Fourth Order Group Iterative Methods with MAOR Scheme	121
4.4	Experimental Results and Discussions	131
5	FRACTIONAL ORDER MAOR ITERATIVE METHOD FOR SOLVING FRACTIONAL POISSON EQUATION	146
5.1	Introduction	146
5.2	Derivation of Sequential Algorithms for Fractional Order MAOR Iterative Methods	146
5.2.1	Sequential Fractional Order Point Iterative Methods with MAOR Scheme	146
5.2.2	Sequential Fractional Order Group Iterative Methods with MAOR Scheme	160
5.3	Construction of Fractional Order Parallel Algorithm	182
5.3.1	Parallel Fractional Order Point Iterative Methods with MAOR Scheme	183
5.3.2	Parallel Fractional Order Group Iterative Methods with MAOR Scheme	187

5.4	Experimental Results and Discussions	195
6	CONCLUSION AND FUTURE WORKS	231
6.1	Conclusion	231
6.2	Future Works	233
	REFERENCES	235
	BIODATA OF STUDENT	241
	LIST OF PUBLICATIONS	243



LIST OF TABLES

Table	Page
3.1 Computational costs for relaxation schemes in second order point iterative methods ($\sigma = M - 1$).	81
3.2 Computational costs for relaxation schemes in second order group iterative methods ($\sigma = M - 1$).	81
3.3 Number of iterations (Ite) and Error for second order iterative methods with various relaxation schemes for Problem 1.	82
3.4 Execution time (seconds), speedup and parallel efficiency for all second order point iterative methods with various relaxation schemes for Problem 1 at $M = 514$.	83
3.5 Execution time (seconds), speedup and parallel efficiency for all second order group iterative methods with various relaxation schemes for Problem 1 at $M = 514$.	84
3.6 Number of iterations (Ite) and Error for second order iterative methods with various relaxation schemes for Problem 2.	87
3.7 Computational costs of second order iterative methods with various relaxation schemes for Problems 1 and 2 at $M = 514$.	90
3.8 Execution time (seconds), speedup and parallel efficiency for all second order point iterative methods with various relaxation schemes for Problem 2 at $M = 514$.	91
3.9 Execution time (seconds), speedup and parallel efficiency for all second order group iterative methods with various relaxation schemes for Problem 2 at $M = 514$.	92
4.1 Computational costs for relaxation schemes in fourth order point iterative methods ($\sigma = M - 1$).	134
4.2 Computational costs for relaxation schemes in fourth order group iterative methods ($\sigma = M - 1$).	134

4.3	Number of iterations (Ite) and Error for fourth order iterative methods with various relaxation schemes for Problem 1.	135
4.4	Execution time (seconds), speedup and parallel efficiency for all fourth order point iterative methods with various relaxation schemes for Problem 1 at $M = 514$.	136
4.5	Execution time (seconds), speedup and parallel efficiency for all fourth order group iterative methods with various relaxation schemes for Problem 1 at $M = 514$.	137
4.6	Number of iterations (Ite) and Error for fourth order iterative methods with various relaxation schemes for Problem 2.	140
4.7	Computational costs of fourth order iterative methods with various relaxation schemes for Problems 1 and 2 at $M = 514$.	143
4.8	Execution time (seconds), speedup and parallel efficiency for all fourth order point iterative methods with various relaxation schemes for Problem 2 at $M = 514$.	144
4.9	Execution time (seconds), speedup and parallel efficiency for all fourth order group iterative methods with various relaxation schemes for Problem 2 at $M = 514$.	145
5.1	Computational costs for relaxation schemes in fractional order point iterative methods at $1 < \alpha, \beta < 2$ ($\sigma = M - 1$).	200
5.2	Computational costs for relaxation schemes in fractional order point iterative methods at $\alpha = \beta = 2$ ($\sigma = M - 1$).	200
5.3	Computational costs for relaxation schemes in fractional order group iterative methods at $1 < \alpha, \beta < 2$ ($\sigma = M - 1$).	201
5.4	Computational costs for relaxation schemes in fractional order group iterative methods at $\alpha = \beta = 2$ ($\sigma = M - 1$).	201
5.5	Number of iterations (Ite) and Error for Fractional FS and Fractional EG iterative methods with various relaxation schemes for Problem 3.	202

5.6	Number of iterations (Ite) and Error for Fractional HS and Fractional EDG iterative methods with various relaxation schemes for Problem 3.	203
5.7	Number of iterations (Ite) and Error for Fractional QS and Fractional MEG iterative methods with various relaxation schemes for Problem 3.	204
5.8	Execution time (seconds), speedup and parallel efficiency for fractional FS iterative methods with various relaxation schemes for Problem 3 at $M = 80$.	205
5.9	Execution time (seconds), speedup and parallel efficiency for fractional HS iterative methods with various relaxation schemes for Problem 3 at $M = 80$.	206
5.10	Execution time (seconds), speedup and parallel efficiency for fractional QS iterative methods with various relaxation schemes for Problem 3 at $M = 80$.	207
5.11	Execution time (seconds), speedup and parallel efficiency for fractional EG iterative methods with various relaxation schemes for Problem 3 at $M = 80$.	208
5.12	Execution time (seconds), speedup and parallel efficiency for fractional EDG iterative methods with various relaxation schemes for Problem 3 at $M = 80$.	209
5.13	Execution time (seconds), speedup and parallel efficiency for fractional MEG iterative methods with various relaxation schemes for Problem 3 at $M = 80$.	210
5.14	Computational costs of fractional order iterative methods with various relaxation schemes for Problem 3 at $M = 80$.	211
5.15	Number of iterations (Ite) and Error for fractional order iterative methods with various relaxation schemes for Problem 4.	220
5.16	Execution time (seconds), speedup and parallel efficiency for all fractional order point iterative methods with various relaxation schemes for Problem 4 at $M = 128$.	221

5.17 Execution time (seconds), speedup and parallel efficiency for all fractional order group iterative methods with various relaxation schemes for Problem 4 at $M = 128$.	222
5.18 Number of iterations (Ite) and Error for fractional order iterative methods with various relaxation schemes for Problem 5.	225
5.19 Computational costs of fractional order iterative methods with various relaxation schemes for Problems 4 and 5 at $M = 128$.	228
5.20 Execution time (seconds), speedup and parallel efficiency for all fractional order point iterative methods with various relaxation schemes for Problem 5 at $M = 128$.	229
5.21 Execution time (seconds), speedup and parallel efficiency for all fractional order group iterative methods with various relaxation schemes for Problem 5 at $M = 128$.	230

LIST OF FIGURES

Figure	Page
1.1 The general discretized solution domain Ω .	2
2.1 The stencil for FS5.	22
2.2 The stencil for HS5.	23
2.3 The stencil for QS5.	23
2.4 The stencil for EG5.	24
2.5 The stencil for EDG5.	25
2.6 The stencil for MEG5.	26
2.7 The stencil for FS9.	28
2.8 The stencil for HS9.	29
2.9 The stencil for QS9.	30
2.10 The stencil for EG9.	31
2.11 The stencil for EDG9.	32
2.12 The stencil for MEG9.	33
2.13 The stencil for FFS.	35
3.1 The solution domain for FS5.	50
3.2 The solution domain for HS5.	52
3.3 The solution domain for QS5.	54
3.4 The solution domain for EG5.	57
3.5 The solution domain for EDG5.	61
3.6 The solution domain for MEG5.	63
3.7 The execution time (in seconds) for (a) FS5; (b) HS5; (c) QS5; (d) EG5; (e) EDG5; and (f) MEG5 iterative methods on all relaxation schemes for Problem 1 at $M = 514$.	85
3.8 The speedup for (in seconds) for (a) FS5; (b) HS5; (c) QS5; (d) EG5; (e) EDG5; and (f) MEG5 iterative methods on all relaxation schemes for Problem 1 at $M = 514$.	86

3.9	The execution time (in seconds) for (a) FS5; (b) HS5; (c) QS5; (d) EG5; (e) EDG5; and (f) MEG5 iterative methods on all relaxation schemes for Problem 2 at $M = 514$.	88
3.10	The speedup for (in seconds) for (a) FS5; (b) HS5; (c) QS5; (d) EG5; (e) EDG5; and (f) MEG5 iterative methods on all relaxation schemes for Problem 2 at $M = 514$.	89
4.1	The solution domain for FS9.	94
4.2	The solution domain for HS9.	97
4.3	The solution domain for QS9.	101
4.4	The solution domain for EG9.	103
4.5	The solution domain for EDG9.	108
4.6	The solution domain for MEG9.	113
4.7	The execution time (in seconds) for (a) FS9; (b) HS9; (c) QS9; (d) EG9; (e) EDG9; and (f) MEG9 iterative methods on all relaxation schemes for Problem 1 at $M = 514$.	138
4.8	The speedup for (in seconds) for (a) FS9; (b) HS9; (c) QS9; (d) EG9; (e) EDG9; and (f) MEG9 iterative methods on all relaxation schemes for Problem 1 at $M = 514$.	139
4.9	The execution time (in seconds) for (a) FS9; (b) HS9; (c) QS9; (d) EG9; (e) EDG9; and (f) MEG9 iterative methods on all relaxation schemes for Problem 1 at $M = 514$.	141
4.10	The speedup for (in seconds) for (a) FS9; (b) HS9; (c) QS9; (d) EG9; (e) EDG9; and (f) MEG9 iterative methods on all relaxation schemes for Problem 1 at $M = 514$.	142
5.1	The solution domain for FFS.	147
5.2	The stencil for FHS.	151
5.3	The solution domain for FHS.	151
5.4	The stencil for FQS.	156
5.5	The solution domain for FQS.	157

5.6	The stencil for FEG.	161
5.7	The solution domain for FEG.	162
5.8	The stencil for FEDG.	169
5.9	The solution domain for FEDG.	170
5.10	The stencil for FMEG.	177
5.11	The solution domain for FMEG.	177
5.12	The execution time (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 1.25$.	212
5.13	The speedup for (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 1.25$.	213
5.14	The execution time (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 1.50$.	214
5.15	The speedup for (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 1.50$.	215
5.16	The execution time (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 1.75$.	216
5.17	The speedup for (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 1.75$.	217
5.18	The execution time (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 2.00$.	218

5.19	The speedup for (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 2.00$.	219
5.20	The execution time (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 4 at $M = 128$.	223
5.21	The speedup for (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 4 at $M = 128$.	224
5.22	The execution time (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 5 at $M = 128$.	226
5.23	The speedup for (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 5 at $M = 128$.	227

LIST OF ABBREVIATIONS

AOR	Accelerated Over Relaxation
CB	Chessboard
EDG	Explicit Decoupled Group
EDG5	Five-point EDG
EDG9	Nine-point EDG
EG	Explicit Group
EG5	Five-point EG
EG9	Nine-point EG
FEDG	Fractional EDG
FEG	Fractional EG
FFS	Fractional FS
FHS	Fractional HS
FMEG	Fractional MEG
FQS	Fractional QS
FS	Full Sweep
FS5	Five-point FS
FS9	Nine-point FS
GS	Gauss Seidel
HS	Half Sweep
HS5	Five-point HS
HS9	Nine-point HS
HZL	Horizontal zebra line
MAOR	Modified AOR
MEG	Modified EG
MEG5	Five-point MEG
MEG9	Nine-point MEG
MSOR	Modified SOR

PDE	Partial Differential Equation
QS	Quarter Sweep
QS5	Five-point QS
QS9	Nine-point QS
SOR	Successive Over Relaxation



CHAPTER 1

INTRODUCTION

1.1 Partial Differential Equation

Partial Differential Equations (PDEs) are a vital tool in understanding and predicting the behaviour of natural and engineered systems. In mathematical modeling of many physical phenomena, a greater number of processes can be represented through PDEs, which are a cornerstone of applied mathematics, providing the mathematical framework for describing processes that involve functions of several variables. The study of PDEs are advancing continuously, especially with the rise of new computational techniques that enhances our ability to solve more complex problems. High-performance computing has played a crucial role by enabling simulations of large-scale systems that were previously computationally infeasible. The increasing power of parallel computing and specialized hardware like a graphics processing unit has accelerated the solving of high-dimensional PDEs in various fields. Another significant development in the study of PDEs is the introduction of fractional PDEs. These extend the classical framework by incorporating non-local interactions and memory effects, which makes them particularly useful for modeling phenomena that cannot be described by traditional PDEs.

A generalised representation of a PDE appears in the form:

$$F \left(x_1, x_2, \dots, x_n, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \dots, \frac{\partial^2 u}{\partial x_i \partial x_j}, \dots \right) = 0,$$

where $u = (x_1, x_2, \dots, x_n)$ is the unknown function, $n \in \mathbb{Z}$, $1 \leq i, j \leq n$, and F is a given function. The unknowns $x_1, x_2, \dots, x_n$ might represent time or spatial variables while u represents the physical quantity of interest, such as temperature, pressure, or displacement. Generally, the PDEs are very complicated to solve analytically, hence

numerical methods are applied. The operational speed of modern computers makes it possible to obtain a fast approximate solution, taking into account a satisfactory computational error. For such equations, the theory of numerical methods has become one of the most sought-after research areas of modern science.

For simplicity, let's assume the solution domain Ω to be considered on the unit square $\Omega = \{(x, y) | (x, y) \in [0, T] \times [0, T]\}$. To solve the problem numerically, the solution domain needs to be uniformly discretized in both x and y directions with a grid size $h = \frac{T}{M}$, where M is an arbitrary positive integer. Figure 1.1 shows the solution domain discretization for partial differential equation.

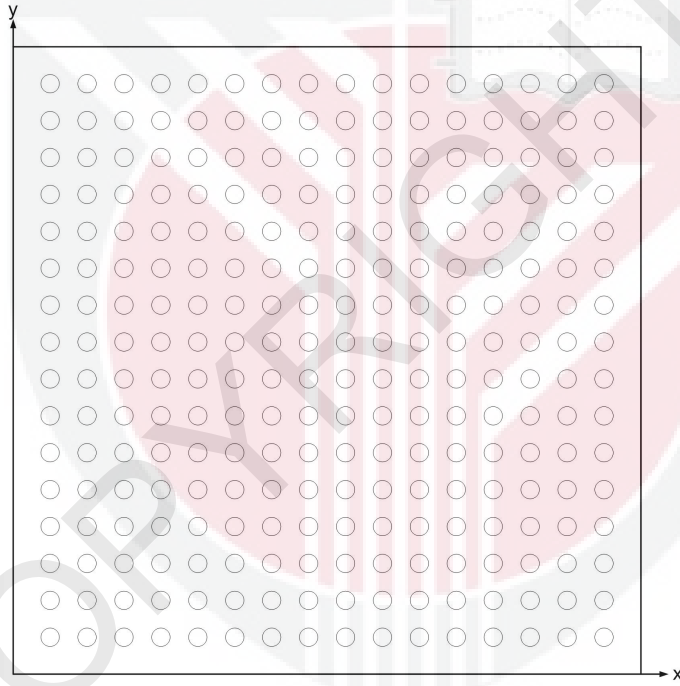


Figure 1.1: The general discretized solution domain Ω .

The points denoted as $\circ$ are points with unknown values and they represent the approximate solution the PDEs. Let's denote the points of the domain as $x_i = ih, y_j = jh$, for all $i, j = 0, 1, \dots, M - 1, M$, let $u_{i,j}$ be an approximation to $u(x_i, y_j)$ at the grid point (x_i, y_j) .

1.1.1 Helmholtz Equation

The Helmholtz equation, named after the German physicist Hermann von Helmholtz, is a fundamental PDE which has a broad application in various scientific fields. This equation is a powerful mathematical tool for analyzing wave phenomena in steady-state conditions and its widespread applicability, coupled with its theoretical depth, makes it an essential equation in the sciences and engineering. While it simplifies the more complex time-dependent wave equation by focusing on its frequency-domain characteristics, it retains essential features for analyzing how waves interact with media and boundaries. Its development is linked to the broader history of mathematical physics in the 19th century, during a period when scientists began to seek mathematical descriptions of natural phenomena involving waves, heat, and light. With modern advancements in computational techniques and fractional calculus, the Helmholtz equation has the potential to breakthrough a wide array of fields, including quantum computing, material science, and environmental studies. As we continue to explore these new frontiers, the Helmholtz equation will remain an indispensable tool for understanding the intricate behaviour of waves in complex systems.

A typical model problem for the two-dimensional Helmholtz equation on the Cartesian coordinates is formulated as follows:

$$\frac{\partial^2 u(x, y)}{\partial x^2} + \frac{\partial^2 u(x, y)}{\partial y^2} - au(x, y) = f(x, y), \quad (1.1)$$

where u and f are real functions and a is a non-negative constant. This equation simplifies the more general wave equation by assuming that the wave is steady-state, meaning it has no explicit time dependence.

1.1.2 Fractional Poisson Equation

Fractional calculus is of great importance in mathematical modeling of various phenomena in the field of engineering (Bagley and Torvik (1984); Mainardi (1997)),

physics (Hilfer (2000); Herrmann (2011)), hydrology (Benson and Wheatcraft (1984)), bio science (Magin (2006); Goyal et al. (2019)), viscoelasticity (Mainardi (2010); Bagley and Torvik (2013)), control system (Agrawal et al. (2010)), and other sciences (Zhou et al. (2010); Kumar et al. (2018); Singh et al. (2018); Tejado et al. (2020)). It is considered a powerful tool for modeling the complex dynamical systems as the fractional derivative which simultaneously possesses memory and nonlocal property can describe different nonlinear phenomena more accurately and efficiently in comparison with the integer-order derivative (Jajarmi and Baleanu (2018)).

The fractional Poisson equation is a generalization of the classical Poisson equation which incorporates the fractional derivatives to model anomalous diffusion, memory effects, or non-local interactions in a system. The two-dimensional fractional Poisson equation is a key equation in fractional calculus, extending the classical model by considering non-integer order derivatives. These types of equations are important in various scientific fields, where traditional models based on integer-order derivatives are unable to comprehensively capture complex, real-world phenomena. As the field of fractional calculus continues to evolve, the fractional Poisson equation is likely to become even more important in accurately describing complex systems across multiple disciplines.

This thesis examines the fractional Poisson equation of the form:

$$\frac{\partial^\alpha u(x, y)}{\partial x^\alpha} + \frac{\partial^\beta u(x, y)}{\partial y^\beta} = f(x, y). \quad (1.2)$$

The parameter α and β are the fractional order of the spatial derivative, where they are considered to be $1 \leq \alpha, \beta \leq 2$, and the $f(x, y)$ function in Equation (1.2) is of real values such that $f : \mathbb{R}^2 \rightarrow \mathbb{R}$.

1.2 Iterative Method

An iterative method is referred to as a wide range of techniques that use successive approximations to obtain a more accurate solutions of a problem (in most cases an equation or linear system of equations). In contrast to the direct methods, which attempt to solve the problem by a finite sequence of operations to get the exact solution, iterative methods solve the problem approximately and, generally, the number of operations is not predefined.

Iterative methods are essential for linear problems, which includes a large number of variables where direct methods would be greatly expensive and, in some cases, impractical even with the best available computing power. However, in an iterative method, to get the final solution with a satisfiable error sometimes require a large number of operations and a huge amount of time to execute.

An iterative method can be formulated as (Rakhimov (2010)):

$$u^{k+1} = Gu^k + b, \quad (1.3)$$

where u^k is the current state of some event, u^{k+1} is the new state, which is yet to be found, G is the iteration parameter and k is the iteration count. If it is considered as a stationary iteration, it means that neither G nor b depends on the number of iterations k . The convergence condition is formulated as $|u^{k+1} - u^k| \leq \varepsilon$, where ε is the error tolerance.

Eventually, Equations (1.1) and (1.2) can be approximated into the following linear system of equations:

$$Au = f, \quad (1.4)$$

where A is a large and sparse square matrix while u and f are column vectors. Matrix A can then be decomposed into:

$$A = D - L - U, \quad (1.5)$$

where D is the diagonal matrix, L is the strictly lower triangular matrix and U is the strictly upper triangular matrix. By substituting Equation (1.5) into Equation (1.4), and with some manipulation, we can obtain the general formula of the following relaxation schemes.

1.3 Relaxation Schemes

Gauss-Seidel (GS) Scheme (Young (1971))

This scheme is a classical iterative technique for solving linear systems of equations, particularly when these systems are large and sparse. It is commonly used in numerical methods to solve PDEs discretized via finite difference approximation. The GS scheme has a poor convergent rate, which is why it is considered as a benchmark scheme. The general formula for the GS scheme is defined as:

$$u^{k+1} = (D - L)^{-1} [Uu^k + b].$$

Successive Overrelaxation (SOR) Scheme (Young (1971))

This scheme is an extension from the GS scheme where it includes an additional relaxation parameter ω (with $0 < \omega < 2$) to accelerate the convergent rate. The scheme works by over-relaxing the update step, meaning that the updates to the column vector u are scaled by ω , hence significantly improving the convergent rate. An optimal relaxation parameter ω can lead to the fastest convergence, however it varies for different problems and it is often determined experimentally. The general formula for the SOR scheme is defined as:

$$u^{k+1} = (D - \omega L)^{-1} [((1 - \omega)D + \omega U) u^k + \omega b].$$

Note that when $\omega = 1$, the scheme reduces to the GS scheme.

Modified SOR (MSOR) Scheme (Kincaid and Young (1972))

The MSOR schemes is an extension of the SOR scheme and is specifically designed to improve the rate of convergence in an iterative method. When the standard SOR scheme struggles with a slow convergent rate, the MSOR scheme attempts to address these limitations by modifying the relaxation process in a more adaptive or refined manner, often adjusting the relaxation factor dynamically or employing different strategies to relax different parts of the system. In this case, the MSOR scheme implements the concept of red-black ordering strategy adds the relaxation parameters ω_1 for all red points and ω_2 for all black points (with $0 < \omega_1, \omega_2 < 2$). This process is further refined to improve the rate of convergence and reduces computational cost. Hence, the general formula for the MSOR scheme is as follows:

$$u^{k+1} = (D - \omega_{1,2}L)^{-1} \left[((1 - \omega_{1,2})D + \omega_{1,2}U) u^k + \omega_{1,2}b \right].$$

Note that when $\omega_1 = \omega_2$, the scheme reduces to the SOR scheme.

The GS, SOR, and MSOR schemes are all important for solving large systems of linear equations. While the GS is a simple and effective relaxation scheme, the SOR and MSOR scheme, in particular, offers significant improvements in the rate of convergence by incorporating relaxation techniques.

1.4 Parallel Computing

Parallel computing is a technique of using multiple processors simultaneously in a singular particular process to perform a separate portion of a task. Numerous tasks can be completed faster by sharing the work load or by executing as many tasks in parallel as possible. Parallel computing includes both parallel algorithms and parallel architectures.

1.4.1 Parallel Algorithm

Parallel algorithm is a collection of independent task modules, some of which can be executed simultaneously. Modules communicate with each other during the execution of an algorithm. The main characteristics for parallel algorithms are:

- (a) **Module granularity:** The number of computations contained in a typical module. The modules can be jobs, tasks or processors, depending upon the level which the parallelism is expressed.
- (b) **Concurrency control:** The scheme of selecting modules for execution. The control scheme must satisfy data and control dependencies so that the overall execution of algorithm is correct.
- (c) **Data mechanism:** The way which the instruction operands are used.
- (d) **Communication geometry:** The interconnection pattern between computational modules. The algorithm is regular when the interconnection pattern repeats over the set of computations, and irregular when the communications are random.
- (e) **Algorithm size:** The total number of computations that the algorithm must perform. It affects both the number of processors and the size of computer memory.

1.4.2 Parallel Architecture

Parallel computer architectures may be characterized using the following criteria:

- (a) **Processor complexity:** The computing power and the internal structure of each processing element. The processors in a homogeneous system have identical capabilities but the processors in a heterogeneous system are not identical. Their complexity varies for different architecture and it is related to algorithm granularity.
- (b) **Mode of operation:** An instruction control and data handling. It is also related to the mode of algorithm concurrence control.

- (c) **Memory structure:** A mode of operation and organization of computer memory. Memories can be accessed by using either address or data content.
- (d) **Interconnection network:** Refers to firstly the hardware connections among processors and secondly between processors and memories. It should match the algorithm communication geometry closely. Computer with simple interconnection networks is efficient only for small number of algorithms. A broad range of applications are configured for complex interconnection networks.
- (e) **Number of processors and memory size:** Indicate the amount of processors the parallel system contain and the size of the main memory. In general, more processors provide more computing power.

A standard way of classifying computer systems has been proposed by Flynn (1972). Machines are categorized according to the way that the data and instructions are inter related. Flynn grouped the parallel computers under four basic categories:

(a) Single instruction stream, single data stream (SISD)

- The traditional SISD processors are an extension of the Von Neumann model, consist of a central processing unit (CPU) and memory unit. These systems execute a single stream of instructions operating on one set of data. While inherently sequential, parallelism in SISD machines can appear through multitasking enabled by the operating system or through parallelism built into the CPU's arithmetic operations.

(b) Single instruction stream, multiple data stream (SIMD)

- In SIMD architecture, a single instruction controls multiple processing elements, each handling different data. These systems typically consist of numerous simple processors, each with its own memory to store the data being processed. SIMD machines, often referred to as massively parallel systems, operate a single program on diverse datasets simultaneously, enabling high efficiency for tasks requiring large-scale parallelism.

(c) Multiple instruction stream, single data stream (MISD)

- This architecture has several streams of instructions operated on a single stream of data simultaneously.

(d) Multiple instruction stream, multiple data stream (MIMD)

- In MIMD architecture, processors operate independently, executing different instructions on separate data. This architecture typically includes a small number of advanced processors and offers greater flexibility than SIMD systems, making it suitable for a broader range of computations. When each processor executes its own program, the system is referred to as Multiple Program Multiple Data (MPMD).

1.4.3 Type of Parallel Computers

The computing platforms needed for parallel computer are as follow:

- Shared Memory Multiprocessor System
- Message-Passing Multicomputer
- Distributed Shared Memory

Here we only introduce the first platform briefly where the *de facto* standard for writing parallel programs is OpenMP. OpenMP is a shared-memory parallel programming model based on the concept of multiple threads. It uses an explicit approach, granting programmers full control over parallelization. Following the *fork-join* model, execution begins with a single master thread running sequentially until it encounters a parallel region. At this point, the master thread forks a team of parallel threads to execute the enclosed code in parallel. Once the parallel execution is complete, the threads synchronize and terminate, leaving only the master thread to continue sequential execution.

OpenMP parallelism is primarily implemented using compiler directives embedded

in C/C++ or FORTRAN code. Its API allows dynamic adjustment of thread counts for different parallel regions. OpenMP does not specify guidelines for parallel I/O, leaving it to the programmer to ensure correct handling in multithreaded programs. Threads can cache data locally, and it is the programmer's responsibility to flush shared variables to maintain consistency when required.

There are five major different categories of OpenMP language constructs:

- **Parallel Control Construct:** This governs the control flow of the program. The key construct in this category is the `parallel` directive, which defines a parallel region. When the master thread encounters this directive, it will fork a team of slave threads that concurrently execute the tasks within the parallel region. This facilitates parallel execution of the code enclosed in the directive.
- **Work Sharing:** The work sharing constructs control how work is distributed among threads in a parallel region. This includes the following key directives: `for` defines how loop iterations are distributed among the threads; `single` requires that the guarded region can only be executed once; `section` decides how different bodies of the program code are executed by different threads; `task` specifies an explicit task that can be scheduled for execution by one of the threads in the team.
- **Data Environment:** These clauses define the "visibility" of the variables across threads. By default, all variables are shared, meaning that multiple threads access the same data in memory. This requires synchronization to ensure mutual exclusion when a shared variable is accessed by more than one thread, especially when one thread is writing to it. By using the `private` clause, each thread gets its own local copy of the variable, eliminating the need for synchronization when accessing it. This ensures that threads can work independently without interfering with each other's data.
- **Synchronization:** Ensures coordinated execution among threads. The `critical`

directive specifies a region of code that only one thread can execute at a time, ensuring mutual exclusion. The `atomic` directive guarantees that a specific memory location is updated atomically, preventing multiple threads from concurrently modifying it. Essentially, it provides a mini-critical section for a single operation. The `barrier` directive synchronizes all threads in the team, causing each thread to wait until all others reach the `barrier` before continuing execution in parallel. These synchronization mechanisms help maintain correct and efficient parallel execution in multithreaded programs.

- **Runtime Functions & Environment Variables:** OpenMP provides a set of runtime functions and environment variables so programmers can use them directly in their OpenMP program. Examples are like thread number or the number of threads in the team. Synchronization directives also have their corresponding runtime functions defined in the OpenMP runtime library.

In OpenMP, the responsibility for parallelizing a sequential program is clearly divided between the programmer and the compiler. The programmer uses simple directives to annotate the code, indicating where and how to parallelize. The compiler then handles the transformation of the sequential program into a multithreaded version, typically by integrating thread library functions into the control-flow. Synchronization directives are also mapped to native thread synchronization functions to ensure correct execution in a multithreaded environment.

1.4.4 Performance for Parallel System

The first criterion taken into consideration when the performances of the parallel systems are analyzed is the speedup used to express how many times a parallel program works faster than a sequential one, where both programs are solving the same problem. The most important reason of parallelization a sequential program is to run the program faster. The speedup formula is

$$S = \frac{T_s}{T_p},$$

where T_s is the execution time of the fastest sequential program that solves the problem. T_p is the execution time of the parallel program used to finalize the same problem.

If a parallel program is executed on a computer having p processors, the highest value that can be obtained for the speedup is equal with the number of processors from the system. The maximum speedup value could be achieved in an ideal multiprocessor system where there are no communication costs and the workload of processors is balance. In such a system, every processor needs $\frac{T_s}{p}$ time units in order to complete its job so the speedup value will be as the following:

$$S = \frac{T_s}{\frac{T_s}{p}} = p.$$

According to the Amdahl law, it is very difficult, even into an ideal parallel system, to obtain a speedup value equal with the number of processors because each program, in terms of running time, has a fraction that cannot be parallelized and has to be executed sequentially by a single processor.

The parallel efficiency quantifies the number of the valuable operations performed by the processors during the parallel program execution. The parallel efficiency could be expressed as the following:

$$E = \frac{S}{p},$$

where S is the speedup and p represent the number of the processors from the system. Due to the fact that the speedup value is lower than the number of processors, the parallel efficiency will be always located between 0 and 1.

1.5 Motivation

Many scientific problems which involve PDEs occur in fast and real time applications. Fast solutions with minimal absolute error require big sizes of discretization which in return requires a large-capacity computational memory. The foundation for solv-

ing PDEs begins with the second order finite difference approximation which includes some point and group iterative methods and has later been expanded to the fourth order finite difference approximation. However, with the growing focus on developing fractional order finite difference approximations, the implementation and performances of fractional order finite difference approximations combined with point and group iterative methods has not been extensively studied.

In addition to the iterative methods mentioned above, relaxation schemes have also been utilized to accelerate the convergence process. These schemes aim to improve the convergent rate by adjusting the relaxation parameters, which helps in speeding up the iterative solution of PDEs. The GS scheme is considered as the benchmark scheme as it does not utilize any relaxation parameter. This has later been expanded to the SOR and MSOR schemes which utilizes some relaxation parameters to increase the rate of convergence. Moreover, Hadjidimos (1978) invented the Accelerated Overrelaxation (AOR) scheme which is a two-parameter generalization of the SOR scheme and has weighted parameters θ and ω . This scheme is also shown to have a faster convergent rate compared to the SOR scheme. This scheme is then extended by Hadjidimos et al. (1992) where they introduced a theory of the Modified AOR (MAOR) scheme which is a generalization of the AOR scheme. By referring to the next chapter, the implementation and performances of MAOR scheme on the fourth order and fractional order finite difference approximations has not received much attention as compared to the second order finite difference approximation.

Therefore, the motivation of this study is to develop the MAOR scheme for the point and group iterative methods on the fourth order and fractional order finite difference approximations. These finite difference approximations includes the full sweep (FS), half sweep (HS), quarter sweep (QS), explicit group (EG), explicit decoupled group (EDG), and modified explicit group (MEG) iterative methods. Hence, this study will develop the MAOR scheme for all of the iterative methods above along with differ-

ent ordering strategies. These ordering strategies are implemented to ensure that the executed points and groups are independent from each other, which will assist in the implementation of parallel computing. Different iterative methods are ordered differently with regards to their order of discretization and will be explained in the later chapters.

1.6 Significant of Study

The Helmholtz equation remains a cornerstone in various fields such as acoustics, electromagnetics, fluid dynamics, and even medical imaging due to its ability to model wave propagation phenomena. In the current context, its significance is underscored by the increasing need for solutions to complex, dynamic systems where wave-like behaviors dominate. The equation is also vital for describing the behavior of electromagnetic waves, especially in scenarios like optical fibers, radar technology, and wireless communications (Shah et al. (2021)). Moreover, as computational power grows and the understanding of complex systems deepens, the Helmholtz equation's role in accurately simulating wave phenomena becomes increasingly crucial. This will allow for more efficient and accurate solving of the equation, particularly for high-dimensional or irregular geometries, which are essential in advanced simulation environments.

The fractional Poisson equation has gained significant attention in recent years due to its applications in complex systems that exhibit non-locality and memory effects. As mentioned earlier, this equation generalizes the classical Poisson equation by incorporating fractional derivatives, allowing for more description of phenomena such as anomalous diffusion, viscoelasticity, and image processing. Recent studies highlight its versatility and potential in solving high-dimensional and complex physical systems (Boev and Georgiev (2023); Bailo et al. (2024)). The equation's significance also extends to quantum mechanics, where it helps describe systems with fractal or irregular structures (Chen et al. (2020)).

The MAOR scheme is an advanced iterative technique that can significantly improve the convergent rate when solving large systems of linear equations, particularly those arising from discretized PDEs. The scheme improves upon traditional relaxation scheme by optimizing the relaxation parameter. By adjusting this parameter dynamically or strategically, the MAOR scheme accelerates the convergence of the iterative solution process (Young (1971)). Additionally, the MAOR scheme can be easily applied to a wide variety of problems in fields such as fluid dynamics, structural analysis, and computational electromagnetics (Saad (2003)). Additionally, the method can be implemented in parallel computing environments, making it highly efficient for large-scale simulations.

1.7 Research Questions

This thesis aims to address the following research questions:

1. How can the MAOR scheme be effectively formulated for second order, fourth order, and fractional order iterative methods?
2. How does the performance of the MAOR scheme compare with existing relaxation schemes for solving second order, fourth order, and fractional order iterative methods?
3. What ordering strategies can be employed during parallelization to optimize the computational efficiency of the MAOR schemes for second order, fourth order, and fractional order iterative methods?
4. How does the parallel performance of the MAOR algorithms scale with increasing problem size and number of processors?

1.8 Problem Statement

The Helmholtz equation and the fractional Poisson equation are foundational in modelling and solving complex problems in various fields. Despite the importance of these

equations, solving them efficiently and accurately remains a challenge, particularly for large-scale problems requiring high-resolution discretization and computational resources. Traditional iterative methods often suffer from slow convergence rates, especially when applied to higher order and fractional formulations of these equations, which can lead to significant computational overheads.

The MAOR scheme has shown promise in improving the convergence rates of iterative solvers for partial differential equations. However, its potential remains underexplored for advanced formulations, such as fourth order approximations of the Helmholtz equation and fractional order approximations of the Poisson equation. Moreover, the increasing demand for high-performance computing highlights the necessity for parallel algorithms to handle these computationally intensive tasks efficiently.

This thesis seeks to address these gaps by developing and optimizing MAOR schemes tailored to fourth order, and fractional order iterative methods. Additionally, constructing parallel algorithms for these schemes is crucial to enhance computational performance and scalability, making them viable for real-world applications. By bridging these methodological and computational challenges, this research aims to contribute to the development of faster and resource-efficient solutions for critical scientific and engineering problems.

1.9 Objectives

This thesis will be focused on the following objectives:

1. to implement the MAOR scheme on second order and fourth order iterative method for solving the Helmholtz equation;
2. to develop the MAOR scheme on fractional order iterative method for solving the fractional Poisson equation;
3. to construct the parallel algorithms for all the iterative methods mentioned above;

1.10 Research Scope and Limitations

This thesis focuses on the implementation, development, and optimization of the MAOR scheme for solving selected PDEs using advanced numerical techniques. The solution domain will be discretized using their respective finite difference approximations and solved using the point and group iterative methods. The thesis investigates the application of the MAOR scheme on second order and fourth order iterative methods to solve the Helmholtz equation, aiming to improve the convergent rate for wave-based PDEs. The thesis then extends the MAOR scheme to fractional-order iterative methods, targeting the fractional Poisson equation, which is essential in modelling anomalous diffusion and other phenomena requiring non-local interactions. Finally, this thesis will construct and analyse the parallel algorithms for all the iterative methods, emphasizing on computational performance and the ability to handle large-scale problems effectively.

Some limitations for this thesis includes the solution domain for second order and fourth order iterative methods will be considered for group cases of MEG iterative method. This means that the EG and EDG solution domain will have some extra blocks of two points. However, for the fractional order iterative methods, the solution domain will be considered according to where the test problem is referred to. Furthermore, this thesis is also limited to the Helmholtz equation and fractional Poisson equation, excluding other PDEs that may also benefit from MAOR schemes and parallel processing. The limited number of test problems presented later reflects the relatively scarce research focus currently directed toward these specific equations.

1.11 Thesis Structure

The remainder of the thesis are organized as follows. Chapter 2 derives the second order, fourth order and fractional order of discretization for the FS, HS, QS, EG, EDG and MEG iterative methods. The fractional order will be discretized with the Shifted Grünwald Estimate. Furthermore, the Accelerated Overrelaxation (AOR) and Modi-

fied AOR (MAOR) Schemes will also be presented. Finally, recent research related to these subjects of this thesis is reviewed and analyzed at the end of the chapter.

Chapter 3 is devoted to the implementation of second order MAOR iterative method for solving Helmholtz equation. The implementation will include several ordering strategies to the iterative methods. Next, the sequential algorithms for all of the second order MAOR iterative methods are presented. This shows how the iterative methods are executed using various ordering strategies. This is followed by the construction of the parallel algorithms for second order MAOR iterative methods. This will include a detail explanation on how the OpenMP parallel programming works. Two problems are used to test and discuss the performance of the second order MAOR iterative methods when compared with other relaxation schemes.

The implementation of fourth order MAOR iterative method for solving Helmholtz equation is given in Chapter 4. The implementation will include several ordering strategies to the iterative methods. Next, the sequential algorithms for all of the fourth order MAOR iterative methods are presented. This shows how the iterative methods are executed using various ordering strategies. This is followed by the construction of the parallel algorithms for fourth order MAOR iterative methods. This will include a detail explanation on how the OpenMP parallel programming works. Two problems are used to test and discuss the performance of the fourth order MAOR iterative methods when compared with other relaxation schemes.

Chapter 5 is focused on the development of fractional order MAOR iterative method for solving fractional Poisson equation. The equation is discretized using shifted Grünwald estimate and will include several ordering strategies to the iterative methods. Next, the sequential algorithms for all of the fractional order MAOR iterative methods are presented. This shows how the iterative methods are executed using various ordering strategies. This is followed by the construction of the parallel algorithms for

fractional order MAOR iterative methods. This will include a detail explanation on how the OpenMP parallel programming works. Three problems are used to test and discuss the performance of the fractional order MAOR iterative methods when compared with other relaxation schemes.

Finally, the conclusions and recommendations for future research are given in Chapter 6.



CHAPTER 2

RESEARCH BACKGROUND AND LITERATURE REVIEW

2.1 Introduction

This chapter is devoted to the discretization of second order, fourth order and fractional order iterative methods. The second and fourth order are discretized using the Taylor series expansion while the fractional order is discretized using fractional finite difference with shifted Grünwald estimate. All order of discretization will derive their respective point and group iterative methods. The second order iterative method is referred to as the five-point iterative method because only five points are utilized in the structure of its finite difference discretization. Similarly, the fourth order iterative method is referred to as the nine-point iterative method because only nine points are utilized in the structure of its finite difference discretization. Their stencils are provided for visual assistant as to which points is utilized with respect to each iterative method. Then, the chapter continues with the scheme derivation of Accelerated Overrelaxation (AOR) and Modified AOR (MAOR) schemes. Furthermore, the chapter presents the comparative analysis on iterative methods by analyzing previous researches that are related to this thesis. These researches are listed and categorized according to the order of discretization. Finally, the chapter ends with the research gap which discuss the iterative methods and schemes that are yet to be done.

2.2 Second Order Iterative Methods

This iterative method is used to solve Equation (1.1). It is developed by using the Taylor series expansion up to the second order. The expansion are applied to the points that are horizontally and vertically adjacent to $u_{i,j}$:

$$\begin{aligned}
u_{i+1,j} &= u_{i,j} + hu_x + \frac{h^2}{2}u_{xx}, & u_{i,j+1} &= u_{i,j} + hu_y + \frac{h^2}{2}u_{yy}, \\
u_{i-1,j} &= u_{i,j} - hu_x + \frac{h^2}{2}u_{xx}, & u_{i,j-1} &= u_{i,j} - hu_y + \frac{h^2}{2}u_{yy}.
\end{aligned}$$

By adding all these terms yields

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} = 4u_{i,j} + h^2(u_{xx} + u_{yy}), \quad (2.1)$$

and taking into consideration Equation (1.1), the second order full sweep iterative method can be obtained.

Five-point Full Sweep (FS5) (Saudi and Sulaiman (2017); Dahalan et al. (2019))

From Equation (2.1), we can get

$$\begin{aligned}
u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} &= 4u_{i,j} + h^2(au_{i,j} + f_{i,j}) \\
&= (4 + \alpha h^2)u_{i,j} + h^2 f_{i,j}.
\end{aligned}$$

After some rearrangement, the equation above can be simplified into

$$u_{i,j} = \frac{1}{\rho_1}(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - h^2 f_{i,j}), \quad (2.2)$$

where $\rho_1 = 4 + \alpha h^2$.

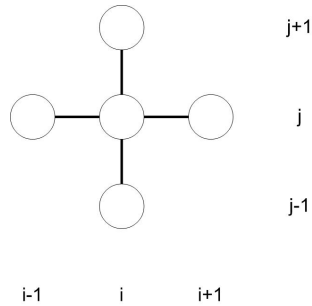


Figure 2.1: The stencil for FS5.

Five-point Half Sweep (HS5) (Akhir et al. (2011b); Eng et al. (2017))

By taking Equation (2.2) and rotating it 45° anti-clockwise, which implies that the grid spacing becomes $\sqrt{2}h$, will yield

$$u_{i,j} = \frac{1}{\rho_2}(u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i+1,j+1} - 2h^2 f_{i,j}), \quad (2.3)$$

where $\rho_2 = 4 + 2\alpha h^2$.

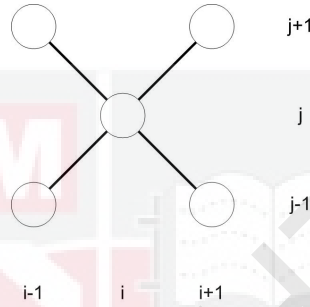


Figure 2.2: The stencil for HS5.

Five-point Quarter Sweep (QS5) (Akhir et al. (2012a); Hoe and Hasan (2013))

Equation (2.1) also can be discretized using the same formulae with grid spacing $2h$ which leads to

$$u_{i,j} = \frac{1}{\rho_3}(u_{i-2,j} + u_{i+2,j} + u_{i,j-2} + u_{i,j+2} - 4h^2 f_{i,j}), \quad (2.4)$$

where $\rho_3 = 4 + 4\alpha h^2$.

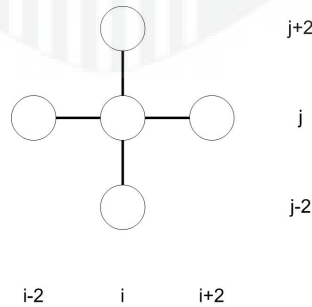


Figure 2.3: The stencil for QS5.

Five-point Explicit Group (EG5) (Akhir et al. (2011c); Saudi and Sulaiman (2012a))

The formulation of EG5 iterative method considers a complete group of four points (4×4) using Equation (2.2), which can be generally expressed as

$$\begin{bmatrix} \rho_1 & -1 & -1 & 0 \\ -1 & \rho_1 & 0 & -1 \\ -1 & 0 & \rho_1 & -1 \\ 0 & -1 & -1 & \rho_1 \end{bmatrix} \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix},$$

where

$$\begin{aligned} S_1 &= u_{i,j-1} + u_{i-1,j} - h^2 f_{i,j}, & S_2 &= u_{i+2,j} + u_{i+1,j-1} - h^2 f_{i+1,j}, \\ S_3 &= u_{i-1,j+1} + u_{i,j+2} - h^2 f_{i,j+1}, & S_4 &= u_{i+2,j+1} + u_{i+1,j+2} - h^2 f_{i+1,j+1}. \end{aligned}$$

By establishing the inverse matrix of the coefficient matrix and manipulating it, the equation above can be rewritten as

$$\begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix} = \frac{1}{Q_1} \begin{bmatrix} \rho_1^2 - 2 & \rho_1 & \rho_1 & 2 \\ \rho_1 & \rho_1^2 - 2 & 2 & \rho_1 \\ \rho_1 & 2 & \rho_1^2 - 2 & \rho_1 \\ 2 & \rho_1 & \rho_1 & \rho_1^2 - 2 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}, \quad (2.5)$$

where $Q_1 = \rho_1 (\rho_1^2 - 4)$.

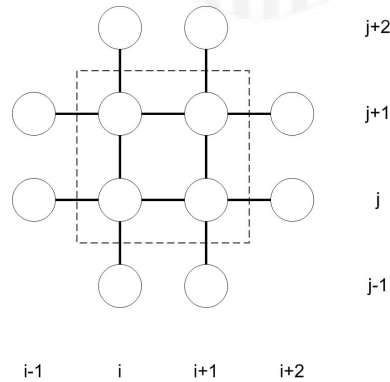


Figure 2.4: The stencil for EG5.

Five-point Explicit Decoupled Group (EDG5) (Yousif and Martins (2008))

The formulation of EDG5 iterative method considers a complete group of two points (2×2) using Equation (2.3), which can be generally expressed as

$$\begin{bmatrix} \rho_2 & -1 \\ -1 & \rho_2 \end{bmatrix} \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix},$$

where

$$S_1 = u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} - 2h^2 f_{i,j},$$

$$S_2 = u_{i+2,j} + u_{i,j+2} + u_{i+2,j+2} - 2h^2 f_{i+1,j+1}.$$

By establishing the inverse matrix of the coefficient matrix and manipulating it, the equation above can be rewritten as

$$\begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix} = \frac{1}{Q_2} \begin{bmatrix} \rho_2 & 1 \\ 1 & \rho_2 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}, \quad (2.6)$$

where $Q_2 = (\rho_2^2 - 1)$.

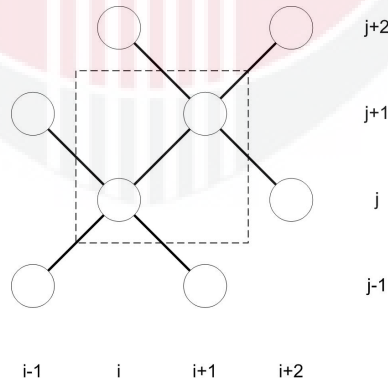


Figure 2.5: The stencil for EDG5.

Five-point Modified EG (MEG5) (Othman and Abdullah (2000a))

The formulation of MEG5 iterative method considers a complete group of four points (4×4) using Equation (2.4), which can be generally expressed as

$$\begin{bmatrix} \rho_3 & -1 & -1 & 0 \\ -1 & \rho_3 & 0 & -1 \\ -1 & 0 & \rho_3 & -1 \\ 0 & -1 & -1 & \rho_3 \end{bmatrix} \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix},$$

where

$$\begin{aligned} S_1 &= u_{i,j-2} + u_{i-2,j} - 4h^2 f_{i,j}, & S_2 &= u_{i+4,j} + u_{i+2,j-2} - 4h^2 f_{i+2,j}, \\ S_3 &= u_{i-2,j+2} + u_{i,j+4} - 4h^2 f_{i,j+2}, & S_4 &= u_{i+4,j+2} + u_{i+2,j+4} - 4h^2 f_{i+2,j+2}. \end{aligned}$$

By establishing the inverse matrix of the coefficient matrix and manipulating it, the equation above can be rewritten as

$$\begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix} = \frac{1}{Q_3} \begin{bmatrix} \rho_3^2 - 2 & \rho_3 & \rho_3 & 2 \\ \rho_3 & \rho_3^2 - 2 & 2 & \rho_3 \\ \rho_3 & 2 & \rho_3^2 - 2 & \rho_3 \\ 2 & \rho_3 & \rho_3 & \rho_3^2 - 2 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}, \quad (2.7)$$

where $Q_3 = \rho_3 (\rho_3^2 - 4)$.

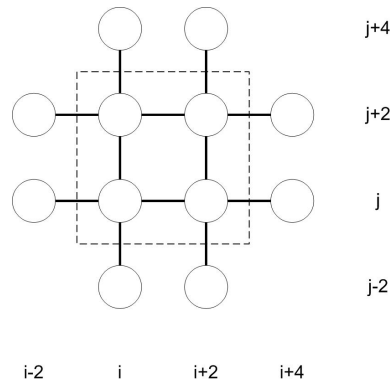


Figure 2.6: The stencil for MEG5.

2.3 Fourth Order Iterative Methods

This iterative method is also used to solve Equation (1.1). It is developed by using the Taylor series expansion up to the fourth order. For simplicity, we will only display the even terms of the expansion. The expansion are applied to the points that are horizontally and vertically adjacent to $u_{i,j}$:

$$\begin{aligned} u_{i+1,j} &= u_{i,j} + \frac{h^2}{2}u_{xx} + \frac{h^4}{24}u_{xxxx}, & u_{i,j+1} &= u_{i,j} + \frac{h^2}{2}u_{yy} + \frac{h^4}{24}u_{yyyy}, \\ u_{i-1,j} &= u_{i,j} + \frac{h^2}{2}u_{xx} + \frac{h^4}{24}u_{xxxx}, & u_{i,j-1} &= u_{i,j} + \frac{h^2}{2}u_{yy} + \frac{h^4}{24}u_{yyyy}. \end{aligned}$$

By adding all these terms and taking into consideration Equation (1.1) yields

$$\begin{aligned} u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} \\ = (4 + \alpha h^2)u_{i,j} + h^2 f_{i,j} + \frac{h^4}{12}(u_{xxxx} + u_{yyyy}). \end{aligned} \quad (2.8)$$

The expansion are also applied to the points that are diagonally adjacent to $u_{i,j}$:

$$\begin{aligned} u_{i+1,j+1} &= u_{i,j} + \frac{h^2}{2}(u_{xx} + 2u_{xy} + u_{yy}) \\ &\quad + \frac{h^4}{24}(u_{xxxx} + 4u_{xxxy} + 6u_{xxyy} + 4u_{xyyy} + u_{yyyy}), \\ u_{i+1,j-1} &= u_{i,j} + \frac{h^2}{2}(u_{xx} - 2u_{xy} + u_{yy}) \\ &\quad + \frac{h^4}{24}(u_{xxxx} - 4u_{xxxy} + 6u_{xxyy} - 4u_{xyyy} + u_{yyyy}), \\ u_{i-1,j+1} &= u_{i,j} + \frac{h^2}{2}(u_{xx} - 2u_{xy} + u_{yy}) \\ &\quad + \frac{h^4}{24}(u_{xxxx} - 4u_{xxxy} + 6u_{xxyy} - 4u_{xyyy} + u_{yyyy}), \\ u_{i-1,j-1} &= u_{i,j} + \frac{h^2}{2}(u_{xx} + 2u_{xy} + u_{yy}) \\ &\quad + \frac{h^4}{24}(u_{xxxx} + 4u_{xxxy} + 6u_{xxyy} + 4u_{xyyy} + u_{yyyy}). \end{aligned}$$

By adding all these terms and taking into consideration Equation (1.1) yields

$$\begin{aligned} & u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1} \\ &= (4 + \alpha h^2)u_{i,j} + 2h^2 f_{i,j} + \frac{h^4}{6}(u_{xxxx} + 6u_{xxyy} + u_{yyyy}). \end{aligned} \quad (2.9)$$

By adding Equations (2.8) and (2.9) and with some manipulation, and taking into consideration Equation (1.1), the fourth order full sweep iterative method can be obtained.

Nine-point Full Sweep (FS9)(Collatz (1960))

From Equations (2.8) and (2.9), we can get

$$\begin{aligned} & u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i+1,j+1} \\ &+ q_4(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) - \rho_4 u_{i,j} \\ &= \frac{1}{2}h^2(8f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}), \end{aligned}$$

where $q_4 = 4 - \frac{\alpha}{2}h^2$ and $\rho_4 = 20 + 4\alpha h^2$. The equation above is called the *Mehrstellenverfahren* and is often referred to as the fourth order compact scheme since only the nearest (compact) nine grid points are involved in the approximation scheme. After some rearrangement, the equation above can be simplified into

$$\begin{aligned} u_{i,j} = \frac{1}{\rho_4} & \left(u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i+1,j+1} \right. \\ & \left. + q_4(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) - \frac{1}{2}h^2 F1_{i,j} \right), \end{aligned} \quad (2.10)$$

where $F1_{i,j} = 8f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}$.

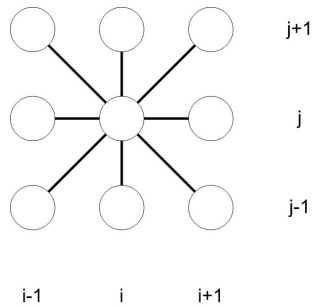


Figure 2.7: The stencil for FS9.

Nine-point Half Sweep (HS9) (Saudi and Sulaiman (2012b); Ling and Ali (2014))

By taking Equation (2.10) and rotating it 45° anti-clockwise, which implies that the grid spacing becomes $\sqrt{2}h$, will yield

$$u_{i,j} = \frac{1}{\rho_5} \left(u_{i-2,j} + u_{i+2,j} + u_{i,j-2} + u_{i,j+2} + q_5(u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i+1,j+1}) - h^2 F_{2i,j} \right), \quad (2.11)$$

where $q_5 = 4 - \alpha h^2$, $\rho_5 = 20 + 8\alpha h^2$ and $F_{2i,j} = 8f_{i,j} + f_{i-1,j-1} + f_{i+1,j-1} + f_{i-1,j+1} + f_{i+1,j+1}$.

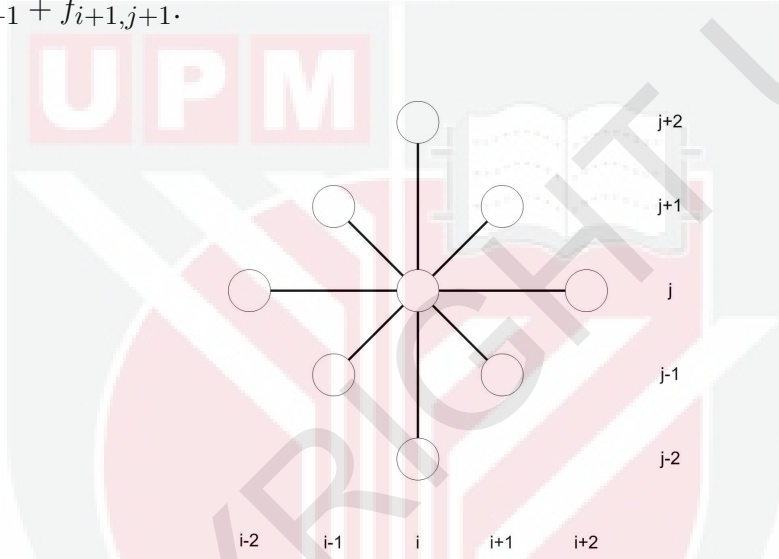


Figure 2.8: The stencil for HS9.

Note that Equation (2.11) cannot solve points that are near the boundary. Hence, by considering a higher derivative, another rotated five point can be derived as

$$u_{i,j} = \frac{1}{q_5^*} \left(u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i+1,j+1} - \frac{h^2}{6} F_{2i,j} \right), \quad (2.12)$$

where $q_5^* = 4 + 2\alpha h^2$. Equation (2.12) is associated with the rotated nine point of fourth order, specifically to solve the points near the boundary.

Nine-point Quarter Sweep (QS9) (Ling and Ali (2013))

Equations (2.8) and (2.9) also can be discretized using the same formulae with grid spacing $2h$ which leads to

$$u_{i,j} = \frac{1}{\rho_6} \left(u_{i-2,j-2} + u_{i+2,j-2} + u_{i-2,j+2} + u_{i+2,j+2} + q_6(u_{i-2,j} + u_{i+2,j} + u_{i,j-2} + u_{i,j+2}) - 2h^2 F3_{i,j} \right), \quad (2.13)$$

where $q_6 = 4 - 2\alpha h^2$, $\rho_6 = 20 + 16\alpha h^2$ and $F3_{i,j} = 8f_{i,j} + f_{i+2,j} + f_{i-2,j} + f_{i,j+2} + f_{i,j-2}$.

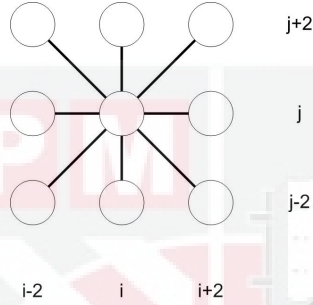


Figure 2.9: The stencil for QS9.

Nine-point Explicit Group (EG9) (Saudi and Sulaiman (2012c,d))

The formulation of EG9 iterative method considers a complete group of four points (4×4) using Equation (2.10), which can be generally expressed as

$$\begin{bmatrix} \rho_4 & -q_4 & -q_4 & -1 \\ -q_4 & \rho_4 & -1 & -q_4 \\ -q_4 & -1 & \rho_4 & -q_4 \\ -1 & -q_4 & -q_4 & \rho_4 \end{bmatrix} \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix},$$

where

$$\begin{aligned} S_1 &= u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} + q_4(u_{i-1,j} + u_{i,j-1}) - \frac{1}{2}h^2 F1_{i,j}, \\ S_2 &= u_{i,j-1} + u_{i+2,j+1} + u_{i+2,j-1} + q_4(u_{i+2,j} + u_{i+1,j-1}) - \frac{1}{2}h^2 F1_{i+1,j}, \\ S_3 &= u_{i-1,j} + u_{i-1,j+2} + u_{i+1,j+2} + q_4(u_{i-1,j+1} + u_{i,j+2}) - \frac{1}{2}h^2 F1_{i,j+1}, \\ S_4 &= u_{i,j+2} + u_{i+2,j+2} + u_{i+2,j} + q_4(u_{i+2,j+1} + u_{i+1,j+2}) - \frac{1}{2}h^2 F1_{i+1,j+1}. \end{aligned}$$

By establishing the inverse matrix of the coefficient matrix and manipulating it, the

equation above can be rewritten as

$$\begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix} = \frac{1}{Q_4} \begin{bmatrix} R_1 & R_2 & R_2 & R_3 \\ R_2 & R_1 & R_3 & R_2 \\ R_2 & R_3 & R_1 & R_2 \\ R_3 & R_2 & R_2 & R_1 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}, \quad (2.14)$$

where $Q_4 = (\rho_4 + 1)(\rho_4^2 - 4q_4^2 - 2\rho_4 + 1)$, $R_1 = \rho_4^2 - 2q_4^2 - \rho_4$, $R_2 = q_4(\rho_4 + 1)$ and $R_3 = 2q_4^2 + \rho_4 - 1$.

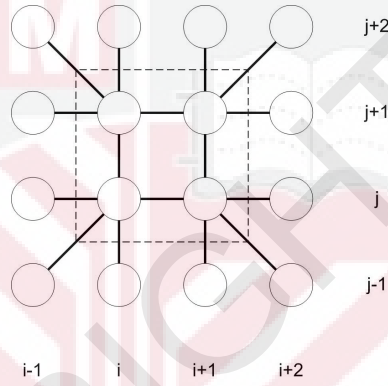


Figure 2.10: The stencil for EG9.

Nine-point Explicit Decoupled Group (EDG9) (Ali and Ping (2016))

The formulation of EDG5 iterative method considers a complete group of two points (2×2) using Equation (2.11), which can be generally expressed as

$$\begin{bmatrix} \rho_5 & -q_5 \\ -q_5 & \rho_5 \end{bmatrix} \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix},$$

where

$$\begin{aligned} S_1 &= u_{i,j+2} + u_{i-2,j} + u_{i,j-2} + u_{i+2,j} \\ &\quad + q_5(u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1}) - h^2 F_{2i,j}, \\ S_2 &= u_{i+1,j+3} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i+3,j+1} \\ &\quad + q_5(u_{i,j+2} + u_{i+2,j} + u_{i+2,j+2}) - h^2 F_{2i+1,j+1} \end{aligned}$$

By establishing the inverse matrix of the coefficient matrix and manipulating it, the equation above can be rewritten as

$$\begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix} = \frac{1}{Q_5} \begin{bmatrix} \rho_5 & q_5 \\ q_5 & \rho_5 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}, \quad (2.15)$$

where $Q_5 = (\rho_5^2 - q_5^2)$.

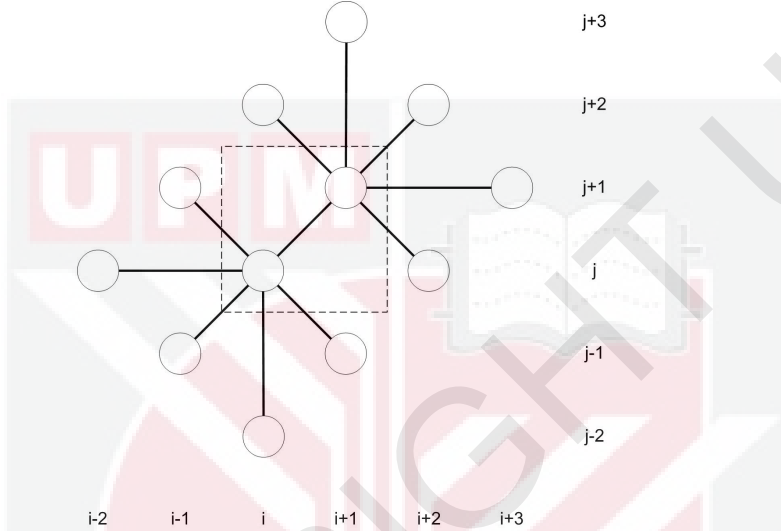


Figure 2.11: The stencil for EDG9.

Note that Equation (2.15) also cannot solve points that are near the boundary. Hence, Equation (2.12) is considered to solve the points near the boundary.

Nine-point Modified EG (MEG9) (Ling and Ali (2013))

The formulation of MEG9 iterative method considers a complete group of four points (4×4) using Equation (2.13), which can be generally expressed as

$$\begin{bmatrix} \rho_6 & -q_6 & -q_6 & -1 \\ -q_6 & \rho_6 & -1 & -q_6 \\ -q_6 & -1 & \rho_6 & -q_6 \\ -1 & -q_6 & -q_6 & \rho_6 \end{bmatrix} \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix},$$

Riemann-Liouville fractional derivatives

The Riemann-Liouville fractional derivative of order α is defined as

$$D_x^\alpha f(x) = \frac{d^\alpha f(x)}{dx^\alpha} = \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dx^n} \int_0^x (x-t)^{n-\alpha-1} f(t) dt,$$

such that the function $f : \mathbb{R} \rightarrow \mathbb{R}$ has continuous derivatives of the integer-order n , where $(n-1) \leq \alpha \leq n$ (Podlubny (1999); Agrawal (2002)).

Shifted Grünwald Estimate

The shifted Grünwald formula for the case of $1 \leq \alpha \leq 2$ is defined as

$$\frac{d^\alpha f}{dx^\alpha} = \lim_{M \rightarrow \infty} \frac{1}{h^\alpha} \sum_{k=0}^M g_{x_k} \cdot f(x - (k-1)h),$$

that which further defines the shifted Grünwald estimate to the fractional derivatives

$$\frac{d^\alpha f}{dx^\alpha} = \frac{1}{h^\alpha} \sum_{k=0}^M g_{x_k} \cdot f(x - (k-1)h) + O(h^\alpha),$$

given that the normalized Grünwald weight $g_{(\alpha,k)}$ is defined as

$$g_{(\alpha,k)} = (-1)^k \frac{\Gamma(\alpha+1)}{\Gamma(k+1)\Gamma(\alpha-k+1)} \quad \text{for } k = 0, 1, 2, \dots,$$

or

$$g_{(\alpha,0)} = 1, g_{(\alpha,1)} = -\alpha \text{ and } g_{(\alpha,k)} = (-1)^k \frac{\alpha(\alpha-1)\dots(\alpha-k+1)}{k!} \quad \text{for } k = 2, 3, 4, \dots,$$

where Γ is the gamma function. The normalized weights are dependent on the index k and the order α (Miller and Ross (1993); Samko et al. (1993); Podlubny (1999)).

Fractional Full Sweep (FFS) (Borhanifar and Valizadeh (2013))

By discretizing the equation, both the x -derivatives and y -derivatives are replaced with

shifted Grünwald finite differences, which yields

$$\frac{1}{h^\alpha} \sum_{s=0}^{i+1} g(\alpha, s) \cdot u_{i-s+1, j} + \frac{1}{h^\beta} \sum_{z=0}^{j+1} g(\beta, z) \cdot u_{i, j-z+1} = f_{i, j}$$

and can be rearranged to:

$$u_{i, j} = \frac{1}{P} \left(h^\beta u_{i+1, j} + h^\beta \sum_{s=2,3,4,\dots}^{i+1} g(\alpha, s) \cdot u_{i-s+1, j} + h^\alpha u_{i, j+1} + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g(\beta, z) \cdot u_{i, j-z+1} - h^\alpha h^\beta f_{i, j} \right). \quad (2.17)$$

where $P = \beta h^\alpha + \alpha h^\beta$.

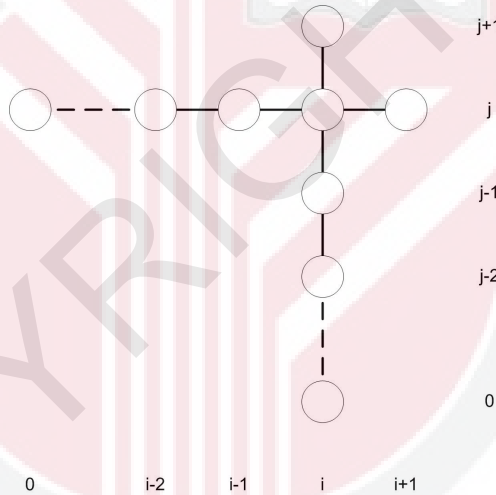


Figure 2.13: The stencil for FFS.

The thesis will extend the foundational work on the FFS method by deriving new fractional iterative methods: Fractional Half Sweep (FHS), Fractional Quarter Sweep (FQS), Fractional Explicit Group (FEG), Fractional Explicit Decoupled Group (FEDG), and Fractional Modified EG (FMEG). These derivations will be presented in the latter part of the thesis, building upon the principles and techniques established in the FFS method. This extension aims to enhance computational efficiency and flexibility, enabling faster convergence rates and broader applicability in solving the Fractional Poisson equation across different fractional orders.

2.5 Accelerated Overrelaxation (AOR) Scheme

2.5.1 Scheme Derivation

The AOR scheme which is a two-parameter generalization of the SOR scheme and has relaxation parameters θ and ω . Hence, the general formula for the AOR scheme is as follows;

$$u^{k+1} = (D - \theta L)^{-1} \left[[(1 - \omega) D + (\omega - \theta) L + \omega U] u^k + \omega b \right]. \quad (2.18)$$

2.5.2 Modified AOR (MAOR) Scheme

The theory of the MAOR scheme has been introduced by Hadjidimos et al. (1992) which is based which is the generalization of the AOR scheme. They stated that, this scheme reduces the extrapolation of previous schemes with unique parameters corresponding to the row blocks of matrices for specific choices of the acceleration and relaxation matrices. Furthermore, they also provide the convergence conditions for two-cyclic matrices. The idea of a two-cyclic matrices for the MAOR scheme is identical to the concept of Red-Black ordering where the relaxation parameters θ and ω are both defined as;

$$\theta = \begin{bmatrix} \theta_1 I_1 & 0 \\ 0 & \theta_2 I_2 \end{bmatrix}, \quad \omega = \begin{bmatrix} \omega_1 I_1 & 0 \\ 0 & \omega_2 I_2 \end{bmatrix}$$

with θ_1 and ω_1 denoted as the relaxation parameters on the red points, θ_2 and ω_2 denoted as the relaxation parameters on the black points, and I_1 and I_2 are identity matrices. Hence, the general formula for the MAOR scheme is as follows;

$$u^{k+1} = (D - \theta_{1,2} L)^{-1} \left[[(1 - \omega_{1,2}) D + (\omega_{1,2} - \theta_{1,2}) L + \omega_{1,2} U] u^k + \omega_{1,2} b \right] \quad (2.19)$$

2.6 Related Works

In the context of iterative methods for solving PDEs, the FS method is commonly regarded as a benchmark due to its utilisation of all domain points during the iterative process. In comparison, the HS method demonstrates faster convergence by employing only half of the domain points during each iteration. Similarly, the QS method exhibits even faster convergence by further reducing the number of active domain points to one-quarter during the iterative process. To enhance the performance of these point iterative methods, they have been extended into group iterative methods tailored to their respective sweeping approaches. Specifically, the EG, EDG, and MEG methods serve as extensions of the FS, HS, and QS methods, respectively. These group methods generally offer an improved convergence rate compared to their corresponding point iterative methods. Furthermore, relaxation schemes such as the GS, SOR, MSOR, AOR, and MAOR have been integrated with these iterative methods. These relaxation techniques are implemented to further enhance the convergence rate of the iterative processes. Comparative analyses of these relaxation schemes across various iterative methods have been conducted to evaluate their effectiveness in improving performance metrics such as convergence speed and computational efficiency. To ease the flow of this section, the related works will be organized according to second order, fourth order and fractional order iterative methods.

2.6.1 Comparative Analysis on Second Order Iterative Methods

The FS5 iterative method has been investigated for various applications, with notable enhancements achieved through the incorporation of advanced relaxation schemes. For instance, Saudi and Sulaiman (2017) utilized the MSOR scheme to the FS5 iterative method for solving the two-dimensional Laplace equation for Robot Path Planning. The study demonstrated that the FS5 MSOR scheme outperformed the FS5 GS and FS5 SOR methods in terms of convergence rate. Moreover, the FS5 MAOR method was developed and utilized to solve the two-dimensional Laplace equation for Robot Path Planning (Dahalan et al. (2019)) and the two-dimensional Poisson Image Blend-

ing Problem (Saad et al. (2021)). Both studies highlighted the superior convergence rate of the FS5 MAOR method when compared to FS5 SOR, FS5 MSOR, and FS5 AOR methods. Despite these improvements, the FS5 method is generally acknowledged to have the slowest convergence rate compared to other iterative methods, underscoring the need for continued refinement and exploration of alternative strategies.

An advancement over the FS5 iterative method is the HS5 iterative method, which has demonstrated a faster convergence rate. This improvement is substantiated by Akhir et al. (2011b), who employed the HS5 MSOR scheme to solve the two-dimensional Helmholtz equation. Their findings showed that the HS5 MSOR method outperformed both the FS5 GS and FS5 MSOR methods in terms of convergence. Additionally, other studies have explored the application of the HS5 AOR scheme for solving the two-dimensional Poisson Image Blending Problem. This approach was compared against the FS5 AOR and FS5 SOR schemes, with results indicating that the HS5 AOR method achieved superior convergence rates. Despite these advancements, there remains a gap in the literature, as no comparative studies have been conducted to determine the relative performance of the MSOR and AOR schemes. It is important to note, however, that while the HS5 iterative method offers improvements over the FS5 method, it is still not regarded as the iterative method with the fastest convergence rate, leaving room for further optimization and exploration of more efficient techniques.

Further advancement over the FS5 and HS5 iterative methods is the QS5 iterative method which demonstrate superior convergence rates across a variety of applications. For example, Eng et al. (2018) applied the QS5 SOR method to solve the two-dimensional Poisson Image Blending Problem, outperforming both FS5 and HS5 SOR methods in terms of convergence. Similarly, Akhir et al. (2012a) and Akhir et al. (2014) utilized the QS5 MSOR method to solve the two-dimensional Helmholtz equation and its homogeneous counterpart, respectively. Both studies reported faster convergence compared to FS5 GS, FS5 MSOR, and HS5 MSOR methods, although Akhir

et al. (2012a) omitted comparisons with the SOR scheme, and Akhir et al. (2014) did not implement the HS5 MSOR method with the Horizontal-Zebra-Line (HZL) ordering strategy. Moreover, Akhir et al. (2016) and Hoe and Hasan (2013), who employed the QS5 AOR method, addresses the non-homogeneous Helmholtz and Poisson equations. These methods exhibited better convergence compared to FS5 and HS5 AOR methods but lacked comparisons with alternative relaxation schemes. Finally, Dahalan et al. (2022) introduced the QS5 MAOR method for solving the two-dimensional Laplace equation in Robot Path Planning. This approach achieved the fastest convergence among all compared methods, including FS5, HS5, and QS5 schemes across various relaxation techniques. However, this research also did not explore the accuracy of the results comprehensively. Together, these studies highlight the potential of the QS5 iterative method while underscoring the need for more comprehensive comparative analyses in terms of accuracy and methodological evaluations.

The EG5 iterative method extends the FS5 iterative method, offering improved convergence rates. Saudi and Sulaiman (2012a) applied the EG5 SOR method to solve the two-dimensional Laplace equation for Robot Path Planning, demonstrating faster convergence compared to FS5 GS, FS5 SOR, and EG5 GS methods. Similarly, Akhir et al. (2011c) utilized the EG5 MSOR scheme for the two-dimensional Helmholtz equation, achieving superior convergence rates relative to FS5 GS, EG5 GS, and EG5 SOR methods. However, in both studies, the EG5 method exhibited singular ungroup points near the domain boundaries, a limitation that could impact solution accuracy. Eng et al. (2019) implemented the EG5 AOR scheme to address the two-dimensional Poisson Image Blending Problem. This method outperformed FS5 SOR and FS5 AOR methods in terms of convergence. However, the study did not provide a comparative analysis of accuracy across the methods, leaving gaps in understanding its overall performance. Despite these advancements, the EG5 method's boundary-related singularities and the absence of comprehensive accuracy evaluations highlight areas for further improvement.

The EDG5 iterative method, an extension of the EG5 iterative method, has shown notable improvements in convergence rates across various relaxation schemes. Akhir et al. (2011a) applied the EDG5 MSOR method to solve the two-dimensional Helmholtz equation, achieving better convergence compared to FS5 GS, EDG5 GS, and EDG5 SOR methods. However, similar to EG5, the EDG5 method also suffers from singular ungroup points near the boundaries of the solution domain. Yousif and Martins (2008) demonstrated the EDG5 AOR scheme's superiority over the EG5 AOR method for the two-dimensional Laplace equation but did not compare the accuracy of the methods. Another contributions includes the EDG5 AOR scheme to solve the two-dimensional Laplace equation for Robot Path Planning (Dahalan and Saudi (2022)). This scheme outperformed EG5 SOR, EG5 AOR, EG5 MAOR, and EDG5 SOR methods in convergence rates but lacked an in-depth comparison of accuracy across methods. Dahalan et al. (2020) extended this work by introducing the EDG5 MAOR scheme for the same application, reporting superior convergence rates relative to all tested EG5 and EDG5 schemes, including SOR, MSOR, AOR, and MAOR. However, the study did not provide detailed accuracy comparisons among these methods, leaving room for further investigation into their overall performance metrics.

The MEG5 iterative method demonstrates significant improvements in convergence rates compared to the previous iterative methods. Othman and Abdullah (2000a) applied the MEG5 SOR method to solve the two-dimensional Poisson equation, while Sulaiman et al. (2010) further extending the application to the Helmholtz equation and the steady convection-diffusion equation. Both studies concluded that MEG5 SOR achieves superior convergence rates compared to the EG5 SOR and EDG5 SOR methods. However, these studies also noted that the solution domains of EG5 and EDG5 methods exhibit singular ungroup points near the boundary, which may also affect the overall accuracy. Additionally, Akhir et al. (2015) evaluated the MEG5 MSOR scheme for solving the two-dimensional Helmholtz equation. The study found that MEG5

MSOR outperformed the EG5 MSOR and EDG5 MSOR methods in terms of convergence rate. Despite these advancements, the inherent boundary issues with singular ungroup points in EG5 and EDG5 domains remain unaddressed, suggesting potential areas for further investigation in MEG5 methodology.

Recent advancements in parallel algorithms for the second order iterative methods have demonstrated improvements in solving two-dimensional PDEs, albeit with some challenges. Othman and Abdullah (2000b) implemented a Parallel QS5 SOR method for the two-dimensional Poisson equation on Symmetric Multi-Processors (SMP) systems. The method, employing Chessboard (CB) ordering strategy for QS5 and FS5, and HZL ordering strategy for HS5, achieved better convergence rates than Parallel FS5 SOR and Parallel HS5 SOR. However, its speedup and efficiency were slightly lower than those of the other methods. Extending from the previous research of Othman and Abdullah (2000a), Othman et al. (2006) developed a Parallel MEG5 SOR method implemented on Shared Memory Multiprocessors. Comparisons with Parallel EG5 SOR and Parallel EDG5 SOR revealed the superiority of MEG5 SOR in convergence rates. However, EG5 and EDG5 methods, ordered with CB and HZL strategies, were also affected by singular ungroup points near the boundaries. Similarly, Fu and Ali (2008) evaluated Parallel EG5 SOR and EDG5 SOR on Distributed Memory Multicomputers, showing that EDG5 with four-colour strategy surpassed EG5 in accuracy, convergence, and efficiency, though results were not systematically tabulated. Additionally, Lopez et al. (2018) explored the Parallel FS5 GS method for the Poisson equation using CUDA programming. This method exhibited strong performance with increased mesh sizes but lacked derivations for ordering strategies and comparisons with other methods. These studies underscore the evolving landscape of parallelised iterative methods while highlighting areas for further exploration, including ordering strategies and comprehensive performance analyses.

Other than researches mentioned above, there has been continuing study that intro-

duces the Modified EDG5 (MEDG) (Ali and Fu (2007); Ali and Pin (2012); Ling and Ali (2016)) and Octo MEG5 (OMEG5) (Akhir et al. (2012b)) iterative methods. So far, OMEG5 iterative method has the fastest convergent rate compared to all of the methods mentioned above.

2.6.2 Comparative Analysis on Fourth Order Iterative Methods

Recent developments in fourth order point iterative methods have demonstrated improvements in solving two-dimensional partial differential equations, with specific methods exhibiting enhanced convergence rates under certain conditions. Ling et al. (2020) introduced the FS9 AOR method for solving the two-dimensional Laplace equation in robot path planning. This method achieved a better convergence rate compared to FS5 SOR, FS5 AOR, and FS9 SOR. However, the study lacked a comprehensive comparison of accuracy across all methods. Furthermore, Saudi and Sulaiman (2012b) proposed the HS9 SOR method for the same equation, showing a superior convergence rate relative to FS5 GS, FS5 SOR, and HS5 SOR. Despite these improvements, the HS9 method's applicability near the boundary was unaddressed, as no formulation was provided for those points, and no accuracy comparisons were included. Similarly, Ling and Ali (2014) applied the HS9 SOR method to the two-dimensional Poisson equation, demonstrating faster convergence compared to FS5 SOR, HS5 SOR, and FS9 SOR. However, the study also introduced varying formulations for points near boundaries, which were not generalized.

Researched in fourth order group iterative methods have shown promising improvements in computational efficiency compared to the fourth order point iterative methods. Saudi and Sulaiman (2012d) introduced the EG9 GS method for the Laplace equation in robot path planning, achieving a faster convergence rate than FS5 GS and EG5 GS methods. However, the study noted that the EG9 method's solution domain contains singular ungroup points near boundaries, a recurring issue in this class of methods. Similarly, Ali and Ping (2015) applied the EG9 GS method to the Helmholtz

equation, outperforming FS5 GS, EG5 GS, and FS9 GS methods in terms of convergence rate. Nonetheless, the presence of singular ungroup points near the boundary in both EG5 and EG9 methods was highlighted as a significant drawback. To further enhance convergence rates, Saudi and Sulaiman (2012c) extended the EG9 GS method to EG9 SOR, demonstrating superior performance over FS5 SOR and EG5 SOR methods. However, the boundary singularity issue persisted in this extension.

Extensions toward the EG9 iterative methods have also been developed. Ali and Ping (2016) introduced the EDG9 GS method, which demonstrated a faster convergence rate compared to FS5 GS, EG5 GS, EDG5 GS, and EG9 GS methods. Despite this improvement, the solution domains of EG5 and EG9 methods continued to exhibit singular ungroup points near the boundary. Building on this, Saudi and Sulaiman (2016) developed the EDG9 SOR method, applying it to the Laplace equation for robot path planning. The EDG9 SOR method outperformed a range of methods, including FS5 SOR, HS5 SOR, EG5 GS, EDG5 GS, FS9 SOR, HS9 SOR, and EG9 GS, in terms of convergence rate. However, like its predecessors, the group methods retained boundary singularities. Lastly, Ling and Ali (2013) presented the MEG9 SOR method for solving the two-dimensional Poisson equation. This method surpassed FS5 SOR, FS9 SOR, and EG9 SOR methods in convergence efficiency, but the EG9 method's boundary singularity issues persisted. These findings highlight the potential of advanced nine-point group methods while emphasizing the necessity of addressing boundary-related challenges to optimize performance and applicability.

Another research investigated the Parallel HS9 SOR method for solving the two dimensional Poisson equation using Shared Memory Multiprocessors (SMP) (Othman and Sulaiman (2008)). The method was compared against the Parallel FS9 SOR method, with both employing a four-colour grid ordering strategy. The Parallel HS9 SOR demonstrated a superior convergence rate compared to its FS9 counterpart, highlighting its efficiency. However, the study also noted a limitation whereby the HS9

method is not applicable for points near the boundary, and the research did not provide a formulation for addressing this issue. These findings suggest the potential of Parallel HS9 SOR in enhancing computational efficiency, although further refinement is needed to address boundary-related constraints.

2.6.3 Comparative Analysis on Fractional Order Iterative Methods

The comparative analysis of previous researches with regards to fractional order iterative methods has been done on two types of fractional equation which are the space-fractional and time-fractional equations. Space-fractional equations is where the fractional derivatives replaces the space derivatives of the equations, while time-fractional equations replaces the time derivatives of the equations. Generally, when comparing these types of equations, the iterative methods for space-fractional equation is shown to converge faster, while the approximate solution of iterative methods for time-fractional equation is shown to be more accurate. Nonetheless, it is clear that fractional derivatives are applicable on both types of equations.

Borhanifar and Valizadeh (2013) introduced the FFS iterative method to solve the two-dimensional Fractional Poisson equation, leveraging the Shifted Grünwald Estimate. The study evaluated the method's performance for fractional orders $\alpha = 1.25, 1.5, 1.75$ and compared the accuracy of the results as α approached 2. Findings revealed an improvement in accuracy when α neared the classical integer value of 2, demonstrating the method's robustness in fractional scenarios. However, the research applied only a direct approach, leaving iterative and advanced schemes unexplored. This study serves as a benchmark for our thesis, which aims to extend the methodology with iterative and parallel algorithms for enhanced computational efficiency and broader applicability.

The progression of the FFS iterative method in solving fractional differential equations is highlighted by Sunarto et al. (2014a) as they introduced the FFS SOR method for one-dimensional space-fractional diffusion equations discretized using Caputo's im-

implicit finite difference method. This scheme, tested at fractional orders $\alpha = 1.2, 1.5, 1.8$, demonstrated superior convergence compared to the FFS GS method. Similarly, Sunarto et al. (2014b) extended the FFS iterative method to time-fractional diffusion equations, proposing the FFS AOR scheme, which was evaluated at $\alpha = 0.25, 0.5, 0.75$. The results showed that the FFS AOR outperformed both the FFS GS and FFS SOR methods in convergence rate. Building on this, Sunarto et al. (2016a) applied the FFS AOR method to space-fractional diffusion equations, confirming its enhanced convergence at $\alpha = 1.2, 1.5, 1.8$, again outperforming the FFS GS and FFS SOR methods. These advancements underscore the efficacy of relaxation schemes in improving the performance of FFS methods for fractional differential equations.

An efficient alternative to the FFS iterative method is the FHS iterative method. Sunarto et al. (2015) proposed the FHS GS scheme to solve one-dimensional space-fractional diffusion equations using Caputo's implicit finite difference, demonstrating better convergence than the FFS GS method for fractional orders $\alpha = 1.2, 1.5, 1.7$. Furthermore, Sunarto and Sulaiman (2017) extended the FHS approach to the SOR scheme from Sunarto et al. (2014a), showcasing improved convergence over the FFS SOR method at $\alpha = 1.2, 1.5, 1.8$. Another contributions were made by Sunarto et al. (2016b) and Sunarto et al. (2016c), who applied the FHS AOR scheme to time- and space-fractional diffusion equations, respectively. These methods consistently outperformed the FFS AOR and FFS SOR approaches, with Sunarto et al. (2016b) focusing on $\alpha = 0.25, 0.5, 0.75$ and Sunarto et al. (2016c) on $\alpha = 1.2, 1.5, 1.8$. Additionally, Ali and Ali (2019b) extended the FHS iterative method to two-dimensional second-order hyperbolic telegraph time-fractional differential equations discretized using the Crank-Nicolson scheme. Tested at $\alpha = 0.65, 0.75, 0.85$, this approach demonstrated superior convergence rates compared to the FFS iterative method. Collectively, these studies highlight the efficiency of the FHS approach across various fractional differential equations and relaxation schemes.

Another promising alternative to the FFS and FHS iterative methods is the FQS iterative method. Sunarto and Sulaiman (2019a) introduced the FQS GS scheme for one-dimensional time-fractional diffusion equations discretized using Caputo's implicit finite difference, demonstrating better convergence rates compared to FFS GS and FHS GS methods for fractional orders $\alpha = 0.25, 0.5, 0.75$. An extension of the work done by Sunarto et al. (2015), Sunarto and Sulaiman (2019b) tested the FQS GS scheme at $\alpha = 1.2, 1.5, 1.8$, achieving similar improvements in convergence over FFS GS and FHS GS methods. Additionally, Sunarto et al. (2018) proposed the FQS SOR method, an extension of Sunarto and Sulaiman (2017), applied to $\alpha = 1.2, 1.5, 1.8$. This method demonstrated superior convergence compared to FFS SOR and FHS SOR techniques. Similarly, Baharuddin et al. (2017) extended the work of Sunarto et al. (2016b) by applying the FQS AOR scheme to fractional orders $\alpha = 0.25, 0.5, 0.75$. This method outperformed the FFS AOR and FHS AOR approaches, highlighting the efficacy of the FQS framework across a range of fractional differential equations.

Further developments in fractional iterative methods have introduced the FEG, FEDG, and FMEG iterative methods to improve convergence rates in solving fractional differential equations. Muhiddin et al. (2019) proposed the FEG SOR method for one-dimensional time-fractional parabolic equations discretized using the Grünwald implicit finite difference. Tested for fractional orders $\alpha = 0.333, 0.666, 0.999$, the method demonstrated faster convergence compared to FFS GS and FEG GS techniques, although it faced challenges due to singular ungroup points near the boundary. Ali and Ali (2019a) and Balasim and Ali (2016) evaluated the FEG and FEDG iterative methods for two-dimensional time-fractional equations. For $\alpha = 0.55, 0.75, 0.95$ and $\alpha = 0.25, 0.75$ respectively, the FEDG method consistently outperformed both FFS and FHS methods in terms of convergence rates. However, like FEG, FEDG methods were affected by singularities near boundary points. Additionally, Balasim and Ali (2017) introduced the FEG and FMEG iterative methods for two-dimensional time-fractional advection-diffusion equations. This approach, tested at $\alpha = 0.75, 0.95$,

showed superior convergence compared to FFS, marking significant progress in iterative solutions for fractional problems. Nevertheless, singular ungroup points remain a limitation in FEG-based methods, emphasizing the need for further refinement in handling boundary conditions.

Recent advancements in numerical methods for fractional differential equations have emphasised the development of fourth order fractional iterative methods. For instance, Khan and Ali (2020) introduced a fourth order FFS SOR method to address the Rayleigh-Stokes problem for heated generalized second-grade fluids incorporating fractional derivatives. By leveraging the fourth order Crank-Nicolson finite difference for spatial discretization and a relationship between the Grünwald-Letnikov and Riemann-Liouville fractional derivatives for the time component, the method demonstrated high accuracy in solving equations for $0 < \alpha < 1$. Comparative analyses showed the Crank-Nicolson compact method to be superior in accuracy over implicit compact and radial basis function meshless methods. Extending this work, Khan et al. (2020) proposed the fourth order FEG SOR method, a group-based adaptation of the previous method. This approach further optimized execution time and reduced computational complexity, enhancing efficiency. However, challenges persist with FEG iterative methods due to singular ungroup points near the boundary, indicating the need for improvements in domain handling for future iterations. These developments reflect the ongoing efforts to refine high-accuracy fractional solvers with practical applicability across complex fluid dynamics scenarios.

2.6.4 Research Gap

Let's analyze the research gap according to the order of discretization. For second order iterative methods, the combinations of iterative methods and relaxation schemes are mostly covered including some iterative methods on MAOR scheme. However, the parallel second order MAOR method has not been covered.

For fourth order iterative methods, it can be seen that the combinations of iterative methods and relaxation schemes are not entirely covered. Also, it noted that the MAOR scheme has not been combined with any fourth order iterative methods and has not been solved sequentially or in parallel. Additionally, as mentioned before, the HS9 and EDG9 formula cannot solve points that are near the boundary. Thus, another rotated five points of higher derivatives is derived specifically to solve those points.

For fractional order iterative methods, it can be seen that the combinations of iterative methods and relaxation schemes are somewhat covered for both time and space fractional equations. The MAOR scheme is currently combined with the FFS iterative method and there not many that utilize parallel computing for fractional equations.

By looking at the group iterative methods for all order of discretization, the EG and EDG methods will have a similar layout in terms of grouping the points in the solution domain. However, they will differ from the MEG method which is why there exist some points that are ungrouped in the EG and EDG methods. These points may group into two points which will require some additional formulation.

Since the all of the iterative methods will be solve using parallel computing, they will need to be ordered such that the point or group of points solved are independent from each other. This will ensure that the problem solved within the same number of iteration but with less execution time.

CHAPTER 3

SECOND ORDER MAOR ITERATIVE METHODS FOR SOLVING HELMHOLTZ EQUATION

3.1 Introduction

This chapter is devoted to the formulation of second order MAOR iterative methods. It starts with the implementation of the MAOR scheme on FS5, HS5, QS5, EG5, EDG5, and MEG5 iterative methods which includes different ordering strategies of the solution domain. The solution domain is ordered such that the tasks executed will be independent from each other. Then, the chapter is continued with the sequential algorithm for the second order MAOR iterative methods. This will describe how each iterative method is formulated with their respective ordering strategies. Next, the chapter provides the construction of parallel algorithm for the second order MAOR iterative methods. With the tasks being independent from each other, it is possible to compute them in a parallel environment.

3.2 Implementation of Sequential Algorithms for Second Order MAOR Iterative Methods

This section presents the implementation of sequential algorithms for MAOR scheme on the second order iterative methods. The inclusion of the solution domain for each iterative method with their respective ordering strategies will assist in the derivation of the scheme. For simplicity, let's denote $v_{i,j} = u_{i,j}^{k+1} - u_{i,j}^k$.

3.2.1 Sequential Second Order Point Iterative Methods with MAOR Scheme

Sequential FS5 MAOR

Figure 3.1 shows the solution domain for FS5 with the chessboard (CB) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

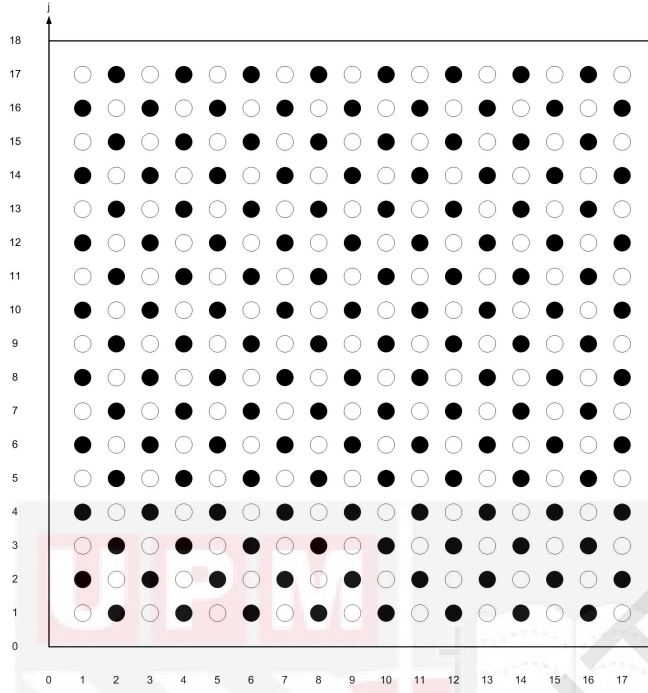


Figure 3.1: The solution domain for FS5.

Equation (2.2) can be generally represented in the following matrix form

$$D = \begin{bmatrix} \rho_1 & & & \\ & \rho_1 & & \\ & & \ddots & \\ & & & \rho_1 \end{bmatrix}, \quad L = \begin{bmatrix} 0 & 0 \\ H_1 & 0 \end{bmatrix}, \quad U = L^T,$$

$$H_1 = \begin{bmatrix} J_1 & -I & & & \\ -I & J_1^T & -I & & \\ & -I & J_1 & -I & \\ & & \ddots & \ddots & \ddots \\ & & & -I & J_1 & -I \\ & & & & -I & J_1^T & -I \\ & & & & & -I & J_1 \end{bmatrix}, \quad b = h^2 \begin{bmatrix} f_{1,1} \\ f_{3,1} \\ \vdots \\ f_{i,j} \\ \vdots \\ f_{M-3,M-1} \\ f_{M-1,M-1} \end{bmatrix},$$

$$J_1 = \begin{bmatrix} -1 & -1 & & & \\ & -1 & -1 & & \\ & & \ddots & \ddots & \\ & & & -1 & -1 \\ & & & -1 & -1 \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential FS5 MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all $\bigcirc$ points using

$$\begin{aligned} \bigcirc \quad u_{i,j}^{k+1} &= \frac{\omega_1}{\rho_1} \left(u_{i-1,j}^k + u_{i+1,j}^k + u_{i,j-1}^k + u_{i,j+1}^k - h^2 f_{i,j} \right) \\ &\quad + (1 - \omega_1) u_{i,j}^k. \end{aligned} \quad (3.1)$$

3. Execute all $\bullet$ points using

$$\begin{aligned} \bullet \quad u_{i,j}^{k+1} &= \frac{\theta_2}{\rho_1} \left(v_{i-1,j} + v_{i+1,j} + v_{i,j-1} + v_{i,j+1} \right) \\ &\quad + \frac{\omega_2}{\rho_1} \left(u_{i-1,j}^k + u_{i+1,j}^k + u_{i,j-1}^k + u_{i,j+1}^k - h^2 f_{i,j} \right) \\ &\quad + (1 - \omega_2) u_{i,j}^k. \end{aligned} \quad (3.2)$$

4. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, go to step (5), otherwise, repeat the iteration cycle (i. e. go to step (2)).
5. Stop.

Sequential HS5 MAOR

Figure 3.2 shows the solution domain for HS5 with the horizontal zebra line (HZL) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

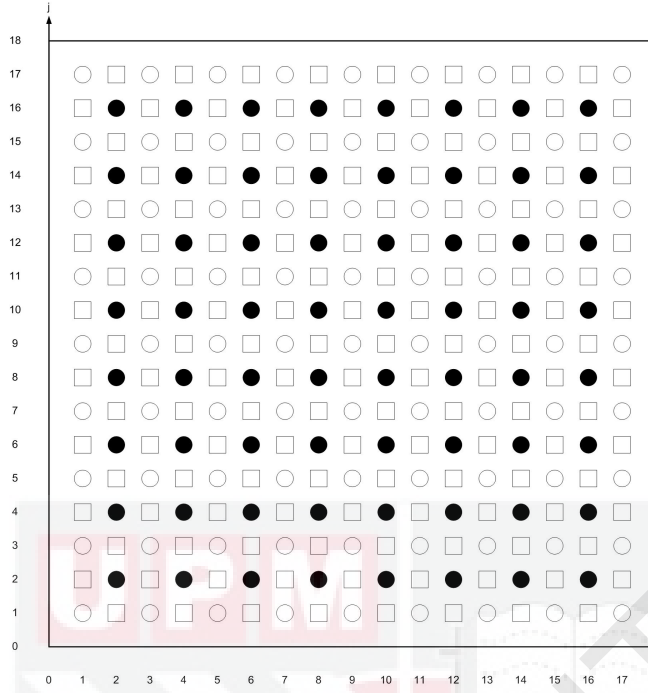


Figure 3.2: The solution domain for HS5.

Equation (2.3) can be generally represented in the following matrix form

$$D = \begin{bmatrix} \rho_2 & & & \\ & \rho_2 & & \\ & & \ddots & \\ & & & \rho_2 \end{bmatrix}, \quad L = \begin{bmatrix} 0 & 0 \\ H_1 & 0 \end{bmatrix}, \quad U = L^T,$$

$$H_1 = \begin{bmatrix} J_1 & J_1 & & & \\ & J_1 & J_1 & & \\ & & \ddots & \ddots & \\ & & & J_1 & J_1 \\ & & & & J_1 & J_1 \end{bmatrix}, \quad b = 2h^2 \begin{bmatrix} f_{1,1} \\ f_{3,1} \\ \vdots \\ f_{i,j} \\ \vdots \\ f_{M-4,M-2} \\ f_{M-2,M-2} \end{bmatrix},$$

$$J_1 = \begin{bmatrix} -1 & -1 & & & \\ & -1 & -1 & & \\ & & \ddots & \ddots & \\ & & & -1 & -1 \\ & & & & -1 & -1 \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential HS5 MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all $\circ$ points using

$$\begin{aligned} \circ \quad u_{i,j}^{k+1} &= \frac{\omega_1}{\rho_2} \left(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k - 2h^2 f_{i,j} \right) \\ &\quad + (1 - \omega_1) u_{i,j}^k. \end{aligned} \quad (3.3)$$

3. Execute all $\bullet$ points using

$$\begin{aligned} \bullet \quad u_{i,j}^{k+1} &= \frac{\theta_2}{\rho_2} (v_{i-1,j-1} + v_{i+1,j-1} + v_{i-1,j+1} + v_{i+1,j+1}) \\ &\quad + \frac{\omega_2}{\rho_2} \left(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k - 2h^2 f_{i,j} \right) \\ &\quad + (1 - \omega_2) u_{i,j}^k. \end{aligned} \quad (3.4)$$

4. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, evaluate the solutions of the remaining $\square$ points using

$$\square \quad u_{i,j}^{k+1} = \frac{1}{\rho_1} \left(u_{i+1,j}^{k+1} + u_{i-1,j}^{k+1} + u_{i,j+1}^{k+1} + u_{i,j-1}^{k+1} - h^2 f_{i,j} \right). \quad (3.5)$$

Otherwise, repeat the iteration cycle (i. e. go to step (2)).

5. Stop.

Sequential QS5 MAOR

Figure 3.3 shows the solution domain for QS5 with the chessboard (CB) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

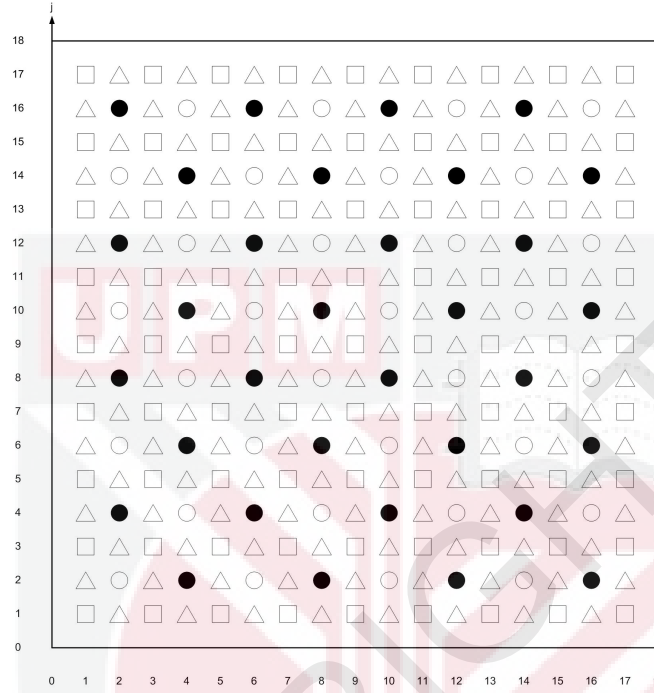


Figure 3.3: The solution domain for QS5.

Equation (2.4) can be generally represented in the following matrix form

$$D = \begin{bmatrix} \rho_3 & & & \\ & \rho_3 & & \\ & & \ddots & \\ & & & \rho_3 \end{bmatrix}, \quad L = \begin{bmatrix} 0 & 0 \\ H_1 & 0 \end{bmatrix}, \quad U = L^T,$$

$$H_1 = \begin{bmatrix} J_1 & -I & & & & \\ -I & J_1^T & -I & & & \\ & -I & J_1 & -I & & \\ & & \ddots & \ddots & \ddots & \\ & & & -I & J_1 & -I \\ & & & & -I & J_1^T & -I \\ & & & & & -I & J_1 \end{bmatrix}, \quad b = 4h^2 \begin{bmatrix} f_{2,2} \\ f_{6,2} \\ \vdots \\ f_{i,j} \\ \vdots \\ f_{M-8,M-2} \\ f_{M-4,M-2} \end{bmatrix},$$

$$J_1 = \begin{bmatrix} -1 & -1 & & & \\ & -1 & -1 & & \\ & & \ddots & \ddots & \\ & & & -1 & -1 \\ & & & & -1 \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential QS5 MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all ○ points using

$$\begin{aligned} \bigcirc \quad u_{i,j}^{k+1} &= \frac{\omega_1}{\rho_3} \left(u_{i-2,j}^k + u_{i+2,j}^k + u_{i,j-2}^k + u_{i,j+2}^k - 4h^2 f_{i,j} \right) \\ &+ (1 - \omega_1) u_{i,j}^k. \end{aligned} \quad (3.6)$$

3. Execute all ● points using

$$\begin{aligned} \bullet \quad u_{i,j}^{k+1} &= \frac{\theta_2}{\rho_3} (v_{i-2,j} + v_{i+2,j} + v_{i,j-2} + v_{i,j+2}) \\ &= \frac{\omega_2}{\rho_3} \left(u_{i-2,j}^k + u_{i+2,j}^k + u_{i,j-2}^k + u_{i,j+2}^k - 4h^2 f_{i,j} \right) \\ &+ (1 - \omega_2) u_{i,j}^k. \end{aligned} \quad (3.7)$$

4. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, evaluate the solutions of the remaining $\square$ points using

$$\square \quad u_{i,j}^{k+1} = \frac{1}{\rho_2} \left(u_{i+1,j+1}^{k+1} + u_{i-1,j-1}^{k+1} + u_{i-1,j+1}^{k+1} + u_{i+1,j-1}^{k+1} - 2h^2 f_{i,j} \right), \quad (3.8)$$

followed by the remaining $\triangle$ points using

$$\triangle \quad u_{i,j}^{k+1} = \frac{1}{\rho_1} \left(u_{i+1,j}^{k+1} + u_{i-1,j}^{k+1} + u_{i,j+1}^{k+1} + u_{i,j-1}^{k+1} - h^2 f_{i,j} \right). \quad (3.9)$$

Otherwise, repeat the iteration cycle (i. e. go to step (2)).

5. Stop.

3.2.2 Sequential Second Order Group Iterative Methods with MAOR Scheme

Sequential EG5 MAOR

Figure 3.4 shows the solution domain for EG5 with the chessboard (CB) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

Let

$$A_1 = \begin{bmatrix} \rho_1^2 - 2 & \rho_1 & \rho_1 & 2 \\ \rho_1 & \rho_1^2 - 2 & 2 & \rho_1 \\ \rho_1 & 2 & \rho_1^2 - 2 & \rho_1 \\ 2 & \rho_1 & \rho_1 & \rho_1^2 - 2 \end{bmatrix},$$

and

$$\begin{aligned} S_1^k &= u_{i,j-1}^k + u_{i-1,j}^k - h^2 f_{i,j}, & S_2^k &= u_{i+2,j}^k + u_{i+1,j-1}^k - h^2 f_{i+1,j}, \\ S_3^k &= u_{i-1,j+1}^k + u_{i,j+2}^k - h^2 f_{i,j+1}, & S_4^k &= u_{i+2,j+1}^k + u_{i+1,j+2}^k - h^2 f_{i+1,j+1}. \end{aligned}$$

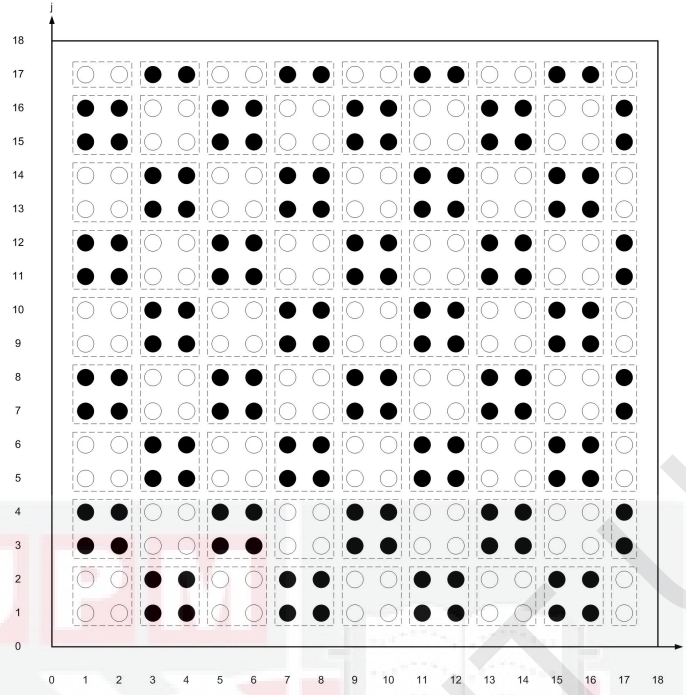


Figure 3.4: The solution domain for EG5.

Equation (2.5) can be generally represented in the following matrix form

$$D = \begin{bmatrix} A_1 & & & & \\ & A_1 & & & \\ & & \ddots & & \\ & & & A_1 & \\ & & & & A_1 \end{bmatrix}, \quad L = \begin{bmatrix} 0 & 0 \\ H_1 & 0 \end{bmatrix}, \quad U = L^T,$$

$$H_1 = \begin{bmatrix} J_1 & J_1^T & J_2^T & & & \\ & J_1 & J_2^T & & & \\ J_2 & & J_1 & J_2^T & & \\ & J_2 & J_1^T & J_1 & J_2^T & \\ & & \ddots & \ddots & \ddots & \ddots \\ & & & J_2 & J_1 & J_1^T & J_2^T \\ & & & & J_2 & J_1 & J_2^T \\ & & & & & J_2 & J_1 \\ & & & & & & J_2 & J_1^T & J_1 \end{bmatrix}, \quad b = \begin{bmatrix} G_1 \\ G_1 \\ \vdots \\ G_1 \\ G_1 \end{bmatrix},$$

$$J_1 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad J_2 = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_1 = h^2 \begin{bmatrix} f_{i,j} \\ f_{i+1,j} \\ f_{i,j+1} \\ f_{i+1,j+1} \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential EG5 MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all groups of four $\odot$ points using

$$\odot \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix}^{k+1} = \frac{\omega_1}{Q_1} A_1 \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}^k + (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix}^k. \quad (3.10)$$

3. Execute the remaining groups of two $\odot$ points at $i = M - 1$ using

$$\odot \begin{bmatrix} u_{i,j} \\ u_{i,j+1} \end{bmatrix}^{k+1} = \frac{\omega_1}{\rho_1^2 - 1} \begin{bmatrix} \rho_1 & 1 \\ 1 & \rho_1 \end{bmatrix} \begin{bmatrix} S_1 + u_{i+1,j} \\ S_3 + u_{i+1,j+1} \end{bmatrix}^k + (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i,j+1} \end{bmatrix}^k, \quad (3.11)$$

and at $j = M - 1$ using

$$\begin{aligned}
\bigcirc \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \end{bmatrix}^{k+1} &= \frac{\omega_1}{\rho_1^2 - 1} \begin{bmatrix} \rho_1 & 1 \\ 1 & \rho_1 \end{bmatrix} \begin{bmatrix} S_1 + u_{i,j+1} \\ S_2 + u_{i+1,j+1} \end{bmatrix}^k \\
&+ (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \end{bmatrix}^k.
\end{aligned} \tag{3.12}$$

Finally, execute the remaining $\bigcirc$ point at $i, j = M - 1$ using Equation (3.1)

4. Execute all groups of four $\bullet$ points using

$$\begin{aligned}
\bullet \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix}^{k+1} &= \frac{\theta_2}{Q_1} A_1 \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \\ S_3^{k+1} - S_3^k \\ S_4^{k+1} - S_4^k \end{bmatrix} + \frac{\omega_2}{Q_1} A_1 \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}^k \\
&+ (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix}^k.
\end{aligned} \tag{3.13}$$

5. Execute the remaining groups of two $\bullet$ points at $i = M - 1$ using

$$\begin{aligned}
\bullet \begin{bmatrix} u_{i,j} \\ u_{i,j+1} \end{bmatrix}^{k+1} &= \frac{\theta_2}{\rho_1^2 - 1} \begin{bmatrix} \rho_1 & 1 \\ 1 & \rho_1 \end{bmatrix} \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_3^{k+1} - S_3^k \end{bmatrix} \\
&+ \frac{\omega_2}{\rho_1^2 - 1} \begin{bmatrix} \rho_1 & 1 \\ 1 & \rho_1 \end{bmatrix} \begin{bmatrix} S_1 + u_{i+1,j} \\ S_3 + u_{i+1,j+1} \end{bmatrix}^k \\
&+ (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i,j+1} \end{bmatrix}^k,
\end{aligned} \tag{3.14}$$

and at $j = M - 1$ using

$$\begin{aligned}
 \bullet \quad \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \end{bmatrix}^{k+1} &= \frac{\theta_2}{\rho_1^2 - 1} \begin{bmatrix} \rho_1 & 1 \\ 1 & \rho_1 \end{bmatrix} \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \end{bmatrix} \\
 &+ \frac{\omega_2}{\rho_1^2 - 1} \begin{bmatrix} \rho_1 & 1 \\ 1 & \rho_1 \end{bmatrix} \begin{bmatrix} S_1 + u_{i,j+1} \\ S_2 + u_{i+1,j+1} \end{bmatrix}^k \\
 &+ (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \end{bmatrix}^k. \tag{3.15}
 \end{aligned}$$

6. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, go to step (5), otherwise, repeat the iteration cycle (i. e. go to step (2)).

7. Stop.

Sequential EDG5 MAOR

Figure 3.5 shows the solution domain for EDG5 with the horizontal zebra line (HZL) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

Let

$$A_2 = \begin{bmatrix} \rho_2 & 1 \\ 1 & \rho_2 \end{bmatrix},$$

and

$$\begin{aligned}
 S_1^k &= u_{i-1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j-1}^k - 2h^2 f_{i,j}, \\
 S_2^k &= u_{i+2,j}^k + u_{i,j+2}^k + u_{i+2,j+2}^k - 2h^2 f_{i+1,j+1}
 \end{aligned}$$

Equation (2.6) can be generally represented in the following matrix form

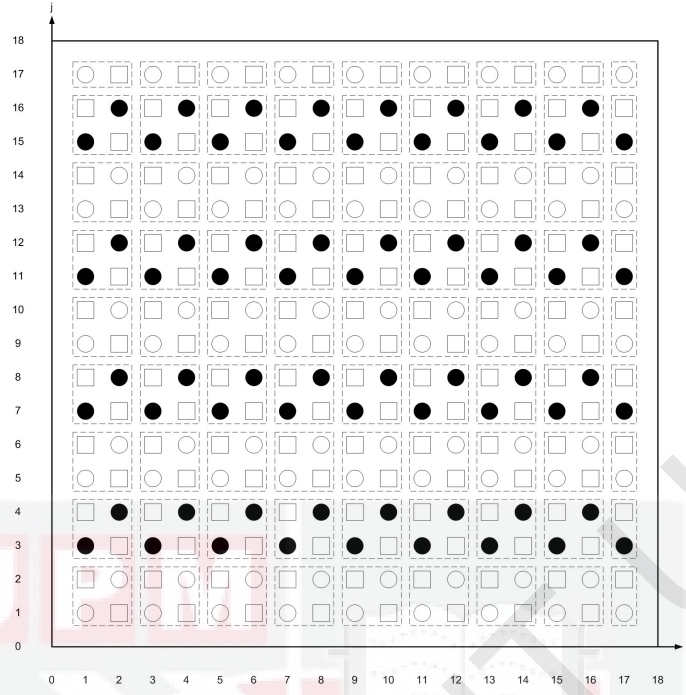


Figure 3.5: The solution domain for EDG5.

$$D = \begin{bmatrix} A_2 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_2 \\ & & & & A_2 \end{bmatrix}, \quad L = \begin{bmatrix} H_2 & 0 \\ H_1 & H_2 \end{bmatrix}, \quad U = L^T,$$

$$H_1 = \begin{bmatrix} J_1 & & J_1^T & J_1^T & & \\ J_1 & J_1 & & J_1^T & J_1^T & \\ & J_1 & J_1 & & \ddots & \ddots \\ & & J_1 & J_1 & & J_1^T & J_1^T \\ & & & \ddots & \ddots & & J_1^T \\ & & & & J_1 & J_1 & \\ & & & & & J_1 & J_1 \\ & & & & & & J_1 & J_1 \end{bmatrix}, \quad b = \begin{bmatrix} G_1 \\ G_1 \\ \vdots \\ G_1 \\ G_1 \end{bmatrix},$$

$$H_2 = \begin{bmatrix} J_1 & & & & \\ & J_1 & & & \\ & & \ddots & & \\ & & & J_1 & \\ & & & & J_1 \end{bmatrix}, \quad J_1 = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}, \quad G_1 = 2h^2 \begin{bmatrix} f_{i,j} \\ f_{i+1,j+1} \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential EDG5 MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all groups of two ○ points using

$$\begin{aligned} \circ \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix}^{k+1} &= \frac{\theta_1}{Q_2} A_2 \begin{bmatrix} v_{i-1,j+1} \\ 0 \end{bmatrix} + \frac{\omega_1}{Q_2} A_2 \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}^k \\ &+ (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix}^k. \end{aligned} \quad (3.16)$$

3. Execute the remaining ○ points at $i = M - 1$ and $j = M - 1$ using Equation (3.3).
4. Execute all groups of two ● points using

$$\begin{aligned} \bullet \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix}^{k+1} &= \frac{\theta_2}{Q_2} A_2 \begin{bmatrix} S_1^{k+1} - S_1^k \\ v_{i,j+2} + v_{i+2,j+2} \end{bmatrix} + \frac{\omega_2}{Q_2} A_2 \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}^k \\ &+ (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix}^k. \end{aligned} \quad (3.17)$$

5. Execute the remaining ● points at $j = M - 1$ using Equation (3.4).

6. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, evaluate the solutions of the remaining $\square$ points using Equation (3.5). Otherwise, repeat the iteration cycle (i. e. go to step (2)).
7. Stop.

Sequential MEG5 MAOR

Figure 3.6 shows the solution domain for MEG5 with the chessboard (CB) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

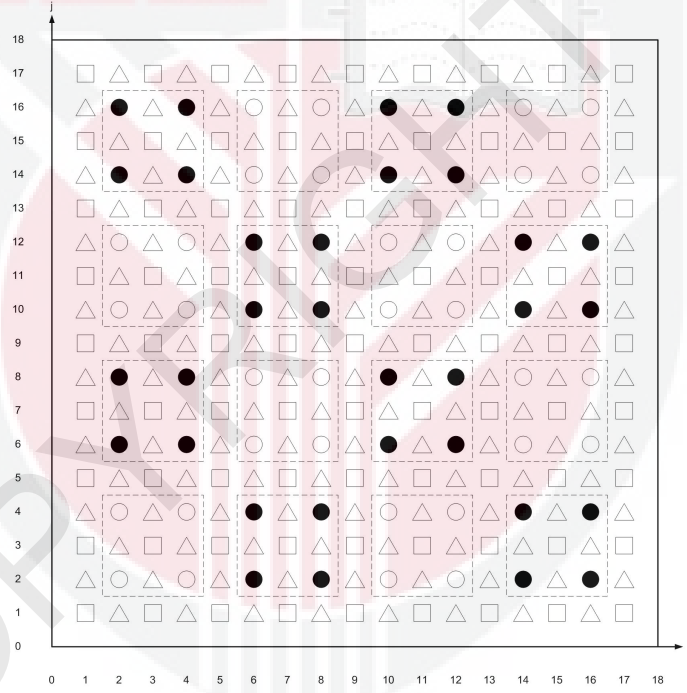


Figure 3.6: The solution domain for MEG5.

Let

$$A_3 = \begin{bmatrix} \rho_3^2 - 2 & \rho_3 & \rho_3 & 2 \\ \rho_3 & \rho_3^2 - 2 & 2 & \rho_3 \\ \rho_3 & 2 & \rho_3^2 - 2 & \rho_3 \\ 2 & \rho_3 & \rho_3 & \rho_3^2 - 2 \end{bmatrix},$$

and

$$\begin{aligned}
S_1^k &= u_{i,j-2}^k + u_{i-2,j}^k - 4h^2 f_{i,j}, & S_2^k &= u_{i+4,j}^k + u_{i+2,j-2}^k - 4h^2 f_{i+2,j}, \\
S_3^k &= u_{i-2,j+2}^k + u_{i,j+4}^k - 4h^2 f_{i,j+2}, & S_4^k &= u_{i+4,j+2}^k + u_{i+2,j+4}^k - 4h^2 f_{i+2,j+2}.
\end{aligned}$$

Equation (2.7) can be generally represented in the following matrix form

$$D = \begin{bmatrix} A_3 & & & \\ & A_3 & & \\ & & \ddots & \\ & & & A_3 \end{bmatrix}, \quad L = \begin{bmatrix} 0 & 0 \\ H_1 & 0 \end{bmatrix}, \quad U = L^T,$$

$$H_1 = \begin{bmatrix} J_1 & J_1^T & J_2^T & & & \\ & J_1 & J_2^T & & & \\ J_2 & & J_1 & J_2^T & & \\ & J_2 & J_1^T & J_1 & J_2^T & \\ & & \ddots & \ddots & \ddots & \ddots \\ & & & J_2 & J_1 & J_1^T & J_2^T \\ & & & & J_2 & J_1 & J_2^T \\ & & & & & J_2 & J_1^T & J_1 \end{bmatrix}, \quad b = \begin{bmatrix} G_1 \\ G_1 \\ \vdots \\ G_1 \\ G_1 \end{bmatrix},$$

$$J_1 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad J_2 = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_1 = h^2 \begin{bmatrix} f_{i,j} \\ f_{i+2,j} \\ f_{i,j+2} \\ f_{i+2,j+2} \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential MEG5 MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all groups of four ○ points using

$$\circ \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix}^{k+1} = \frac{\omega_1}{Q_3} A_3 \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}^k + (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix}^k. \quad (3.18)$$

3. Execute all groups of four ● points using

$$\bullet \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix}^{k+1} = \frac{\theta_2}{Q_3} A_3 \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \\ S_3^{k+1} - S_3^k \\ S_4^{k+1} - S_4^k \end{bmatrix} + \frac{\omega_2}{Q_3} A_3 \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}^k + (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix}^k. \quad (3.19)$$

4. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, evaluate the solutions of the remaining □ points using Equation (3.8) followed by the remaining △ points using Equation (3.9). Otherwise, repeat the iteration cycle (i. e. go to step (2)).
5. Stop.

3.3 Construction of Second Order Parallel Algorithm

From Equations (3.1), (3.3), (3.6), (3.10), (3.16) and (3.18), it is clear that all the ○ tasks in $u_{i,j}^{k+1}$ are independent of each other and can be computed in parallel. After

that, all the ● tasks of Equations (3.2), (3.4), (3.7), (3.13), (3.17) and (3.19) can be calculated simultaneously using the updated ○ values since these calculations are also independent. At the end of each stage, a synchronization call is executed to ensure the updated values are used in the subsequent iteration. Each processor independently iterates on its own task and then the convergence test is performed. If the convergence criteria is achieved, the HS5, QS5, EDG5 and MEG5 iterative methods will evaluate the remaining tasks directly. Otherwise increase the number of iteration and repeat the iteration cycle for all iterative methods.

As an example, lets choose four number of processors which is denoted as P0, P1, P2 and P3 for the first, second, third and fourth processor respectively. The parallel algorithms for solving the second order MAOR iterative methods will be constructed using multicore processors using the OpenMP technology and C++ Compiler.

3.3.1 Parallel Second Order Point Iterative Methods with MAOR Scheme

Parallel FS5 MAOR

Step 1: Solve the ○ points. From Figure 3.1, assign each processor to a row of ○ points ($P0 \rightarrow j = 1$, $P1 \rightarrow j = 2$, $P2 \rightarrow j = 3$, $P3 \rightarrow j = 4$). All processor will solve the ○ points using Equation (3.1). After the processors have finish solving their respective rows, they will be assigned to the next row of ○ points ($P0 \rightarrow j = 5$, $P1 \rightarrow j = 6$, $P2 \rightarrow j = 7$, $P3 \rightarrow j = 8$). This process will continue until there are no available row of ○ points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 2: Solve the ● points. From Figure 3.1, assign each processor to a row of ● points ($P0 \rightarrow j = 1$, $P1 \rightarrow j = 2$, $P2 \rightarrow j = 3$, $P3 \rightarrow j = 4$). All processor will solve the ● points using Equation (3.2). After the processors have finish solving the respective rows, they will be assigned to the next row of block ($P0 \rightarrow j = 5$, $P1 \rightarrow j = 6$, $P2 \rightarrow j = 7$, $P3 \rightarrow j = 8$). This process will continue until there are no available row

of ● points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 3: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

These steps above can be summarized in Algorithm 1.

Algorithm 1: The iterative process for the parallel FS5 MAOR iterative method.

```

1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into ○ and ● as shown in Figure 3.1;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
4   Get thread number  $p$ , and maximum number of threads  $np$ ;
5   while  $|u^{k+1} - u^k| > \varepsilon$  do
6     #pragma omp parallel
7     {
8       for  $\{j = p + 1; j < M; j = j + np\}$  do
9         Compute all ○ points using Equation (3.1);
10      end for
11      #pragma omp barrier
12      for  $\{j = p + 1; j < M; j = j + np\}$  do
13        Compute all ● points using Equation (3.2);
14      end for
15      #pragma omp barrier
16       $ite = ite + 1$ ;
17    }
18  end while
19  Display  $ite$ , error, and execution time;

```

Parallel HS5 MAOR

Step 1: Solve the ○ points. From Figure 3.2, assign each processor to a row of ○ points ($P0 \rightarrow j = 1, P1 \rightarrow j = 3, P2 \rightarrow j = 5, P3 \rightarrow j = 7$). All processor will solve the ○ points using Equation (3.3). After the processors have finish solving their respective rows, they will be assigned to the next row of ○ points ($P0 \rightarrow j = 9, P1 \rightarrow j = 11, P2 \rightarrow j = 13, P3 \rightarrow j = 15$). This process will continue until there are no available row of ○ points left to be solved. Then the `#pragma omp barrier` command is called to

synchronize all of the processors.

Step 2: Solve the ● points. From Figure 3.2, assign each processor to a row of ● points ($P0 \rightarrow j = 2$, $P1 \rightarrow j = 4$, $P2 \rightarrow j = 6$, $P3 \rightarrow j = 8$). All processor will solve the ● points using Equation (3.4). After the processors have finish solving the respective rows, they will be assigned to the next row of block ($P0 \rightarrow j = 10$, $P1 \rightarrow j = 12$, $P2 \rightarrow j = 14$, $P3 \rightarrow j = 16$). This process will continue until there are no available row of ● points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 3: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

Step 4: Evaluate the solutions of the remaining □ points using Equation (3.5) directly.

These steps above can be summarized in Algorithm 2.

Parallel QS5 MAOR

Step 1: Solve the ○ points. From Figure 3.3, assign each processor to a row of ○ points ($P0 \rightarrow j = 2$, $P1 \rightarrow j = 4$, $P2 \rightarrow j = 6$, $P3 \rightarrow j = 8$). All processor will solve the ○ points using Equation (3.6). After the processors have finish solving their respective rows, they will be assigned to the next row of ○ points ($P0 \rightarrow j = 10$, $P1 \rightarrow j = 12$, $P2 \rightarrow j = 14$, $P3 \rightarrow j = 16$). This process will continue until there are no available row of ○ points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 2: Solve the ● points. From Figure 3.3, assign each processor to a row of ● points ($P0 \rightarrow j = 2$, $P1 \rightarrow j = 4$, $P2 \rightarrow j = 6$, $P3 \rightarrow j = 8$). All processor will solve the ● points using Equation (3.7). After the processors have finish solving the

Algorithm 2: The iterative process for the parallel HS5 MAOR iterative method.

```
1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into  $\bigcirc$  and  $\bullet$  as shown in Figure 3.2;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
4   Get thread number  $p$ , and maximum number of threads  $np$ ;
5   while  $|u^{k+1} - u^k| > \varepsilon$  do
6     #pragma omp parallel
7     {
8       for  $\{j = 2(p+1) - 1; j < M; j = j + 2np\}$  do
9         Compute all  $\bigcirc$  points using Equation (3.3);
10      end for
11      #pragma omp barrier
12      for  $\{j = 2(p+1); j < M; j = j + 2np\}$  do
13        Compute all  $\bullet$  points using Equation (3.4);
14      end for
15      #pragma omp barrier
16       $ite = ite + 1$ ;
17    }
18  end while
19  Compute all  $\square$  points using Equation (3.5) directly;
20  Display  $ite$ , error, and execution time;
```

respective rows, they will be assigned to the next row of block ($P0 \rightarrow j = 10$, $P1 \rightarrow j = 12$, $P2 \rightarrow j = 14$, $P3 \rightarrow j = 16$). This process will continue until there are no available row of $\bullet$ points left to be solved. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 3: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

Step 4: Evaluate the solutions of the remaining $\square$ points using Equation (3.8) and $\triangle$ points using Equation (3.9) directly.

These steps above can be summarized in Algorithm 3.

Algorithm 3: The iterative process for the parallel QS5 MAOR iterative method.

```
1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into  $\bigcirc$  and  $\bullet$  as shown in Figure 3.3;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
4   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
5   Get thread number  $p$ , and maximum number of threads  $np$ ;
6   while  $|u^{k+1} - u^k| > \varepsilon$  do
7     #pragma omp parallel
8     {
9       for  $\{j = 2(p + 1); j < M; j = j + 2np\}$  do
10        Compute all  $\bigcirc$  points using Equation (3.6);
11      end for
12      #pragma omp barrier
13      for  $\{j = 2(p + 1); j < M; j = j + 2np\}$  do
14        Compute all  $\bullet$  points using Equation (3.7);
15      end for
16      #pragma omp barrier
17       $ite = ite + 1$ ;
18    }
19  end while
20  Compute all  $\square$  points using Equation (3.8) directly;
21  Compute all  $\triangle$  points using Equation (3.9) directly;
22  Display  $ite$ , error, and execution time;
```

3.3.2 Parallel Second Order Group Iterative Methods with MAOR Scheme

Parallel EG5 MAOR

Step 1: Solve the group of four $\bigcirc$ points. From Figure 3.4, assign each processor to a row of block $\bigcirc$ points ($P0 \rightarrow j = 1, P1 \rightarrow j = 3, P2 \rightarrow j = 5, P3 \rightarrow j = 7$). All processor will solve the block of four $\bigcirc$ points using Equation (3.10). After the processors have finish solving their respective rows, they will be assigned to the next row of block $\bigcirc$ points ($P0 \rightarrow j = 9, P1 \rightarrow j = 11, P2 \rightarrow j = 13, P3 \rightarrow j = 15$). This process will continue until there are no available row of block left to be solved. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 2: Solve the remaining group of $\bigcirc$ points at $i = M - 1$ and $j = M - 1$. For $i = M - 1$, each processor will be assigned to a block of $\bigcirc$ points consecutively ($P0 \rightarrow j = 1, P1 \rightarrow j = 5, P2 \rightarrow j = 9, P3 \rightarrow j = 13$). All processor will solve

their respective block of $\bigcirc$ points using Equation (3.11). After the processors have finish solving their respective blocks, they will be assigned to the next block of $\bigcirc$ points consecutively ($P0 \rightarrow j = 17$, $P1 \rightarrow j = 21$, $P2 \rightarrow j = 25$, $P3 \rightarrow j = 29$). This process will continue until there are no available row of block left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Similarly, for $j = M - 1$, each processor will be assigned to a block of $\bigcirc$ points consecutively ($P0 \rightarrow i = 1$, $P1 \rightarrow i = 5$, $P2 \rightarrow i = 9$, $P3 \rightarrow i = 13$). All processor will solve their respective block of $\bigcirc$ points using Equation (3.12). After the processors have finish solving their respective blocks, they will be assigned to the next block of $\bigcirc$ points consecutively ($P0 \rightarrow i = 17$, $P1 \rightarrow i = 21$, $P2 \rightarrow i = 25$, $P3 \rightarrow i = 29$). This process will continue until there are no available row of block left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Finally, the remaining $\bigcirc$ point at $i, j = M - 1$ is solved using Equation (3.1). Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 3: Solve the group of four $\bullet$ points. From Figure 3.4, assign each processor to a row of block $\bullet$ points ($P0 \rightarrow j = 1$, $P1 \rightarrow j = 3$, $P2 \rightarrow j = 5$, $P3 \rightarrow j = 7$). All processor will solve the block of four $\bullet$ points using Equation (3.13). After the processors have finish solving their respective rows, they will be assigned to the next row of block $\bullet$ points ($P0 \rightarrow j = 9$, $P1 \rightarrow j = 11$, $P2 \rightarrow j = 13$, $P3 \rightarrow j = 15$). This process will continue until there are no available row of block left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 4: Solve the remaining group of $\bullet$ points at $i = M - 1$ and $j = M - 1$. For $i = M - 1$, each processor will be assigned to a block of $\bullet$ points consecutively ($P0 \rightarrow j = 3$, $P1 \rightarrow j = 7$, $P2 \rightarrow j = 11$, $P3 \rightarrow j = 15$). All processor will solve their respective block of $\bullet$ points using Equation (3.14). After the processors have fin-

ish solving their respective blocks, they will be assigned to the next block of ● points consecutively ($P0 \rightarrow j = 19, P1 \rightarrow j = 23, P2 \rightarrow j = 27, P3 \rightarrow j = 31$). This process will continue until there are no available row of block left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Similarly, for $j = M - 1$, each processor will be assigned to a block of ● points consecutively ($P0 \rightarrow i = 3, P1 \rightarrow i = 7, P2 \rightarrow i = 11, P3 \rightarrow i = 15$). All processor will solve their respective block of ● points using Equation (3.15). After the processors have finish solving their respective blocks, they will be assigned to the next block of ● points consecutively ($P0 \rightarrow i = 19, P1 \rightarrow i = 23, P2 \rightarrow i = 27, P3 \rightarrow i = 31$). This process will continue until there are no available row of block left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 5: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

These steps above can be summarized in Algorithm 4.

Algorithm 4: The iterative process for the parallel EG5 MAOR iterative method.

```

1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into ○ and ● as shown in Figure 3.4;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
4   Get thread number  $p$ , and maximum number of threads  $np$ ;
5   while  $|u^{k+1} - u^k| > \varepsilon$  do
6     #pragma omp parallel
7     {
8       for  $\{j = 2(p + 1) - 1; j < M - 1; j = j + 2np\}$  do
9         Compute all groups of ○ points using Equation (3.10);
10      end for
11      #pragma omp barrier
12
```

```

13
14      $i = M - 1;$ 
15     for  $\{j = 4(p + 1) - 3; j < M - 1; j = j + 4np\}$  do
16         Compute all groups of  $\bigcirc$  points using Equation (3.11);
17     end for
18     #pragma omp barrier
19      $j = M - 1;$ 
20     for  $\{i = 4(p + 1) - 3; i < M - 1; i = i + 4np\}$  do
21         Compute all groups of  $\bigcirc$  points using Equation (3.12);
22     end for
23     #pragma omp barrier
24      $i = M - 1, j = M - 1$ 
25     Compute  $\bigcirc$  point using Equation (3.1);
26     #pragma omp barrier
27     for  $\{j = 2(p + 1) - 1; j < M - 1; j = j + 2np\}$  do
28         Compute all groups of  $\bullet$  points using Equation (3.13);
29     end for
30     #pragma omp barrier
31      $i = M - 1;$ 
32     for  $\{j = 4(p + 1) - 1; j < M - 1; j = j + 4np\}$  do
33         Compute all groups of  $\bullet$  points using Equation (3.14);
34     end for
35     #pragma omp barrier
36      $j = M - 1;$ 
37     for  $\{i = 4(p + 1) - 1; i < M - 1; i = i + 4np\}$  do
38         Compute all groups of  $\bullet$  points using Equation (3.15);
39     end for
40     #pragma omp barrier
41      $ite = ite + 1;$ 
42     }
43 end while
44 Display  $ite$ , error, and execution time;

```

Parallel EDG5 MAOR

Step 1: Solve the group of two $\bigcirc$ points. From Figure 3.5, assign each processor to a row of block $\bigcirc$ points ($P0 \rightarrow j = 1, P1 \rightarrow j = 5, P2 \rightarrow j = 9, P3 \rightarrow j = 13$). All processor will solve the block of four $\bigcirc$ points using Equation (3.16). After the processors have finish solving their respective rows, they will be assigned to the next row of block $\bigcirc$ points ($P0 \rightarrow j = 17, P1 \rightarrow j = 21, P2 \rightarrow j = 25, P3 \rightarrow j = 29$). This process will continue until there are no available row of block left to be solved. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 2: Solve the remaining $\bigcirc$ points at $i = M - 1$ and $j = M - 1$. For $i = M - 1$, each processor will be assigned to a $\bigcirc$ points consecutively ($P_0 \rightarrow j = 1, P_1 \rightarrow j = 5, P_2 \rightarrow j = 9, P_3 \rightarrow j = 13$). All processor will solve their respective $\bigcirc$ points using Equation (3.3). After the processors have finish solving their respective points, they will be assigned to the next $\bigcirc$ points consecutively ($P_0 \rightarrow j = 17, P_1 \rightarrow j = 21, P_2 \rightarrow j = 25, P_3 \rightarrow j = 29$). This process will continue until there are no available points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Similarly, for $j = M - 1$, each processor will be assigned to a block of $\bigcirc$ points consecutively ($P_0 \rightarrow i = 1, P_1 \rightarrow i = 3, P_2 \rightarrow i = 5, P_3 \rightarrow i = 7$). All processor will solve their respective block of $\bigcirc$ points using Equation (3.3). After the processors have finish solving their respective points, they will be assigned to the next $\bigcirc$ points consecutively ($P_0 \rightarrow i = 9, P_1 \rightarrow i = 11, P_2 \rightarrow i = 13, P_3 \rightarrow i = 15$). This process will continue until there are no available points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 3: Solve the group of two $\bullet$ points. From Figure 3.5, assign each processor to a row of block $\bullet$ points ($P_0 \rightarrow j = 3, P_1 \rightarrow j = 7, P_2 \rightarrow j = 11, P_3 \rightarrow j = 15$). All processor will solve the block of four $\bullet$ points using Equation (3.17). After the processors have finish solving their respective points, they will be assigned to the next row $\bullet$ points ($P_0 \rightarrow j = 19, P_1 \rightarrow j = 23, P_2 \rightarrow j = 27, P_3 \rightarrow j = 31$). This process will continue until there are no available points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 4: Solve the remaining $\bullet$ points at $i = M - 1$. Each processor will be assign to a $\bullet$ point consecutively ($P_0 \rightarrow j = 3, P_1 \rightarrow j = 7, P_2 \rightarrow j = 11, P_3 \rightarrow j = 15$). All processor will solve their respective block of $\bullet$ points using Equation (3.4). After the

processors have finish solving their respective points, they will be assigned to the next

● points consecutively ($P0 \rightarrow j = 19$, $P1 \rightarrow j = 23$, $P2 \rightarrow j = 27$, $P3 \rightarrow j = 31$).

This process will continue until there are no available points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 5: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

Step 6: Evaluate the solutions of the remaining □ points using Equation (3.5) directly.

These steps above can be summarized in Algorithm 5.

Algorithm 5: The iterative process for the parallel EDG5 MAOR iterative method.

```

1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into ○ and ● as shown in Figure 3.5;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
4   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
5   Get thread number  $p$ , and maximum number of threads  $np$ ;
6   while  $|u^{k+1} - u^k| > \varepsilon$  do
7     #pragma omp parallel
8     {
9       for  $\{j = 4(p + 1) - 3; j < M - 1; j = j + 4np\}$  do
10        Compute all groups of ○ points using Equation (3.16);
11      end for
12      #pragma omp barrier
13       $i = M - 1$ ;
14      for  $\{j = 4(p + 1) - 3; j < M - 1; j = j + 4np\}$  do
15        Compute all ○ points using Equation (3.3);
16      end for
17      #pragma omp barrier
18       $j = M - 1$ ;
19      for  $\{i = 2(p + 1) - 1; i < M; i = i + 2np\}$  do
20        Compute all ○ points using Equation (3.3);
21      end for
22      #pragma omp barrier

```

```

23
24   for { $j = 4(p + 1) - 1; j < M - 1; j = j + 4np$ } do
25       Compute all groups of ● points using Equation (3.17);
26   end for
27   #pragma omp barrier
28    $i = M - 1;$ 
29   for { $j = 4(p + 1) - 1; j < M - 1; j = j + 4np$ } do
30       Compute all ● points using Equation (3.4);
31   end for
32   #pragma omp barrier
33    $ite = ite + 1;$ 
34   }
35 end while
36 Compute all □ points using Equation (3.5) directly;
37 Display  $ite$ , error, and execution time;

```

Parallel MEG5 MAOR

Step 1: Solve the block of four ○ points. From Figure 3.6, assign each processor to a row of block ○ points ($P0 \rightarrow j = 2, P1 \rightarrow j = 6, P2 \rightarrow j = 10, P3 \rightarrow j = 14$). All processor will solve the block of four ○ points using Equation (3.18). After the processors have finish solving their respective rows, they will be assigned to the next row of block ○ points ($P0 \rightarrow j = 18, P1 \rightarrow j = 22, P2 \rightarrow j = 26, P3 \rightarrow j = 30$). This process will continue until there are no available row of block left to be solved. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 2: Solve the block of four ● points. From Figure 3.6, assign each processor to a row of block ● points ($P0 \rightarrow j = 2, P1 \rightarrow j = 6, P2 \rightarrow j = 10, P3 \rightarrow j = 14$). All processor will solve the block of four ● points using Equation (3.19). After the processors have finish solving their respective rows, they will be assigned to the next row of block ● points ($P0 \rightarrow j = 18, P1 \rightarrow j = 22, P2 \rightarrow j = 26, P3 \rightarrow j = 30$). This process will continue until there are no available row of block left to be solved. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 3: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

Step 4: Evaluate the solutions of the remaining $\square$ points using Equation (3.8) and $\triangle$ points using Equation (3.9) directly.

These steps above can be summarized in Algorithm 6.

Algorithm 6: The iterative process for the parallel MEG5 MAOR iterative method.

```

1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into  $\circ$  and  $\bullet$  as shown in Figure 3.6;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
4   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
5   Get thread number  $p$ , and maximum number of threads  $np$ ;
6   while  $|u^{k+1} - u^k| > \varepsilon$  do
7     #pragma omp parallel
8     {
9       for  $\{j = 4(p + 1) - 2; j < M; j = j + 4np\}$  do
10        Compute all groups of  $\circ$  points using Equation (3.18);
11      end for
12      #pragma omp barrier
13      for  $\{j = 4(p + 1) - 2; j < M; j = j + 4np\}$  do
14        Compute all groups of  $\bullet$  points using Equation (3.19);
15      end for
16      #pragma omp barrier
17       $ite = ite + 1$ ;
18    }
19  end while
20  Compute all  $\square$  points using Equation (3.8) directly;
21  Compute all  $\triangle$  points using Equation (3.9) directly;
22  Display  $ite$ , error, and execution time;

```

3.4 Experimental Results and Discussions

Two problems will be tested in order to investigate the competency of the proposed method. The tests were performed on various sizes of M while the number of processors ranging from 1 to 8. The performance of the second order MAOR iterative methods in solving the problems are observed in terms of number of iterations, accuracy, execution time, speedup, parallel efficiency and computational costs. The computational cost for each iterative method with the relaxation schemes is measured by

computing the total of arithmetic operations involved which is presented in Tables 3.1 and 3.2.

Problem 1 (Akhir et al. (2011c))

$$u_{xx}(x, y) + u_{yy}(x, y) - 10u(x, y) = 6 - 10(2x^2 + y^2),$$

where $\Omega = [0, 1] \times [0, 1]$, subject to the Dirichlet boundary conditions and satisfying the exact solution $u(x, y) = 2x^2 + y^2$. The test were performed on $M = 66, 130, 258, 514$ with a tolerance of $\varepsilon = 10^{-12}$.

Problem 2 (Othman et al. (2006); Ali and Fu (2007))

$$u_{xx}(x, y) + u_{yy}(x, y) = (x^2 + y^2)e^{xy},$$

where $\Omega = [0, 1] \times [0, 1]$, subject to the Dirichlet boundary conditions and satisfying the exact solution $u(x, y) = e^{xy}$. The test were performed on $M = 66, 130, 258, 514$ with a tolerance of $\varepsilon = 10^{-12}$.

Table 3.3 presents the number of iteration and accuracy of all iterative methods with various relaxation schemes for Problem 1. From the table, the MAOR scheme for all iterative methods reduces the number of iterations by 90.30 – 99.45% when compared with GS scheme, 0.20 – 34.99% when compared with SOR scheme, 0.20 – 23.85% when compared with MSOR scheme, and 0.00 – 31.16% when compared with AOR scheme. Similarly, Table 3.6 presents the number of iteration and accuracy of all iterative methods with various relaxation schemes for Problem 2. From the table, the MAOR scheme for all iterative methods also reduces the number of iterations by 92.35 – 99.47% when compared with GS scheme, 2.61 – 30.82% when compared with SOR scheme, 2.61 – 29.47% when compared with MSOR scheme, and 0.00 – 30.82% when compared with AOR scheme. The MAOR scheme incorporates additional re-

laxation parameters designed to minimize the spectral radius of the iteration matrix. This reduction in the spectral radius leads to faster convergence by dampening error components more effectively, thus requiring fewer iterations to reach a solution.

In terms of accuracy, while the MAOR scheme is not the most accurate solver, it maintains competitive error levels when compared to other relaxation schemes. The reason lies in its prioritisation of minimising the iteration count through the optimisation of relaxation parameters. However, this optimisation does not necessarily guarantee the smallest possible residual error. As a result, although the MAOR scheme converges quickly, the final error can plateau at a higher level than some other schemes that may achieve greater accuracy at the expense of more iterations.

The execution time, speedup and parallel efficiency are shown in Tables 3.4 and 3.5 for Problem 1 and Tables 3.8 and 3.9 for Problem 2. From these tables, we can see that the MAOR scheme does not achieve the fastest execution times compared to some other relaxation schemes but remains competitive. The slower convergence of the MAOR scheme can be attributed to its design, which focuses on minimizing the spectral radius of the iteration matrix. This approach effectively reduces the number of iterations required for convergence. However, it prioritises iteration reduction over minimising residual errors, which can result in slower overall convergence in certain cases. Additionally, the MAOR scheme introduces extra acceleration parameters to enhance convergence, but these parameters can complicate the tuning process. If not optimally chosen, the parameters may lead to less effective convergence. Furthermore, the inclusion of additional computations associated with these parameters increases the per iteration computational cost. This increased cost can offset the benefits of a reduced iteration count, making the MAOR scheme slower than other relaxation methods in terms of total execution time. When parallel algorithms are implemented, the execution time generally decreases as the number of processors increases, and the speedup tends to increase exponentially. This shows that as more processors are added, the

workload can be divided more efficiently, leading to reduced execution time. With an increase in the number of processors, the parallel efficiency of iterative methods improves. This is because the computational load is distributed across multiple processors, which decreases the overall time required for each iteration and increases the efficiency of parallel computation. Figures 3.7 and 3.9 show the execution time while figures 3.8 and 3.10 show the speedup of the second order iterative methods on all relaxation schemes for Problems 1 and 2 at $M = 514$.

The computational costs for all second order iterative methods with various relaxation schemes for both problems are shown in Table 3.7. From this table, the MAOR scheme has a 0.00 – 31.16% reduction in computational costs when compared to the AOR scheme. However, the MAOR scheme has a 1.08 – 45.09% increase in computational costs when compared to the MSOR scheme. The higher computational cost of the MAOR scheme is attributed to the introduction of additional acceleration parameters designed to minimise the spectral radius of the iteration matrix. Although this reduces the number of iterations required for convergence, it increases the computational workload per iteration. Compared to the simpler mechanisms in SOR and MSOR schemes, the MAOR scheme requires more arithmetic operations per iteration, which increases its complexity and computational overhead. This effect is particularly pronounced in large systems, where the added complexity can significantly impact overall performance.

Table 3.1: Computational costs for relaxation schemes in second order point iterative methods ($\sigma = M - 1$).

Method	FS5	HS5	QS5
GS	$6\sigma^2 * \text{ite}$	$\frac{7}{2}(\sigma^2 + 1) * \text{ite} + 3(\sigma^2 - 1)$	$\frac{7}{4}(\sigma - 1)^2 * \text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
SOR	$10\sigma^2 * \text{ite}$	$\frac{11}{2}(\sigma^2 + 1) * \text{ite} + 3(\sigma^2 - 1)$	$\frac{11}{4}(\sigma - 1)^2 * \text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MSOR			
AOR	$5(3\sigma^2 - 1) * \text{ite}$	$(8\sigma^2 - 5\sigma + 8) * \text{ite} + 3(\sigma^2 - 1)$	$4(\sigma - 1)^2 * \text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MAOR			

Table 3.2: Computational costs for relaxation schemes in second order group iterative methods ($\sigma = M - 1$).

Method	EG5	EDG5	MEG5
GS	$(11\sigma^2 - 6\sigma + 1) * \text{ite}$	$\frac{1}{2}(9\sigma^2 - 4\sigma + 9) * \text{ite} + 3(\sigma^2 - 1)$	$3(\sigma - 1)^2 * \text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
SOR	$(15\sigma^2 - 6\sigma + 1) * \text{ite}$	$\frac{1}{2}(13\sigma^2 - 4\sigma + 13) * \text{ite} + 3(\sigma^2 - 1)$	$4(\sigma - 1)^2 * \text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MSOR			
AOR	$\frac{1}{2}(43\sigma^2 - 20\sigma - 3) * \text{ite}$	$\frac{1}{8}(81\sigma^2 - 54\sigma + 61) * \text{ite} + 3(\sigma^2 - 1)$	$\frac{45}{8}(\sigma - 1)^2 * \text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MAOR			

Table 3.3: Number of iterations (Ite) and Error for second order iterative methods with various relaxation schemes for Problem 1.

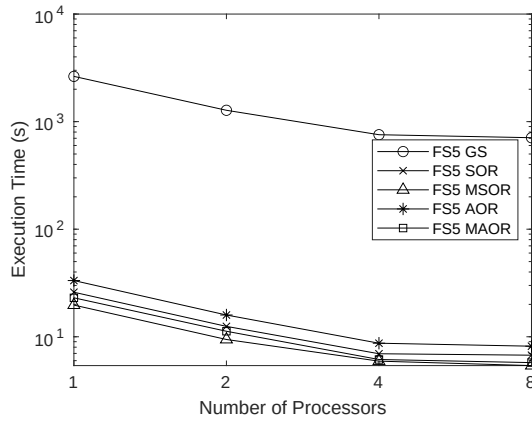
Method	M	Ite	Error	Ite	Error	Ite	Error
FS5				HS5		QS5	
GS	66	6539	2.28E-12	3372	1.13E-12	1736	5.64E-13
	130	23827	4.44E-12	12308	2.22E-12	6351	1.10E-12
	258	87710	8.74E-12	45407	4.37E-12	23479	2.18E-12
	514	323633	1.73E-11	167974	8.68E-12	87066	4.34E-12
SOR	66	247	3.29E-14	180	6.95E-15	129	3.57E-15
	130	544	8.35E-15	345	4.16E-15	243	9.36E-15
	258	1035	7.57E-14	950	3.17E-15	543	4.59E-15
	514	2732	1.90E-15	2723	4.93E-16	1015	3.60E-14
MSOR	66	247	4.18E-14	179	1.34E-14	125	7.74E-15
	130	494	5.34E-15	342	2.70E-15	243	9.36E-15
	258	1035	7.57E-14	706	3.28E-15	492	3.31E-15
	514	2131	3.74E-15	1932	4.93E-16	1015	3.60E-14
AOR	66	244	5.66E-14	175	2.32E-14	128	5.63E-15
	130	530	1.38E-14	326	1.26E-14	241	1.79E-14
	258	1035	7.57E-14	659	2.80E-14	532	1.05E-14
	514	2580	1.58E-14	1299	1.84E-14	1015	3.60E-14
MAOR	66	238	2.15E-14	170	9.86E-15	122	7.09E-15
	130	463	2.26E-14	326	1.26E-14	234	2.40E-14
	258	912	1.88E-14	643	1.67E-14	466	3.01E-14
	514	1776	1.88E-14	1299	1.84E-14	1013	3.63E-14
EG5				EDG5		MEG5	
GS	66	3375	1.13E-12	2564	8.48E-13	897	2.75E-13
	130	12311	2.22E-12	9358	1.66E-12	3277	5.50E-13
	258	45410	4.37E-12	34545	3.28E-12	12131	1.09E-12
	514	167977	8.68E-12	127908	6.51E-12	45075	2.17E-12
SOR	66	178	1.17E-14	172	3.27E-14	89	8.74E-15
	130	340	4.13E-15	330	2.59E-14	178	6.14E-15
	258	694	4.22E-15	744	1.36E-15	347	4.61E-15
	514	1399	2.98E-15	1487	2.78E-15	705	1.22E-15
MSOR	66	178	1.17E-14	172	3.42E-14	89	8.74E-15
	130	340	4.13E-15	330	2.79E-14	178	6.14E-15
	258	694	4.22E-15	674	1.35E-15	347	4.61E-15
	514	1399	2.98E-15	1451	5.57E-14	705	1.22E-15
AOR	66	176	2.55E-14	157	5.82E-15	87	2.56E-14
	130	329	1.19E-14	317	1.77E-14	175	1.80E-14
	258	661	2.49E-14	636	6.99E-15	336	1.37E-14
	514	1308	1.97E-14	1270	2.53E-15	660	1.40E-14
MAOR	66	172	1.59E-14	148	1.04E-14	87	4.98E-15
	130	329	1.52E-14	288	9.68E-15	170	9.40E-15
	258	641	1.80E-14	561	1.33E-14	327	9.36E-15
	514	1284	2.99E-14	1105	6.66E-15	641	7.40E-15

Table 3.4: Execution time (seconds), speedup and parallel efficiency for all second order point iterative methods with various relaxation schemes for Problem 1 at $M = 514$.

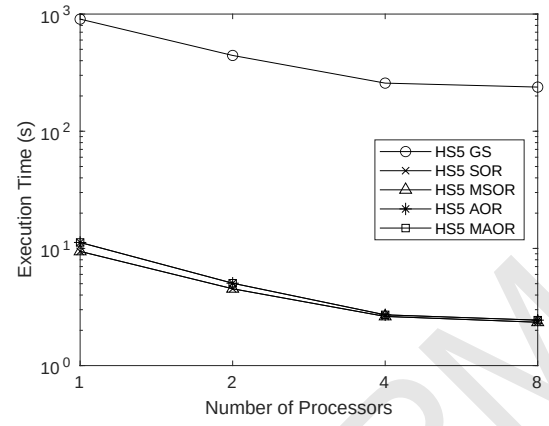
Methods	Execution Time				Speedup				Efficiency			
	T1	T2	T4	T8	S1	S2	S4	S8	E1	E2	E4	E8
FS5 GS	2633.980	1274.870	755.683	708.760	1.000	2.066	3.486	3.716	1.000	1.033	0.871	0.465
FS5 SOR	25.944	12.490	6.952	6.740	1.000	2.077	3.732	3.849	1.000	1.039	0.933	0.481
FS5 MSOR	19.708	9.434	5.934	5.395	1.000	2.089	3.321	3.653	1.000	1.045	0.830	0.457
FS5 AOR	33.312	15.931	8.710	8.172	1.000	2.091	3.825	4.076	1.000	1.046	0.956	0.510
FS5 MAOR	23.020	11.249	6.146	5.751	1.000	2.046	3.746	4.003	1.000	1.023	0.936	0.500
HS5 GS	905.132	443.371	257.401	238.119	1.000	2.041	3.516	3.801	1.000	1.021	0.879	0.475
HS5 SOR	9.433	4.525	2.628	2.344	1.000	2.085	3.589	4.025	1.000	1.042	0.897	0.503
HS5 MSOR	9.433	4.525	2.628	2.344	1.000	2.085	3.589	4.025	1.000	1.042	0.897	0.503
HS5 AOR	11.243	5.047	2.711	2.442	1.000	2.228	4.147	4.604	1.000	1.114	1.037	0.576
HS5 MAOR	11.243	5.047	2.711	2.442	1.000	2.228	4.147	4.604	1.000	1.114	1.037	0.576
QS5 GS	162.809	84.207	49.751	46.752	1.000	1.933	3.272	3.482	1.000	0.967	0.818	0.435
QS5 SOR	2.216	1.154	0.640	0.628	1.000	1.921	3.464	3.528	1.000	0.960	0.866	0.441
QS5 MSOR	2.216	1.154	0.640	0.628	1.000	1.921	3.464	3.528	1.000	0.960	0.866	0.441
QS5 AOR	2.999	1.559	0.935	0.811	1.000	1.924	3.209	3.699	1.000	0.962	0.802	0.462
QS5 MAOR	2.972	1.516	0.939	0.813	1.000	1.961	3.164	3.655	1.000	0.980	0.791	0.457

Table 3.5: Execution time (seconds), speedup and parallel efficiency for all second order group iterative methods with various relaxation schemes for Problem 1 at $M = 514$.

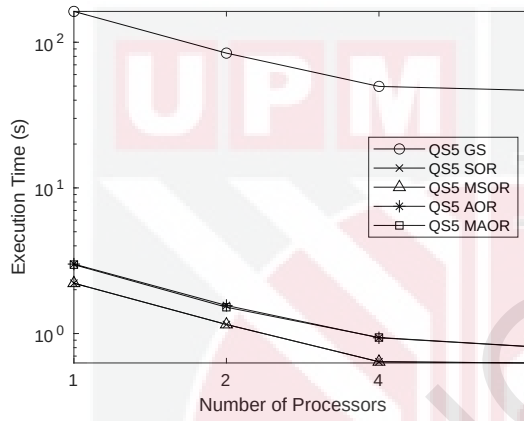
Methods	Execution Time					Speedup				Efficiency			
	T1	T2	T4	T8		S1	S2	S4	S8	E1	E2	E4	E8
EG5 GS	1271.120	609.631	371.026	340.286		1.000	2.085	3.426	3.735	1.000	1.043	0.856	0.467
EG5 SOR	11.862	5.739	3.368	3.048		1.000	2.067	3.522	3.892	1.000	1.033	0.880	0.486
EG5 MSOR	11.862	5.739	3.368	3.048		1.000	2.067	3.522	3.892	1.000	1.033	0.880	0.486
EG5 AOR	15.982	7.197	3.961	3.715		1.000	2.221	4.035	4.302	1.000	1.110	1.009	0.538
EG5 MAOR	15.383	7.102	3.910	3.660		1.000	2.166	3.934	4.203	1.000	1.083	0.984	0.525
EDG5 GS	679.820	310.424	195.467	184.572		1.000	2.190	3.478	3.683	1.000	1.095	0.869	0.460
EDG5 SOR	9.387	4.130	2.345	2.169		1.000	2.273	4.003	4.328	1.000	1.136	1.001	0.541
EDG5 MSOR	9.084	4.113	2.344	2.035		1.000	2.209	3.875	4.463	1.000	1.104	0.969	0.558
EDG5 AOR	10.979	5.036	2.939	2.640		1.000	2.180	3.735	4.158	1.000	1.090	0.934	0.520
EDG5 MAOR	9.966	4.493	2.426	2.352		1.000	2.218	4.107	4.237	1.000	1.109	1.027	0.530
MEG5 GS	69.123	35.821	23.322	20.296		1.000	1.930	2.964	3.406	1.000	0.965	0.741	0.426
MEG5 SOR	1.238	0.667	0.431	0.389		1.000	1.856	2.869	3.108	1.000	0.928	0.717	0.386
MEG5 MSOR	1.238	0.667	0.431	0.389		1.000	1.856	2.869	3.108	1.000	0.928	0.717	0.386
MEG5 AOR	1.474	0.765	0.464	0.441		1.000	1.928	3.178	3.343	1.000	0.964	0.795	0.418
MEG5 MAOR	1.454	0.754	0.452	0.419		1.000	1.929	3.214	3.467	1.000	0.964	0.804	0.433



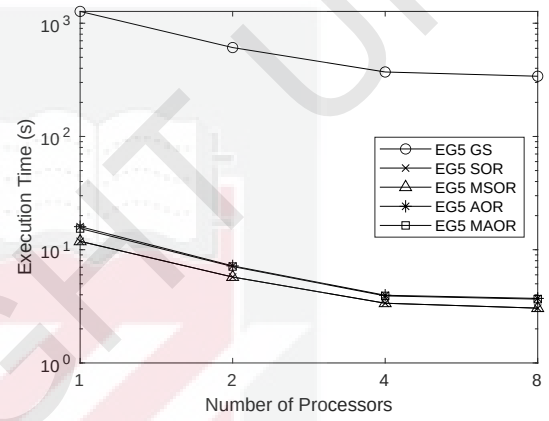
(a)



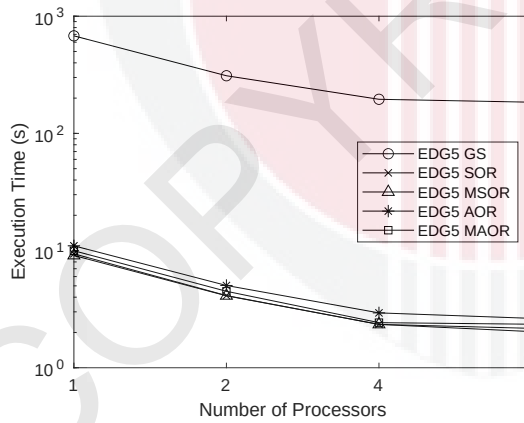
(b)



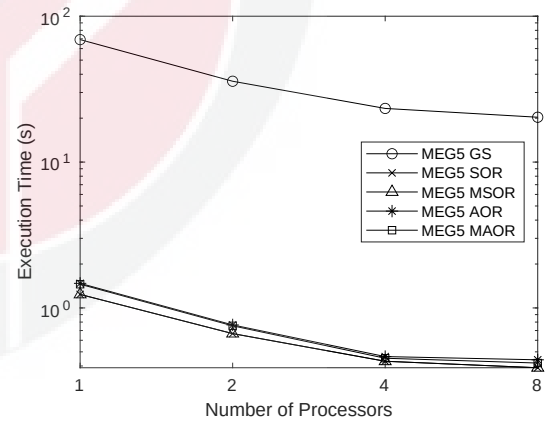
(c)



(d)

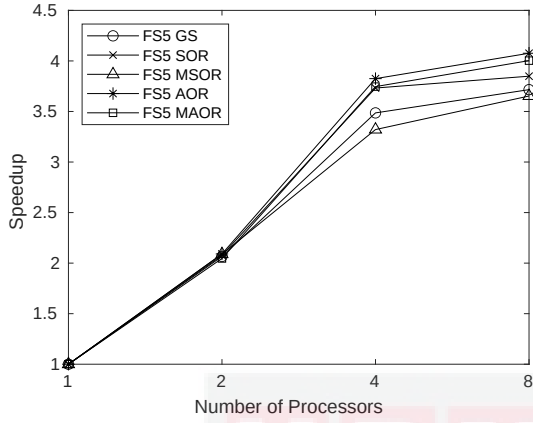


(e)

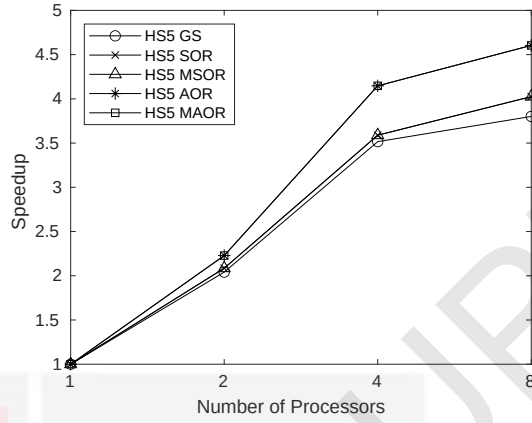


(f)

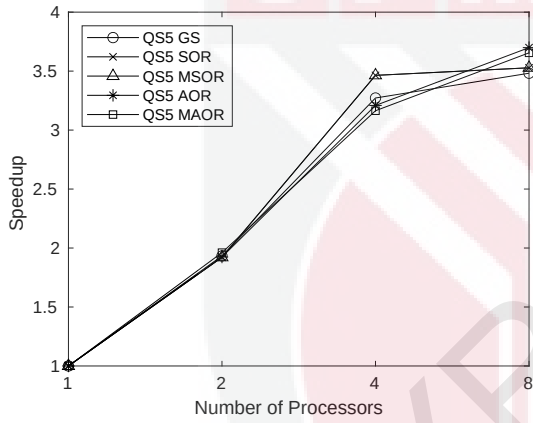
Figure 3.7: The execution time (in seconds) for (a) FS5; (b) HS5; (c) QS5; (d) EG5; (e) EDG5; and (f) MEG5 iterative methods on all relaxation schemes for Problem 1 at $M = 514$.



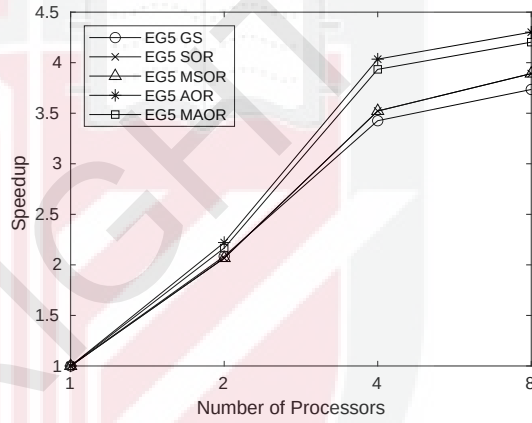
(a)



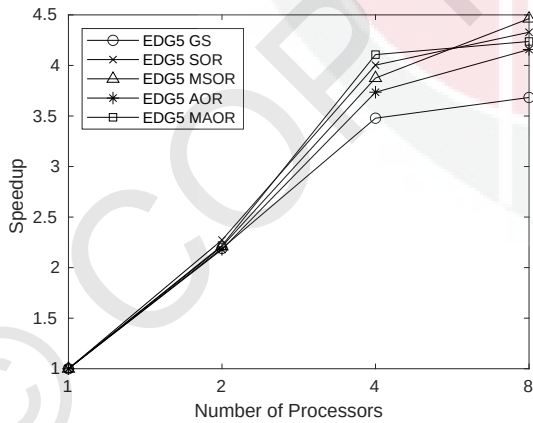
(b)



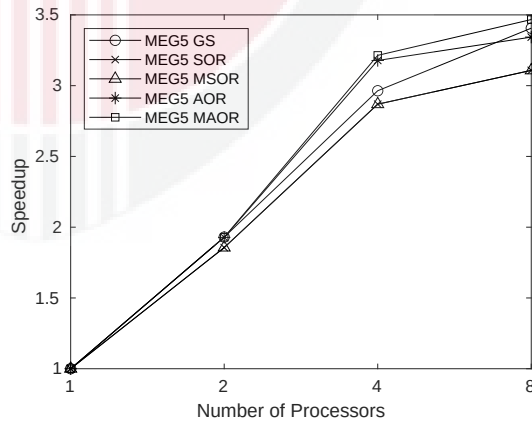
(c)



(d)



(e)



(f)

Figure 3.8: The speedup for (in seconds) for (a) FS5; (b) HS5; (c) QS5; (d) EG5; (e) EDG5; and (f) MEG5 iterative methods on all relaxation schemes for Problem 1 at $M = 514$.

Table 3.6: Number of iterations (Ite) and Error for second order iterative methods with various relaxation schemes for Problem 2.

Method	M	Ite	Error	Ite	Error	Ite	Error
FS5				HS5		QS5	
GS	66	9835	5.59E-09	5071	3.22E-07	2609	2.27E-08
	130	35840	7.16E-10	18514	4.15E-08	9552	2.90E-09
	256	131927	8.00E-11	68301	5.26E-09	35317	3.64E-10
	514	486726	1.66E-11	252640	6.50E-10	130956	4.04E-11
SOR	66	304	5.59E-09	223	3.22E-07	153	2.27E-08
	130	678	7.21E-10	458	4.15E-08	298	2.90E-09
	256	1358	9.16E-11	945	5.26E-09	684	3.67E-10
	514	2703	1.15E-11	2476	6.63E-10	1363	4.62E-11
MSOR	66	304	5.59E-09	216	3.22E-07	153	2.27E-08
	130	605	7.21E-10	425	4.15E-08	298	2.90E-09
	256	1358	9.16E-11	945	5.26E-09	604	3.67E-10
	514	2703	1.15E-11	1924	6.63E-10	1353	4.61E-11
AOR	66	298	5.59E-09	217	3.22E-07	150	2.27E-08
	130	653	7.22E-10	438	4.15E-08	291	2.90E-09
	256	1292	9.16E-11	881	5.26E-09	659	3.67E-10
	514	2560	1.16E-11	2476	6.63E-10	1289	4.62E-11
MAOR	66	288	5.59E-09	205	3.22E-07	149	2.27E-08
	130	568	7.21E-10	399	4.15E-08	288	2.90E-09
	256	1167	9.15E-11	802	5.26E-09	563	3.67E-10
	514	2560	1.16E-11	1713	6.63E-10	1277	4.62E-11
EG5				EDG5		MEG5	
GS	66	5074	5.59E-09	3853	3.22E-07	1346	2.27E-08
	130	18517	7.18E-10	14073	4.15E-08	4927	2.90E-09
	256	68304	8.58E-11	51955	5.26E-09	18246	3.65E-10
	514	252643	5.85E-12	192370	6.54E-10	67801	4.33E-11
SOR	66	223	5.59E-09	204	3.22E-07	111	2.27E-08
	130	456	7.21E-10	416	4.15E-08	226	2.90E-09
	256	925	9.16E-11	902	5.26E-09	465	3.67E-10
	514	2477	1.15E-11	1686	6.63E-10	946	4.62E-11
MSOR	66	216	5.59E-09	204	3.22E-07	107	2.27E-08
	130	446	7.21E-10	416	4.15E-08	217	2.90E-09
	256	925	9.16E-11	841	5.26E-09	447	3.67E-10
	514	2477	1.15E-11	1686	6.63E-10	946	4.62E-11
AOR	66	217	5.59E-09	189	3.22E-07	110	2.27E-08
	130	439	7.21E-10	358	4.15E-08	219	2.90E-09
	256	883	9.16E-11	823	5.26E-09	430	3.67E-10
	514	2477	1.15E-11	1686	6.63E-10	882	4.62E-11
MAOR	66	205	5.59E-09	176	3.22E-07	103	2.27E-08
	130	398	7.22E-10	341	4.15E-08	206	2.90E-09
	256	802	9.16E-11	667	5.26E-09	395	3.67E-10
	514	1747	1.15E-11	1326	6.63E-10	847	4.62E-11

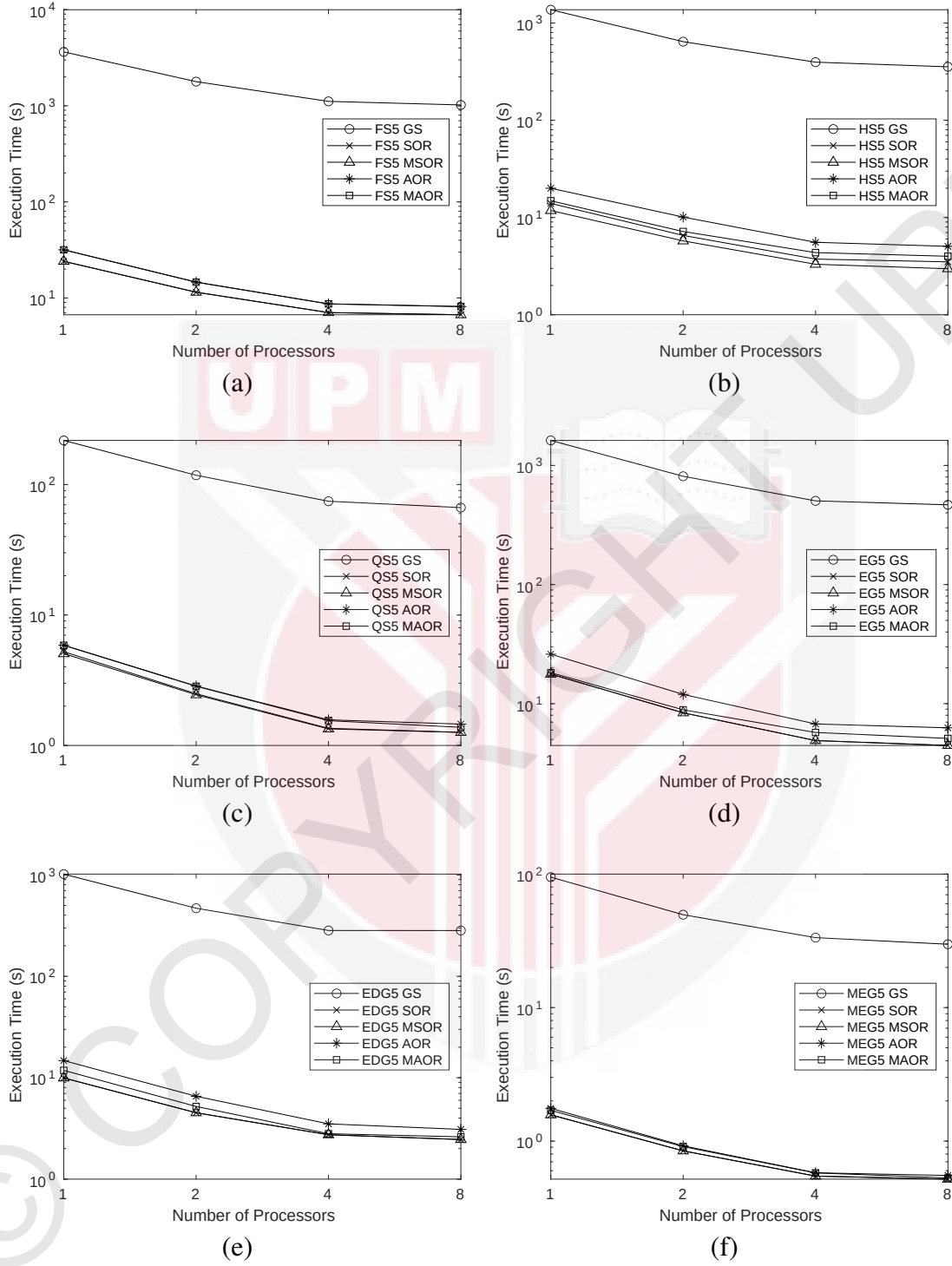


Figure 3.9: The execution time (in seconds) for (a) FS5; (b) HS5; (c) QS5; (d) EG5; (e) EDG5; and (f) MEG5 iterative methods on all relaxation schemes for Problem 2 at $M = 514$.

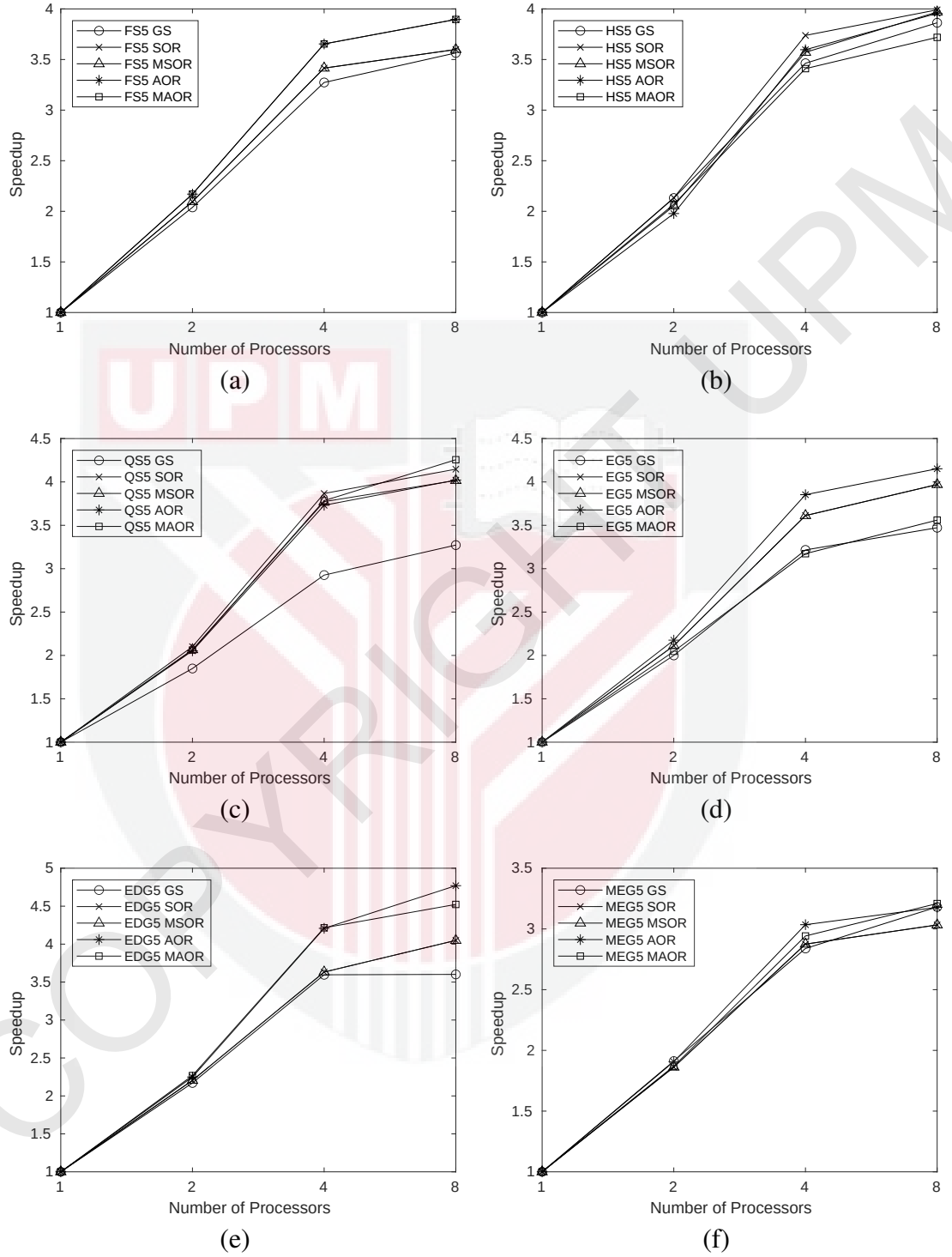


Figure 3.10: The speedup for (in seconds) for (a) FS5; (b) HS5; (c) QS5; (d) EG5; (e) EDG5; and (f) MEG5 iterative methods on all relaxation schemes for Problem 2 at $M = 514$.

Table 3.7: Computational costs of second order iterative methods with various relaxation schemes for Problems 1 and 2 at $M = 514$.

Methods	Computational Cost	
	Problem 1	Problem 2
FS5 GS	511,021,037,862	768,547,168,164
FS5 SOR	7,189,777,080	7,113,458,070
FS5 MSOR	5,608,131,390	7,113,458,070
FS5 AOR	10,184,627,400	10,105,676,800
FS5 MAOR	7,010,813,280	10,105,676,800
HS5 GS	154,720,801,034	232,706,230,304
HS5 SOR	3,942,155,009	3,584,638,564
HS5 MSOR	2,797,233,924	2,785,654,444
HS5 AOR	2,732,320,209	5,207,309,924
HS5 MAOR	2,732,320,209	3,602,877,339
QS5 GS	39,942,953,479	60,077,578,759
QS5 SOR	732,961,287	983,833,095
QS5 MSOR	732,961,287	976,624,135
QS5 AOR	1,065,556,487	1,352,866,311
QS5 MAOR	1,063,459,335	1,340,283,399
EG5 GS	485,752,865,014	730,588,479,826
EG5 SOR	5,518,296,742	9,770,422,466
EG5 MSOR	5,518,296,742	9,770,422,466
EG5 AOR	7,394,126,616	14,002,485,954
EG5 MAOR	7,258,454,568	9,875,794,494
EDG5 GS	151,346,523,516	227,620,475,934
EDG5 SOR	2,542,933,477	2,883,139,698
EDG5 MSOR	2,481,388,633	2,883,139,698
EDG5 AOR	3,380,425,874	4,487,456,370
EDG5 MAOR	2,941,339,259	3,529,449,210
MEG5 GS	35,449,674,247	53,322,127,879
MEG5 SOR	740,497,927	993,204,743
MEG5 MSOR	740,497,927	993,204,743
MEG5 AOR	974,461,447	1,301,813,767
MEG5 MAOR	946,444,807	1,250,204,167

Table 3.8: Execution time (seconds), speedup and parallel efficiency for all second order point iterative methods with various relaxation schemes for Problem 2 at $M = 514$.

Methods	Execution Time				Speedup				Efficiency			
	T1	T2	T4	T8	S1	S2	S4	S8	E1	E2	E4	E8
FS5 GS	3639.700	1784.750	1111.970	1020.980	1.000	2.039	3.273	3.565	1.000	1.020	0.818	0.446
FS5 SOR	24.078	11.486	7.050	6.690	1.000	2.096	3.415	3.599	1.000	1.048	0.854	0.450
FS5 MSOR	24.078	11.486	7.050	6.690	1.000	2.096	3.415	3.599	1.000	1.048	0.854	0.450
FS5 AOR	31.726	14.621	8.682	8.143	1.000	2.170	3.654	3.896	1.000	1.085	0.914	0.487
FS5 MAOR	31.726	14.621	8.682	8.143	1.000	2.170	3.654	3.896	1.000	1.085	0.914	0.487
HS5 GS	1370.800	643.352	395.855	354.877	1.000	2.131	3.463	3.863	1.000	1.065	0.866	0.483
HS5 SOR	13.969	6.546	3.737	3.500	1.000	2.134	3.738	3.991	1.000	1.067	0.935	0.499
HS5 MSOR	11.805	5.749	3.306	2.974	1.000	2.053	3.571	3.969	1.000	1.027	0.893	0.496
HS5 AOR	19.996	10.109	5.559	5.055	1.000	1.978	3.597	3.956	1.000	0.989	0.899	0.494
HS5 MAOR	14.857	7.180	4.357	3.995	1.000	2.069	3.410	3.719	1.000	1.035	0.853	0.465
QS5 GS	217.896	117.879	74.479	66.591	1.000	1.849	2.926	3.272	1.000	0.924	0.731	0.409
QS5 SOR	5.234	2.491	1.353	1.262	1.000	2.101	3.867	4.147	1.000	1.050	0.967	0.518
QS5 MSOR	5.055	2.447	1.340	1.258	1.000	2.066	3.773	4.017	1.000	1.033	0.943	0.502
QS5 AOR	5.864	2.856	1.572	1.458	1.000	2.053	3.731	4.021	1.000	1.027	0.933	0.503
QS5 MAOR	5.861	2.842	1.549	1.377	1.000	2.062	3.784	4.256	1.000	1.031	0.946	0.532

Table 3.9: Execution time (seconds), speedup and parallel efficiency for all second order group iterative methods with various relaxation schemes for Problem 2 at $M = 514$.

Methods	Execution Time					Speedup				Efficiency			
	T1	T2	T4	T8		S1	S2	S4	S8	E1	E2	E4	E8
EG5 GS	1621.970	811.106	504.322	467.145		1.000	2.000	3.216	3.472	1.000	1.000	0.804	0.434
EG5 SOR	17.703	8.381	4.901	4.459		1.000	2.112	3.612	3.970	1.000	1.056	0.903	0.496
EG5 MSOR	17.703	8.381	4.901	4.459		1.000	2.112	3.612	3.970	1.000	1.056	0.903	0.496
EG5 AOR	26.033	11.964	6.757	6.270		1.000	2.176	3.853	4.152	1.000	1.088	0.963	0.519
EG5 MAOR	18.190	8.884	5.732	5.110		1.000	2.047	3.173	3.560	1.000	1.024	0.793	0.445
EDG5 GS	1018.200	469.069	283.132	282.734		1.000	2.171	3.596	3.601	1.000	1.085	0.899	0.450
EDG5 SOR	9.951	4.514	2.740	2.458		1.000	2.205	3.632	4.049	1.000	1.102	0.908	0.506
EDG5 MSOR	9.951	4.514	2.740	2.458		1.000	2.205	3.632	4.049	1.000	1.102	0.908	0.506
EDG5 AOR	14.756	6.562	3.507	3.094		1.000	2.249	4.208	4.770	1.000	1.124	1.052	0.596
EDG5 MAOR	11.815	5.209	2.803	2.612		1.000	2.268	4.215	4.523	1.000	1.134	1.054	0.565
MEG5 GS	94.879	49.632	33.423	29.830		1.000	1.912	2.839	3.181	1.000	0.956	0.710	0.398
MEG5 SOR	1.574	0.845	0.547	0.519		1.000	1.862	2.875	3.033	1.000	0.931	0.719	0.379
MEG5 MSOR	1.574	0.845	0.547	0.519		1.000	1.862	2.875	3.033	1.000	0.931	0.719	0.379
MEG5 AOR	1.759	0.922	0.579	0.553		1.000	1.907	3.035	3.178	1.000	0.954	0.759	0.397
MEG5 MAOR	1.700	0.909	0.578	0.530		1.000	1.870	2.942	3.209	1.000	0.935	0.735	0.401

CHAPTER 4

FOURTH ORDER MAOR ITERATIVE METHOD FOR SOLVING HELMHOLTZ EQUATION

4.1 Introduction

This chapter is devoted to the formulation of fourth order MAOR iterative methods. It starts with the implementation of the MAOR scheme on FS9, HS9, QS9, EG9, EDG9, and MEG9 iterative methods which includes different ordering strategies of the solution domain. The solution domain is ordered such that the tasks executed will be independent from each other. Then, the chapter is continued with the sequential algorithm for the fourth order MAOR iterative methods. This will describe how each iterative method is formulated with their respective ordering strategies. Next, the chapter provides the construction of parallel algorithm for the fourth order MAOR iterative methods. With the tasks being independent from each other, it is possible to compute them in a parallel environment.

4.2 Implementation of Sequential Algorithms for Fourth Order MAOR Iterative Methods

This section presents the implementation of sequential algorithms for MAOR scheme on the fourth order iterative methods. The inclusion of the solution domain for each iterative method with their respective ordering strategies will assist in the derivation of the scheme. For simplicity, let's denote $v_{i,j} = u_{i,j}^{k+1} - u_{i,j}^k$.

4.2.1 Sequential Fourth Order Point Iterative Methods with MAOR Scheme

Sequential FS9 MAOR

Figure 4.1 shows the solution domain for FS9 with the horizontal zebra line (HZL) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

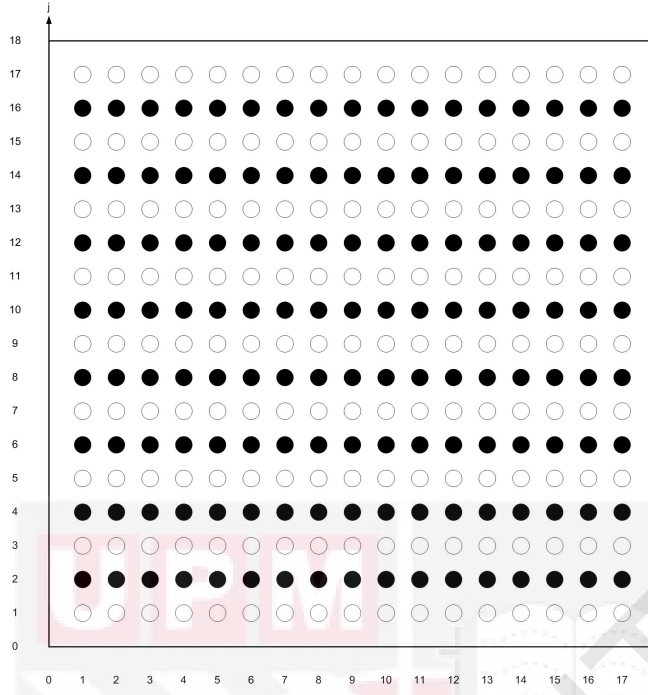


Figure 4.1: The solution domain for FS9.

Equation (2.10) can be generally represented in the following matrix form

$$D = \begin{bmatrix} \rho_4 & & & \\ & \rho_4 & & \\ & & \ddots & \\ & & & \rho_4 \end{bmatrix}, \quad L = \begin{bmatrix} H_2 & 0 \\ H_1 & H_2 \end{bmatrix}, \quad U = L^T,$$

$$H_1 = \begin{bmatrix} J_1 & J_1 & & & \\ & J_1 & J_1 & & \\ & & \ddots & \ddots & \\ & & & J_1 & J_1 \\ & & & J_1 & J_1 \end{bmatrix}, \quad H_2 = \begin{bmatrix} J_2 & & & & \\ & J_2 & & & \\ & & \ddots & & \\ & & & J_2 & \\ & & & & J_2 \end{bmatrix},$$

$$J_1 = \begin{bmatrix} -q_4 & -1 & & & \\ -1 & -q_4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & -q_4 & -1 \\ & & & -1 & -q_4 \end{bmatrix}, \quad J_2 = \begin{bmatrix} -q_4 & & & & \\ & -q_4 & & & \\ & & \ddots & & \\ & & & -q_4 & \\ & & & & -q_4 \end{bmatrix},$$

$$b = \frac{1}{2}h^2 \begin{bmatrix} F1_{1,1} \\ F1_{2,1} \\ \vdots \\ F1_{i,j} \\ \vdots \\ F1_{M-2,M-1} \\ F1_{M-1,M-1} \end{bmatrix}$$

Thus, by implementing the above matrices into Equation (2.19), the sequential FS9 MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all $\odot$ points using

$$\begin{aligned} \odot \quad u_{i,j}^{k+1} &= \frac{\theta_1}{\rho_4} (q_4(v_{i-1,j})) \\ &+ \frac{\omega_1}{\rho_4} \left(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k \right. \\ &\quad \left. + q_4(u_{i+1,j}^k + u_{i-1,j}^k + u_{i,j+1}^k + u_{i,j-1}^k) - \frac{1}{2}h^2 F1_{i,j} \right) \\ &+ (1 - \omega_1)u_{i,j}^k. \end{aligned} \tag{4.1}$$

3. Execute all ● points using

$$\begin{aligned}
 \bullet \quad u_{i,j}^{k+1} = & \frac{\theta_2}{\rho_4} (v_{i-1,j-1} + v_{i+1,j-1} + v_{i-1,j+1} + v_{i+1,j+1} \\
 & + q_4(v_{i-1,j} + v_{i,j+1} + v_{i,j-1})) \\
 & + \frac{\omega_2}{\rho_4} (u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k \\
 & + q_4(u_{i+1,j}^k + u_{i-1,j}^k + u_{i,j+1}^k + u_{i,j-1}^k) - \frac{1}{2}h^2 F_{1_{i,j}}) \\
 & + (1 - \omega_2)u_{i,j}^k.
 \end{aligned} \tag{4.2}$$

4. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, go to step (5), otherwise, repeat the iteration cycle (i. e. go to step (2)).

5. Stop.

Sequential HS9 MAOR

Figure 4.2 shows the solution domain for HS9 with the 3 color horizontal zebra line (HZL) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

Equation (2.11) can be generally represented in the following matrix form

$$D = \begin{bmatrix} \rho_5 & & & & \\ & \rho_5 & & & \\ & & \ddots & & \\ & & & \rho_5 & \\ & & & & \rho_5 \end{bmatrix}, \quad L = \begin{bmatrix} H_4 & 0 & 0 \\ H_2 & H_4 & 0 \\ H_1 & H_3 & H_4 \end{bmatrix}, \quad U = L^T,$$

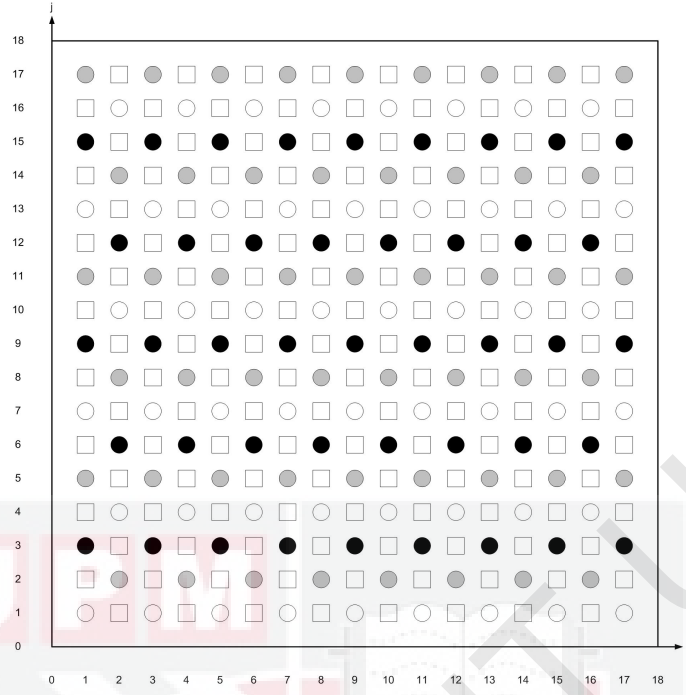


Figure 4.2: The solution domain for HS9.

$$H_1 = \begin{bmatrix} J_1 & & & & \\ & J_1^T & & & \\ -I & & \ddots & & \\ & \ddots & & J_1 & \\ & & & -I & J_1^T \end{bmatrix}, \quad H_2 = \begin{bmatrix} -I & & & & \\ & -I & & & \\ J_1 & & \ddots & & \\ & J_1^T & & -I & \\ & & \ddots & & -I \\ & & & J_1 & J_1^T \end{bmatrix},$$

$$H_3 = \begin{bmatrix} J_1 & & -I & & \\ & J_1^T & & -I & \\ & & \ddots & & \ddots \\ & & & J_1 & -I \\ & & & J_1^T & -I \end{bmatrix}, \quad H_4 = \begin{bmatrix} J_2 & & & & \\ & J_2 & & & \\ & & \ddots & & \\ & & & J_2 & \\ & & & & J_2 \end{bmatrix},$$

$$J_1 = \begin{bmatrix} -q_5 & -q_5 & & & \\ & -q_5 & -q_5 & & \\ & & \ddots & \ddots & \\ & & & -q_5 & -q_5 \\ & & & -q_5 & -q_5 \end{bmatrix}, \quad J_2 = \begin{bmatrix} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & -1 & \\ & & & & -1 \end{bmatrix},$$

$$b = h^2 \begin{bmatrix} F^{2,4} \\ F^{4,4} \\ \vdots \\ F^{2i,j} \\ \vdots \\ F^{M-3,M-3} \\ F^{M-1,M-3} \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential HS9 MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all $\odot$ points using

$$\begin{aligned} \odot \quad u_{i,j}^{k+1} &= \frac{\theta_1}{\rho_5} (v_{i-2,j}) \\ &+ \frac{\omega_1}{\rho_5} \left(u_{i-2,j}^k + u_{i+2,j}^k + u_{i,j-2}^k + u_{i,j+2}^k \right. \\ &+ q_5 (u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k) - h^2 F^{2i,j} \Big) \\ &+ (1 - \omega_1) u_{i,j}^k, \end{aligned} \quad (4.3)$$

and

$$\begin{aligned} \bigcirc \quad u_{i,j}^{k+1} = & \frac{\omega_1}{q_5^*} \left(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k - \frac{h^2}{6} F_{2i,j} \right) \\ & + (1 - \omega_1) u_{i,j}^k, \end{aligned} \quad (4.4)$$

for the $\bigcirc$ points near the boundary.

3. Execute all $\bigcirc$ points using

$$\begin{aligned} \bigcirc \quad u_{i,j}^{k+1} = & \frac{\theta_2}{\rho_5} (v_{i-2,j} + v_{i,j+2} + q_5(v_{i-1,j-1} + v_{i+1,j-1})) \\ & + \frac{\omega_2}{\rho_5} \left(u_{i-2,j}^k + u_{i+2,j}^k + u_{i,j-2}^k + u_{i,j+2}^k \right. \\ & \left. + q_5(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k) - h^2 F_{2i,j} \right) \\ & + (1 - \omega_2) u_{i,j}^k, \end{aligned} \quad (4.5)$$

and

$$\begin{aligned} \bigcirc \quad u_{i,j}^{k+1} = & \frac{\theta_2}{q_5^*} (v_{i-1,j-1} + v_{i+1,j-1}) \\ & + \frac{\omega_2}{q_5^*} \left(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k - \frac{h^2}{6} F_{2i,j} \right) \\ & + (1 - \omega_2) u_{i,j}^k, \end{aligned} \quad (4.6)$$

for the $\bigcirc$ points near the boundary.

4. Execute all $\bullet$ points using

$$\begin{aligned} \bullet \quad u_{i,j}^{k+1} = & \frac{\theta_3}{\rho_5} (v_{i-2,j} + v_{i,j-2} + v_{i,j+2} \\ & + q_5(v_{i-1,j-1} + v_{i+1,j-1} + v_{i-1,j+1} + v_{i+1,j+1})) \\ & + \frac{\omega_3}{\rho_5} \left(u_{i-2,j}^k + u_{i+2,j}^k + u_{i,j-2}^k + u_{i,j+2}^{(k)} \right. \\ & \left. + q_5(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k) - h^2 F_{2i,j} \right) \\ & + (1 - \omega_3) u_{i,j}^k, \end{aligned} \quad (4.7)$$

and

$$\begin{aligned}
 \bullet \quad u_{i,j}^{k+1} &= \frac{\theta_3}{q_5^*} (v_{i-1,j-1} + v_{i+1,j-1} + v_{i-1,j+1} + v_{i+1,j+1}) \\
 &+ \frac{\omega_3}{q_5^*} \left(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k - \frac{h^2}{6} F_{2i,j} \right) \\
 &+ (1 - \omega_3) u_{i,j}^k,
 \end{aligned} \tag{4.8}$$

for the $\bullet$ points near the boundary.

5. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, evaluate the solutions of the remaining $\square$ points using Equation (3.5). Otherwise, repeat the iteration cycle (i. e. go to step (2)).
6. Stop.

Sequential QS9 MAOR

Figure 4.3 shows the solution domain for QS9 with the horizontal zebra line (HZL) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

Equation (2.13) can be generally represented in the following matrix form

$$D = \begin{bmatrix} \rho_6 & & & & \\ & \rho_6 & & & \\ & & \ddots & & \\ & & & \rho_6 & \\ & & & & \rho_6 \end{bmatrix}, \quad L = \begin{bmatrix} H_2 & 0 \\ H_1 & H_2 \end{bmatrix}, \quad U = L^T,$$

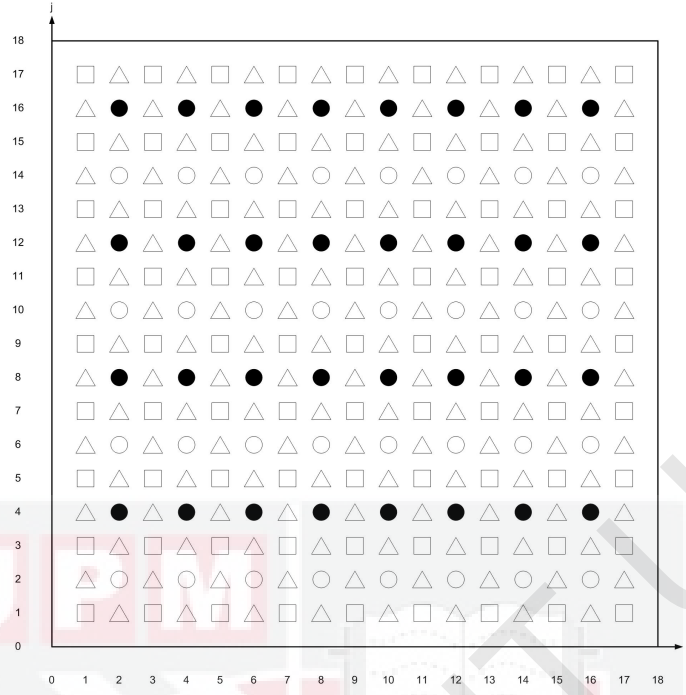


Figure 4.3: The solution domain for QS9.

$$H_1 = \begin{bmatrix} J_1 & J_1 & & & \\ & J_1 & J_1 & & \\ & & \ddots & \ddots & \\ & & & J_1 & J_1 \\ & & & & J_1 \end{bmatrix}, \quad H_2 = \begin{bmatrix} J_2 & & & & \\ & J_2 & & & \\ & & \ddots & & \\ & & & J_2 & \\ & & & & J_2 \end{bmatrix},$$

$$J_1 = \begin{bmatrix} -q_6 & -1 & & & \\ -1 & -q_6 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & -q_6 & -1 \\ & & & -1 & -q_6 \end{bmatrix}, \quad J_2 = \begin{bmatrix} -q_6 & & & & \\ & -q_6 & & & \\ & & \ddots & & \\ & & & -q_6 & \\ & & & & -q_6 \end{bmatrix},$$

$$b = 2h^2 \begin{bmatrix} F3_{2,2} \\ F3_{4,2} \\ \vdots \\ F3_{i,j} \\ \vdots \\ F3_{M-4,M-2} \\ F3_{M-2,M-2} \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential QS9 MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all $\bigcirc$ points using

$$\begin{aligned} \bigcirc \quad u_{i,j}^{k+1} &= \frac{\theta_1}{\rho_6} (q_6(v_{i-2,j})) \\ &+ \frac{\omega_1}{\rho_6} (u_{i-2,j-2}^k + u_{i+2,j-2}^k + u_{i-2,j+2}^k + u_{i+2,j+2}^k \\ &+ q_6(u_{i+2,j}^k + u_{i-2,j}^k + u_{i,j+2}^k + u_{i,j-2}^k) - 2h^2 F3_{i,j}) \\ &+ (1 - \omega_1)u_{i,j}^k. \end{aligned} \quad (4.9)$$

3. Execute all $\bullet$ points using

$$\begin{aligned} \bullet \quad u_{i,j}^{k+1} &= \frac{\theta_2}{\rho_6} (v_{i-2,j-2} + v_{i+2,j-2} + v_{i-2,j+2} + v_{i+2,j+2} \\ &+ q_6(v_{i-2,j} + v_{i,j+2} + v_{i,j-2})) \\ &+ \frac{\omega_2}{\rho_6} (u_{i-2,j-2}^k + u_{i+2,j-2}^k + u_{i-2,j+2}^k + u_{i+2,j+2}^k \\ &+ q_6(u_{i+2,j}^k + u_{i-2,j}^k + u_{i,j+2}^k + u_{i,j-2}^k) - 2h^2 F3_{i,j}) \\ &+ (1 - \omega_2)u_{i,j}^k. \end{aligned} \quad (4.10)$$

4. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges,

evaluate the solutions of the remaining $\square$ points using Equation (3.8) followed by the remaining $\triangle$ points using Equation (3.9). Otherwise, repeat the iteration cycle (i. e. go to step (2)).

5. Stop.

4.2.2 Sequential Fourth Order Group Iterative Methods with MAOR Scheme

Sequential EG9 MAOR

Figure 4.4 shows the solution domain for EG9 with the horizontal zebra line (HZL) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

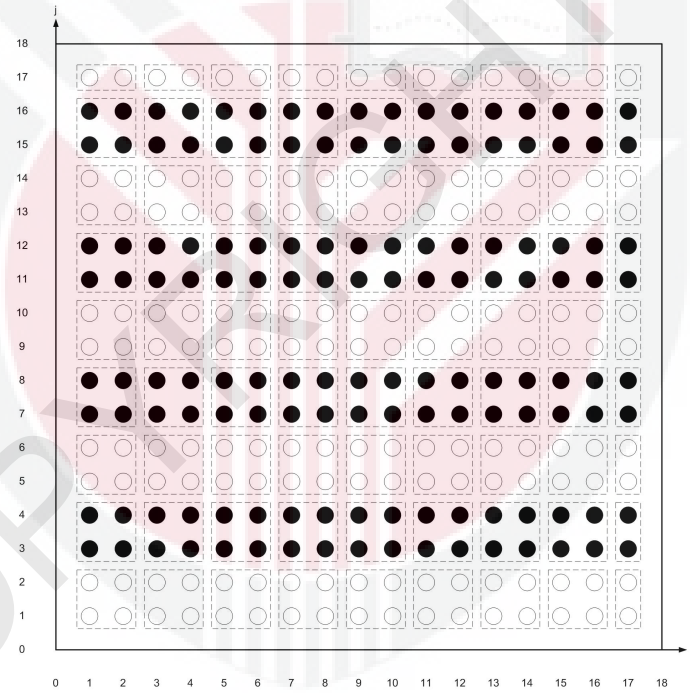


Figure 4.4: The solution domain for EG9.

Let

$$A_4 = \begin{bmatrix} R_1 & R_2 & R_2 & R_3 \\ R_2 & R_1 & R_3 & R_2 \\ R_2 & R_3 & R_1 & R_2 \\ R_3 & R_2 & R_2 & R_1 \end{bmatrix},$$

and

$$\begin{aligned}
S_1^k &= u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + q_4(u_{i-1,j}^k + u_{i,j-1}^k) - \frac{1}{2}h^2 F_{1i,j}, \\
S_2^k &= u_{i,j-1}^k + u_{i+2,j+1}^k + u_{i+2,j-1}^k + q_4(u_{i+2,j}^k + u_{i+1,j-1}^k) - \frac{1}{2}h^2 F_{1i+1,j}, \\
S_3^k &= u_{i-1,j}^k + u_{i-1,j+2}^k + u_{i+1,j+2}^k + q_4(u_{i-1,j+1}^k + u_{i,j+2}^k) - \frac{1}{2}h^2 F_{1i,j+1}, \\
S_4^k &= u_{i,j+2}^k + u_{i+2,j+2}^k + u_{i+2,j}^k + q_4(u_{i+2,j+1}^k + u_{i+1,j+2}^k) - \frac{1}{2}h^2 F_{1i+1,j+1}.
\end{aligned}$$

Equation (2.14) can be generally represented in the following matrix form

$$D = \begin{bmatrix} A_4 & & & \\ & A_4 & & \\ & & \ddots & \\ & & & A_4 \\ & & & & A_4 \end{bmatrix}, \quad L = \begin{bmatrix} H_2 & 0 \\ H_1 & H_2 \end{bmatrix}, \quad U = L^T,$$

$$H_1 = \begin{bmatrix} J_1 & J_3 & & & J_1^T & J_2^T \\ J_2 & J_1 & J_3 & & J_3^T & \ddots & \ddots \\ & J_2 & J_1 & J_3 & & \ddots & \ddots & J_2^T \\ & & J_2 & J_1 & \ddots & & J_3^T & J_1^T \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \ddots & J_1 & J_3 & \\ & & & & & J_2 & J_1 & J_3 \\ & & & & & & J_2 & J_1 \end{bmatrix}, \quad b = \begin{bmatrix} G_1 \\ G_1 \\ \vdots \\ G_1 \\ G_1 \end{bmatrix},$$

$$H_2 = \begin{bmatrix} J_4 & & & \\ & J_4 & & \\ & & \ddots & \\ & & & J_4 \\ & & & & J_4 \end{bmatrix}, \quad G_1 = \frac{1}{2}h^2 \begin{bmatrix} F1_{i,j} \\ F1_{i+1,j} \\ F1_{i,j+1} \\ F1_{i+1,j+1} \end{bmatrix},$$

$$J_1 = \begin{bmatrix} 0 & 0 & -q_4 & -1 \\ 0 & 0 & -1 & -q_4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad J_2 = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$J_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad J_4 = \begin{bmatrix} 0 & -q_4 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -q_4 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential EG9 MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all groups of $\odot$ points using

$$\begin{aligned}
\bigcirc \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix}^{k+1} &= \frac{\theta_1}{Q_4} A_4 \begin{bmatrix} S_1^{k+1} - S_1^k \\ 0 \\ S_3^{k+1} - S_3^k \\ 0 \end{bmatrix} + \frac{\omega_1}{Q_4} A_4 \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}^k \\
&+ (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix}^k.
\end{aligned} \tag{4.11}$$

3. Execute the remaining groups of two $\bigcirc$ points at $i = M - 1$ using

$$\begin{aligned}
\bigcirc \begin{bmatrix} u_{i,j} \\ u_{i,j+1} \end{bmatrix}^{k+1} &= \frac{\theta_1}{\rho_4^2 - q_4^2} \begin{bmatrix} \rho_4 & q_4 \\ q_4 & \rho_4 \end{bmatrix} \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_3^{k+1} - S_3^k \end{bmatrix} \\
&+ \frac{\omega_1}{\rho_4^2 - q_4^2} \begin{bmatrix} \rho_4 & q_4 \\ q_4 & \rho_4 \end{bmatrix} \begin{bmatrix} S_1 + q_4 u_{i+1,j} + u_{i+1,j+1} \\ S_3 + q_4 u_{i+1,j+1} + u_{i+1,j} \end{bmatrix}^k \\
&+ (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i,j+1} \end{bmatrix}^k.
\end{aligned} \tag{4.12}$$

and at $j = M - 1$ using

$$\begin{aligned}
\bigcirc \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \end{bmatrix}^{k+1} &= \frac{\theta_1}{\rho_4^2 - q_4^2} \begin{bmatrix} \rho_4 & q_4 \\ q_4 & \rho_4 \end{bmatrix} \begin{bmatrix} S_1^{k+1} - S_1^k \\ 0 \end{bmatrix} \\
&+ \frac{\omega_1}{\rho_4^2 - q_4^2} \begin{bmatrix} \rho_4 & q_4 \\ q_4 & \rho_4 \end{bmatrix} \begin{bmatrix} S_1 + q_4 u_{i,j+1} + u_{i+1,j+1} \\ S_2 + q_4 u_{i+1,j+1} + u_{i,j+1} \end{bmatrix}^k \\
&+ (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \end{bmatrix}^k.
\end{aligned} \tag{4.13}$$

Finally, execute the remaining $\circ$ point at $i, j = M - 1$ using Equation (4.1)

4. Execute all groups of $\bullet$ points using

$$\bullet \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix}^{k+1} = \frac{\theta_2}{Q_4} A_4 \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \\ S_3^{k+1} - S_3^k \\ S_4^{k+1} - S_4^k \end{bmatrix} + \frac{\omega_2}{Q_4} A_4 \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}^k + (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix}^k. \quad (4.14)$$

5. Execute the remaining groups of two $\bullet$ points at $i = M - 1$ using

$$\bullet \begin{bmatrix} u_{i,j} \\ u_{i,j+1} \end{bmatrix}^{k+1} = \frac{\theta_2}{\rho_4^2 - q_4^2} \begin{bmatrix} \rho_4 & q_4 \\ q_4 & \rho_4 \end{bmatrix} \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_3^{k+1} - S_3^k \end{bmatrix} + \frac{\omega_2}{\rho_4^2 - q_4^2} \begin{bmatrix} \rho_4 & q_4 \\ q_4 & \rho_4 \end{bmatrix} \begin{bmatrix} S_1 + q_4 u_{i+1,j} + u_{i+1,j+1} \\ S_3 + q_4 u_{i+1,j+1} + u_{i+1,j} \end{bmatrix}^k + (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i,j+1} \end{bmatrix}^k. \quad (4.15)$$

6. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, go to step (5), otherwise, repeat the iteration cycle (i. e. go to step (2)).

7. Stop.

Sequential EDG9 MAOR

Figure 4.5 shows the solution domain for EDG9 with the horizontal zebra line (HZL) ordering strategy that is slightly modified. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

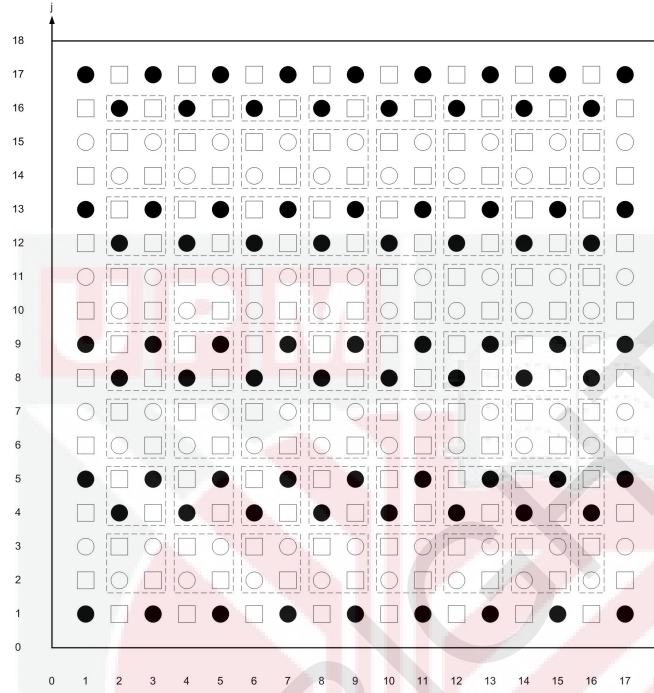


Figure 4.5: The solution domain for EDG9.

Let

$$A_5 = \begin{bmatrix} \rho_5 & q_5 \\ q_5 & \rho_5 \end{bmatrix},$$

and

$$\begin{aligned} S_1^k &= u_{i,j+2}^k + u_{i-2,j}^k + u_{i,j-2}^k + u_{i+2,j}^k \\ &\quad + q_5(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k) - h^2 F_{2i,j}, \\ S_2^k &= u_{i+1,j+3}^k + u_{i-1,j+1}^k + u_{i+1,j-1}^k + u_{i+3,j+1}^k \\ &\quad + q_5(u_{i,j+2}^k + u_{i+2,j}^k + u_{i+2,j+2}^k) - h^2 F_{2i+1,j+1}. \end{aligned}$$

Equation (2.15) can be generally represented in the following matrix form

$$D = \begin{bmatrix} A_5 & & & & \\ & A_5 & & & \\ & & \ddots & & \\ & & & A_5 & \\ & & & & A_5 \end{bmatrix}, \quad L = \begin{bmatrix} H_2 & 0 \\ H_1 & H_2 \end{bmatrix}, \quad U = L^T,$$

$$H_1 = \begin{bmatrix} J_1 & J_1^T & J_2^T & & \\ J_2 & J_1 & J_1^T & \ddots & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & J_1 & J_1^T & J_2^T \\ & & & J_2 & J_1 & J_1^T \end{bmatrix}, \quad b = \begin{bmatrix} G_1 \\ G_1 \\ \vdots \\ G_1 \\ G_1 \end{bmatrix},$$

$$H_2 = \begin{bmatrix} J_1 & & & & \\ & J_1 & & & \\ & & \ddots & & \\ & & & J_1 & \\ & & & & J_1 \end{bmatrix}, \quad J_1 = \begin{bmatrix} -1 & -q_5 \\ 0 & -1 \end{bmatrix}, \quad J_2 = \begin{bmatrix} 0 & -q_5 \\ 0 & 0 \end{bmatrix},$$

$$G_1 = h^2 \begin{bmatrix} F2_{i,j} \\ F2_{i+1,j+1} \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential EDG9 MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute the $\odot$ points at $i = 1$ using

$$\begin{aligned} \odot \quad u_{i,j}^{k+1} &= \frac{\omega_1}{q_5^*} \left(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k - \frac{h^2}{6} F2_{i,j} \right) \\ &\quad + (1 - \omega_1) u_{i,j}^k. \end{aligned} \quad (4.16)$$

3. Execute all groups of $\bigcirc$ points using

$$\begin{aligned} \bigcirc \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix}^{k+1} &= \frac{\theta_1}{Q_5} A_5 \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \end{bmatrix} + \frac{\omega_1}{Q_5} A_5 \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}^k \\ &+ (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix}^k. \end{aligned} \quad (4.17)$$

4. Execute the remaining $\bigcirc$ points at $i = M - 2$ using

$$\begin{aligned} \bigcirc \quad u_{i,j}^{k+1} &= \frac{\theta_1}{\rho_5} (v_{i-2,j} + q_5 v_{i-1,j+1}) \\ &+ \frac{\omega_1}{\rho_5} (u_{i-2,j}^k + u_{i+2,j}^k + u_{i,j-2}^k + u_{i,j+2}^k \\ &+ q_5 (u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k) - h^2 F_{2i,j}^2) \\ &+ (1 - \omega_1) u_{i,j}^k, \end{aligned} \quad (4.18)$$

and at $i = M - 1$ using

$$\begin{aligned} \bigcirc \quad u_{i,j}^{k+1} &= \frac{\theta_1}{q_5^*} (v_{i-1,j-1}) \\ &+ \frac{\omega_1}{q_5^*} \left(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k - \frac{h^2}{6} F_{2i,j}^2 \right) \\ &+ (1 - \omega_1) u_{i,j}^k. \end{aligned} \quad (4.19)$$

5. Execute the $\bullet$ points at $j = 1$ using

$$\begin{aligned} \bullet \quad u_{i,j}^{k+1} &= \frac{\theta_2}{q_5^*} (v_{i+1,j+1} + v_{i-1,j+1}) \\ &+ \frac{\omega_2}{q_5^*} \left(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k - \frac{h^2}{6} F_{2i,j}^2 \right) \\ &+ (1 - \omega_2) u_{i,j}^k, \end{aligned} \quad (4.20)$$

and at $i = 1$ using

$$\begin{aligned}
 \bullet \quad u_{i,j}^{k+1} &= \frac{\theta_2}{q_5^*} (v_{i+1,j+1}) \\
 &+ \frac{\omega_2}{q_5^*} \left(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k - \frac{h^2}{6} F_{2i,j} \right) \\
 &+ (1 - \omega_2) u_{i,j}^k.
 \end{aligned} \tag{4.21}$$

6. Execute all groups of $\bullet$ points using

$$\begin{aligned}
 \bullet \quad \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix}^{k+1} &= \frac{\theta_2}{Q_5} A_5 \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \end{bmatrix} + \frac{\omega_2}{Q_5} A_5 \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}^k \\
 &+ (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix}^k.
 \end{aligned} \tag{4.22}$$

7. Execute the remaining $\bullet$ points at $i = M - 2$ using

$$\begin{aligned}
 \bullet \quad u_{i,j}^{k+1} &= \frac{\theta_2}{\rho_5} (v_{i-2,j} + v_{i,j-2} + v_{i,j+2} \\
 &+ q_5(v_{i-1,j-1} + v_{i+1,j-1} + v_{i-1,j+1})) \\
 &+ \frac{\omega_2}{\rho_5} \left(u_{i-2,j}^k + u_{i+2,j}^k + u_{i,j-2}^k + u_{i,j+2}^k \right. \\
 &\left. + q_5(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k) - h^2 F_{2i,j} \right) \\
 &+ (1 - \omega_2) u_{i,j}^k,
 \end{aligned} \tag{4.23}$$

and at $j = M - 2$ using

- $$\begin{aligned}
u_{i,j}^{k+1} = & \frac{\theta_2}{\rho_5} (v_{i-2,j} + v_{i,j-2} \\
& + q_5(v_{i-1,j-1} + v_{i+1,j-1})) \\
& + \frac{\omega_2}{\rho_5} (u_{i-2,j}^k + u_{i+2,j}^k + u_{i,j-2}^k + u_{i,j+2}^k \\
& + q_5(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k) - h^2 F_{2i,j}) \\
& + (1 - \omega_2)u_{i,j}^k.
\end{aligned}
\tag{4.24}$$

8. Execute the remaining $\bullet$ points at $i = M - 1$ using

- $$\begin{aligned}
u_{i,j}^{k+1} = & \frac{\theta_2}{q_5^*} (v_{i-1,j-1} + v_{i-1,j+1}) \\
& + \frac{\omega_2}{q_5^*} \left(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k - \frac{h^2}{6} F_{2i,j} \right) \\
& + (1 - \omega_2)u_{i,j}^k,
\end{aligned}
\tag{4.25}$$

and at $j = M - 1$ using

- $$\begin{aligned}
u_{i,j}^{k+1} = & \frac{\theta_2}{q_5^*} (v_{i-1,j-1} + v_{i+1,j-1}) \\
& + \frac{\omega_2}{q_5^*} \left(u_{i-1,j-1}^k + u_{i+1,j-1}^k + u_{i-1,j+1}^k + u_{i+1,j+1}^k - \frac{h^2}{6} F_{2i,j} \right) \\
& + (1 - \omega_2)u_{i,j}^k.
\end{aligned}
\tag{4.26}$$

9. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, evaluate the solutions of the remaining $\square$ points using Equation (3.5). Otherwise, repeat the iteration cycle (i. e. go to step (2)).

10. Stop.

Sequential MEG9 MAOR

Figure 4.6 shows the solution domain for MEG9 with the horizontal zebra line (HZL) ordering strategy.

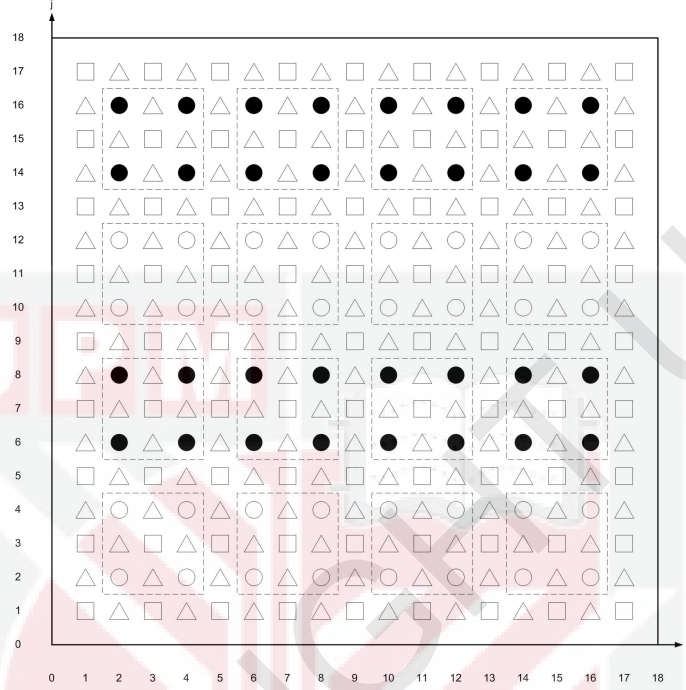


Figure 4.6: The solution domain for MEG9.

Let

$$A_6 = \begin{bmatrix} R_1 & R_2 & R_2 & R_3 \\ R_2 & R_1 & R_3 & R_2 \\ R_2 & R_3 & R_1 & R_2 \\ R_3 & R_2 & R_2 & R_1 \end{bmatrix},$$

and

$$\begin{aligned} S_1^k &= u_{i-2,j-2}^k + u_{i+2,j-2}^k + u_{i-2,j+2}^k + q_6(u_{i-2,j}^k + u_{i,j-2}^k) - 2h^2 F_3 u_{i,j}, \\ S_2^k &= u_{i,j-2}^k + u_{i+4,j+2}^k + u_{i+4,j-2}^k + q_6(u_{i+4,j}^k + u_{i+2,j-2}^k) - 2h^2 F_3 u_{i+2,j}, \\ S_3^k &= u_{i-2,j}^k + u_{i-2,j+4}^k + u_{i+2,j+4}^k + q_6(u_{i-2,j+2}^k + u_{i,j+4}^k) - 2h^2 F_3 u_{i,j+2}, \\ S_4^k &= u_{i,j+4}^k + u_{i+4,j+4}^k + u_{i+4,j}^k + q_6(u_{i+4,j+2}^k + u_{i+2,j+4}^k) - 2h^2 F_3 u_{i+2,j+2}. \end{aligned}$$

Equation (2.7) can be generally represented in the following matrix form

$$D = \begin{bmatrix} A_6 & & & & \\ & A_6 & & & \\ & & \ddots & & \\ & & & A_6 & \\ & & & & A_6 \end{bmatrix}, \quad L = \begin{bmatrix} H_2 & 0 \\ H_1 & H_2 \end{bmatrix}, \quad U = L^T,$$

$$H_1 = \begin{bmatrix} J_1 & J_3 & & J_1^T & J_2^T & & \\ J_2 & J_1 & J_3 & J_3^T & \ddots & \ddots & \\ & J_2 & J_1 & J_3 & & \ddots & J_2^T \\ & & J_2 & J_1 & \ddots & & J_3^T & J_1^T \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \ddots & J_1 & J_3 & \\ & & & & & J_2 & J_1 & J_3 \\ & & & & & & J_2 & J_1 & J_3 \\ & & & & & & & J_2 & J_1 \end{bmatrix}, \quad b = \begin{bmatrix} G_1 \\ G_1 \\ \vdots \\ G_1 \\ G_1 \end{bmatrix},$$

$$H_2 = \begin{bmatrix} J_4 & & & & \\ & J_4 & & & \\ & & \ddots & & \\ & & & J_4 & \\ & & & & J_4 \end{bmatrix}, \quad G_1 = 2h^2 \begin{bmatrix} F3_{i,j} \\ F3_{i+2,j} \\ F3_{i,j+2} \\ F3_{i+2,j+2} \end{bmatrix},$$

$$J_1 = \begin{bmatrix} 0 & 0 & -q_6 & -1 \\ 0 & 0 & -1 & -q_6 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad J_2 = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$J_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad J_4 = \begin{bmatrix} 0 & -q_6 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -q_6 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential MEG9 MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all groups of $\bigcirc$ points using

$$\bigcirc \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix}^{k+1} = \frac{\theta_1}{Q_6} A_6 \begin{bmatrix} S_1^{k+1} - S_1^k \\ 0 \\ S_3^{k+1} - S_3^k \\ 0 \end{bmatrix} + \frac{\omega_1}{Q_6} A_6 \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}^k + (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix}^k. \quad (4.27)$$

3. Execute all groups of $\bullet$ points using

$$\begin{aligned}
\bullet \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix}^{k+1} &= \frac{\theta_2}{Q_6} A_6 \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \\ S_3^{k+1} - S_3^k \\ S_4^{k+1} - S_4^k \end{bmatrix} + \frac{\omega_2}{Q_6} A_6 \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}^k \\
&+ (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix}^k.
\end{aligned} \tag{4.28}$$

4. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, evaluate the solutions of the remaining $\square$ points using Equation (3.8) followed by the remaining $\triangle$ points using Equation (3.9). Otherwise, repeat the iteration cycle (i. e. go to step (2)).
5. Stop.

4.3 Construction of Fourth Order Parallel Algorithm

From Equations (4.1), (4.9), (4.11), (4.17) and (4.27), it is clear that all the $\circ$ tasks in $u_{i,j}^{k+1}$ are independent of each other and can be computed in parallel. After that, all the $\bullet$ tasks of Equations (4.2), (4.10), (4.14), (4.22) and (4.28) can be calculated simultaneously using the updated $\circ$ values since these calculations are also independent. Similarly, for Equation (4.3), all the $\circ$ tasks in $u_{i,j}^{k+1}$ are independent of each other and can be computed in parallel. This is then followed with the $\bullet$ tasks of Equation (4.5) which can be calculated simultaneously using the updated $\circ$ values since these calculations are also independent. Finally, all the remaining $\bullet$ tasks are calculated using the updated $\circ$ and $\bullet$ values. At the end of each stage, a synchronization call is executed to ensure the updated values are used in the subsequent iteration. Each processor independently iterates on its own task and then the convergence test is performed. If the

convergence criteria is achieved, the HS9, QS9, EDG9 and MEG9 iterative methods will evaluate the remaining tasks directly. Otherwise increase the number of iteration and repeat the iteration cycle for all iterative methods.

Similar to the previous chapter, four number of processors are chosen as an example. The parallel algorithms for solving the fourth order MAOR iterative methods will be constructed using multicore processors using the OpenMP technology and C++ Compiler.

4.3.1 Parallel Fourth Order Point Iterative Methods with MAOR Scheme

Parallel FS9 MAOR

Step 1: Solve the $\circ$ points. From Figure 4.1, assign each processor to a row of $\circ$ points ($P0 \rightarrow j = 1, P1 \rightarrow j = 3, P2 \rightarrow j = 5, P3 \rightarrow j = 7$). All processor will solve the $\circ$ points using Equation (4.1). After the processors have finish solving their respective rows, they will be assigned to the next row of $\circ$ points ($P0 \rightarrow j = 9, P1 \rightarrow j = 11, P2 \rightarrow j = 13, P3 \rightarrow j = 15$). This process will continue until there are no available row of $\circ$ points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 2: Solve the $\bullet$ points. From Figure 4.1, assign each processor to a row of $\bullet$ points ($P0 \rightarrow j = 2, P1 \rightarrow j = 4, P2 \rightarrow j = 6, P3 \rightarrow j = 8$). All processor will solve the $\bullet$ points using Equation (4.2). After the processors have finish solving the respective rows, they will be assigned to the next row of block ($P0 \rightarrow j = 10, P1 \rightarrow j = 12, P2 \rightarrow j = 14, P3 \rightarrow j = 16$). This process will continue until there are no available row of $\bullet$ points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 3: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

These steps above can be summarized in Algorithm 7.

Algorithm 7: The iterative process for the parallel FS9 MAOR iterative method.

```

1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into  $\bigcirc$  and  $\bullet$  as shown in Figure 4.1;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
4   Get thread number  $p$ , and maximum number of threads  $np$ ;
5   while  $|u^{k+1} - u^k| > \varepsilon$  do
6     #pragma omp parallel
7     {
8       for  $\{j = 2(p+1) - 1; j < M; j = j + 2np\}$  do
9         Compute all  $\bigcirc$  points using Equation (4.1);
10      end for
11      #pragma omp barrier
12      for  $\{j = 2(p+1); j < M; j = j + 2np\}$  do
13        Compute all  $\bullet$  points using Equation (4.2);
14      end for
15      #pragma omp barrier
16       $ite = ite + 1$ ;
17    }
18  end while
19  Display  $ite$ , error, and execution time;

```

Parallel HS9 MAOR

Step 1: Solve the $\bigcirc$ points. From Figure 4.2, assign each processor to a row of $\bigcirc$ points ($P0 \rightarrow j = 1, P1 \rightarrow j = 4, P2 \rightarrow j = 7, P3 \rightarrow j = 10$). All processor will solve the $\bigcirc$ points using either Equations (4.3) or (4.4). After the processors have finish solving their respective rows, they will be assigned to the next row of $\bigcirc$ points ($P0 \rightarrow j = 13, P1 \rightarrow j = 16, P2 \rightarrow j = 19, P3 \rightarrow j = 22$). This process will continue until there are no available row of $\bigcirc$ points left to be solved. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 2: Solve the $\bullet$ points. From Figure 4.2, assign each processor to a row of $\bullet$ points ($P0 \rightarrow j = 2, P1 \rightarrow j = 5, P2 \rightarrow j = 8, P3 \rightarrow j = 11$). All processor will solve the $\bullet$ points using either Equations (4.5) or (4.6). After the processors have

finish solving the respective rows, they will be assigned to the next row of block ($P0 \rightarrow j = 14$, $P1 \rightarrow j = 17$, $P2 \rightarrow j = 20$, $P3 \rightarrow j = 23$). This process will continue until there are no available row of $\bullet$ points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 3: Solve the $\bullet$ points. From Figure 4.2, assign each processor to a row of $\bullet$ points ($P0 \rightarrow j = 3$, $P1 \rightarrow j = 6$, $P2 \rightarrow j = 9$, $P3 \rightarrow j = 12$). All processor will solve the $\bullet$ points using either Equations (4.7) or (4.8). After the processors have finish solving the respective rows, they will be assigned to the next row of block ($P0 \rightarrow j = 15$, $P1 \rightarrow j = 18$, $P2 \rightarrow j = 21$, $P3 \rightarrow j = 24$). This process will continue until there are no available row of $\bullet$ points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 4: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

Step 5: Evaluate the solutions of the remaining $\square$ points using Equation (3.5) directly.

These steps above can be summarized in Algorithm 8.

Parallel QS9 MAOR

Step 1: Solve the $\circ$ points. From Figure 4.3, assign each processor to a row of $\circ$ points ($P0 \rightarrow j = 2$, $P1 \rightarrow j = 6$, $P2 \rightarrow j = 10$, $P3 \rightarrow j = 14$). All processor will solve the $\circ$ points using Equation (4.9). After the processors have finish solving their respective rows, they will be assigned to the next row of $\circ$ points ($P0 \rightarrow j = 18$, $P1 \rightarrow j = 22$, $P2 \rightarrow j = 26$, $P3 \rightarrow j = 30$). This process will continue until there are no available row of $\circ$ points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Algorithm 8: The iterative process for the parallel HS9 MAOR iterative method.

```
1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into  $\circ$ ,  $\odot$  and  $\bullet$  as shown in Figure 4.2;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \omega_3, \theta_1, \theta_2$  and  $\theta_3$ ,
4   and error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
5   Get thread number  $p$ , and maximum number of threads  $np$ ;
6   while  $|u^{k+1} - u^k| > \varepsilon$  do
7     #pragma omp parallel
8     {
9       for  $\{j = 3(p + 1) - 2; j < M; j = j + 3np\}$  do
10        Compute all  $\circ$  points using either Equations (4.3) or (4.4);
11      end for
12      #pragma omp barrier
13      for  $\{j = 3(p + 1) - 1; j < M; j = j + 3np\}$  do
14        Compute all  $\odot$  points using either Equations (4.5) or (4.6);
15      end for
16      #pragma omp barrier
17      for  $\{j = 3(p + 1); j < M; j = j + 3np\}$  do
18        Compute all  $\bullet$  points using either Equations (4.7) or (4.8);
19      end for
20      #pragma omp barrier
21       $ite = ite + 1$ ;
22    }
23  end while
24  Compute all  $\square$  points using Equation (3.5) directly;
25  Display  $ite$ , error, and execution time;
```

Step 2: Solve the $\bullet$ points. From Figure 4.3, assign each processor to a row of $\bullet$ points ($P0 \rightarrow j = 4, P1 \rightarrow j = 8, P2 \rightarrow j = 12, P3 \rightarrow j = 16$). All processor will solve the $\bullet$ points using Equation (4.10). After the processors have finish solving the respective rows, they will be assigned to the next row of block ($P0 \rightarrow j = 20, P1 \rightarrow j = 24, P2 \rightarrow j = 28, P3 \rightarrow j = 32$). This process will continue until there are no available row of $\bullet$ points left to be solved. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 3: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

Step 4: Evaluate the solutions of the remaining $\square$ points using Equation (3.8) and

$\triangle$ points using Equation (3.9) directly.

These steps above can be summarized in Algorithm 9.

Algorithm 9: The iterative process for the parallel QS9 MAOR iterative method.

```

1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into  $\circ$  and  $\bullet$  as shown in Figure 4.3;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
4   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
5   Get thread number  $p$ , and maximum number of threads  $np$ ;
6   while  $|u^{k+1} - u^k| > \varepsilon$  do
7     #pragma omp parallel
8     {
9       for  $\{j = 4(p + 1) - 2; j < M; j = j + 4np\}$  do
10        Compute all  $\circ$  points using Equation (4.9);
11      end for
12      #pragma omp barrier
13      for  $\{j = 4(p + 1); j < M; j = j + 4np\}$  do
14        Compute all  $\bullet$  points using Equation (4.10);
15      end for
16      #pragma omp barrier
17       $ite = ite + 1$ ;
18    }
19  end while
20  Compute all  $\square$  points using Equation (3.8) directly;
21  Compute all  $\triangle$  points using Equation (3.9) directly;
22  Display  $ite$ , error, and execution time;

```

4.3.2 Parallel Fourth Order Group Iterative Methods with MAOR Scheme

Parallel EG9 MAOR

Step 1: Solve the group of four $\circ$ points. From Figure 4.4, assign each processor to a row of block $\circ$ points ($P0 \rightarrow j = 1, P1 \rightarrow j = 5, P2 \rightarrow j = 9, P3 \rightarrow j = 13$). All processor will solve the block of four $\circ$ points using Equation (4.11). After the processors have finish solving their respective rows, they will be assigned to the next row of block $\circ$ points ($P0 \rightarrow j = 17, P1 \rightarrow j = 21, P2 \rightarrow j = 25, P3 \rightarrow j = 29$). This process will continue until there are no available row of block left to be solved. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 2: Solve the remaining group of $\bigcirc$ points at $i = M - 1$ and $j = M - 1$. For $i = M - 1$, each processor will be assigned to a block of $\bigcirc$ points consecutively ($P0 \rightarrow j = 1, P1 \rightarrow j = 5, P2 \rightarrow j = 9, P3 \rightarrow j = 13$). All processor will solve their respective block of $\bigcirc$ points using Equation (4.12). After the processors have finish solving their respective blocks, they will be assigned to the next block of $\bigcirc$ points consecutively ($P0 \rightarrow j = 17, P1 \rightarrow j = 21, P2 \rightarrow j = 25, P3 \rightarrow j = 29$). This process will continue until there are no available row of block left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

However, for $j = M - 1$, it can be seen that the block of points are not independent from each other. Thus, only the master processor (i. e. $P0$) will solve the block of $\bigcirc$ points using Equation (4.13). Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Finally, the remaining $\bigcirc$ point at $i, j = M - 1$ is solved using Equation (4.1). Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 3: Solve the group of four $\bullet$ points. From Figure 4.4, assign each processor to a row of block $\bullet$ points ($P0 \rightarrow j = 3, P1 \rightarrow j = 7, P2 \rightarrow j = 11, P3 \rightarrow j = 15$). All processor will solve the block of four $\bullet$ points using Equation (4.14). After the processors have finish solving their respective rows, they will be assigned to the next row of block $\bullet$ points ($P0 \rightarrow j = 19, P1 \rightarrow j = 23, P2 \rightarrow j = 27, P3 \rightarrow j = 31$). This process will continue until there are no available row of block left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 4: Solve the remaining group of $\bullet$ points at $i = M - 1$. For $i = M - 1$, each processor will be assigned to a block of $\bullet$ points consecutively ($P0 \rightarrow j = 3, P1 \rightarrow j = 7, P2 \rightarrow j = 11, P3 \rightarrow j = 15$). All processor will solve their respective block of $\bullet$ points using Equation (4.15). After the processors have finish solving their

respective blocks, they will be assigned to the next block of ● points consecutively ($P_0 \rightarrow j = 19$, $P_1 \rightarrow j = 23$, $P_2 \rightarrow j = 27$, $P_3 \rightarrow j = 31$). This process will continue until there are no available row of block left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 5: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

These steps above can be summarized in Algorithm 10.

Algorithm 10: The iterative process for the parallel EG9 MAOR iterative method.

```

1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into ○ and ● as shown in Figure 4.4;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
4   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
5   Get thread number  $p$ , and maximum number of threads  $np$ ;
6   while  $|u^{k+1} - u^k| > \varepsilon$  do
7     #pragma omp parallel
8     {
9       for  $\{j = 4(p + 1) - 3; j < M - 1; j = j + 4np\}$  do
10        Compute all groups of ○ points using Equation (4.11);
11      end for
12      #pragma omp barrier
13       $i = M - 1$ ;
14      for  $\{j = 4(p + 1) - 3; j < M - 1; j = j + 4np\}$  do
15        Compute all groups of ○ points using Equation (4.12);
16      end for
17      #pragma omp barrier
18       $j = M - 1$ ;
19      for  $\{i = 1; i < M - 1; i = i + 2\}$  do
20        Compute all groups of ○ points using Equation (4.13);
21      end for
22      #pragma omp barrier
23       $i = M - 1, j = M - 1$ 
24      Compute ○ point using Equation (4.1);
25      #pragma omp barrier

```

```

26
27   for { $j = 4(p + 1) - 1; j < M - 1; j = j + 4np$ } do
28       Compute all groups of ● points using Equation (4.14);
29   end for
30   #pragma omp barrier
31    $i = M - 1;$ 
32   for { $j = 4(p + 1) - 1; j < M - 1; j = j + 4np$ } do
33       Compute all groups of ● points using Equation (4.15);
34   end for
35   #pragma omp barrier
36    $ite = ite + 1;$ 
37   }
38 end while
39 Display  $ite$ , error, and execution time;

```

Parallel EDG9 MAOR

Step 1: Solve the ○ points at $i = 1$. From Figure 4.5, each processor will be assigned to a ○ points consecutively ($P_0 \rightarrow j = 3, P_1 \rightarrow j = 7, P_2 \rightarrow j = 11, P_3 \rightarrow j = 15$). All processor will solve their respective block of ○ points using Equation (4.16). After the processors have finish solving their respective points, they will be assigned to the next ○ points consecutively ($P_0 \rightarrow j = 19, P_1 \rightarrow j = 23, P_2 \rightarrow j = 27, P_3 \rightarrow j = 31$). This process will continue until there are no available point left to be solved. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 2: Solve the group of two ○ points. From Figure 4.5, assign each processor to a row of block ○ points ($P_0 \rightarrow j = 2, P_1 \rightarrow j = 6, P_2 \rightarrow j = 10, P_3 \rightarrow j = 14$). All processor will solve the block of four ○ points using Equation (4.17). After the processors have finish solving their respective rows, they will be assigned to the next row of block ○ points ($P_0 \rightarrow j = 18, P_1 \rightarrow j = 22, P_2 \rightarrow j = 26, P_3 \rightarrow j = 30$). This process will continue until there are no available row of block left to be solved. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 3: Solve the remaining ○ points at $i = M - 2$ and $i = M - 1$. For $i = M - 2$,

each processor will be assigned to a $\bigcirc$ points consecutively ($P_0 \rightarrow j = 2, P_1 \rightarrow j = 6, P_2 \rightarrow j = 10, P_3 \rightarrow j = 14$). All processor will solve their respective $\bigcirc$ points using Equation (4.18). After the processors have finish solving their respective points, they will be assigned to the next $\bigcirc$ points consecutively ($P_0 \rightarrow j = 18, P_1 \rightarrow j = 22, P_2 \rightarrow j = 26, P_3 \rightarrow j = 30$). This process will continue until there are no available point left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

For $i = M - 1$, each processor will be assigned to a $\bigcirc$ points consecutively ($P_0 \rightarrow j = 3, P_1 \rightarrow j = 7, P_2 \rightarrow j = 11, P_3 \rightarrow j = 15$). All processor will solve their respective $\bigcirc$ points using Equation (4.19). After the processors have finish solving their respective points, they will be assigned to the next $\bigcirc$ points consecutively ($P_0 \rightarrow j = 19, P_1 \rightarrow j = 23, P_2 \rightarrow j = 27, P_3 \rightarrow j = 31$). This process will continue until there are no available point left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 4: Solve the $\bullet$ points at $j = 1$ and $i = 1$. For $j = 1$, each processor will be assigned to a $\bullet$ points consecutively ($P_0 \rightarrow i = 1, P_1 \rightarrow i = 3, P_2 \rightarrow i = 5, P_3 \rightarrow i = 7$). All processor will solve their respective block of $\bullet$ points using Equation (4.20). After the processors have finish solving their respective points, they will be assigned to the next $\bullet$ points consecutively ($P_0 \rightarrow i = 9, P_1 \rightarrow i = 11, P_2 \rightarrow i = 13, P_3 \rightarrow i = 15$). This process will continue until there are no available point left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

For $i = 1$, each processor will be assign to a $\bullet$ point consecutively ($P_0 \rightarrow j = 5, P_1 \rightarrow j = 9, P_2 \rightarrow j = 13, P_3 \rightarrow j = 17$). All processor will solve their respective block of $\bullet$ points using Equation (4.21). After the processors have finish solving their respective points, they will be assigned to the next $\bullet$ points consecutively (P_0

$\rightarrow j = 21, P1 \rightarrow j = 25, P2 \rightarrow j = 29, P3 \rightarrow j = 33$). This process will continue until there are no available point left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 5: Solve the group of two ● points. From Figure 4.5, assign each processor to a row of block ● points ($P0 \rightarrow j = 4, P1 \rightarrow j = 8, P2 \rightarrow j = 12, P3 \rightarrow j = 16$). All processor will solve the block of four ● points using Equation (4.22). After the processors have finish solving their respective rows, they will be assigned to the next row of block ● points ($P0 \rightarrow j = 20, P1 \rightarrow j = 24, P2 \rightarrow j = 28, P3 \rightarrow j = 32$). This process will continue until there are no available row of block left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 6: Solve the remaining ● points at $i = M - 2$ and $j = M - 2$. For $i = M - 2$, each processor will be assigned to a ● points consecutively ($P0 \rightarrow j = 4, P1 \rightarrow j = 8, P2 \rightarrow j = 12, P3 \rightarrow j = 16$). All processor will solve their respective ● points using Equation (4.23). After the processors have finish solving their respective points, they will be assigned to the next ● points consecutively ($P0 \rightarrow j = 20, P1 \rightarrow j = 24, P2 \rightarrow j = 28, P3 \rightarrow j = 32$). This process will continue until there are no available point left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

For $j = M - 2$, only the main processor is utilized to solved these points. The ● points are solved using Equation (4.24). Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 7: Solve the remaining ● points at $i = M - 1$ and $j = M - 1$. For $i = M - 1$, each processor will be assigned to a ● points consecutively ($P0 \rightarrow j = 5, P1 \rightarrow j = 9, P2 \rightarrow j = 13, P3 \rightarrow j = 17$). All processor will solve their respective ● points using Equation (4.25). After the processors have finish solving their respective points, they

will be assigned to the next ● points consecutively ($P0 \rightarrow j = 21$, $P1 \rightarrow j = 25$, $P2 \rightarrow j = 29$, $P3 \rightarrow j = 33$). This process will continue until there are no available point left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

For $j = M - 1$, each processor will be assigned to a ● points consecutively ($P0 \rightarrow i = 1$, $P1 \rightarrow i = 3$, $P2 \rightarrow i = 5$, $P3 \rightarrow i = 7$). All processor will solve their respective ● points using Equation (4.26). After the processors have finish solving their respective points, they will be assigned to the next ● points consecutively ($P0 \rightarrow j = 9$, $P1 \rightarrow j = 11$, $P2 \rightarrow j = 13$, $P3 \rightarrow j = 15$). This process will continue until there are no available point left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 8: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

Step 9: Evaluate the solutions of the remaining □ points using Equation (3.5) directly.

These steps above can be summarized in Algorithm 11.

Algorithm 11: The iterative process for the parallel EDG9 MAOR iterative method.

```
1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into  $\bigcirc$  and  $\bullet$  as shown in Figure 4.5;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
4   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
5   Get thread number  $p$ , and maximum number of threads  $np$ ;
6   while  $|u^{k+1} - u^k| > \varepsilon$  do
7     #pragma omp parallel
8     {
9        $i = 1$ ;
10      for  $\{j = 4(p+1) - 1; j < M - 1; j = j + 4np\}$  do
11        Compute all  $\bigcirc$  points using Equation (4.16);
12      end for
13      #pragma omp barrier
14      for  $\{j = 4(p+1) - 2; j < M; j = j + 4np\}$  do
15        Compute all groups of  $\bigcirc$  points using Equation (4.17);
16      end for
17      #pragma omp barrier
18       $i = M - 2$ ;
19      for  $\{j = 4(p+1) - 2; j < M - 2; j = j + 4np\}$  do
20        Compute all  $\bigcirc$  points using Equation (4.18);
21      end for
22      #pragma omp barrier
23       $i = M - 1$ ;
24      for  $\{j = 4(p+1) - 1; j < M - 1; j = j + 4np\}$  do
25        Compute all  $\bigcirc$  points using Equation (4.19);
26      end for
27      #pragma omp barrier
28       $j = 1$ ;
29      for  $\{i = 2(p+1) - 1; i < M - 1; i = i + 2np\}$  do
30        Compute all  $\bullet$  points using Equation (4.20);
31      end for
32      #pragma omp barrier
33       $i = 1$ ;
34      for  $\{j = 4(p+1) + 1; j < M; j = j + 4np\}$  do
35        Compute all  $\bullet$  points using Equation (4.21);
36      end for
37      #pragma omp barrier
38      for  $\{j = 4(p+1); j < M; j = j + 4np\}$  do
39        Compute all groups of  $\bullet$  points using Equation (4.22);
40      end for
41      #pragma omp barrier
```

```

42
43      $i = M - 2;$ 
44     for  $\{j = 4(p + 1); j < M - 2; j = j + 4np\}$  do
45         Compute all ● points using Equation (4.23);
46     end for
47     #pragma omp barrier
48      $j = M - 2;$ 
49     for  $\{i = 2; i < M - 1; i = i + 2\}$  do
50         Compute all ● points using Equation (4.24);
51     end for
52     #pragma omp barrier
53      $i = M - 1;$ 
54     for  $\{j = 4(p + 1) + 1; j < M; j = j + 4np\}$  do
55         Compute all ● points using Equation (4.25);
56     end for
57     #pragma omp barrier
58      $j = M - 1;$ 
59     for  $\{i = 2(p + 1) - 1; i < M; i = i + 2np\}$  do
60         Compute all ● points using Equation (4.26);
61     end for
62     #pragma omp barrier
63      $ite = ite + 1;$ 
64     }
65 end while
66 Compute all □ points using Equation (3.5) directly;
67 Display  $ite$ , error, and execution time;

```

Parallel MEG9 MAOR

Step 1: Solve the block of four ○ points. From Figure 4.6, assign each processor to a row of block ○ points ($P0 \rightarrow j = 2, P1 \rightarrow j = 10, P2 \rightarrow j = 18, P3 \rightarrow j = 26$). All processor will solve the block of four ○ points using Equation (4.27). After the processors have finish solving their respective rows, they will be assigned to the next row of block ○ points ($P0 \rightarrow j = 34, P1 \rightarrow j = 42, P2 \rightarrow j = 50, P3 \rightarrow j = 58$). This process will continue until there are no available row of block left to be solved. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 2: Solve the block of four ● points. From Figure 4.6, assign each processor to a row of block ● points ($P0 \rightarrow j = 6, P1 \rightarrow j = 14, P2 \rightarrow j = 22, P3 \rightarrow j = 30$). All processor will solve the block of four ● points using Equation (4.28). After the

processors have finish solving their respective rows, they will be assigned to the next row of block $\bullet$ points ($P0 \rightarrow j = 38, P1 \rightarrow j = 46, P2 \rightarrow j = 54, P3 \rightarrow j = 62$). This process will continue until there are no available row of block left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 3: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

Step 4: Evaluate the solutions of the remaining $\square$ points using Equation (3.8) and $\triangle$ points using Equation (3.9) directly.

These steps above can be summarized in Algorithm 12.

Algorithm 12: The iterative process for the parallel MEG9 MAOR iterative method.

```

1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into  $\circ$  and  $\bullet$  as shown in Figure 4.6;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
4   Get thread number  $p$ , and maximum number of threads  $np$ ;
5   while  $|u^{k+1} - u^k| > \varepsilon$  do
6     #pragma omp parallel
7     {
8       for  $\{j = 8(p+1) - 6; j < M; j = j + 8np\}$  do
9         Compute all groups of  $\circ$  points using Equation (4.27);
10      end for
11      #pragma omp barrier
12      for  $\{j = 8(p+1) - 2; j < M; j = j + 8np\}$  do
13        Compute all groups of  $\bullet$  points using Equation (4.28);
14      end for
15      #pragma omp barrier
16       $ite = ite + 1$ ;
17    }
18  end while
19  Compute all  $\square$  points using Equation (3.8) directly;
20  Compute all  $\triangle$  points using Equation (3.9) directly;
21  Display  $ite$ , error, and execution time;

```

4.4 Experimental Results and Discussions

Two problems, similar to the problems in Chapter 3, will be tested in order to investigate the competency of the proposed method. The tests were performed on various sizes of M while the number of processors ranging from 1 to 8. The performance of the second order MAOR iterative methods in solving the problems are observed in terms of number of iterations, accuracy, execution time, speedup, parallel efficiency and computational costs. The computational cost for each iterative method with the relaxation schemes is measured by computing the total of arithmetic operations involved which is presented in Tables 4.1 and 4.2.

Table 4.3 presents the number of iteration and accuracy of all iterative methods with various relaxation schemes for Problem 1. From the table, the MAOR scheme for all iterative methods reduces the number of iterations by 89.84 – 99.40% when compared with GS scheme, 6.67 – 50.74% when compared with SOR scheme, 6.67 – 41.71% when compared with MSOR scheme, and 0.00 – 50.22% when compared with AOR scheme. Similarly, Table 4.6 presents the number of iteration and accuracy of all iterative methods with various relaxation schemes for Problem 2. From the table, the MAOR scheme for all iterative methods also reduces the number of iterations by 91.94 – 99.51% when compared with GS scheme, 10.71 – 62.54% when compared with SOR scheme, 7.41 – 57.62% when compared with MSOR scheme, and 2.65 – 36.16% when compared with AOR scheme. The MAOR scheme incorporates additional relaxation parameters designed to minimize the spectral radius of the iteration matrix. This reduction in the spectral radius leads to faster convergence by dampening error components more effectively, thus requiring fewer iterations to reach a solution.

In terms of accuracy, while the MAOR scheme is not the most accurate solver, it maintains competitive error levels when compared to other relaxation schemes. The reason lies in its prioritisation of minimising the iteration count through the optimisation of relaxation parameters. However, this optimisation does not necessarily guarantee the

smallest possible residual error. As a result, although the MAOR scheme converges quickly, the final error can plateau at a higher level than some other schemes that may achieve greater accuracy at the expense of more iterations.

The execution time, speedup and parallel efficiency are shown in Tables 4.4 and 4.5 for Problem 1 and Tables 4.8 and 4.9 for Problem 2. From these tables, we can see that the MAOR scheme does not achieve the fastest execution times compared to some other relaxation schemes but remains competitive. The slower convergence of the MAOR scheme can be attributed to its design, which focuses on minimizing the spectral radius of the iteration matrix. This approach effectively reduces the number of iterations required for convergence. However, it prioritises iteration reduction over minimising residual errors, which can result in slower overall convergence in certain cases. Additionally, the MAOR scheme introduces extra acceleration parameters to enhance convergence, but these parameters can complicate the tuning process. If not optimally chosen, the parameters may lead to less effective convergence. Furthermore, the inclusion of additional computations associated with these parameters increases the per iteration computational cost. This increased cost can offset the benefits of a reduced iteration count, making the MAOR scheme slower than other relaxation methods in terms of total execution time. When parallel algorithms are implemented, the execution time generally decreases as the number of processors increases, and the speedup tends to increase exponentially. This shows that as more processors are added, the workload can be divided more efficiently, leading to reduced execution time. With an increase in the number of processors, the parallel efficiency of iterative methods improves. This is because the computational load is distributed across multiple processors, which decreases the overall time required for each iteration and increases the efficiency of parallel computation. Figures 4.7 and 4.9 show the execution time while figures 4.8 and 4.10 show the speedup of the fourth order iterative methods on all relaxation schemes for Problems 1 and 2 at $M = 514$.

The computational costs for all fourth order iterative methods with various relaxation schemes for both problems are shown in Table 4.7. From this table, the MAOR scheme has a 0.00 – 50.22% reduction in computational costs when compared to the AOR scheme. However, the MAOR scheme has a 17.53 – 31.24% increase in computational costs when compared to the MSOR scheme. The higher computational cost of the MAOR scheme is attributed to the introduction of additional acceleration parameters designed to minimise the spectral radius of the iteration matrix. Although this reduces the number of iterations required for convergence, it increases the computational workload per iteration. Compared to the simpler mechanisms in SOR and MSOR schemes, the MAOR scheme requires more arithmetic operations per iteration, which increases its complexity and computational overhead. This effect is particularly pronounced in large systems, where the added complexity can significantly impact overall performance.

Table 4.1: Computational costs for relaxation schemes in fourth order point iterative methods ($\sigma = M - 1$).

Method	FS9	HS9	QS9
GS	$17\sigma^2*\text{ite}$	$(8\sigma^2 - 8\sigma + 16)*\text{ite} + 3(\sigma^2 - 1)$	$\frac{17}{4}(\sigma - 1)^2*\text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
SOR	$21\sigma^2*\text{ite}$	$(10\sigma^2 - 8\sigma + 18)*\text{ite} + 3(\sigma^2 - 1)$	$\frac{21}{4}(\sigma - 1)^2*\text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MSOR			
AOR	$2\sigma(16\sigma - 3)*\text{ite}$	$\frac{1}{6}(92\sigma^2 - 165\sigma - 357)*\text{ite} + 3(\sigma^2 - 1)$	$8(\lambda - 1)^2*\text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MAOR			

Table 4.2: Computational costs for relaxation schemes in fourth order group iterative methods ($\sigma = M - 1$).

Method	EG9	EDG9	MEG9
GS	$(21\sigma^2 + 38\sigma + 17)\text{ite}$	$(9\sigma^2 - 14\sigma + 25)*\text{ite} + 3(\sigma^2 - 1)$	$\frac{21}{4}(\sigma - 1)^2*\text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
SOR	$(25\sigma^2 + 46\sigma + 21)*\text{ite}$	$(11\sigma^2 - 14\sigma + 27)*\text{ite} + 3(\sigma^2 - 1)$	$\frac{25}{4}(\sigma - 1)^2*\text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MSOR			
AOR	$(38\sigma^2 + 63\sigma + 26)*\text{ite}$	$\frac{1}{8}(141\sigma^2 - 304\sigma + 539)*\text{ite} + 3(\sigma^2 - 1)$	$\frac{19}{2}(\sigma - 1)^2*\text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MAOR			

Table 4.3: Number of iterations (Ite) and Error for fourth order iterative methods with various relaxation schemes for Problem 1.

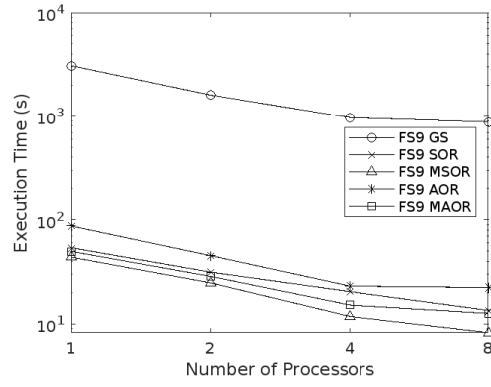
Method	M	Ite	Error	Ite	Error	Ite	Error
FS9				HS9		QS9	
GS	66	5500	1.90E-12	2834	5.09E-09	1462	4.65E-13
	130	20040	3.70E-12	10348	1.71E-10	5342	9.19E-13
	256	73795	7.28E-12	38188	4.00E-12	19748	1.82E-12
	514	272437	1.45E-11	141346	7.11E-12	73253	3.61E-12
SOR	66	281	1.15E-15	186	5.09E-09	139	1.94E-14
	130	529	5.28E-15	385	1.72E-10	275	2.99E-14
	256	1053	1.63E-15	797	5.59E-12	524	8.39E-16
	514	2960	1.18E-13	2290	1.31E-13	1057	2.13E-16
MSOR	66	270	1.46E-14	186	5.09E-09	137	1.85E-15
	130	527	6.43E-15	363	1.72E-10	267	7.14E-16
	256	1053	1.63E-15	725	5.58E-12	524	9.62E-16
	514	2119	2.77E-15	1935	1.35E-13	1057	2.13E-16
AOR	66	223	1.30E-14	159	5.09E-09	115	1.79E-14
	130	482	1.23E-14	299	1.72E-10	225	8.57E-15
	256	851	3.82E-14	782	5.59E-12	469	2.74E-14
	514	2384	1.06E-14	2264	1.75E-13	830	5.40E-15
MAOR	66	220	3.00E-14	150	5.09E-09	111	2.01E-14
	130	427	2.37E-14	291	1.72E-10	216	9.01E-15
	256	833	9.83E-15	573	5.58E-12	423	1.28E-14
	514	1635	1.17E-14	1128	1.70E-13	830	5.40E-15
EG9				EDG9		MEG9	
GS	66	3109	1.04E-12	2292	5.09E-09	827	2.53E-13
	130	11338	2.03E-12	8365	1.71E-10	3019	5.04E-13
	256	41818	4.00E-12	30887	4.11E-12	11172	9.98E-13
	514	154716	7.95E-12	114403	5.66E-12	41510	1.99E-12
SOR	66	189	1.36E-15	171	5.09E-09	90	2.63E-15
	130	365	7.29E-16	344	1.72E-10	181	6.43E-16
	256	739	1.85E-15	661	5.57E-12	354	2.48E-16
	514	2538	1.04E-13	1753	1.40E-13	726	9.06E-17
MSOR	66	189	3.74E-14	170	5.09E-09	90	2.63E-15
	130	365	7.29E-16	324	1.72E-10	177	3.97E-15
	256	739	1.85E-15	661	5.57E-12	354	2.48E-16
	514	1946	7.73E-14	1753	1.40E-13	726	9.06E-17
AOR	66	170	8.49E-15	143	5.09E-09	87	1.16E-14
	130	327	2.07E-14	279	1.72E-10	171	8.71E-15
	256	658	1.04E-14	618	5.59E-12	325	7.17E-15
	514	1306	6.04E-15	1224	1.74E-13	656	5.13E-15
MAOR	66	167	2.37E-14	140	5.09E-09	84	7.65E-15
	130	323	2.50E-14	272	1.72E-10	163	9.85E-15
	256	632	3.15E-10	531	5.58E-12	320	7.25E-15
	514	1259	1.08E-14	1073	1.78E-13	627	5.41E-15

Table 4.4: Execution time (seconds), speedup and parallel efficiency for all fourth order point iterative methods with various relaxation schemes for Problem 1 at $M = 514$.

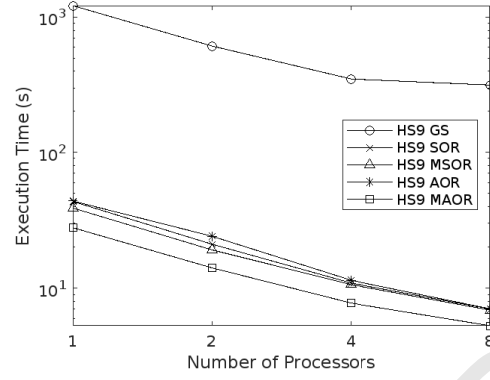
Methods	Execution Time					Speedup				Efficiency			
	T1	T2	T4	T8		S1	S2	S4	S8	E1	E2	E4	E8
FS9 GS	3093.100	1610.040	971.646	881.115		1.000	1.921	3.183	3.510	1.000	0.961	0.796	0.439
FS9 SOR	54.140	31.539	20.488	13.529		1.000	1.717	2.642	4.002	1.000	0.858	0.661	0.500
FS9 MSOR	43.930	24.857	11.824	8.263		1.000	1.767	3.715	5.317	1.000	0.884	0.929	0.665
FS9 AOR	87.639	45.405	23.151	22.391		1.000	1.930	3.786	3.914	1.000	0.965	0.946	0.489
FS9 MAOR	49.934	28.689	15.223	12.652		1.000	1.741	3.280	3.947	1.000	0.870	0.820	0.493
HS9 GS	1216.070	613.522	351.558	317.160		1.000	1.982	3.459	3.834	1.000	0.991	0.865	0.479
HS9 SOR	43.450	21.065	10.921	7.041		1.000	2.063	3.979	6.171	1.000	1.031	0.995	0.771
HS9 MSOR	38.872	19.122	10.633	6.906		1.000	2.033	3.656	5.628	1.000	1.016	0.914	0.704
HS9 AOR	43.781	24.108	11.442	7.057		1.000	1.816	3.826	6.204	1.000	0.908	0.957	0.776
HS9 MAOR	27.861	14.115	7.779	5.300		1.000	1.974	3.582	5.257	1.000	0.987	0.895	0.657
QS9 GS	228.112	120.422	65.303	56.359		1.000	1.894	3.493	4.048	1.000	0.947	0.873	0.506
QS9 SOR	6.391	2.511	1.237	1.143		1.000	2.545	5.167	5.591	1.000	1.273	1.292	0.699
QS9 MSOR	6.391	2.511	1.237	1.143		1.000	2.545	5.167	5.591	1.000	1.273	1.292	0.699
QS9 AOR	15.996	8.390	4.088	1.913		1.000	1.907	3.913	8.362	1.000	0.953	0.978	1.045
QS9 MAOR	15.996	8.390	4.088	1.913		1.000	1.907	3.913	8.362	1.000	0.953	0.978	1.045

Table 4.5: Execution time (seconds), speedup and parallel efficiency for all fourth order group iterative methods with various relaxation schemes for Problem 1 at $M = 514$.

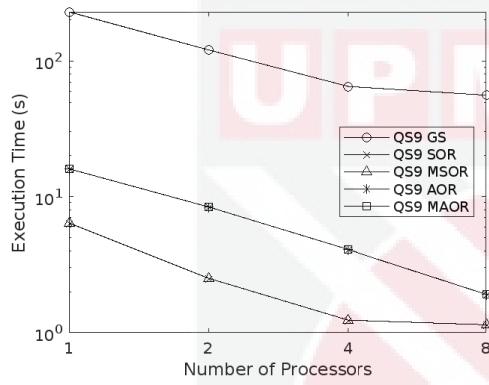
Methods	Execution Time					Speedup					Efficiency			
	T1	T2	T4	T8		S1	S2	S4	S8		E1	E2	E4	E8
EG9 GS	1779.940	811.489	455.979	432.450		1.000	2.193	3.904	4.116		1.000	1.097	0.976	0.514
EG9 SOR	60.279	33.225	23.008	19.703		1.000	1.814	2.620	3.059		1.000	0.907	0.655	0.382
EG9 MSOR	45.639	25.760	12.255	9.619		1.000	1.772	3.724	4.745		1.000	0.886	0.931	0.593
EG9 AOR	46.574	27.869	14.327	9.723		1.000	1.671	3.251	4.790		1.000	0.836	0.813	0.599
EG9 MAOR	45.538	24.918	12.147	8.912		1.000	1.828	3.749	5.110		1.000	0.914	0.937	0.639
EDG9 GS	933.469	442.251	252.414	227.346		1.000	2.111	3.698	4.106		1.000	1.055	0.925	0.513
EDG9 SOR	33.918	18.020	8.783	6.124		1.000	1.882	3.862	5.539		1.000	0.941	0.965	0.692
EDG9 MSOR	33.918	18.020	8.783	6.124		1.000	1.882	3.862	5.539		1.000	0.941	0.965	0.692
EDG9 AOR	40.250	19.623	10.687	7.004		1.000	2.051	3.766	5.747		1.000	1.026	0.942	0.718
EDG9 MAOR	29.078	14.561	8.283	5.830		1.000	1.997	3.511	4.988		1.000	0.999	0.878	0.623
MEG9 GS	115.554	59.744	30.785	27.309		1.000	1.934	3.754	4.231		1.000	0.967	0.938	0.529
MEG9 SOR	4.705	1.860	0.969	0.587		1.000	2.529	4.854	8.011		1.000	1.265	1.213	1.001
MEG9 MSOR	4.705	1.860	0.969	0.587		1.000	2.529	4.854	8.011		1.000	1.265	1.213	1.001
MEG9 AOR	13.744	8.132	2.117	1.718		1.000	1.690	6.491	8.002		1.000	0.845	1.623	1.000
MEG9 MAOR	8.230	5.586	2.035	1.214		1.000	1.473	4.044	6.782		1.000	0.737	1.011	0.848



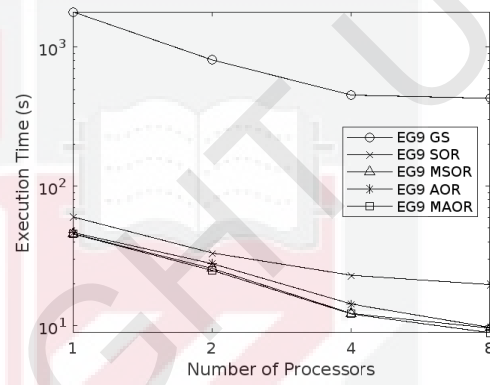
(a)



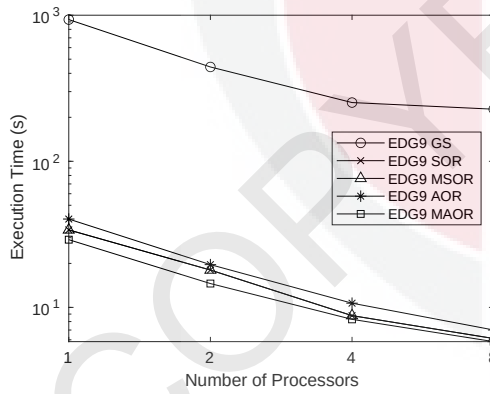
(b)



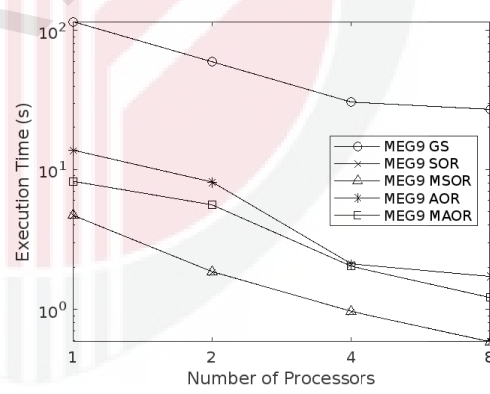
(c)



(d)

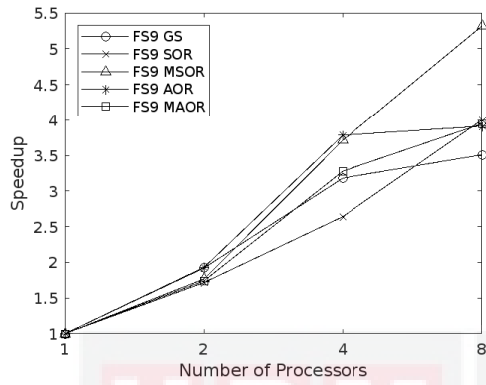


(e)

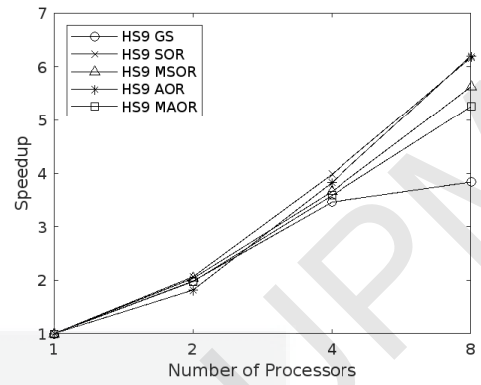


(f)

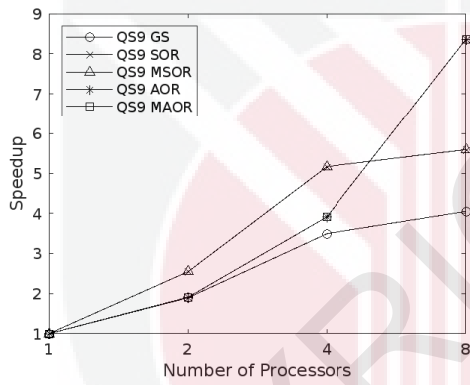
Figure 4.7: The execution time (in seconds) for (a) FS9; (b) HS9; (c) QS9; (d) EG9; (e) EDG9; and (f) MEG9 iterative methods on all relaxation schemes for Problem 1 at $M = 514$.



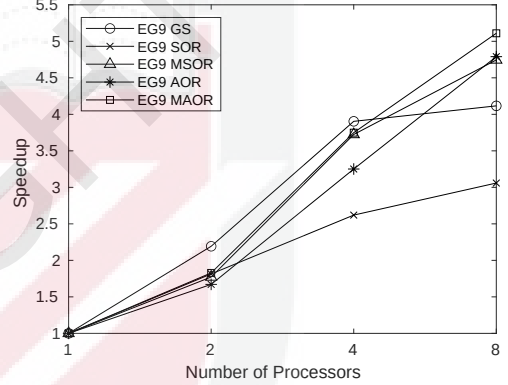
(a)



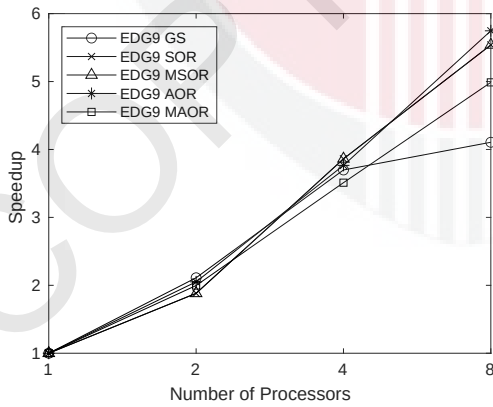
(b)



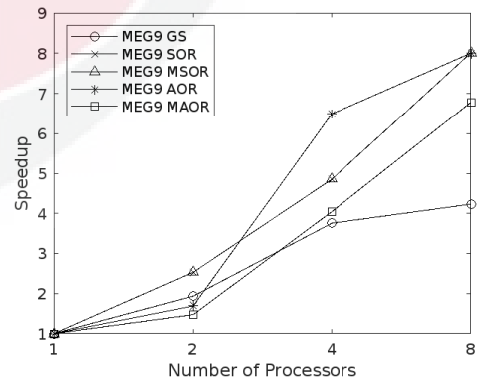
(c)



(d)



(e)



(f)

Figure 4.8: The speedup for (in seconds) for (a) FS9; (b) HS9; (c) QS9; (d) EG9; (e) EDG9; and (f) MEG9 iterative methods on all relaxation schemes for Problem 1 at $M = 514$.

Table 4.6: Number of iterations (Ite) and Error for fourth order iterative methods with various relaxation schemes for Problem 2.

Method	M	Ite	Error	Ite	Error	Ite	Error
FS9				HS9		QS9	
GS	66	8266	1.20E-12	4260	5.28E-10	2194	7.16E-10
	130	30132	5.47E-12	15561	1.62E-11	8028	2.32E-11
	256	110971	1.10E-11	57433	5.11E-12	29692	2.40E-12
	514	409687	2.18E-11	212574	1.09E-11	110156	5.43E-12
SOR	66	336	3.16E-12	231	5.29E-10	162	7.16E-10
	130	629	6.60E-14	444	1.79E-11	339	2.39E-11
	256	1519	3.40E-15	1031	5.85E-13	621	7.64E-13
	514	4703	1.92E-13	3341	1.87E-14	1537	2.45E-14
MSOR	66	322	3.13E-12	230	5.29E-10	162	7.16E-10
	130	629	6.40E-14	444	1.79E-11	319	2.39E-11
	256	1238	3.16E-15	939	5.85E-13	620	7.63E-13
	514	4703	1.92E-13	2634	9.49E-14	1248	2.40E-14
AOR	66	264	3.12E-12	195	5.29E-10	142	7.16E-10
	130	601	1.01E-13	392	1.79E-11	268	2.39E-11
	256	1197	1.24E-14	779	5.88E-13	601	7.70E-13
	514	2368	1.10E-14	2077	7.04E-14	1194	2.27E-14
MAOR	66	257	3.18E-12	179	5.29E-10	131	7.17E-10
	130	501	9.00E-14	345	1.80E-11	254	2.39E-11
	256	987	2.47E-14	670	5.98E-13	499	7.71E-13
	514	1993	1.31E-14	1326	4.04E-14	988	3.78E-14
EG9				EDG9		MEG9	
GS	66	4670	1.81E-12	3443	5.28E-10	1240	7.16E-10
	130	17045	2.96E-12	12577	1.65E-11	4535	2.35E-11
	256	62885	6.03E-12	46449	4.02E-12	16795	1.25E-12
	514	232666	1.20E-11	172050	8.70E-12	62423	2.98E-12
SOR	66	237	3.16E-12	211	5.29E-10	112	7.17E-10
	130	487	1.05E-13	418	1.79E-11	224	2.39E-11
	256	983	3.38E-15	924	5.85E-13	476	7.72E-13
	514	3991	1.63E-13	2843	1.02E-13	972	2.46E-14
MSOR	66	227	3.16E-12	205	5.29E-10	108	7.17E-10
	130	449	1.05E-13	413	1.80E-11	219	2.39E-11
	256	899	3.38E-15	919	5.85E-13	450	7.72E-13
	514	3195	1.29E-13	2342	1.95E-14	911	2.13E-14
AOR	66	205	3.17E-12	174	5.29E-10	107	7.16E-10
	130	418	8.07E-14	347	1.80E-11	208	2.39E-11
	256	842	7.56E-15	744	5.56E-13	421	7.68E-13
	514	2305	8.81E-14	1356	3.74E-14	841	2.37E-14
MAOR	66	196	3.20E-12	164	5.29E-10	100	7.17E-10
	130	380	1.27E-13	316	1.80E-11	193	2.39E-11
	256	746	3.33E-14	633	5.78E-13	378	7.80E-13
	514	1495	6.11E-15	1252	1.82E-14	748	2.60E-14

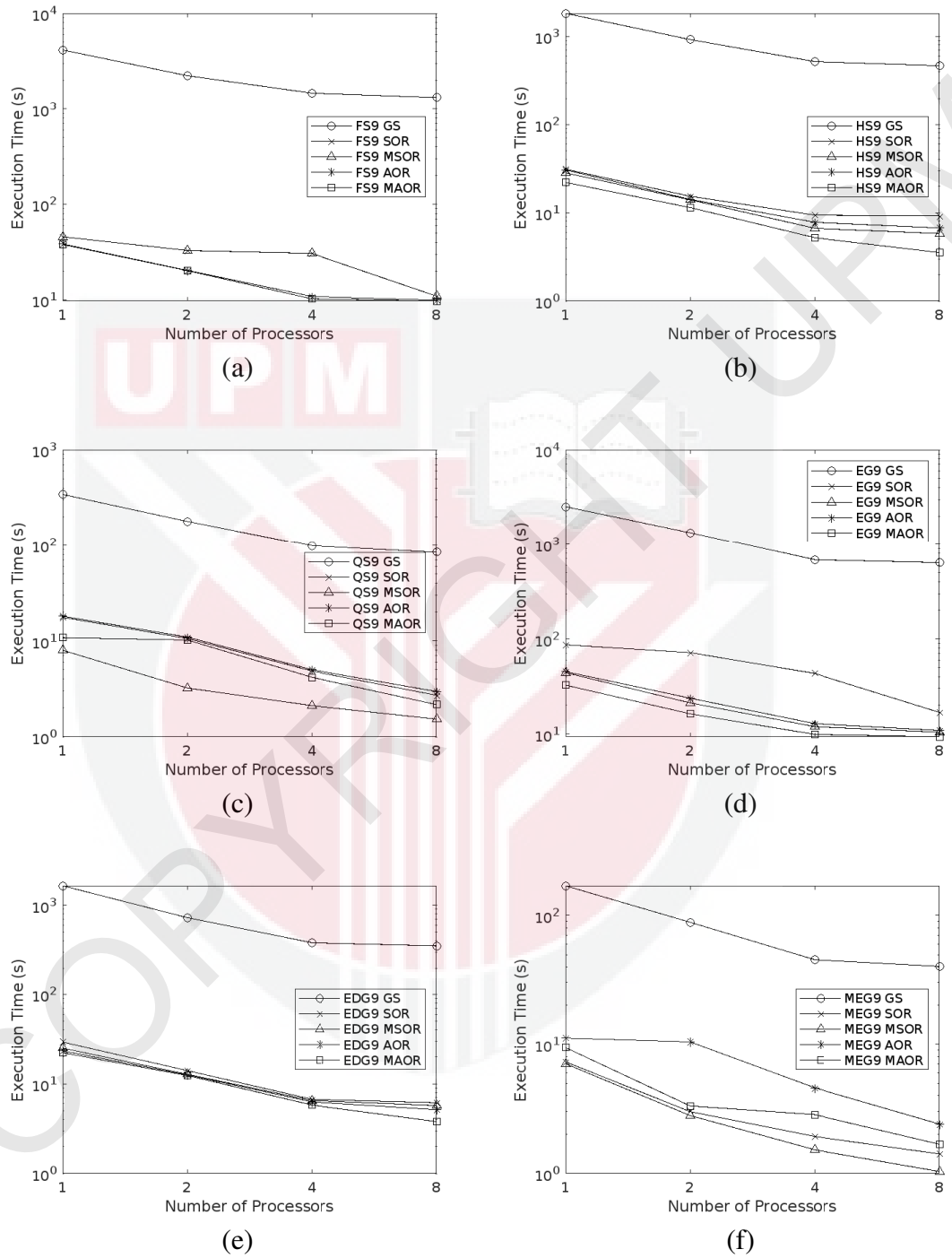
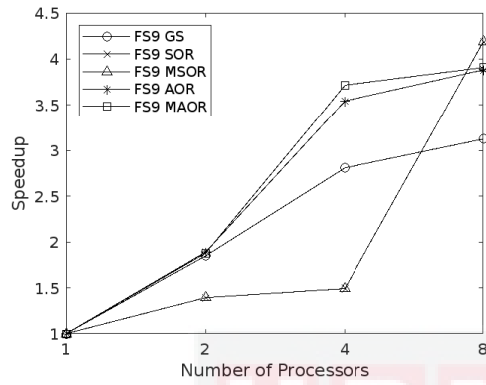
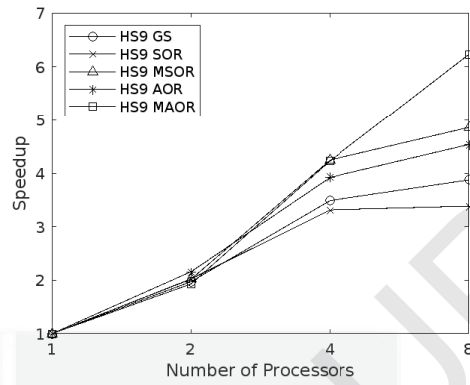


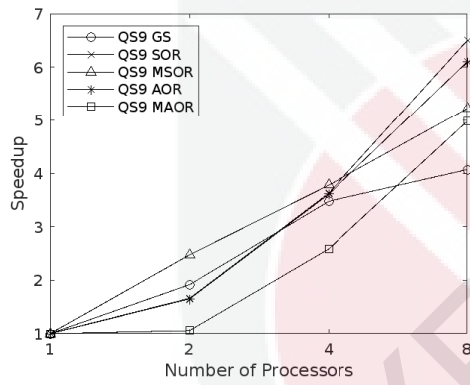
Figure 4.9: The execution time (in seconds) for (a) FS9; (b) HS9; (c) QS9; (d) EG9; (e) EDG9; and (f) MEG9 iterative methods on all relaxation schemes for Problem 1 at $M = 514$.



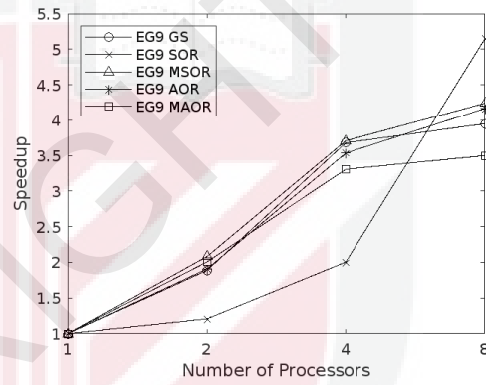
(a)



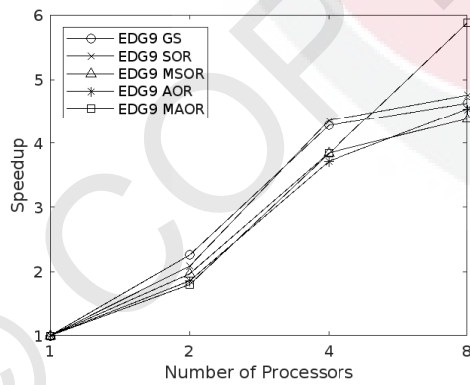
(b)



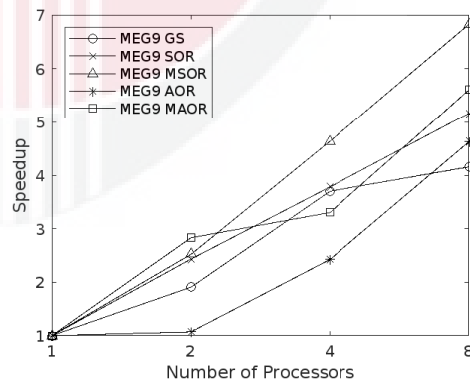
(c)



(d)



(e)



(f)

Figure 4.10: The speedup for (in seconds) for (a) FS9; (b) HS9; (c) QS9; (d) EG9; (e) EDG9; and (f) MEG9 iterative methods on all relaxation schemes for Problem 1 at $M = 514$.

Table 4.7: Computational costs of fourth order iterative methods with various relaxation schemes for Problems 1 and 2 at $M = 514$.

Methods	Computational Cost	
	Problem 1	Problem 2
FS9 GS	1,218,848,538,501	1,832,887,607,751
FS9 SOR	16,358,585,040	25,991,359,947
FS9 MSOR	11,710,757,331	25,991,359,947
FS9 AOR	20,069,298,720	19,934,605,440
FS9 MAOR	13,763,969,550	16,777,731,690
HS9 GS	297,006,050,848	446,674,883,040
HS9 SOR	6,018,002,664	8,779,614,468
HS9 MSOR	5,085,203,244	6,921,898,440
HS9 AOR	9,104,539,528	8,352,595,511
HS9 MAOR	4,536,580,152	5,332,756,170
QS9 GS	81,613,298,183	122,727,373,319
QS9 SOR	1,455,954,439	2,116,557,319
QS9 MSOR	1,455,954,439	1,718,819,335
QS9 AOR	1,741,888,007	2,505,251,335
QS9 MAOR	1,741,888,007	2,073,238,023
EG9 GS	858,064,218,960	1,290,379,595,960
EG9 SOR	16,758,018,072	26,351,950,404
EG9 MSOR	12,849,134,424	21,096,086,580
EG9 AOR	13,102,793,702	23,125,527,935
EG9 MAOR	12,631,253,653	14,998,986,665
EDG9 GS	270,147,915,196	406,273,465,704
EDG9 SOR	5,062,934,616	8,210,531,976
EDG9 MSOR	5,062,934,616	6,763,792,272
EDG9 AOR	5,654,356,152	6,264,054,516
EDG9 MAOR	4,956,898,175	5,783,686,108
MEG9 GS	57,129,638,407	85,911,280,135
MEG9 SOR	1,190,730,247	1,593,776,647
MEG9 MSOR	1,190,730,247	1,493,834,247
MEG9 AOR	1,634,933,255	2,095,651,335
MEG9 MAOR	1,562,712,583	1,864,047,111

Table 4.8: Execution time (seconds), speedup and parallel efficiency for all fourth order point iterative methods with various relaxation schemes for Problem 2 at $M = 514$.

Methods	Execution Time				Speedup				Efficiency			
	T1	T2	T4	T8	S1	S2	S4	S8	E1	E2	E4	E8
FS9 GS	4159.730	2249.240	1481.010	1331.040	1.000	1.849	2.809	3.125	1.000	0.925	0.702	0.391
FS9 SOR	45.635	32.769	30.580	10.898	1.000	1.393	1.492	4.187	1.000	0.696	0.373	0.523
FS9 MSOR	45.635	32.769	30.580	10.898	1.000	1.393	1.492	4.187	1.000	0.696	0.373	0.523
FS9 AOR	38.348	20.274	10.820	9.879	1.000	1.891	3.544	3.882	1.000	0.946	0.886	0.485
FS9 MAOR	37.895	20.170	10.198	9.698	1.000	1.879	3.716	3.908	1.000	0.939	0.929	0.488
HS9 GS	1811.320	918.543	519.277	467.383	1.000	1.972	3.488	3.875	1.000	0.986	0.872	0.484
HS9 SOR	31.039	15.339	9.366	9.166	1.000	2.024	3.314	3.386	1.000	1.012	0.828	0.423
HS9 MSOR	28.209	13.975	6.645	5.806	1.000	2.019	4.245	4.859	1.000	1.009	1.061	0.607
HS9 AOR	30.467	14.119	7.772	6.713	1.000	2.158	3.920	4.538	1.000	1.079	0.980	0.567
HS9 MAOR	22.012	11.409	5.204	3.533	1.000	1.929	4.229	6.231	1.000	0.965	1.057	0.779
QS9 GS	342.790	178.782	98.606	84.266	1.000	1.917	3.476	4.068	1.000	0.959	0.869	0.508
QS9 SOR	17.582	10.576	4.842	2.706	1.000	1.662	3.631	6.498	1.000	0.831	0.908	0.812
QS9 MSOR	7.979	3.225	2.111	1.528	1.000	2.474	3.779	5.220	1.000	1.237	0.945	0.653
QS9 AOR	18.070	10.952	5.004	2.964	1.000	1.650	3.611	6.097	1.000	0.825	0.903	0.762
QS9 MAOR	10.782	10.203	4.169	2.166	1.000	1.057	2.587	4.977	1.000	0.528	0.647	0.622

Table 4.9: Execution time (seconds), speedup and parallel efficiency for all fourth order group iterative methods with various relaxation schemes for Problem 2 at $M = 514$.

Methods	Execution Time					Speedup				Efficiency			
	T1	T2	T4	T8		S1	S2	S4	S8	E1	E2	E4	E8
EG9 GS	2515.250	1332.690	683.572	636.797		1.000	1.887	3.680	3.950	1.000	0.944	0.920	0.494
EG9 SOR	86.464	71.579	43.172	16.795		1.000	1.208	2.003	5.148	1.000	0.604	0.501	0.644
EG9 MSOR	44.025	21.144	11.869	10.386		1.000	2.082	3.709	4.239	1.000	1.041	0.927	0.530
EG9 AOR	45.187	23.694	12.782	10.848		1.000	1.907	3.535	4.165	1.000	0.954	0.884	0.521
EG9 MAOR	32.544	16.222	9.838	9.299		1.000	2.006	3.308	3.500	1.000	1.003	0.827	0.437
EDG9 GS	1638.230	725.807	382.791	353.734		1.000	2.257	4.280	4.631	1.000	1.129	1.070	0.579
EDG9 SOR	29.332	14.085	6.727	6.167		1.000	2.083	4.360	4.756	1.000	1.041	1.090	0.595
EDG9 MSOR	25.115	12.797	6.527	5.730		1.000	1.963	3.848	4.383	1.000	0.981	0.962	0.548
EDG9 AOR	23.390	12.616	6.293	5.149		1.000	1.854	3.717	4.543	1.000	0.927	0.929	0.568
EDG9 MAOR	22.285	12.407	5.804	3.792		1.000	1.796	3.839	5.876	1.000	0.898	0.960	0.735
MEG9 GS	168.808	88.501	45.614	40.602		1.000	1.907	3.701	4.158	1.000	0.954	0.925	0.520
MEG9 SOR	7.314	3.006	1.934	1.415		1.000	2.433	3.782	5.168	1.000	1.217	0.946	0.646
MEG9 MSOR	7.064	2.802	1.523	1.035		1.000	2.521	4.639	6.825	1.000	1.261	1.160	0.853
MEG9 AOR	11.156	10.417	4.600	2.410		1.000	1.071	2.425	4.629	1.000	0.536	0.606	0.579
MEG9 MAOR	9.421	3.328	2.852	1.681		1.000	2.831	3.303	5.605	1.000	1.415	0.826	0.701

CHAPTER 5

FRACTIONAL ORDER MAOR ITERATIVE METHOD FOR SOLVING FRACTIONAL POISSON EQUATION

5.1 Introduction

This chapter is devoted to the formulation of fractional order MAOR iterative methods. It starts with the derivation of the MAOR scheme on FFS, FHS, FQS, FEG, FEDG, and FMEG iterative methods which includes different ordering strategies of the solution domain. The solution domain is ordered such that the tasks executed will be independent from each other. Then, the chapter is continued with the sequential algorithm for the fractional order MAOR iterative methods. This will describe how each iterative method is formulated with their respective ordering strategies. Next, the chapter provides the construction of parallel algorithm for the fractional order MAOR iterative methods. With the tasks being independent from each other, it is possible to compute them in a parallel environment.

5.2 Derivation of Sequential Algorithms for Fractional Order MAOR Iterative Methods

This section presents the derivation of sequential algorithms for MAOR scheme on the fractional order iterative methods. The inclusion of the solution domain for each iterative method with their respective ordering strategies will assist in the derivation of the scheme. For simplicity, let's denote $v_{i,j} = u_{i,j}^{k+1} - u_{i,j}^k$.

5.2.1 Sequential Fractional Order Point Iterative Methods with MAOR Scheme

Sequential FFS MAOR

Figure 5.1 shows the solution domain for FFS with the chessboard (CB) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

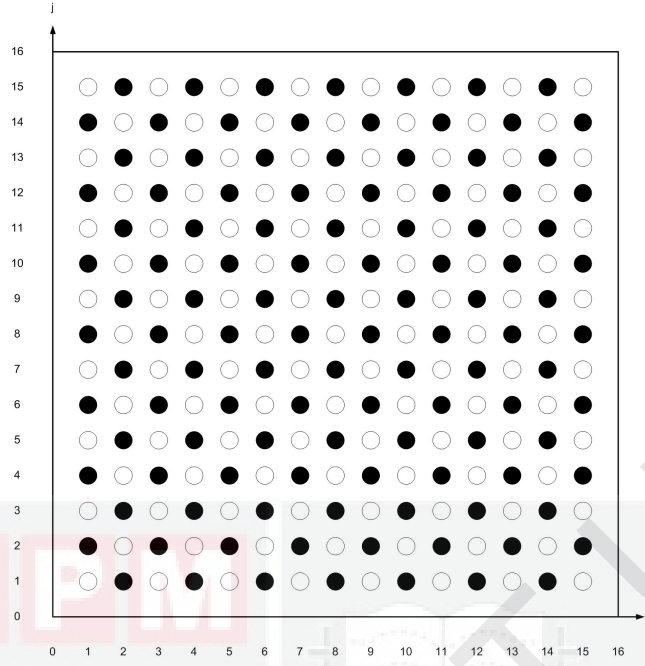


Figure 5.1: The solution domain for FFS.

For this case, to represent Equation (2.17) in the following matrix form, let the mesh size be defined as shown in Figure 5.1.

$$D = \begin{bmatrix} P & & & \\ & P & & \\ & & \ddots & \\ & & & P \end{bmatrix}, \quad L = \begin{bmatrix} H_2 & 0 \\ H_1 & H_2 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & H_3 \\ 0 & 0 \end{bmatrix},$$

$$H_1 = \begin{bmatrix} J_1 & J_3 & & & & \\ K_2 & J_2 & J_3 & & & \\ & K_2 & J_1 & \ddots & & \\ K_4 & & \ddots & \ddots & \ddots & \\ \vdots & \ddots & & \ddots & J_1 & J_3 \\ K_{14} & & \ddots & & K_2 & J_2 & J_3 \\ & K_{14} & \cdots & K_4 & & K_2 & J_1 \end{bmatrix}, \quad b = h^\alpha h^\beta \begin{bmatrix} f_{1,1} \\ f_{3,1} \\ \vdots \\ f_{i,j} \\ \vdots \\ f_{12,15} \\ f_{14,15} \end{bmatrix},$$

$$H_2 = \begin{bmatrix} J_4 & & & & \\ & J_4 & & & \\ K_3 & & J_4 & & \\ & K_3 & & \ddots & \\ K_5 & & \ddots & & J_4 \\ \vdots & \ddots & & K_3 & J_4 \\ K_{15} & \cdots & K_5 & K_3 & J_4 \end{bmatrix},$$

$$H_3 = \begin{bmatrix} J_2 & J_3 & & & \\ K_2 & J_1 & J_3 & & \\ & K_2 & J_2 & \ddots & \\ K_4 & & \ddots & \ddots & \ddots \\ \vdots & \ddots & & \ddots & J_2 & J_3 \\ K_{14} & & \ddots & & K_2 & J_1 & J_3 \\ & K_{14} & \cdots & K_4 & & K_2 & J_2 \end{bmatrix},$$

$$J_1 = \begin{bmatrix} -h^\beta g_{(\alpha,2)} & -h^\beta & & & \\ -h^\beta g_{(\alpha,4)} & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \\ -h^\beta g_{(\alpha,14)} & \cdots & -h^\beta g_{(\alpha,4)} & -h^\beta g_{(\alpha,2)} & -h^\beta \end{bmatrix},$$

$$J_2 = \begin{bmatrix} -h^\beta & & & & \\ -h^\beta g_{(\alpha,2)} & -h^\beta & & & \\ -h^\beta g_{(\alpha,4)} & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & -h^\beta \\ -h^\beta g_{(\alpha,14)} & \cdots & -h^\beta g_{(\alpha,4)} & -h^\beta g_{(\alpha,2)} & \end{bmatrix},$$

$$J_3 = \begin{bmatrix} -h^\alpha & & & & \\ & -h^\alpha & & & \\ & & \ddots & & \\ & & & -h^\alpha & \\ & & & & -h^\alpha \end{bmatrix},$$

$$J_4 = \begin{bmatrix} -h^\beta g_{(\alpha,3)} & & & & \\ -h^\beta g_{(\alpha,5)} & \ddots & & & \\ \vdots & \ddots & \ddots & & \\ -h^\beta g_{(\alpha,15)} & \cdots & -h^\beta g_{(\alpha,5)} & -h^\beta g_{(\alpha,3)} & \end{bmatrix},$$

$$K_i = \begin{bmatrix} -h^\alpha g_{(\beta,i)} & & & & \\ & -h^\alpha g_{(\beta,i)} & & & \\ & & \ddots & & \\ & & & -h^\alpha g_{(\beta,i)} & \\ & & & & -h^\alpha g_{(\beta,i)} \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential FFS MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all $\odot$ points using

$$\begin{aligned} \odot \quad u_{i,j}^{k+1} = & \frac{\theta_1}{P} \left(h^\beta \sum_{s=2,3,4,\dots}^{i+1} g_{(\alpha,s)} \cdot v_{i-s+1,j} + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g_{(\beta,z)} \cdot v_{i,j-z+1} \right) \\ & + \frac{\omega_1}{P} \left(h^\beta u_{i+1,j}^k + h^\beta \sum_{s=2,3,4,\dots}^{i+1} g_{(\alpha,s)} \cdot u_{i-s+1,j}^k \right. \\ & \left. + h^\alpha u_{i,j+1}^k + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g_{(\beta,z)} \cdot u_{i,j-z+1}^k - h^\alpha h^\beta f_{i,j} \right) \\ & + (1 - \omega_1) u_{i,j}^k. \end{aligned} \quad (5.1)$$

3. Execute all ● points using

$$\begin{aligned}
 \bullet \quad u_{i,j}^{k+1} = & \frac{\theta_2}{P} \left(h^\beta v_{i+1,j} + h^\beta \sum_{s=2,3,4,\dots}^{i+1} g_{(\alpha,s)} \cdot v_{i-s+1,j} \right. \\
 & \left. + h^\alpha v_{i,j+1} + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g_{(\beta,z)} \cdot v_{i,j-z+1} \right) \\
 & + \frac{\omega_2}{P} \left(h^\beta u_{i+1,j}^k + h^\beta \sum_{s=2,3,4,\dots}^{i+1} g_{(\alpha,s)} \cdot u_{i-s+1,j}^k \right. \\
 & \left. + h^\alpha u_{i,j+1}^k + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g_{(\beta,z)} \cdot u_{i,j-z+1}^k - h^\alpha h^\beta f_{i,j} \right) \\
 & + (1 - \omega_2) u_{i,j}^{(k)}. \tag{5.2}
 \end{aligned}$$

4. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, go to step (5), otherwise, repeat the iteration cycle (i. e. go to step (2)).

5. Stop.

Sequential FHS MAOR

The derivation of the FHS iterative method builds upon the foundational FFS iterative method. By rotating Equation (2.17) 45° anti-clockwise, the grid spacing is effectively transformed to $\sqrt{2}h$. This geometric adjustment forms the basis for the FHS iterative method, which optimises the computational process by modifying the grid structure. This transformation allows the method to take advantage of reduced computational nodes, leading to potentially more efficient calculations while maintaining the integrity of the fractional derivative discretization. Hence, the FHS iterative method can be derive as

$$\begin{aligned}
 u_{i,j} = & \frac{1}{P} \left(h^\beta u_{i+1,j+1} + h^\beta \sum_{s=2,3,4,\dots}^{i+1} g_{(\alpha,s)} \cdot u_{i-s+1,j-s+1} \right. \\
 & \left. + h^\alpha u_{i-1,j+1} + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g_{(\beta,z)} \cdot u_{i+z-1,j-z+1} - h^\alpha h^\beta f_{i,j} \right). \tag{5.3}
 \end{aligned}$$

Figure 5.3 shows the solution domain for FHS with the horizontal zebra line (HZL)

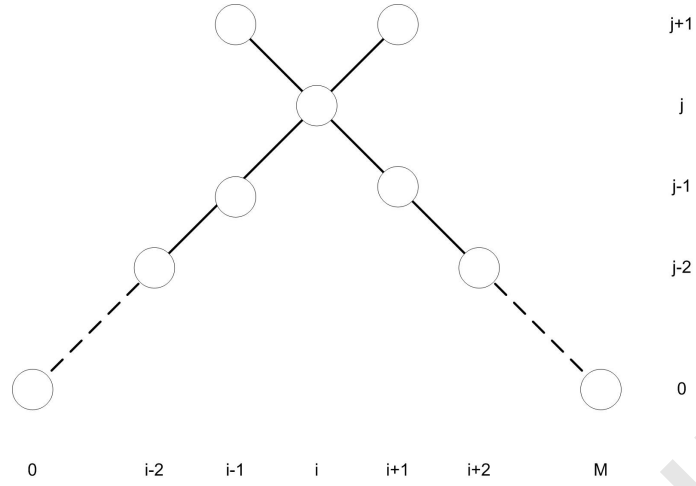


Figure 5.2: The stencil for FHS.

ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

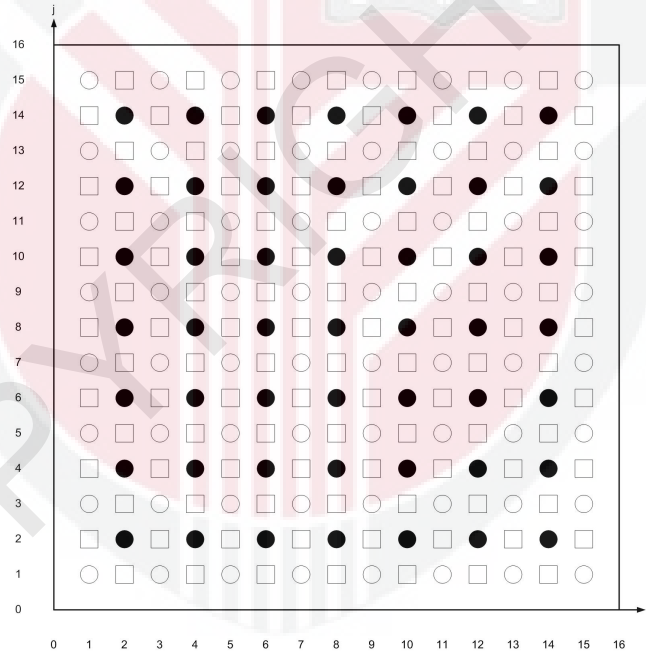


Figure 5.3: The solution domain for FHS.

For this case, to represent Equation (5.3) in the following matrix form, let the mesh size be defined as shown in Figure 5.3.

$$D = \begin{bmatrix} P & & & \\ & P & & \\ & & \ddots & \\ & & & P \\ & & & & P \end{bmatrix}, \quad L = \begin{bmatrix} H_2 & 0 \\ H_1 & H_2 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & H_1 \\ 0 & 0 \end{bmatrix},$$

$$H_1 = \begin{bmatrix} K_2 & J_1 & & \\ K_4 & \ddots & J_1 & \\ \vdots & \ddots & \ddots & \ddots \\ K_{14} & \cdots & K_4 & K_2 & J_1 \end{bmatrix}, \quad H_2 = \begin{bmatrix} K_3 & & & \\ K_5 & \ddots & & \\ \vdots & \ddots & \ddots & \\ K_{15} & \cdots & K_5 & K_3 \end{bmatrix},$$

$$J_1 = \begin{bmatrix} h^\alpha & h^\beta & & & \\ & h^\alpha & h^\beta & & \\ & & \ddots & \ddots & \\ & & & h^\alpha & h^\beta \\ & & & & h^\alpha & h^\beta \end{bmatrix}, \quad b = h^\alpha h^\beta \begin{bmatrix} f_{1,1} \\ f_{3,1} \\ \vdots \\ f_{i,j} \\ \vdots \\ f_{12,14} \\ f_{14,14} \end{bmatrix},$$

$$K_2 = \begin{bmatrix} h^\beta g_{(\alpha,2)} & h^\alpha g_{(\beta,2)} & & & \\ & h^\beta g_{(\alpha,2)} & h^\alpha g_{(\beta,2)} & & \\ & & \ddots & \ddots & \\ & & & h^\beta g_{(\alpha,2)} & h^\alpha g_{(\beta,2)} \\ & & & h^\beta g_{(\alpha,2)} & h^\alpha g_{(\beta,2)} \end{bmatrix},$$

$$K_4 = \begin{bmatrix} & & & h^\alpha g_{(\beta,4)} \\ h^\beta g_{(\alpha,4)} & & & h^\alpha g_{(\beta,4)} \\ & h^\beta g_{(\alpha,4)} & & \ddots \\ & & \ddots & h^\alpha g_{(\beta,4)} \\ & & & h^\beta g_{(\alpha,4)} \end{bmatrix},$$

$$\vdots$$

$$K_{14} = \begin{bmatrix} & & h^\alpha g_{(\beta,14)} \\ & & \\ h^\beta g_{(\alpha,14)} & & \end{bmatrix},$$

$$\begin{aligned}
K_3 &= \begin{bmatrix} & & h^\alpha g_{(\beta,3)} & & \\ h^\beta g_{(\alpha,2)} & & h^\alpha g_{(\beta,3)} & & \\ & h^\beta g_{(\alpha,2)} & & \ddots & \\ & & \ddots & & h^\alpha g_{(\beta,3)} \\ & & & h^\beta g_{(\alpha,2)} & h^\alpha g_{(\beta,3)} \\ & & & & h^\beta g_{(\alpha,2)} \end{bmatrix}, \\
K_5 &= \begin{bmatrix} & & & h^\alpha g_{(\beta,5)} & \\ h^\beta g_{(\alpha,5)} & & & h^\alpha g_{(\beta,5)} & \\ & h^\beta g_{(\alpha,5)} & & & h^\alpha g_{(\beta,5)} \\ & & \ddots & & \\ & & & h^\beta g_{(\alpha,5)} & \end{bmatrix}, \\
&\vdots \\
K_{15} &= \begin{bmatrix} & & h^\alpha g_{(\beta,15)} \\ & & \\ & & \\ & & \\ h^\beta g_{(\alpha,15)} \end{bmatrix}.
\end{aligned}$$

Thus, by implementing the above matrices into Equation (2.19), the sequential FHS MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all $\odot$ points using

$$\begin{aligned}
\bigcirc \quad u_{i,j}^{k+1} = & \frac{\theta_1}{P} \left(h^\beta \sum_{s=2,3,4,\dots}^{i+1} g(\alpha,s) \cdot v_{i-s+1,j-s+1} \right. \\
& + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g(\beta,z) \cdot v_{i+z-1,j-z+1} \Big) \\
& + \frac{\omega_1}{P} \left(h^\beta u_{i+1,j+1}^k + h^\beta \sum_{s=2,3,4,\dots}^{i+1} g(\alpha,s) \cdot u_{i-s+1,j-s+1}^k \right. \\
& + h^\alpha u_{i-1,j+1}^k + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g(\beta,z) \cdot u_{i+z-1,j-z+1}^k - h^\alpha h^\beta f_{i,j} \Big) \\
& + (1 - \omega_1) u_{i,j}^k.
\end{aligned} \tag{5.4}$$

3. Execute all $\bullet$ points using

$$\begin{aligned}
\bullet \quad u_{i,j}^{k+1} = & \frac{\theta_2}{P} \left(h^\beta v_{i+1,j+1} + h^\beta \sum_{s=2,3,4,\dots}^{i+1} g(\alpha,s) \cdot v_{i-s+1,j-s+1} \right. \\
& + h^\alpha v_{i-1,j+1} + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g(\beta,z) \cdot v_{i+z-1,j-z+1} \Big) \\
& + \frac{\omega_2}{P} \left(h^\beta u_{i+1,j+1}^k + h^\beta \sum_{s=2,3,4,\dots}^{i+1} g(\alpha,s) \cdot u_{i-s+1,j-s+1}^k \right. \\
& + h^\alpha u_{i-1,j+1}^k + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g(\beta,z) \cdot u_{i+z-1,j-z+1}^k - h^\alpha h^\beta f_{i,j} \Big) \\
& + (1 - \omega_2) u_{i,j}^k.
\end{aligned} \tag{5.5}$$

4. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, evaluate the solutions of the remaining $\square$ points using Equation (3.5). Otherwise, repeat the iteration cycle (i. e. go to step (2)).

5. Stop.

Sequential FQS MAOR

The derivation of the FQS iterative method originates from applying fractional discreti-

sation principles while modifying the grid structure to further reduce computational effort. Similar to the FFS and FHS approaches, the fractional derivative can be discretized using the same formulae, now with a grid spacing of $2h$. This wider grid spacing results in a quarter-sweep computational approach, wherein only a quarter of the computational domain points is updated per iteration. By employing this approach, the FQS method optimises the efficiency of fractional partial differential equation solvers. This is achieved by minimising the number of active computational points in each iteration while maintaining the accuracy required to resolve the fractional derivatives. Hence, the FQS iterative method can be derive as

$$u_{i,j} = \frac{1}{P} \left(h^\beta u_{i+2,j} + h^\beta \sum_{s=4,6,8,\dots}^{i+2} g(\alpha, \frac{s}{2}) \cdot u_{i-s+2,j} + h^\alpha u_{i,j+2} + h^\alpha \sum_{z=4,6,8,\dots}^{j+2} g(\beta, \frac{z}{2}) \cdot u_{i,j-z+2} - h^\alpha h^\beta f_{i,j} \right). \quad (5.6)$$

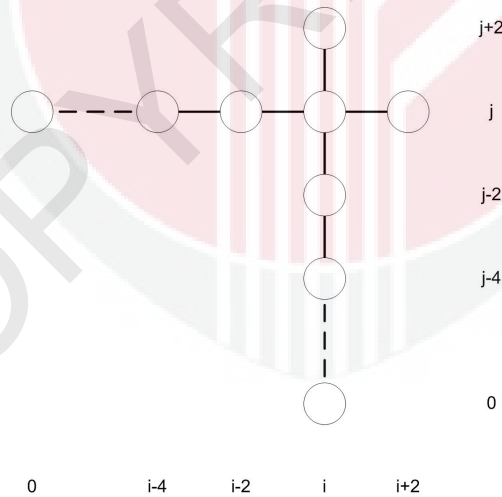


Figure 5.4: The stencil for FQS.

Figure 5.5 shows the solution domain for FQS with the chessboard (CB) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

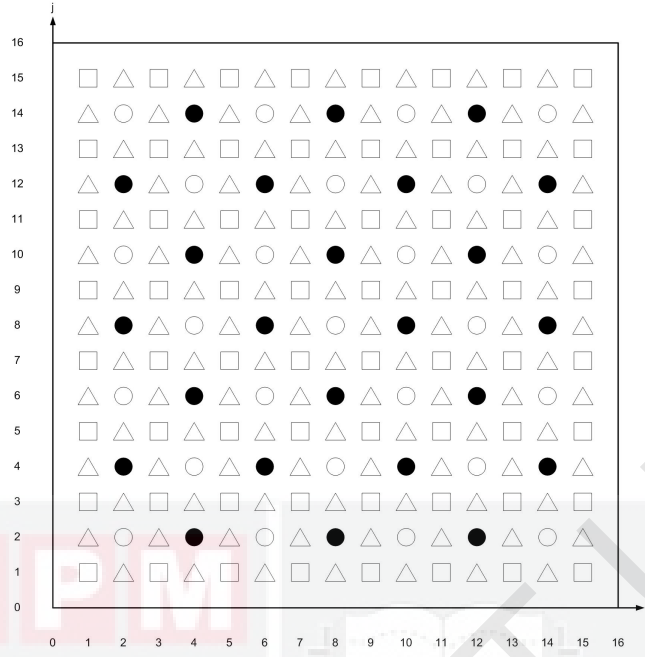


Figure 5.5: The solution domain for FQS.

For this case, to represent Equation (5.6) in the following matrix form, let the mesh size be defined as shown in Figure 5.5.

$$D = \begin{bmatrix} P & & & \\ & P & & \\ & & \ddots & \\ & & & P \end{bmatrix}, \quad L = \begin{bmatrix} H_2 & 0 \\ H_1 & H_2 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & H_3 \\ 0 & 0 \end{bmatrix},$$

$$H_1 = \begin{bmatrix} J_1 & J_3 & & & & \\ K_2 & J_2 & J_3 & & & \\ & K_2 & J_1 & J_3 & & \\ K_4 & & K_2 & J_2 & J_3 & \\ & K_4 & & K_2 & J_1 & J_3 \\ K_6 & & K_4 & & K_2 & J_2 & J_3 \\ & K_6 & & K_4 & & K_2 & J_1 \end{bmatrix}, \quad b = h^\alpha h^\beta \begin{bmatrix} f_{2,2} \\ f_{6,2} \\ \vdots \\ f_{i,j} \\ \vdots \\ f_{8,14} \\ f_{12,14} \end{bmatrix},$$

$$H_2 = \begin{bmatrix} J_4 & & & & \\ & J_4 & & & \\ K_3 & & J_4 & & \\ & K_3 & & J_4 & \\ K_5 & & K_3 & & J_4 \\ & K_5 & & K_3 & J_4 \\ K_7 & & K_5 & & K_3 & J_4 \end{bmatrix},$$

$$H_3 = \begin{bmatrix} J_2 & J_3 & & & \\ K_2 & J_1 & J_3 & & \\ & K_2 & J_2 & J_3 & \\ K_4 & & K_2 & J_1 & J_3 \\ & K_4 & & K_2 & J_2 & J_3 \\ K_6 & & K_4 & & K_2 & J_1 & J_3 \\ & K_6 & & K_4 & & K_2 & J_2 \end{bmatrix},$$

$$J_1 = \begin{bmatrix} -h^\beta g_{(\alpha,2)} & -h^\beta & & & \\ -h^\beta g_{(\alpha,4)} & -h^\beta g_{(\alpha,2)} & -h^\beta & & \\ -h^\beta g_{(\alpha,6)} & -h^\beta g_{(\alpha,4)} & -h^\beta g_{(\alpha,2)} & -h^\beta & \\ -h^\beta g_{(\alpha,8)} & -h^\beta g_{(\alpha,6)} & -h^\beta g_{(\alpha,4)} & -h^\beta g_{(\alpha,2)} & \end{bmatrix},$$

$$J_2 = \begin{bmatrix} -h^\beta & & & & \\ -h^\beta g_{(\alpha,2)} & -h^\beta & & & \\ -h^\beta g_{(\alpha,4)} & -h^\beta g_{(\alpha,2)} & -h^\beta & & \\ -h^\beta g_{(\alpha,6)} & -h^\beta g_{(\alpha,4)} & -h^\beta g_{(\alpha,2)} & -h^\beta & \end{bmatrix},$$

$$J_3 = \begin{bmatrix} -h^\alpha & & & & \\ & -h^\alpha & & & \\ & & -h^\alpha & & \\ & & & -h^\alpha & \\ & & & & -h^\alpha \end{bmatrix}, \quad J_4 = \begin{bmatrix} -h^\beta g_{(\alpha,3)} & & & & \\ -h^\beta g_{(\alpha,5)} & -h^\beta g_{(\alpha,3)} & & & \\ -h^\beta g_{(\alpha,7)} & -h^\beta g_{(\alpha,5)} & -h^\beta g_{(\alpha,3)} & & \end{bmatrix},$$

$$K_i = \begin{bmatrix} -h^\alpha g_{(\beta,i)} & & & \\ & -h^\alpha g_{(\beta,i)} & & \\ & & -h^\alpha g_{(\beta,i)} & \\ & & & -h^\alpha g_{(\beta,i)} \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential FQS MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all $\circ$ points using

$$\begin{aligned} \circ \quad u_{i,j}^{k+1} = & \frac{\theta_1}{P} \left(h^\beta \sum_{s=4,6,8,\dots}^{i+2} g(\alpha, \frac{s}{2}) \cdot v_{i-s+2,j} + h^\alpha \sum_{z=4,6,8,\dots}^{j+2} g(\beta, \frac{z}{2}) \cdot v_{i,j-z+2} \right) \\ & + \frac{\omega_1}{P} \left(h^\beta u_{i+2,j}^k + h^\beta \sum_{s=4,6,8,\dots}^{i+2} g(\alpha, \frac{s}{2}) \cdot u_{i-s+2,j}^k \right. \\ & \left. + h^\alpha u_{i,j+2}^k + h^\alpha \sum_{z=4,6,8,\dots}^{j+2} g(\beta, \frac{z}{2}) \cdot u_{i,j-z+2}^k - h^\alpha h^\beta f_{i,j} \right) \\ & + (1 - \omega_1) u_{i,j}^k. \end{aligned} \quad (5.7)$$

3. Execute all $\bullet$ points using

$$\begin{aligned} \bullet \quad u_{i,j}^{k+1} = & \frac{\theta_2}{P} \left(h^\beta v_{i+2,j} + h^\beta \sum_{s=4,6,8,\dots}^{i+2} g(\alpha, \frac{s}{2}) \cdot v_{i-s+2,j} \right. \\ & \left. + h^\alpha v_{i,j+2} + h^\alpha \sum_{z=4,6,8,\dots}^{j+2} g(\beta, \frac{z}{2}) \cdot v_{i,j-z+2} \right) \\ & + \frac{\omega_2}{P} \left(h^\beta u_{i+2,j}^k + h^\beta \sum_{s=4,6,8,\dots}^{i+2} g(\alpha, \frac{s}{2}) \cdot u_{i-s+2,j}^k \right. \\ & \left. + h^\alpha u_{i,j+2}^k + h^\alpha \sum_{z=4,6,8,\dots}^{j+2} g(\beta, \frac{z}{2}) \cdot u_{i,j-z+2}^k - h^\alpha h^\beta f_{i,j} \right) \\ & + (1 - \omega_2) u_{i,j}^k. \end{aligned} \quad (5.8)$$

4. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, evaluate the solutions of the remaining $\square$ points using Equation (3.8) followed by the remaining $\triangle$ points using Equation (3.9). Otherwise, repeat the iteration cycle (i. e. go to step (2)).
5. Stop.

5.2.2 Sequential Fractional Order Group Iterative Methods with MAOR Scheme

Sequential FEG MAOR

The derivation of the FEG iterative method is based on extending the fractional iterative schemes to encompass a complete group of four points arranged in a (4×4) configuration. This approach leverages Equation (2.17), enabling simultaneous updates to a cluster of grid points during each iteration. The structure of this method ensures efficient utilisation of computational resources by processing grouped data in parallel while maintaining the accuracy of fractional derivative calculations. Hence, the FEG iterative method can be generally expressed as

$$\begin{bmatrix} P & -h^\beta & -h^\alpha & 0 \\ -h^\beta g_{(\alpha,2)} & P & 0 & -h^\alpha \\ -h^\alpha g_{(\beta,2)} & 0 & P & -h^\beta \\ 0 & -h^\alpha g_{(\beta,2)} & -h^\beta g_{(\alpha,2)} & P \end{bmatrix} \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix},$$

where

$$S_1 = h^\beta \sum_{s=2,3,4,\dots}^{i+1} g_{(\alpha,s)} \cdot u_{i-s+1,j} + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g_{(\beta,z)} \cdot u_{i,j-z+1} - h^\alpha h^\beta f_{i,j},$$

$$S_2 = h^\beta u_{i+2,j} + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g_{(\beta,z)} \cdot u_{i+1,j-z+1} - h^\alpha h^\beta f_{i+1,j},$$

$$S_3 = h^\beta \sum_{s=2,3,4,\dots}^{i+1} g_{(\alpha,s)} \cdot u_{i-s+1,j+1} + h^\alpha u_{i,j+2} - h^\alpha h^\beta f_{i,j+1},$$

$$S_4 = h^\beta u_{i+2,j+1} + h^\alpha u_{i+1,j+2} - h^\alpha h^\beta f_{i+1,j+1}.$$

By establishing the inverse matrix of the coefficient matrix and manipulating it, the equation above can be rewritten as

$$\begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix} = \frac{1}{Q_7} \begin{bmatrix} PR_1 & R_2 & R_3 & R_4 \\ g_{(\alpha,2)}R_2 & PR_1 & g_{(\alpha,2)}R_4 & R_3 \\ g_{(\beta,2)}R_3 & g_{(\beta,2)}R_4 & PR_1 & R_2 \\ g_{(\alpha,2)}g_{(\beta,2)}R_4 & g_{(\beta,2)}R_3 & g_{(\alpha,2)}R_2 & PR_1 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}, \quad (5.9)$$

where $Q_7 = g_{(\alpha,2)}^2(h^\beta)^4 - 2g_{(\alpha,2)}g_{(\beta,2)}(h^\alpha)^2(h^\beta)^2 + g_{(\beta,2)}^2(h^\alpha)^4 - 2P^2g_{(\alpha,2)}(h^\beta)^2 - 2P^2g_{(\beta,2)}(h^\alpha)^2 + P^4$ and

$$\begin{aligned} R_1 &= P^2 - g_{(\alpha,2)}(h^\beta)^2 - g_{(\beta,2)}^2(h^\alpha)^2, & R_3 &= h^\alpha(2g_{(\alpha,2)}(h^\beta)^2 + R_1), \\ R_2 &= h^\beta(2g_{(\beta,2)}(h^\alpha)^2 + R_1), & R_4 &= 2h^\alpha h^\beta P. \end{aligned}$$

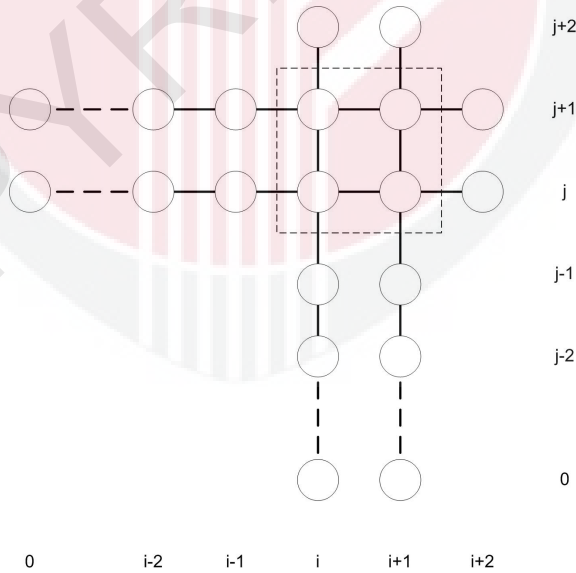


Figure 5.6: The stencil for FEG.

Figure 5.7 shows the solution domain for FEG with the chessboard (CB) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

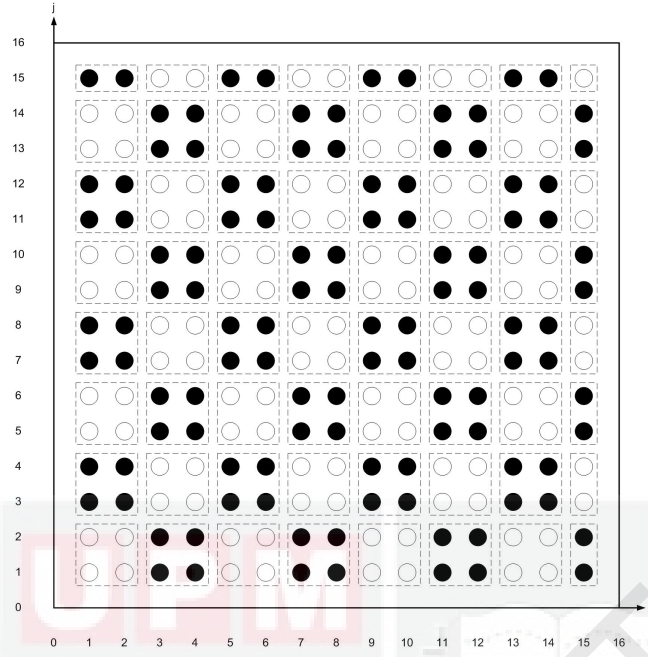


Figure 5.7: The solution domain for FEG.

Let

$$A_7 = \begin{bmatrix} PR_1 & R_2 & R_3 & R_4 \\ g_{(\alpha,2)} R_2 & PR_1 & g_{(\alpha,2)} R_4 & R_3 \\ g_{(\beta,2)} R_3 & g_{(\beta,2)} R_4 & PR_1 & R_2 \\ g_{(\alpha,2)} g_{(\beta,2)} R_4 & g_{(\beta,2)} R_3 & g_{(\alpha,2)} R_2 & PR_1 \end{bmatrix},$$

and

$$S_1^k = h^\beta \sum_{s=2,3,4,\dots}^{i+1} g_{(\alpha,s)} \cdot u_{i-s+1,j}^k + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g_{(\beta,z)} \cdot u_{i,j-z+1}^k - h^\alpha h^\beta f_{i,j},$$

$$S_2^k = h^\beta u_{i+2,j}^k + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g_{(\beta,z)} \cdot u_{i+1,j-z+1}^k - h^\alpha h^\beta f_{i+1,j},$$

$$S_3^k = h^\beta \sum_{s=2,3,4,\dots}^{i+1} g_{(\alpha,s)} \cdot u_{i-s+1,j+1}^k + h^\alpha u_{i,j+2}^k - h^\alpha h^\beta f_{i,j+1},$$

$$S_4^k = h^\beta u_{i+2,j+1}^k + h^\alpha u_{i+1,j+2}^k - h^\alpha h^\beta f_{i+1,j+1}.$$

For this case, to represent Equation (5.9) in the following matrix form, let the mesh size be defined as shown in Figure 5.7.

$$D = \begin{bmatrix} A_7 & & & & \\ & A_7 & & & \\ & & \ddots & & \\ & & & A_7 & \\ & & & & A_7 \end{bmatrix}, \quad L = \begin{bmatrix} H_2 & 0 \\ H_1 & H_2 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & H_3 \\ 0 & 0 \end{bmatrix},$$

$$H_1 = \begin{bmatrix} J_1 & J_3 & & & \\ Y_2 & J_2 & J_3 & & \\ & Y_2 & J_1 & \ddots & \\ Y_6 & & \ddots & \ddots & \ddots \\ & \ddots & & \ddots & J_1 & J_3 \\ Y_{10} & & \ddots & & Y_2 & J_2 & J_3 \\ & Y_{10} & & Y_6 & Y_2 & J_1 \end{bmatrix}, \quad b = \begin{bmatrix} G_1 \\ G_1 \\ \vdots \\ G_1 \\ G_1 \end{bmatrix},$$

$$H_2 = \begin{bmatrix} J_4 & & & & \\ & J_4 & & & \\ Y_4 & & J_4 & & \\ & Y_4 & & \ddots & \\ Y_8 & & \ddots & \ddots & \\ & \ddots & & \ddots & J_4 \\ Y_{12} & & \ddots & Y_4 & J_4 \\ & Y_{12} & Y_8 & Y_4 & J_4 \end{bmatrix}, \quad G_1 = h^\alpha h^\beta \begin{bmatrix} f_{i,j} \\ f_{i+1,j} \\ f_{i,j+1} \\ f_{i+1,j+1} \end{bmatrix},$$

$$H_3 = \begin{bmatrix} J_2 & J_3 & & & & \\ Y_2 & J_1 & J_3 & & & \\ & Y_2 & J_2 & \ddots & & \\ Y_6 & & \ddots & \ddots & \ddots & \\ & \ddots & & \ddots & J_2 & J_3 \\ Y_{10} & & \ddots & & Y_2 & J_1 & J_3 \\ & Y_{10} & & Y_6 & Y_2 & J_2 \end{bmatrix},$$

$$J_1 = \begin{bmatrix} KB_2 & ZB \\ KB_6 & KB_2 & ZB \\ KB_{10} & KB_6 & KB_2 & ZB \end{bmatrix}, \quad J_2 = \begin{bmatrix} ZB \\ KB_2 & ZB \\ KB_6 & KB_2 & ZB \\ KB_{10} & KB_6 & KB_2 \end{bmatrix},$$

$$J_3 = \begin{bmatrix} ZA \\ & ZA \\ & & \ddots \\ & & & ZA \\ & & & & ZA \end{bmatrix}, \quad J_4 = \begin{bmatrix} KB_4 \\ KB_8 & KB_4 \\ KB_{12} & KB_8 & KB_4 \end{bmatrix},$$

$$ZA = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -h^\alpha & 0 & 0 & 0 \\ 0 & -h^\alpha & 0 & 0 \end{bmatrix}, \quad ZB = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -h^\beta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -h^\beta & 0 \end{bmatrix},$$

$$Y_i = \begin{bmatrix} KA_i & & & & \\ & KA_i & & & \\ & & \ddots & & \\ & & & KA_i & \\ & & & & KA_i \end{bmatrix},$$

$$KA_i = \begin{bmatrix} -h^\alpha g_{(\beta,i+1)} & 0 & -h^\alpha g_{(\beta,i)} & 0 \\ 0 & -h^\alpha g_{(\beta,i+1)} & 0 & -h^\alpha g_{(\beta,i)} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$KB_i = \begin{bmatrix} -h^\beta g_{(\alpha,i+1)} & -h^\beta g_{(\alpha,i)} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -h^\beta g_{(\alpha,i+1)} & -h^\beta g_{(\alpha,i)} \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential FEG MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all groups of $\bigcirc$ points using

$$\bigcirc \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix}^{k+1} = \frac{\theta_1}{Q_7} A_7 \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \\ S_3^{k+1} - S_3^k \\ 0 \end{bmatrix} + \frac{\omega_1}{Q_7} A_7 \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}^k + (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix}^k. \quad (5.10)$$

3. Execute the remaining groups of two $\bigcirc$ points at $i = M - 1$ using

$$\begin{aligned} \bigcirc \begin{bmatrix} u_{i,j} \\ u_{i,j+1} \end{bmatrix}^{k+1} &= \frac{\theta_1}{T1} \begin{bmatrix} P & h^\alpha \\ h^\alpha g_{(\beta,2)} & P \end{bmatrix} \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_3^{k+1} - S_3^k \end{bmatrix} \\ &+ \frac{\omega_1}{T1} \begin{bmatrix} P & h^\alpha \\ h^\alpha g_{(\beta,2)} & P \end{bmatrix} \begin{bmatrix} S_1 + h^\beta u_{i+1,j} \\ S_3 + h^\beta u_{i+1,j+1} \end{bmatrix}^k \\ &+ (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i,j+1} \end{bmatrix}^k, \end{aligned} \quad (5.11)$$

where $T1 = P^2 - g_{(\beta,2)}(h^\alpha)^2$, and at $j = M - 1$ using

$$\begin{aligned} \bigcirc \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \end{bmatrix}^{k+1} &= \frac{\theta_1}{T2} \begin{bmatrix} P & h^\beta \\ h^\beta g_{(\alpha,2)} & P \end{bmatrix} \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \end{bmatrix} \\ &+ \frac{\omega_1}{T2} \begin{bmatrix} P & h^\beta \\ h^\beta g_{(\alpha,2)} & P \end{bmatrix} \begin{bmatrix} S_1 + h^\alpha u_{i,j+1} \\ S_2 + h^\alpha u_{i+1,j+1} \end{bmatrix}^k \\ &+ (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \end{bmatrix}^k, \end{aligned} \quad (5.12)$$

where $T2 = P^2 - g_{(\alpha,2)}(h^\beta)^2$. Finally, execute the remaining $\bigcirc$ point at $i, j = M - 1$ using Equation (5.1).

4. Execute all groups of ● points using

$$\bullet \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix}^{k+1} = \frac{\theta_2}{Q_7} A_7 \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \\ S_3^{k+1} - S_3^k \\ S_4^{k+1} - S_4^k \end{bmatrix} + \frac{\omega_2}{Q_7} A_7 \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}^k \quad (5.13)$$

$$+ (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{bmatrix}^k.$$

5. Execute the remaining groups of two ● points at $i = M - 1$ using

$$\bullet \begin{bmatrix} u_{i,j} \\ u_{i,j+1} \end{bmatrix}^{k+1} = \frac{\theta_2}{T_1} \begin{bmatrix} P & h^\alpha \\ h^\alpha g_{(\beta,2)} & P \end{bmatrix} \begin{bmatrix} S_1^{k+1} - S_1^k + h^\beta v_{i+1,j} \\ S_3^{k+1} - S_3^k + h^\beta v_{i+1,j+1} \end{bmatrix}$$

$$+ \frac{\omega_2}{T_1} \begin{bmatrix} P & h^\alpha \\ h^\alpha g_{(\beta,2)} & P \end{bmatrix} \begin{bmatrix} S_1 + h^\beta u_{i+1,j} \\ S_3 + h^\beta u_{i+1,j+1} \end{bmatrix}^k \quad (5.14)$$

$$+ (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i,j+1} \end{bmatrix}^k,$$

and at $j = M - 1$ using

$$\begin{aligned}
\bullet \quad \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \end{bmatrix}^{k+1} &= \frac{\theta_2}{T^2} \begin{bmatrix} P & h^\beta \\ h^\beta g_{(\alpha,2)} & P \end{bmatrix} \begin{bmatrix} S_1^{k+1} - S_1^k + h^\alpha v_{i,j+1} \\ S_2^{k+1} - S_2^k + h^\alpha v_{i+1,j+1} \end{bmatrix} \\
&+ \frac{\omega_2}{T^2} \begin{bmatrix} P & h^\beta \\ h^\beta g_{(\alpha,2)} & P \end{bmatrix} \begin{bmatrix} S_1 + h^\alpha u_{i,j+1} \\ S_2 + h^\alpha u_{i+1,j+1} \end{bmatrix}^k \\
&+ (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i+1,j} \end{bmatrix}^k, \quad (5.15)
\end{aligned}$$

6. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, go to step (5), otherwise, repeat the iteration cycle (i. e. go to step (2)).

7. Stop.

Sequential FEDG MAOR

The derivation of the FEDG iterative method is based on extending the fractional iterative schemes to encompass a complete group of two points arranged in a (2×2) configuration. This approach leverages Equation (5.3), enabling simultaneous updates to a cluster of grid points during each iteration. The structure of this method ensures efficient utilisation of computational resources by processing grouped data in parallel while maintaining the accuracy of fractional derivative calculations. Hence, the FEDG iterative method can be generally expressed as

$$\begin{bmatrix} P & -h^\beta \\ -g_{(\alpha,2)}h^\beta & P \end{bmatrix} \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix},$$

where

$$\begin{aligned}
S_1 &= h^\alpha u_{i-1,j+1} + h^\beta \sum_{s=2,3,4,\dots}^{i+1} g_{(\alpha,s)} \cdot u_{i-s+1,j-s+1} \\
&\quad + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g_{(\beta,z)} \cdot u_{i+z-1,j-z+1} - h^\alpha h^\beta f_{i,j}, \\
S_2 &= h^\alpha u_{i,j+2} + h^\beta u_{i+2,j+2} + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g_{(\beta,z)} \cdot u_{i+z-1,j-z+1} \\
&\quad - h^\alpha h^\beta f_{i+1,j+1}.
\end{aligned}$$

By establishing the inverse matrix of the coefficient matrix and manipulating it, the equation above can be rewritten as

$$\begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix} = \frac{1}{Q_8} \begin{bmatrix} P & h^\beta \\ g_{(\alpha,2)} h^\beta & P \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}, \quad (5.16)$$

where $Q_8 = P^2 - g_{(\alpha,2)}(h^\beta)^2$.

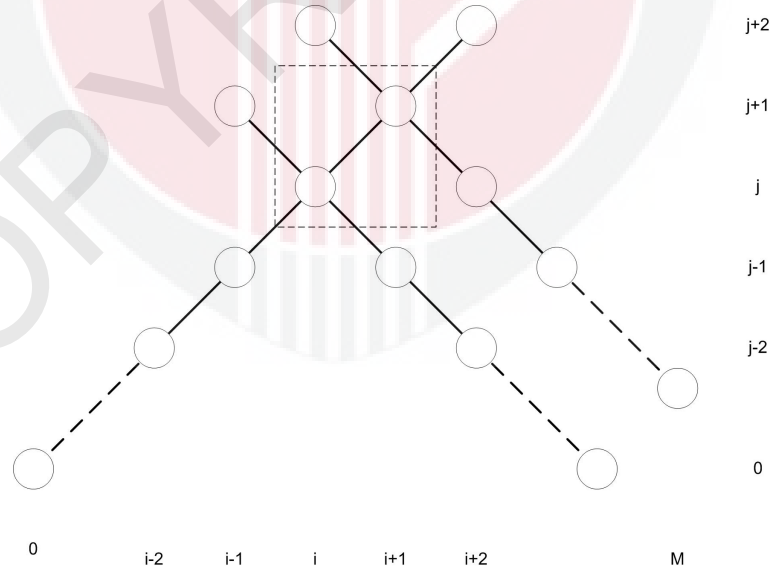


Figure 5.8: The stencil for FEDG.

Figure 5.9 shows the solution domain for FEDG with the horizontal zebra line (HZL) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

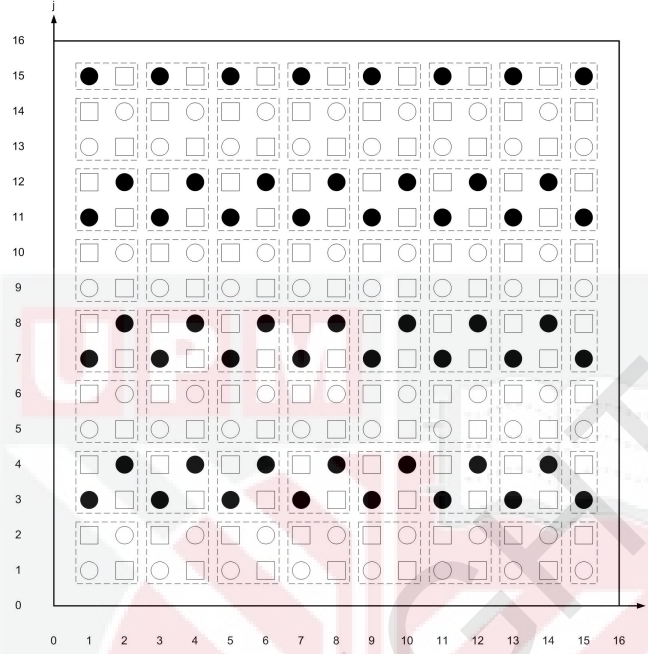


Figure 5.9: The solution domain for FEDG.

Let

$$A_8 = \begin{bmatrix} P & h^\beta \\ g_{(\alpha,2)}h^\beta & P \end{bmatrix},$$

and

$$\begin{aligned} S_1^k &= h^\alpha u_{i-1,j-1}^k + h^\beta \sum_{s=2,3,4,\dots}^{i+1} g_{(\alpha,s)} \cdot u_{i-s+1,j-s+1}^k \\ &\quad + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g_{(\beta,z)} \cdot u_{i+z-1,j-z+1}^k - h^\alpha h^\beta f_{i,j}, \\ S_2^k &= h^\alpha u_{i,j+2}^k + h^\beta u_{i+2,j+2}^k + h^\alpha \sum_{z=2,3,4,\dots}^{j+1} g_{(\beta,z)} \cdot u_{i+z-1,j-z+1}^k \\ &\quad - h^\alpha h^\beta f_{i+1,j+1}. \end{aligned}$$

For this case, to represent Equation (5.16) in the following matrix form, let the mesh size be defined as shown in Figure 5.9.

$$\begin{aligned}
 D &= \begin{bmatrix} A_8 & & & & \\ & A_8 & & & \\ & & \ddots & & \\ & & & A_8 & \\ & & & & A_8 \end{bmatrix}, \quad L = \begin{bmatrix} H_2 & 0 \\ H_1 & H_3 \end{bmatrix}, \quad U = \begin{bmatrix} H_5 & H_4 \\ 0 & H_6 \end{bmatrix}, \\
 H_1 &= \begin{bmatrix} J_2 & J_1 \\ J_3 & J_2 & J_1 \\ J_4 & J_3 & J_2 & J_1 \end{bmatrix}, \quad H_2 = \begin{bmatrix} J_5 \\ J_6 & J_5 \\ J_7 & J_6 & J_5 \\ J_8 & J_7 & J_6 & J_5 \end{bmatrix}, \quad H_3 = \begin{bmatrix} J_5 \\ J_6 & J_5 \\ J_7 & J_6 & J_5 \end{bmatrix}, \\
 H_4 &= \begin{bmatrix} J_1 \\ J_2 & J_1 \\ J_3 & J_2 & J_1 \\ J_4 & J_3 & J_2 \end{bmatrix}, \quad H_5 = \begin{bmatrix} J_9 \\ J_9 \\ J_9 \\ J_9 \end{bmatrix}, \quad H_6 = \begin{bmatrix} J_9 \\ J_9 \\ J_9 \end{bmatrix}, \\
 J_1 &= \begin{bmatrix} ZA & ZB \\ & ZA & ZB \\ & & ZA & ZB \\ & & & ZA & ZB \\ & & & & ZA & ZB \\ & & & & & ZA \end{bmatrix}, \quad b = \begin{bmatrix} G_1 \\ G_1 \\ \vdots \\ G_1 \\ G_1 \end{bmatrix},
 \end{aligned}$$

$$J_2 = \begin{bmatrix} KA1_2 & KA2_3 & KA1_4^T & & & & \\ KB_2 & KA1_2 & KA2_3 & KA1_4^T & & & \\ & KB_2 & KA1_2 & KA2_3 & KA1_4^T & & \\ & & KB_2 & KA1_2 & KA2_3 & KA1_4^T & \\ & & & KB_2 & KA1_2 & KA2_3 & KA1_4^T \\ & & & & KB_2 & KA1_2 & KA2_3 \\ & & & & & KB_2 & KA1_2 \end{bmatrix},$$

$$J_3 = \begin{bmatrix} & KA1_6 & KA2_7 & KA1_8^T & & & \\ & & KA1_6 & KA2_7 & KA1_8^T & & \\ & & & KA1_6 & KA2_7 & KA1_8^T & \\ KB_6 & & & & KA1_6 & KA2_7 & \\ & KB_6 & & & & & KA1_6 \\ & & KB_6 & & & & \\ & & & KB_6 & & & \end{bmatrix},$$

$$J_4 = \begin{bmatrix} & & KA1_{10} & KA2_{11} & KA1_{12}^T & \\ & & & KA1_{10} & KA2_{11} & \\ & & & & KA1_{10} & \\ KB_6 & & & & & \\ & KB_6 & & & & \end{bmatrix},$$

$$J_5 = \begin{bmatrix} ZA^T & & & & \\ & ZA^T & & & \\ & & ZA^T & & \\ & & & ZA^T & \\ & & & & ZA^T \end{bmatrix},$$

$$J_6 = \begin{bmatrix} KA1_4 & KA2_5 & KA1_6^T & & \\ & KA1_4 & KA2_5 & KA1_6^T & \\ KB_4 & & KA1_4 & KA2_5 & KA1_6^T \\ & KB_4 & & KA1_4 & KA2_5 & KA1_6^T \\ & & KB_4 & & KA1_4 & KA2_5 \\ & & & KB_4 & & KA1_4 \\ & & & & KB_4 & \end{bmatrix},$$

$$J_7 = \begin{bmatrix} & & KA1_8 & KA2_9 & KA1_{10}^T & \\ & & & KA1_8 & KA2_9 & KA1_{10}^T \\ & & & & KA1_8 & KA2_9 \\ & KB_8 & & & & KA1_8 \\ & & KB_8 & & & \\ & & & KB_8 & & \end{bmatrix},$$

$$J_8 = \begin{bmatrix} & KA_{12} & KA_{23} \\ & KA_{12} & \\ KB_{12} & & \end{bmatrix}, \quad G_1 = h^\alpha h^\beta \begin{bmatrix} f_{i,j} \\ f_{i+1,j+1} \end{bmatrix},$$

$$J_9 = \begin{bmatrix} KA_{12}^T & & & & \\ & KA_{12}^T & & & \\ & & KA_{12}^T & & \\ & & & KA_{12}^T & \\ & & & & KA_{12}^T \end{bmatrix},$$

$$ZA = \begin{bmatrix} 0 \\ -h^\alpha & 0 \end{bmatrix}, \quad ZB = \begin{bmatrix} 0 \\ -h^\beta & 0 \end{bmatrix}, \quad KB_i = \begin{bmatrix} -h^\beta g_{(\alpha,i+1)} & -h^\beta g_{(\alpha,i)} \\ 0 & 0 \end{bmatrix},$$

$$KA_{1i} = \begin{bmatrix} 0 & -h^\alpha g_{(\beta,i)} \\ 0 & 0 \end{bmatrix}, \quad KA_{2i} = \begin{bmatrix} -h^\alpha g_{(\beta,i)} & 0 \\ 0 & -h^\alpha g_{(\beta,i)} \end{bmatrix},$$

Thus, by implementing the above matrices into Equation (2.19), the sequential FEDG MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.
2. Execute all groups of $\odot$ points using

$$\begin{aligned} \bigcirc \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix}^{k+1} &= \frac{\theta_1}{Q_8} A_8 \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \end{bmatrix} + \frac{\omega_1}{Q_8} A_8 \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}^k \\ &+ (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix}^k. \end{aligned} \quad (5.17)$$

3. Execute the remaining $\bigcirc$ points at $i = M - 1$ using Equation (5.4).

4. Execute all groups of $\bullet$ points using

$$\begin{aligned} \bullet \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix}^{k+1} &= \frac{\theta_2}{Q_8} A_8 \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \end{bmatrix} + \frac{\omega_2}{Q_8} A_8 \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}^k \\ &+ (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i+1,j+1} \end{bmatrix}^k. \end{aligned} \quad (5.18)$$

5. Execute the remaining $\bullet$ points at $i = M - 1$ and $j = M - 1$ using Equation (5.5).

6. Check for convergence where $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, evaluate the solutions of the remaining $\square$ points using Equation (3.5). Otherwise, repeat the iteration cycle (i. e. go to step (2)).

7. Stop.

Sequential FMEG MAOR

The derivation of the FMEG iterative method is based on extending the fractional iterative schemes to encompass a complete group of four points arranged in a (4×4) configuration. This approach leverages Equation (5.6), enabling simultaneous updates to a cluster of grid points during each iteration. The structure of this method ensures efficient utilisation of computational resources by processing grouped data in parallel while maintaining the accuracy of fractional derivative calculations. Hence, the FMEG

iterative method can be generally expressed as

$$\begin{bmatrix} P & -h^\beta & -h^\alpha & 0 \\ -h^\beta g_{(\alpha,2)} & P & 0 & -h^\alpha \\ -h^\alpha g_{(\beta,2)} & 0 & P & -h^\beta \\ 0 & -h^\alpha g_{(\beta,2)} & -h^\beta g_{(\alpha,2)} & P \end{bmatrix} \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix},$$

where

$$\begin{aligned} S_1 &= h^\beta \sum_{s=4,6,8,\dots}^{i+2} g_{(\alpha, \frac{s}{2})} \cdot u_{i-s+2,j} + h^\alpha \sum_{z=4,6,8,\dots}^{j+2} g_{(\beta, \frac{z}{2})} \cdot u_{i,j-z+2} - h^\alpha h^\beta f_{i,j}, \\ S_2 &= h^\beta u_{i+4,j} + h^\alpha \sum_{z=4,6,8,\dots}^{j+2} g_{(\beta, \frac{z}{2})} \cdot u_{i+2,j-z+2} - h^\alpha h^\beta f_{i+2,j}, \\ S_3 &= h^\beta \sum_{s=4,6,8,\dots}^{i+2} g_{(\alpha, \frac{s}{2})} \cdot u_{i-s+2,j+2} + h^\alpha u_{i,j+4} - h^\alpha h^\beta f_{i,j+2}, \\ S_4 &= h^\beta u_{i+4,j+2} + h^\alpha u_{i+2,j+4} - h^\alpha h^\beta f_{i+2,j+2}. \end{aligned}$$

By establishing the inverse matrix of the coefficient matrix and manipulating it, the equation above can be rewritten as

$$\begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix} = \frac{1}{Q_9} \begin{bmatrix} PR_1 & R_2 & R_3 & R_4 \\ g_{(\alpha,2)}R_2 & PR_1 & g_{(\alpha,2)}R_4 & R_3 \\ g_{(\beta,2)}R_3 & g_{(\beta,2)}R_4 & PR_1 & R_2 \\ g_{(\alpha,2)}g_{(\beta,2)}R_4 & g_{(\beta,2)}R_3 & g_{(\alpha,2)}R_2 & PR_1 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}, \quad (5.19)$$

where $Q_9 = g_{(\alpha,2)}^2(h^\beta)^4 - 2g_{(\alpha,2)}g_{(\beta,2)}(h^\alpha)^2(h^\beta)^2 + g_{(\beta,2)}^2(h^\alpha)^4 - 2P^2g_{(\alpha,2)}(h^\beta)^2 - 2P^2g_{(\beta,2)}(h^\alpha)^2 + P^4$ and

$$\begin{aligned} R_1 &= P^2 - g_{(\alpha,2)}(h^\beta)^2 - g_{(\beta,2)}^2(h^\alpha)^2, & R_3 &= h^\alpha(2g_{(\alpha,2)}(h^\beta)^2 + R_1), \\ R_2 &= h^\beta(2g_{(\beta,2)}(h^\alpha)^2 + R_1), & R_4 &= 2h^\alpha h^\beta P. \end{aligned}$$

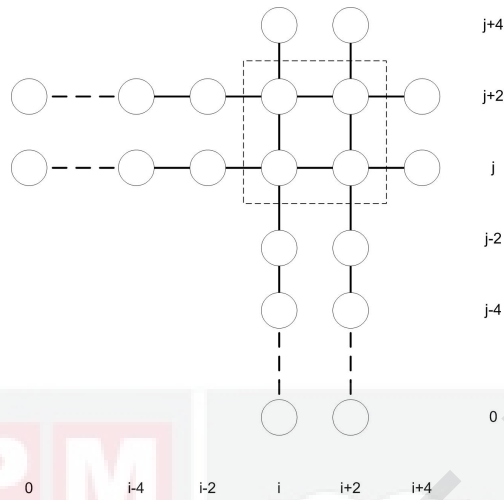


Figure 5.10: The stencil for FMEG.

Figure 5.11 shows the solution domain for FMEG with the chessboard (CB) ordering strategy. This ordering strategy ensures that the points solved in the iterative process are independent from each other.

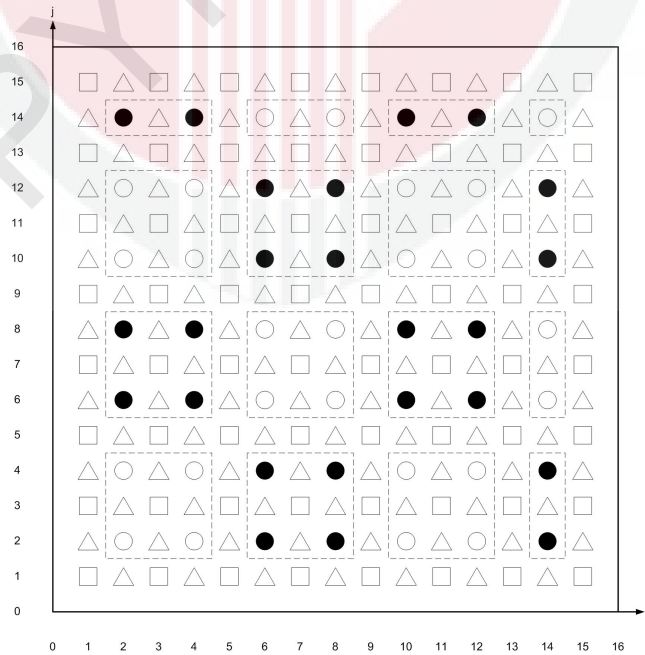


Figure 5.11: The solution domain for FMEG.

Let

$$A_9 = \begin{bmatrix} PR_1 & R_2 & R_3 & R_4 \\ g_{(\alpha,2)}R_2 & PR_1 & g_{(\alpha,2)}R_4 & R_3 \\ g_{(\beta,2)}R_3 & g_{(\beta,2)}R_4 & PR_1 & R_2 \\ g_{(\alpha,2)}g_{(\beta,2)}R_4 & g_{(\beta,2)}R_3 & g_{(\alpha,2)}R_2 & PR_1 \end{bmatrix},$$

and

$$\begin{aligned} S_1^k &= h^\beta \sum_{s=4,6,8,\dots}^{i+2} g_{(\alpha, \frac{s}{2})} \cdot u_{i-s+2,j}^k + h^\alpha \sum_{z=4,6,8,\dots}^{j+2} g_{(\beta, \frac{z}{2})} \cdot u_{i,j-z+2}^k - h^\alpha h^\beta f_{i,j}, \\ S_2^k &= h^\beta u_{i+4,j}^k + h^\alpha \sum_{z=4,6,8,\dots}^{j+2} g_{(\beta, \frac{z}{2})} \cdot u_{i+2,j-z+2}^k - h^\alpha h^\beta f_{i+2,j}, \\ S_3^k &= h^\beta \sum_{s=4,6,8,\dots}^{i+2} g_{(\alpha, \frac{s}{2})} \cdot u_{i-s+2,j+2}^k + h^\alpha u_{i,j+4}^k - h^\alpha h^\beta f_{i,j+2}, \\ S_4^k &= h^\beta u_{i+4,j+2}^k + h^\alpha u_{i+2,j+4}^k - h^\alpha h^\beta f_{i+2,j+2}. \end{aligned}$$

For this case, to represent Equation (5.19) in the following matrix form, let the mesh size be defined as shown in Figure 5.11.

$$D = \begin{bmatrix} A_9 & & & & \\ & A_9 & & & \\ & & \ddots & & \\ & & & A_9 & \\ & & & & A_9 \end{bmatrix}, \quad L = \begin{bmatrix} H_2 & 0 \\ H_1 & H_3 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & H_4 \\ 0 & 0 \end{bmatrix},$$

$$H_1 = \begin{bmatrix} KB_2 & J_2 & J_1 & 0 & 0 \\ KA_2 & 0 & J_2 & J_1 & 0 \\ 0 & KA_2 & KB_2 & 0 & J_1 \\ 0 & 0 & KA_2 & KB_2 & J_2 \end{bmatrix}, \quad b = \begin{bmatrix} G_1 \\ G_1 \\ \vdots \\ G_1 \\ G_1 \end{bmatrix},$$

$$H_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ KB_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ KA_4 & 0 & 0 & 0 & 0 \\ 0 & KA_4 & 0 & KB_4 & 0 \end{bmatrix}, \quad H_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & KB_4 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$H_4 = \begin{bmatrix} J_2 & J_1 & 0 & 0 \\ KB_2 & 0 & J_1 & 0 \\ KA_2 & KB_2 & J_2 & J_1 \\ 0 & KA_2 & 0 & J_2 \\ 0 & 0 & KA_2 & KB_2 \end{bmatrix}, \quad G_1 = h^\alpha h^\beta \begin{bmatrix} f_{i,j} \\ f_{i+2,j} \\ f_{i,j+2} \\ f_{i+2,j+2} \end{bmatrix},$$

$$J_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ h^\alpha & 0 & 0 & 0 \\ 0 & h^\alpha & 0 & 0 \end{bmatrix}, \quad J_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ h^\beta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & h^\beta & 0 \end{bmatrix},$$

$$KA_i = \begin{bmatrix} h^\alpha g_{(\beta,i+1)} & 0 & h^\alpha g_{(\beta,i)} & 0 \\ 0 & h^\alpha g_{(\beta,i+1)} & 0 & h^\alpha g_{(\beta,i)} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$KB_i = \begin{bmatrix} h^\beta g_{(\alpha,i+1)} & h^\beta g_{(\alpha,i)} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & h^\beta g_{(\alpha,i+1)} & h^\beta g_{(\alpha,i)} \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Thus, by implementing the above matrices into Equation (2.19), the sequential FMEG MAOR iterative method can be derived as follows

1. Set the initial values for all matrices and relaxation parameter.

2. Execute all groups of $\bigcirc$ points using

$$\begin{aligned} \bigcirc \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix}^{k+1} &= \frac{\theta_1}{Q_9} A_9 \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \\ S_3^{k+1} - S_3^k \\ 0 \end{bmatrix} + \frac{\omega_1}{Q_9} A_9 \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}^k \\ &+ (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix}^k. \end{aligned} \quad (5.20)$$

3. Execute the remaining groups of two $\bigcirc$ points at $i = M - 2$ using

$$\begin{aligned} \bigcirc \begin{bmatrix} u_{i,j} \\ u_{i,j+2} \end{bmatrix}^{k+1} &= \frac{\theta_1}{T1} \begin{bmatrix} P & h^\alpha \\ h^\alpha g_{(\beta,2)} & P \end{bmatrix} \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_3^{k+1} - S_3^k \end{bmatrix} \\ &+ \frac{\omega_1}{T1} \begin{bmatrix} P & h^\alpha \\ h^\alpha g_{(\beta,2)} & P \end{bmatrix} \begin{bmatrix} S_1 + h^\beta u_{i+2,j} \\ S_3 + h^\beta u_{i+2,j+2} \end{bmatrix}^k \\ &+ (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i,j+2} \end{bmatrix}^k, \end{aligned} \quad (5.21)$$

where $T1 = P^2 - g_{(\beta,2)}(h^\alpha)^2$, and at $j = M - 2$ using

$$\begin{aligned}
\bigcirc \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \end{bmatrix}^{k+1} &= \frac{\theta_1}{T2} \begin{bmatrix} P & h^\beta \\ h^\beta g_{(\alpha,2)} & P \end{bmatrix} \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \end{bmatrix} \\
&+ \frac{\omega_1}{T2} \begin{bmatrix} P & h^\beta \\ h^\beta g_{(\alpha,2)} & P \end{bmatrix} \begin{bmatrix} S_1 + h^\alpha u_{i,j+2} \\ S_2 + h^\alpha u_{i+2,j+2} \end{bmatrix}^k \\
&+ (1 - \omega_1) \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \end{bmatrix}^k, \tag{5.22}
\end{aligned}$$

where $T2 = P^2 - g_{(\alpha,2)}(h^\beta)^2$. Finally, execute the remaining $\bigcirc$ point at $i, j = M - 2$ using Equation (5.7)

4. Execute all groups of $\bullet$ points using

$$\begin{aligned}
\bullet \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix}^{k+1} &= \frac{\theta_2}{Q_9} A_9 \begin{bmatrix} S_1^{k+1} - S_1^k \\ S_2^{k+1} - S_2^k \\ S_3^{k+1} - S_3^k \\ S_4^{k+1} - S_4^k \end{bmatrix} + \frac{\omega_2}{Q_9} A_9 \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}^k \\
&+ (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \\ u_{i,j+2} \\ u_{i+2,j+2} \end{bmatrix}^k. \tag{5.23}
\end{aligned}$$

5. Execute the remaining groups of two $\bullet$ points at $i = M - 2$ using

$$\begin{aligned}
\bullet \quad \begin{bmatrix} u_{i,j} \\ u_{i,j+2} \end{bmatrix}^{k+1} &= \frac{\theta_2}{T_1} \begin{bmatrix} P & h^\alpha \\ h^\alpha g_{(\beta,2)} & P \end{bmatrix} \begin{bmatrix} S_1^{k+1} - S_1^k + h^\beta v_{i+2,j} \\ S_3^{k+1} - S_3^k + h^\beta v_{i+2,j+2} \end{bmatrix} \\
&+ \frac{\omega_2}{T_1} \begin{bmatrix} P & h^\alpha \\ h^\alpha g_{(\beta,2)} & P \end{bmatrix} \begin{bmatrix} S_1 + h^\beta u_{i+2,j} \\ S_3 + h^\beta u_{i+2,j+2} \end{bmatrix}^k \\
&+ (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i,j+2} \end{bmatrix}^k, \quad (5.24)
\end{aligned}$$

and at $j = M - 2$ using

$$\begin{aligned}
\bullet \quad \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \end{bmatrix}^{k+1} &= \frac{\theta_2}{T_2} \begin{bmatrix} P & h^\beta \\ h^\beta g_{(\alpha,2)} & P \end{bmatrix} \begin{bmatrix} S_1^{k+1} - S_1^k + h^\alpha v_{i,j+2} \\ S_2^{k+1} - S_2^k + h^\alpha v_{i+2,j+2} \end{bmatrix} \\
&+ \frac{\omega_2}{T_2} \begin{bmatrix} P & h^\beta \\ h^\beta g_{(\alpha,2)} & P \end{bmatrix} \begin{bmatrix} S_1 + h^\alpha u_{i,j+2} \\ S_2 + h^\alpha u_{i+2,j+2} \end{bmatrix}^k \\
&+ (1 - \omega_2) \begin{bmatrix} u_{i,j} \\ u_{i+2,j} \end{bmatrix}^k, \quad (5.25)
\end{aligned}$$

6. Check for convergence $|u^{k+1} - u^k| \leq \varepsilon$. If the iterative process converges, evaluate the solutions of the remaining $\square$ points using Equation (3.8) followed by the remaining $\triangle$ points using Equation (3.9). Otherwise, repeat the iteration cycle (i. e. go to step (2)).

7. Stop.

5.3 Construction of Fractional Order Parallel Algorithm

From Equations (5.1), (5.4), (5.7), (5.10), (5.17) and (5.20), it is clear that all the $\circ$ tasks in $u_{i,j}^{k+1}$ are independent of each other and can be computed in parallel. After that, all the $\bullet$ tasks of Equations (5.2), (5.5), (5.8), (5.13), (5.18) and (5.23) can be

calculated simultaneously using the updated $\odot$ values since these calculations are also independent. At the end of each stage, a synchronization call is executed to ensure the updated values are used in the subsequent iteration. Each processor independently iterates on its own task and then the convergence test is performed. If the convergence criteria is achieved, the FHS, FQS, FEDG and FMEG iterative methods will evaluate the remaining tasks directly. Otherwise increase the number of iteration and repeat the iteration cycle for all iterative methods.

Similar to the previous chapters, four number of processors are chosen as an example. The parallel algorithms for solving the fourth order MAOR iterative methods will be constructed using multicore processors using the OpenMP technology and C++ Compiler.

5.3.1 Parallel Fractional Order Point Iterative Methods with MAOR Scheme

Parallel FFS MAOR

Step 1: Solve the $\odot$ points. From Figure 5.1, assign each processor to a row of $\odot$ points ($P0 \rightarrow j = 1, P1 \rightarrow j = 2, P2 \rightarrow j = 3, P3 \rightarrow j = 4$). All processor will solve the $\odot$ points using Equation (5.1). After the processors have finish solving their respective rows, they will be assigned to the next row of $\odot$ points ($P0 \rightarrow j = 5, P1 \rightarrow j = 6, P2 \rightarrow j = 7, P3 \rightarrow j = 8$). This process will continue until there are no available row of $\odot$ points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 2: Solve the $\bullet$ points. From Figure 5.1, assign each processor to a row of $\bullet$ points ($P0 \rightarrow j = 1, P1 \rightarrow j = 2, P2 \rightarrow j = 3, P3 \rightarrow j = 4$). All processor will solve the $\bullet$ points using Equation (5.2). After the processors have finish solving the respective rows, they will be assigned to the next row of block ($P0 \rightarrow j = 5, P1 \rightarrow j = 6, P2 \rightarrow j = 7, P3 \rightarrow j = 8$). This process will continue until there are no available row of $\bullet$ points left to be solved. Then the `#pragma omp barrier` command is called to

synchronize all of the processors.

Step 3: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

These steps above can be summarized in Algorithm 13.

Algorithm 13: The iterative process for the parallel FFS MAOR iterative method.

```

1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into  $\bigcirc$  and  $\bullet$  as shown in Figure 5.1;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
4   Get thread number  $p$ , and maximum number of threads  $np$ ;
5   while  $|u^{k+1} - u^k| > \varepsilon$  do
6     #pragma omp parallel
7     {
8       for  $\{j = p + 1; j < M; j = j + np\}$  do
9         Compute all  $\bigcirc$  points using Equation (5.1);
10      end for
11      #pragma omp barrier
12      for  $\{j = p + 1; j < M; j = j + np\}$  do
13        Compute all  $\bullet$  points using Equation (5.2);
14      end for
15      #pragma omp barrier
16       $ite = ite + 1$ ;
17    }
18  end while
19  Display  $ite$ , error, and execution time;

```

Parallel FHS MAOR

Step 1: Solve the $\bigcirc$ points. From Figure 5.3, assign each processor to a $\bigcirc$ point at the first row ($P0 \rightarrow i = 1, P1 \rightarrow i = 3, P2 \rightarrow i = 5, P3 \rightarrow i = 7$). All processor will solve the $\bigcirc$ points using Equation (5.4). After the processors have finish solving their respective points, they will be assigned to the next $\bigcirc$ points ($P0 \rightarrow j = 9, P1 \rightarrow j = 11, P2 \rightarrow j = 13, P3 \rightarrow j = 15$) until there are no available $\bigcirc$ points left to be solved. This process will continue to the next rows of $\bigcirc$ points. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 2: Solve the ● points. From Figure 5.3, assign each processor to a ● point at the second row ($P0 \rightarrow j = 2$, $P1 \rightarrow j = 4$, $P2 \rightarrow j = 6$, $P3 \rightarrow j = 8$). All processor will solve the ● points using Equation (5.5). After the processors have finish solving their respective points, they will be assigned to the next ● points ($P0 \rightarrow j = 10$, $P1 \rightarrow j = 12$, $P2 \rightarrow j = 14$, $P3 \rightarrow j = 16$) until there are no available ● points left to be solved. This process will continue to the next rows of ● points. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 3: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

Step 4: Evaluate the solutions of the remaining □ points using Equation (3.5) directly.

These steps above can be summarized in Algorithm 14.

Parallel FQS MAOR

Step 1: Solve the ○ points. From Figure 5.5, assign each processor to a row of ○ points ($P0 \rightarrow j = 2$, $P1 \rightarrow j = 4$, $P2 \rightarrow j = 6$, $P3 \rightarrow j = 8$). All processor will solve the ○ points using Equation (5.7). After the processors have finish solving their respective rows, they will be assigned to the next row of ○ points ($P0 \rightarrow j = 10$, $P1 \rightarrow j = 12$, $P2 \rightarrow j = 14$, $P3 \rightarrow j = 16$). This process will continue until there are no available row of ○ points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 2: Solve the ● points. From Figure 5.5, assign each processor to a row of ● points ($P0 \rightarrow j = 2$, $P1 \rightarrow j = 4$, $P2 \rightarrow j = 6$, $P3 \rightarrow j = 8$). All processor will solve the ● points using Equation (5.8). After the processors have finish solving the respective rows, they will be assigned to the next row of block ($P0 \rightarrow j = 10$, $P1 \rightarrow j = 12$, $P2 \rightarrow j = 14$, $P3 \rightarrow j = 16$). This process will continue until there

Algorithm 14: The iterative process for the parallel FHS MAOR iterative method.

```
1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into  $\circ$  and  $\bullet$  as shown in Figure 5.3;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
4   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
5   Get thread number  $p$ , and maximum number of threads  $np$ ;
6   while  $|u^{k+1} - u^k| > \varepsilon$  do
7     #pragma omp parallel
8     {
9       for  $\{j = 1; j < M; j = j + 2\}$  do
10        for  $\{i = 2p + 1; i < M; i = i + 2np\}$  do
11          Compute all  $\circ$  points using Equation (5.4);
12        end for
13      end for
14      #pragma omp barrier
15      for  $\{j = 2; j < M; j = j + 2\}$  do
16        for  $\{i = 2(p + 1); i < M; i = i + 2np\}$  do
17          Compute all  $\bullet$  points using Equation (5.5);
18        end for
19      end for
20      #pragma omp barrier
21       $ite = ite + 1$ ;
22    }
23  end while
24  Compute all  $\square$  points using Equation (3.5) directly;
25  Display  $ite$ , error, and execution time;
```

are no available row of $\bullet$ points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 3: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

Step 4: Evaluate the solutions of the remaining $\square$ points using Equation (3.8) and $\triangle$ points using Equation (3.9) directly.

These steps above can be summarized in Algorithm 15.

Algorithm 15: The iterative process for the parallel FQS MAOR iterative method.

```

1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into  $\bigcirc$  and  $\bullet$  as shown in Figure 5.5;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
4   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
5   Get thread number  $p$ , and maximum number of threads  $np$ ;
6   while  $|u^{k+1} - u^k| > \varepsilon$  do
7     #pragma omp parallel
8     {
9       for  $\{j = 2(p + 1); j < M; j = j + 2np\}$  do
10        Compute all  $\bigcirc$  points using Equation (5.7);
11      end for
12      #pragma omp barrier
13      for  $\{j = 2(p + 1); j < M; j = j + 2np\}$  do
14        Compute all  $\bullet$  points using Equation (5.8);
15      end for
16      #pragma omp barrier
17       $ite = ite + 1$ ;
18    }
19  end while
20  Compute all  $\square$  points using Equation (3.8) directly;
21  Compute all  $\triangle$  points using Equation (3.9) directly;
22  Display  $ite$ , error, and execution time;

```

5.3.2 Parallel Fractional Order Group Iterative Methods with MAOR Scheme

Parallel FEG MAOR

Step 1: Solve the group of four $\bigcirc$ points. From Figure 5.7, assign each processor to a row of block $\bigcirc$ points ($P0 \rightarrow j = 1, P1 \rightarrow j = 3, P2 \rightarrow j = 5, P3 \rightarrow j = 7$). All processor will solve the block of four $\bigcirc$ points using Equation (3.10). After the processors have finish solving their respective rows, they will be assigned to the next row of block $\bigcirc$ points ($P0 \rightarrow j = 9, P1 \rightarrow j = 11, P2 \rightarrow j = 13, P3 \rightarrow j = 15$). This process will continue until there are no available row of block left to be solved. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 2: Solve the remaining group of $\bigcirc$ points at $i = M - 1$ and $j = M - 1$. Here, only the main processor is utilized to solved these block of points. The block of $\bigcirc$ points at $i = M - 1$ is solved using Equation (5.11). Then the **#pragma omp**

barrier command is called to synchronize all of the processors.

Similarly, the block of $\circ$ points at $j = M - 1$ is solved using Equation (5.12). Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Finally, the remaining $\circ$ point at $i, j = M - 1$ is solved using Equation (5.1). Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 3: Solve the group of four $\bullet$ points. From Figure 5.7, assign each processor to a row of block $\bullet$ points ($P0 \rightarrow j = 1, P1 \rightarrow j = 3, P2 \rightarrow j = 5, P3 \rightarrow j = 7$). All processor will solve the block of four $\bullet$ points using Equation (5.13). After the processors have finish solving their respective rows, they will be assigned to the next row of block $\bullet$ points ($P0 \rightarrow j = 9, P1 \rightarrow j = 11, P2 \rightarrow j = 13, P3 \rightarrow j = 15$). This process will continue until there are no available row of block left to be solved. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 4: Solve the remaining group of $\bullet$ points at $i = M - 1$ and $j = M - 1$. Here, only the main processor is utilized to solved these block of points. The block of $\bullet$ points at $i = M - 1$ is solved using Equation (5.14). Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Similarly, the block of $\bullet$ points at $j = M - 1$ is solved using Equation (5.15). Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 5: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

These steps above can be summarized in Algorithm 16.

Algorithm 16: The iterative process for the parallel FEG MAOR iterative method.

```
1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into  $\bigcirc$  and  $\bullet$  as shown in Figure 5.7;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
4   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
5   Get thread number  $p$ , and maximum number of threads  $np$ ;
6   while  $|u^{k+1} - u^k| > \varepsilon$  do
7     #pragma omp parallel
8     {
9       for  $\{j = 2(p+1) - 1; j < M - 1; j = j + 2np\}$  do
10        Compute all groups of  $\bigcirc$  points using Equation (5.10);
11      end for
12      #pragma omp barrier
13       $i = M - 1$ ;
14      for  $\{j = 3; j < M - 1; j = j + 4\}$  do
15        Compute all groups of  $\bigcirc$  points using Equation (5.11);
16      end for
17      #pragma omp barrier
18       $j = M - 1$ ;
19      for  $\{i = 3; i < M - 1; i = i + 4\}$  do
20        Compute all groups of  $\bigcirc$  points using Equation (5.12);
21      end for
22      #pragma omp barrier
23       $i = M - 1, j = M - 1$ 
24      Compute  $\bigcirc$  point using Equation (5.1);
25      #pragma omp barrier
26      for  $\{j = 2(p+1) - 1; j < M - 1; j = j + 2np\}$  do
27        Compute all groups of  $\bullet$  points using Equation (5.13);
28      end for
29      #pragma omp barrier
30       $i = M - 1$ ;
31      for  $\{j = 1; j < M - 1; j = j + 4\}$  do
32        Compute all groups of  $\bullet$  points using Equation (5.14);
33      end for
34      #pragma omp barrier
35       $j = M - 1$ ;
36      for  $\{i = 1; i < M - 1; i = i + 4\}$  do
37        Compute all groups of  $\bullet$  points using Equation (5.15);
38      end for
39      #pragma omp barrier
40       $ite = ite + 1$ ;
41    }
42  end while
43  Display  $ite$ , error, and execution time;
```

Step 1: Solve the group of two $\bigcirc$ points and $\bigcirc$ points at $i = M - 1$. From Figure 5.9, assign each processor to a row of block $\bigcirc$ points ($P0 \rightarrow j = 1$, $P1 \rightarrow j = 5$, $P2 \rightarrow j = 9$, $P3 \rightarrow j = 13$). All processor will solve the block of four $\bigcirc$ points using Equation (5.17). However, if $i = M - 1$, the assigned processor will solve the $\bigcirc$ point using Equation (5.4). After the processors have finish solving their respective rows, they will be assigned to the next row of block $\bigcirc$ points ($P0 \rightarrow j = 17$, $P1 \rightarrow j = 21$, $P2 \rightarrow j = 25$, $P3 \rightarrow j = 29$). This process will continue until there are no available row of block left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 2: Solve the group of two $\bullet$ points and $\bullet$ points at $i = M - 1$. From Figure 5.9, assign each processor to a row of block $\bullet$ points ($P0 \rightarrow j = 3$, $P1 \rightarrow j = 7$, $P2 \rightarrow j = 11$, $P3 \rightarrow j = 15$). All processor will solve the block of four $\bullet$ points using Equation (5.18). However, if $i = M - 1$, the assigned processor will solve the $\bullet$ point using Equation (5.5). After the processors have finish solving their respective rows, they will be assigned to the next row $\bullet$ points ($P0 \rightarrow j = 19$, $P1 \rightarrow j = 23$, $P2 \rightarrow j = 27$, $P3 \rightarrow j = 31$). This process will continue until there are no available points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 3: Solve the remaining $\bullet$ points at $j = M - 1$. Each processor will be assigned to a $\bullet$ points consecutively ($P0 \rightarrow i = 1$, $P1 \rightarrow i = 3$, $P2 \rightarrow i = 5$, $P3 \rightarrow i = 7$). All processor will solve their respective $\bullet$ points using Equation (5.5). After the processors have finish solving their respective points, they will be assigned to the next $\bullet$ points consecutively ($P0 \rightarrow i = 9$, $P1 \rightarrow i = 11$, $P2 \rightarrow i = 13$, $P3 \rightarrow i = 15$). This process will continue until there are no available points left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 4: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

Step 5: Evaluate the solutions of the remaining $\square$ points using Equation (3.5) directly.

These steps above can be summarized in Algorithm 17.

Algorithm 17: The iterative process for the parallel FEDG MAOR iterative method.

```

1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into  $\circ$  and  $\bullet$  as shown in Figure 5.9;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
4   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
5   Get thread number  $p$ , and maximum number of threads  $np$ ;
6   while  $|u^{k+1} - u^k| > \varepsilon$  do
7     #pragma omp parallel
8     {
9       for  $\{j = 4(p+1) - 3; j < M - 1; j = j + 4np\}$  do
10        if  $i == M - 1$  then
11          Compute all  $\circ$  points using Equation (5.4);
12        else
13          Compute all groups of  $\circ$  points using Equation (5.17);
14        end if
15      end for
16      #pragma omp barrier
17       $i = M - 1$ ;
18      for  $\{j = 4(p+1) - 3; j < M - 1; j = j + 4np\}$  do
19        Compute all  $\circ$  points using Equation (3.3);
20      end for
21      #pragma omp barrier
22       $j = M - 1$ ;
23      for  $\{i = 2(p+1) - 1; i < M; i = i + 2np\}$  do
24        Compute all  $\circ$  points using Equation (3.3);
25      end for
26      #pragma omp barrier
27      for  $\{j = 4(p+1) - 1; j < M - 1; j = j + 4np\}$  do
28        Compute all groups of  $\bullet$  points using Equation (3.17);
29      end for
30      #pragma omp barrier

```

```

31
32      $i = M - 1;$ 
33     for  $\{j = 4(p + 1) - 1; j < M - 1; j = j + 4np\}$  do
34         Compute all  $\bullet$  points using Equation (3.4);
35     end for
36     #pragma omp barrier
37      $ite = ite + 1;$ 
38 }
39 end while
40 Compute all  $\square$  points using Equation (3.5) directly;
41 Display  $ite$ , error, and execution time;

```

Parallel FMEG MAOR

Step 1: Solve the block of four $\circ$ points. From Figure 5.11, assign each processor to a row of block $\circ$ points ($P0 \rightarrow j = 2, P1 \rightarrow j = 6, P2 \rightarrow j = 10, P3 \rightarrow j = 14$). All processor will solve the block of four $\circ$ points using Equation (5.20). After the processors have finish solving their respective rows, they will be assigned to the next row of block $\circ$ points ($P0 \rightarrow j = 18, P1 \rightarrow j = 22, P2 \rightarrow j = 26, P3 \rightarrow j = 30$). This process will continue until there are no available row of block left to be solved. Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 2: Solve the remaining group of $\circ$ points at $i = M - 2$ and $j = M - 2$. Here, only the main processor is utilized to solved these block of points. The block of $\circ$ points at $i = M - 2$ is solved using Equation (5.21). Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Similarly, the block of $\circ$ points at $j = M - 2$ is solved using Equation (5.22). Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Finally, the remaining $\circ$ point at $i, j = M - 2$ is solved using Equation (5.7). Then the **#pragma omp barrier** command is called to synchronize all of the processors.

Step 3: Solve the block of four ● points. From Figure 5.11, assign each processor to a row of block ● points ($P0 \rightarrow j = 2$, $P1 \rightarrow j = 6$, $P2 \rightarrow j = 10$, $P3 \rightarrow j = 14$). All processor will solve the block of four ● points using Equation (5.23). After the processors have finish solving their respective rows, they will be assigned to the next row of block ● points ($P0 \rightarrow j = 18$, $P1 \rightarrow j = 22$, $P2 \rightarrow j = 26$, $P3 \rightarrow j = 30$). This process will continue until there are no available row of block left to be solved. Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 4: Solve the remaining group of ● points at $i = M - 2$ and $j = M - 2$. Here, only the main processor is utilized to solved these block of points. The block of ● points at $i = M - 2$ is solved using Equation (5.24). Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Similarly, the block of ● points at $j = M - 2$ is solved using Equation (5.25). Then the `#pragma omp barrier` command is called to synchronize all of the processors.

Step 5: The convergence test is performed. The process ends if convergence is achieved, otherwise the iteration cycle is repeated (i. e. go to step 1).

Step 6: Evaluate the solutions of the remaining □ points using Equation (3.8) and △ points using Equation (3.9) directly.

These steps above can be summarized in Algorithm 18.

Algorithm 18: The iterative process for the parallel FMEG MAOR iterative method.

```
1 Begin
2   All points in the domain  $\mathbb{R}^2$  are defined into  $\circ$  and  $\bullet$  as shown in Figure 5.11;
3   The initial values for all matrices, weighted parameters  $\omega_1, \omega_2, \theta_1$  and  $\theta_2$ , and
4   error tolerance  $\varepsilon$  are set. Also, set the number of iterations  $ite = 0$ ;
5   Get thread number  $p$ , and maximum number of threads  $np$ ;
6   while  $|u^{k+1} - u^k| > \varepsilon$  do
7     #pragma omp parallel
8     {
9       for  $\{j = 4(p + 1) - 2; j < M; j = j + 4np\}$  do
10        Compute all groups of  $\circ$  points using Equation (5.20);
11      end for
12      #pragma omp barrier
13       $i = M - 2$ ;
14      for  $\{j = 6; j < M - 2; j = j + 8\}$  do
15        Compute all groups of  $\circ$  points using Equation (5.21);
16      end for
17      #pragma omp barrier
18       $j = M - 2$ ;
19      for  $\{i = 6; i < M - 2; i = i + 8\}$  do
20        Compute all groups of  $\circ$  points using Equation (5.22);
21      end for
22      #pragma omp barrier
23       $i = M - 2, j = M - 2$ 
24      Compute  $\circ$  point using Equation (5.7);
25      #pragma omp barrier
26      for  $\{j = 4(p + 1) - 2; j < M; j = j + 4np\}$  do
27        Compute all groups of  $\bullet$  points using Equation (5.23);
28      end for
29      #pragma omp barrier
30       $i = M - 1$ ;
31      for  $\{j = 4(p + 1) - 1; j < M - 1; j = j + 4np\}$  do
32        Compute all groups of  $\bullet$  points using Equation (3.14);
33      end for
34      #pragma omp barrier
35       $j = M - 1$ ;
36      for  $\{i = 4(p + 1) - 1; i < M - 1; i = i + 4np\}$  do
37        Compute all groups of  $\bullet$  points using Equation (3.15);
38      end for
39      #pragma omp barrier
40       $ite = ite + 1$ ;
41    }
42  end while
43  Display  $ite$ , error, and execution time;
```

5.4 Experimental Results and Discussions

Three problems will be tested in order to investigate the competency of the proposed method. The tests were performed on various sizes of M while the number of processors ranging from 1 to 8. The performance of the second order MAOR iterative methods in solving the problems are observed in terms of number of iterations, accuracy, execution time, speedup, parallel efficiency and computational costs. The computational cost for each iterative method with the relaxation schemes is measured by computing the total of arithmetic operations involved which is presented in Tables 5.1 until 5.4.

Problem 3 (Borhanifar and Valizadeh (2013))

$$\frac{\partial^\alpha u}{\partial x^\alpha} + \frac{\partial^\alpha u}{\partial y^\alpha} = \Gamma(\alpha + 1)(x^\alpha + y^\alpha),$$

where $\Omega = [0, 1] \times [0, 1]$, subject to the Dirichlet boundary conditions $u(x, 0) = 0$, $u(0, y) = 0$, $u(x, 1) = x^\alpha$, $u(0, 1) = y^\alpha$ and satisfying the exact solution $u(x, y) = (xy)^\alpha$. The test were performed on $M = 10, 20, 40, 80$, $\alpha = 1.25, 1.50, 1.75, 2.00$ with a tolerance of $\varepsilon = 10^{-10}$.

Problem 4 (Xie et al. (2017))

$$\frac{\partial^{\frac{4}{3}} u}{\partial x^{\frac{4}{3}}} + \frac{\partial^{\frac{3}{2}} u}{\partial y^{\frac{3}{2}}} = \frac{3x^{\frac{2}{3}}y(y-2)}{\Gamma(\frac{2}{3})} + \frac{4\sqrt{y}x(x-2)}{\sqrt{\pi}},$$

where $\Omega = [0, 2] \times [0, 2]$, subject to the homogeneous boundary conditions $u(x, 0) = u(0, y) = u(x, 2) = u(0, 2) = 0$ and satisfying the exact solution $u(x, y) = xy(x-2)(y-2)$. The test were performed on $M = 16, 32, 64, 128$ with a tolerance of $\varepsilon = 10^{-10}$.

Problem 5 (Heydari et al. (2013))

$$\frac{\partial^{\frac{3}{2}}u}{\partial x^{\frac{3}{2}}} + \frac{\partial^{\frac{4}{3}}u}{\partial y^{\frac{4}{3}}} = \frac{8x^{\frac{3}{2}}y^3(-5y + 5 + 8xy - 8x)}{\sqrt{5\pi}} + \frac{27y^{\frac{5}{3}}x^3(-2x + 2 + 3xy - 3y)}{10\Gamma(\frac{2}{3})},$$

where $\Omega = [0, 1] \times [0, 1]$, subject to the homogeneous boundary conditions $u(x, 0) = u(0, y) = u(x, 1) = u(0, 1) = 0$ and satisfying the exact solution $u(x, y) = x^3y^3(1 - x)(1 - y)$. The test were performed on $M = 16, 32, 64, 128$ with a tolerance of $\varepsilon = 10^{-10}$.

Tables 5.5 to 5.7 presents the number of iteration and accuracy of all iterative methods with various relaxation schemes for Problem 3. From the table, the MAOR scheme for all iterative methods reduces the number of iterations by 35.71 – 97.15% when compared with GS scheme, 0.00 – 57.37% when compared with SOR scheme, 0.00 – 55.53% when compared with MSOR scheme, and 0.00 – 13.33% when compared with AOR scheme. Similarly, Table 5.15 presents the number of iteration and accuracy of all iterative methods with various relaxation schemes for Problem 4. From the table, the MAOR scheme for all iterative methods also reduces the number of iterations by 50.43 – 80.94% when compared with GS scheme, 4.35 – 43.05% when compared with SOR scheme, 0.00 – 42.32% when compared with MSOR scheme, and 0.00 – 9.09% when compared with AOR scheme. Also, Table 5.18 presents the number of iteration and accuracy of all iterative methods with various relaxation schemes for Problem 5. From the table, the MAOR scheme for all iterative methods also reduces the number of iterations by 50.52 – 79.47% when compared with GS scheme, 3.85 – 39.55% when compared with SOR scheme, 0.00 – 38.65% when compared with MSOR scheme, and 0.00 – 7.69% when compared with AOR scheme. The MAOR scheme incorporates additional relaxation parameters designed to minimize the spectral radius of the iteration matrix. This reduction in the spectral radius leads to faster convergence by dampening error components more effectively, thus requiring fewer iterations to reach a solution.

In terms of accuracy, while the MAOR scheme is not the most accurate solver, it main-

tains competitive error levels when compared to other relaxation schemes. The reason lies in its prioritisation of minimising the iteration count through the optimisation of relaxation parameters. However, this optimisation does not necessarily guarantee the smallest possible residual error. As a result, although the MAOR scheme converges quickly, the final error can plateau at a higher level than some other schemes that may achieve greater accuracy at the expense of more iterations.

The execution time, speedup and parallel efficiency are shown in Tables 5.8 to 5.13 for Problem 3, Tables 5.16 and 5.17 for Problem 4 and Tables 5.20 and 5.21 for Problem 5. From these tables, we can see that as the number of processors increases, the execution time decreases and the speedup increases exponentially. From these tables, we can see that the MAOR scheme does not achieve the fastest execution times compared to some other relaxation schemes but remains competitive. The slower convergence of the MAOR scheme can be attributed to its design, which focuses on minimizing the spectral radius of the iteration matrix. This approach effectively reduces the number of iterations required for convergence. However, it prioritises iteration reduction over minimising residual errors, which can result in slower overall convergence in certain cases. Additionally, the MAOR scheme introduces extra acceleration parameters to enhance convergence, but these parameters can complicate the tuning process. If not optimally chosen, the parameters may lead to less effective convergence. Furthermore, the inclusion of additional computations associated with these parameters increases the per iteration computational cost. This increased cost can offset the benefits of a reduced iteration count, making the MAOR scheme slower than other relaxation methods in terms of total execution time. When parallel algorithms are implemented, the execution time generally decreases as the number of processors increases, and the speedup tends to increase exponentially. This shows that as more processors are added, the workload can be divided more efficiently, leading to reduced execution time. With an increase in the number of processors, the parallel efficiency of iterative methods improves. This is because the computational load is distributed across multiple pro-

processors, which decreases the overall time required for each iteration and increases the efficiency of parallel computation. Figures 5.12, 5.14, 5.16, and 5.18 show the execution time while figures 5.13, 5.15, 5.17, and 5.19 show the execution time of the fractional order iterative methods on all relaxation schemes for Problem 3 at $M = 80$ with various α . Figures 5.20 and 5.22 show the execution time while figures 5.21 and 5.23 show the speedup of the fractional order iterative methods on all relaxation schemes for Problems 4 and 5 at $M = 128$.

Moreover, Figures 5.14 and 5.15 show the speedup of the fractional order point and group iterative methods with GS and MAOR schemes for Problem 3 at $M = 80$ with various α , while Figures 5.17 and 5.19 show the execution time of the fractional order point and group iterative methods with GS and MAOR schemes for Problems 4 and 5 at $M = 128$. From the figures, as the number of processor increases, the speedup also increases linearly.

The computational costs for all fractional order iterative methods with various relaxation schemes for Problem 3 are shown in Table 5.14. From this table, the MAOR scheme has a 0.00 – 6.25% reduction in computational costs when compared to the AOR scheme. However, the MAOR scheme has a 28.52 – 161.05% increase in computational costs when compared to the SOR scheme. Similarly, the computational costs for all fractional order iterative methods with various relaxation schemes for Problems 4 and 5 are shown in Table 5.19. From this table, the MAOR scheme has a 0.00 – 4.44% reduction in computational costs when compared to the AOR scheme. However, the MAOR scheme has a 79.72 – 166.77% increase in computational costs when compared to the MSOR scheme. This is due to the MAOR scheme utilizing more points during the iterative process which also contributes in the MAOR scheme having a higher execution time than the MSOR scheme. The higher computational cost of the MAOR scheme is attributed to the introduction of additional acceleration parameters designed to minimise the spectral radius of the iteration matrix. Although this reduces the num-

ber of iterations required for convergence, it increases the computational workload per iteration. Compared to the simpler mechanisms in SOR and MSOR schemes, the MAOR scheme requires more arithmetic operations per iteration, which increases its complexity and computational overhead. This effect is particularly pronounced in large systems, where the added complexity can significantly impact overall performance.



Table 5.1: Computational costs for relaxation schemes in fractional order point iterative methods at $1 < \alpha, \beta < 2$ ($\sigma = M - 1$).

Method	FFS	FHS	FQS
GS	$(2\sigma^3 + 11\sigma^2)*ite$	$\frac{1}{6}(32\sigma^3 + 21\sigma^2 - 8\sigma + 33)*ite$ $+3(\sigma^2 - 1)$	$\frac{1}{4}(\sigma^3 + 8\sigma^2 - 19\sigma + 10)*ite$ $+ \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
SOR	$(2\sigma^3 + 15\sigma^2)*ite$	$\frac{1}{6}(32\sigma^3 + 33\sigma^2 - 8\sigma + 45)*ite$ $+3(\sigma^2 - 1)$	$\frac{1}{4}(\sigma^3 + 12\sigma^2 - 27\sigma + 14)*ite$ $+ \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MSOR			
AOR	$(6\sigma^3 + 26\sigma^2 - 3)*ite$	$(16\sigma^3 + 7\sigma^2 - 7\sigma + 13)*ite$ $+3(\sigma^2 - 1)$	$\frac{1}{4}(3\sigma^3 + 17\sigma^2 - 43\sigma + 11)*ite$ $+ \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MAOR			

Table 5.2: Computational costs for relaxation schemes in fractional order point iterative methods at $\alpha = \beta = 2$ ($\sigma = M - 1$).

Method	FFS	FHS	FQS
GS	$6\sigma^2*ite$	$\frac{7}{2}(\sigma^2 - 1)*ite + 3(\sigma^2 - 1)$	$\frac{7}{4}(\sigma - 1)^2*ite + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
SOR	$10\sigma^2*ite$	$\frac{11}{2}(\sigma^2 - 1)*ite + 3(\sigma^2 - 1)$	$\frac{11}{4}(\sigma - 1)^2*ite + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MSOR			
AOR	$5(3\sigma^2 - 1)*ite$	$(8\sigma^2 - 5\sigma - 3)*ite + 3(\sigma^2 - 1)$	$(4(\sigma - 1)^2 - 5)*ite + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MAOR			

Table 5.3: Computational costs for relaxation schemes in fractional order group iterative methods at $1 < \alpha, \beta < 2$ ($\sigma = M - 1$).

Method	FEG	FEDG	FMEG
GS	$\frac{1}{4}(4\sigma^3 + 71\sigma^2 - 34\sigma + 11)*\text{ite}$	$\frac{1}{3}(2\sigma^3 + 15\sigma^2 + 4\sigma + 30)*\text{ite} + 3(\sigma^2 - 1)$	$\frac{1}{16}(2\sigma^3 + 65\sigma^2 - 204\sigma + 181)*\text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
SOR	$\frac{1}{4}(4\sigma^3 + 87\sigma^2 - 34\sigma + 11)*\text{ite}$	$\frac{1}{3}(2\sigma^3 + 21\sigma^2 + 4\sigma + 42)*\text{ite} + 3(\sigma^2 - 1)$	$\frac{1}{16}(2\sigma^3 + 81\sigma^2 - 236\sigma + 197)*\text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MSOR			
AOR	$\frac{1}{2}(5\sigma^3 + 83\sigma^2 - 40\sigma - 30)*\text{ite}$	$\frac{1}{4}(8\sigma^3 + 47\sigma^2 + 2\sigma + 87)*\text{ite} + 3(\sigma^2 - 1)$	$\frac{1}{16}(5\sigma^3 + 151\sigma^2 - 477\sigma + 81)*\text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MAOR			

Table 5.4: Computational costs for relaxation schemes in fractional order group iterative methods at $\alpha = \beta = 2$ ($\sigma = M - 1$).

Method	FEG	FEDG	FMEG
GS	$(11\sigma^2 - 6\sigma + 1)*\text{ite}$	$\frac{1}{2}(9\sigma^2 - 4\sigma + 9)*\text{ite} + 3(\sigma^2 - 1)$	$(3\sigma^2 - 9\sigma + 7)*\text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
SOR	$(15\sigma^2 - 6\sigma + 1)*\text{ite}$	$\frac{1}{2}(13\sigma^2 - 4\sigma + 13)*\text{ite} + 3(\sigma^2 - 1)$	$(4\sigma^2 - 11\sigma + 8)*\text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MSOR			
AOR	$\frac{1}{2}(43\sigma^2 - 20\sigma - 19)*\text{ite}$	$\frac{1}{8}(77\sigma^2 + 8\sigma + 27)*\text{ite} + 3(\sigma^2 - 1)$	$\frac{1}{8}(45\sigma^2 - 130\sigma + 9)*\text{ite} + \frac{1}{4}(19\sigma^2 + 14\sigma - 5)$
MAOR			

Table 5.5: Number of iterations (Ite) and Error for Fractional FS and Fractional EG iterative methods with various relaxation schemes for Problem 3.

Method	M	$\alpha = 1.25$		$\alpha = 1.50$		$\alpha = 1.75$		$\alpha = 2.00$	
		Ite	Error	Ite	Error	Ite	Error	Ite	Error
Fractional FS									
GS	10	55	1.16E-05	91	2.46E-06	135	1.82E-06	188	5.63E-11
	20	129	2.00E-06	249	4.27E-07	433	2.33E-07	703	1.08E-10
	40	297	3.36E-07	675	6.97E-08	1370	2.95E-08	2592	2.11E-10
	80	682	5.51E-08	1814	1.09E-08	4298	3.61E-09	9472	4.15E-10
SOR	10	28	1.16E-05	27	2.46E-06	32	1.82E-06	37	2.32E-12
	20	63	2.00E-06	56	4.27E-07	60	2.33E-07	74	4.47E-12
	40	156	3.36E-07	152	6.97E-08	123	2.95E-08	148	1.12E-12
	80	388	5.51E-08	509	1.09E-08	286	3.69E-09	300	5.54E-13
MSOR	10	28	1.16E-05	27	2.46E-06	30	1.82E-06	37	1.23E-12
	20	63	2.00E-06	56	4.27E-07	59	2.33E-07	74	2.53E-12
	40	156	3.36E-07	151	6.97E-08	121	2.95E-08	144	3.90E-12
	80	388	5.51E-08	488	1.09E-08	280	3.69E-09	281	9.23E-13
AOR	10	24	1.16E-05	27	2.46E-06	30	1.82E-06	37	3.67E-12
	20	49	2.00E-06	49	4.27E-07	58	2.33E-07	73	5.30E-12
	40	104	3.36E-07	98	6.97E-08	111	2.95E-08	143	1.97E-12
	80	232	5.51E-08	227	1.09E-08	213	3.69E-09	288	2.96E-12
MAOR	10	23	1.16E-05	25	2.46E-06	29	1.82E-06	35	1.58E-11
	20	47	2.00E-06	48	4.27E-07	56	2.33E-07	69	9.04E-12
	40	102	3.36E-07	92	6.97E-08	106	2.95E-08	136	5.35E-12
	80	231	5.51E-08	217	1.09E-08	209	3.69E-09	270	2.51E-12
Fractional EG									
GS	10	28	2.54E-03	40	2.53E-03	59	1.64E-03	101	2.65E-11
	20	46	3.78E-03	66	3.78E-03	108	2.84E-03	369	5.24E-11
	40	72	3.50E-03	93	3.36E-03	148	2.76E-03	1355	1.05E-10
	80	103	2.45E-03	111	2.27E-03	165	1.95E-03	4964	2.07E-10
SOR	10	16	2.54E-03	19	2.53E-03	22	1.64E-03	27	6.64E-13
	20	25	3.78E-03	28	3.78E-03	30	2.84E-03	53	1.04E-12
	40	41	3.50E-03	39	3.36E-03	41	2.76E-03	106	4.82E-12
	80	62	2.45E-03	52	2.27E-03	50	1.95E-03	215	4.89E-13
MSOR	10	15	2.54E-03	18	2.53E-03	21	1.64E-03	27	1.95E-12
	20	24	3.78E-03	27	3.78E-03	30	2.84E-03	53	1.83E-12
	40	40	3.50E-03	38	3.36E-03	41	2.76E-03	105	1.08E-12
	80	61	2.45E-03	50	2.27E-03	49	1.95E-03	208	1.62E-12
AOR	10	15	2.54E-03	18	2.53E-03	20	1.64E-03	27	2.51E-12
	20	25	3.78E-03	26	3.78E-03	30	2.84E-03	52	1.96E-12
	40	39	3.50E-03	39	3.36E-03	40	2.76E-03	104	3.16E-12
	80	60	2.45E-03	51	2.27E-03	50	1.95E-03	200	3.13E-12
MAOR	10	14	2.54E-03	17	2.53E-03	20	1.64E-03	26	9.51E-12
	20	23	3.78E-03	25	3.78E-03	29	2.84E-03	50	4.08E-12
	40	37	3.50E-03	36	3.36E-03	40	2.76E-03	98	4.50E-12
	80	58	2.45E-03	50	2.27E-03	49	1.95E-03	192	3.42E-12

Table 5.6: Number of iterations (Ite) and Error for Fractional HS and Fractional EDG iterative methods with various relaxation schemes for Problem 3.

Method	M	$\alpha = 1.25$		$\alpha = 1.50$		$\alpha = 1.75$		$\alpha = 2.00$	
		Ite	Error	Ite	Error	Ite	Error	Ite	Error
Fractional HS									
GS	10	35	7.94E-03	54	4.28E-03	75	1.77E-03	98	1.01E-04
	20	82	4.43E-03	148	2.55E-03	239	1.14E-03	365	1.14E-05
	40	189	2.44E-03	399	1.49E-03	756	7.09E-04	1352	1.36E-06
	80	435	1.33E-03	1072	8.59E-04	2369	4.32E-04	4961	1.65E-07
SOR	10	19	7.94E-03	21	4.28E-03	24	1.77E-03	27	1.01E-04
	20	39	4.43E-03	42	2.55E-03	43	1.14E-03	53	1.14E-05
	40	90	2.44E-03	96	1.49E-03	88	7.09E-04	105	1.36E-06
	80	227	1.33E-03	253	8.59E-04	211	4.32E-04	211	1.65E-07
MSOR	10	18	7.94E-03	21	4.28E-03	23	1.77E-03	26	1.01E-04
	20	39	4.43E-03	42	2.55E-03	43	1.14E-03	51	1.14E-05
	40	89	2.44E-03	95	1.49E-03	85	7.09E-04	103	1.36E-06
	80	225	1.33E-03	248	8.59E-04	208	4.32E-04	202	1.65E-07
AOR	10	18	7.94E-03	21	4.28E-03	24	1.77E-03	26	1.01E-04
	20	36	4.43E-03	38	2.55E-03	43	1.14E-03	52	1.14E-05
	40	75	2.44E-03	79	1.49E-03	81	7.09E-04	103	1.36E-06
	80	163	1.33E-03	180	8.59E-04	162	4.32E-04	200	1.65E-07
MAOR	10	17	7.94E-03	19	4.28E-03	22	1.77E-03	25	1.01E-04
	20	35	4.43E-03	37	2.55E-03	41	1.14E-03	49	1.14E-05
	40	74	2.44E-03	77	1.49E-03	78	7.09E-04	98	1.36E-06
	80	162	1.33E-03	179	8.59E-04	157	4.32E-04	191	1.65E-07
Fractional EDG									
GS	10	25	7.95E-03	39	4.41E-03	56	1.91E-03	77	1.01E-04
	20	45	4.73E-03	51	3.11E-03	77	1.70E-03	281	1.14E-05
	40	60	3.10E-03	79	2.45E-03	114	1.68E-03	1034	1.36E-06
	80	85	2.09E-03	102	1.81E-03	157	1.43E-03	3793	1.65E-07
SOR	10	14	7.95E-03	17	4.41E-03	20	1.91E-03	24	1.01E-04
	20	23	4.73E-03	27	3.11E-03	27	1.70E-03	47	1.14E-05
	40	39	3.10E-03	37	2.45E-03	38	1.68E-03	94	1.36E-06
	80	59	2.09E-03	46	1.81E-03	45	1.43E-03	189	1.65E-07
MSOR	10	14	7.95E-03	17	4.41E-03	20	1.91E-03	24	1.01E-04
	20	23	4.73E-03	26	3.11E-03	28	1.70E-03	47	1.14E-05
	40	38	3.10E-03	36	2.45E-03	35	1.68E-03	93	1.36E-06
	80	57	2.09E-03	44	1.81E-03	44	1.43E-03	179	1.65E-07
AOR	10	13	7.95E-03	17	4.41E-03	19	1.91E-03	23	1.01E-04
	20	23	4.73E-03	25	3.11E-03	26	1.70E-03	46	1.14E-05
	40	37	3.10E-03	34	2.45E-03	35	1.68E-03	88	1.36E-06
	80	49	2.09E-03	45	1.81E-03	45	1.43E-03	172	1.65E-07
MAOR	10	13	7.95E-03	16	4.41E-03	19	1.91E-03	22	1.01E-04
	20	22	4.73E-03	24	3.11E-03	25	1.70E-03	43	1.14E-05
	40	36	3.10E-03	33	2.45E-03	34	1.68E-03	84	1.36E-06
	80	47	2.09E-03	43	1.81E-03	44	1.43E-03	165	1.65E-07

Table 5.7: Number of iterations (Ite) and Error for Fractional QS and Fractional MEG iterative methods with various relaxation schemes for Problem 3.

Method	M	$\alpha = 1.25$		$\alpha = 1.50$		$\alpha = 1.75$		$\alpha = 2.00$	
		Ite	Error	Ite	Error	Ite	Error	Ite	Error
Fractional QS									
GS	10	23	3.23E-04	32	2.26E-04	41	1.23E-04	49	7.30E-06
	20	55	4.49E-05	91	3.30E-05	135	1.71E-05	188	2.08E-07
	40	129	6.49E-06	249	4.72E-06	433	2.34E-06	703	6.20E-09
	80	297	9.58E-07	675	6.64E-07	1370	3.12E-07	2592	1.45E-10
SOR	10	13	3.23E-04	15	2.26E-04	17	1.23E-04	19	7.30E-06
	20	28	4.49E-05	27	3.30E-05	32	1.71E-05	37	2.08E-07
	40	63	6.49E-06	56	4.72E-06	60	2.34E-06	74	6.24E-09
	80	156	9.58E-07	152	6.64E-07	123	3.12E-07	148	1.91E-10
MSOR	10	13	3.23E-04	15	2.26E-04	16	1.23E-04	18	7.30E-06
	20	28	4.49E-05	27	3.30E-05	30	1.71E-05	37	2.08E-07
	40	63	6.49E-06	56	4.72E-06	59	2.34E-06	74	6.24E-09
	80	156	9.58E-07	151	6.64E-07	121	3.12E-07	144	1.90E-10
AOR	10	13	3.23E-04	15	2.26E-04	16	1.23E-04	18	7.30E-06
	20	24	4.49E-05	27	3.30E-05	30	1.71E-05	37	2.08E-07
	40	49	6.49E-06	49	4.72E-06	58	2.34E-06	73	6.24E-09
	80	104	9.58E-07	98	6.64E-07	111	3.12E-07	143	1.91E-10
MAOR	10	12	3.23E-04	13	2.26E-04	15	1.23E-04	17	7.30E-06
	20	23	4.49E-05	25	3.30E-05	29	1.71E-05	35	2.08E-07
	40	47	6.49E-06	48	4.72E-06	56	2.34E-06	71	6.24E-09
	80	102	9.58E-07	93	6.64E-07	108	3.12E-07	136	1.92E-10
Fractional MEG									
GS	10	14	7.53E-04	18	5.70E-04	22	2.79E-04	26	7.30E-06
	20	28	1.13E-03	40	1.12E-03	59	7.18E-04	101	2.08E-07
	40	46	1.78E-03	66	1.78E-03	108	1.34E-03	369	6.22E-09
	80	72	1.70E-03	93	1.63E-03	148	1.34E-03	1355	1.61E-10
SOR	10	10	7.53E-04	11	5.70E-04	12	2.79E-04	13	7.30E-06
	20	16	1.13E-03	19	1.12E-03	22	7.18E-04	27	2.08E-07
	40	25	1.78E-03	28	1.78E-03	30	1.34E-03	53	6.24E-09
	80	41	1.70E-03	39	1.63E-03	41	1.34E-03	106	1.89E-10
MSOR	10	9	7.53E-04	10	5.70E-04	12	2.79E-04	13	7.30E-06
	20	16	1.13E-03	18	1.12E-03	21	7.18E-04	27	2.08E-07
	40	24	1.78E-03	27	1.78E-03	30	1.34E-03	53	6.24E-09
	80	40	1.70E-03	38	1.63E-03	41	1.34E-03	105	1.91E-10
AOR	10	9	7.53E-04	11	5.70E-04	12	2.79E-04	13	7.30E-06
	20	15	1.13E-03	18	1.12E-03	20	7.18E-04	27	2.08E-07
	40	25	1.78E-03	26	1.78E-03	30	1.34E-03	52	6.24E-09
	80	39	1.70E-03	39	1.63E-03	40	1.34E-03	104	1.90E-10
MAOR	10	9	7.53E-04	10	5.70E-04	12	2.79E-04	12	7.30E-06
	20	15	1.13E-03	17	1.12E-03	20	7.18E-04	26	2.08E-07
	40	23	1.78E-03	25	1.78E-03	29	1.34E-03	51	6.23E-09
	80	37	1.70E-03	37	1.63E-03	40	1.34E-03	98	1.92E-10

Table 5.8: Execution time (seconds), speedup and parallel efficiency for fractional FS iterative methods with various relaxation schemes for Problem 3 at $M = 80$.

α	Methods	Execution Time				Speedup				Efficiency			
		T1	T2	T4	T8	S1	S2	S4	S8	E1	E2	E4	E8
1.25	FFS GS	93.703	49.262	32.889	25.697	1.000	1.902	2.849	3.646	1.000	0.951	0.712	0.456
	FFS SOR	75.798	40.038	22.990	18.755	1.000	1.893	3.297	4.041	1.000	0.947	0.824	0.505
	FFS MSOR	75.798	40.038	22.990	18.755	1.000	1.893	3.297	4.041	1.000	0.947	0.824	0.505
	FFS AOR	124.313	64.845	37.428	29.460	1.000	1.917	3.321	4.220	1.000	0.959	0.830	0.527
	FFS MAOR	108.959	57.429	34.023	28.876	1.000	1.897	3.202	3.773	1.000	0.949	0.801	0.472
1.50	FFS GS	325.862	177.400	101.508	65.842	1.000	1.837	3.210	4.949	1.000	0.918	0.803	0.619
	FFS SOR	77.345	40.616	23.313	18.859	1.000	1.904	3.318	4.101	1.000	0.952	0.829	0.513
	FFS MSOR	76.096	40.241	23.000	18.774	1.000	1.891	3.308	4.053	1.000	0.945	0.827	0.507
	FFS AOR	83.148	44.197	28.497	23.764	1.000	1.881	2.918	3.499	1.000	0.941	0.729	0.437
	FFS MAOR	82.205	43.945	26.481	23.696	1.000	1.871	3.104	3.469	1.000	0.935	0.776	0.434
1.75	FFS GS	842.305	407.539	214.700	166.793	1.000	2.067	3.923	5.050	1.000	1.033	0.981	0.631
	FFS SOR	53.822	29.638	19.370	15.580	1.000	1.816	2.779	3.454	1.000	0.908	0.695	0.432
	FFS MSOR	50.515	26.725	17.588	12.993	1.000	1.890	2.872	3.888	1.000	0.945	0.718	0.486
	FFS AOR	81.871	43.669	24.601	22.257	1.000	1.875	3.328	3.678	1.000	0.937	0.832	0.460
	FFS MAOR	77.549	41.138	24.415	21.728	1.000	1.885	3.176	3.569	1.000	0.943	0.794	0.446
2.00	FFS GS	1.172	1.002	0.897	0.853	1.000	1.170	1.307	1.373	1.000	0.585	0.327	0.172
	FFS SOR	0.040	0.032	0.027	0.023	1.000	1.231	1.460	1.712	1.000	0.615	0.365	0.214
	FFS MSOR	0.036	0.032	0.026	0.023	1.000	1.129	1.420	1.568	1.000	0.565	0.355	0.196
	FFS AOR	0.086	0.081	0.069	0.068	1.000	1.067	1.248	1.260	1.000	0.533	0.312	0.158
	FFS MAOR	0.052	0.038	0.032	0.028	1.000	1.370	1.648	1.843	1.000	0.685	0.412	0.230

Table 5.9: Execution time (seconds), speedup and parallel efficiency for fractional HS iterative methods with various relaxation schemes for Problem 3 at $M = 80$.

α	Methods	Execution Time					Speedup					Efficiency			
		T1	T2	T4	T8		S1	S2	S4	S8		E1	E2	E4	E8
1.25	FHS GS	19.433	10.359	6.616	5.540		1.000	1.876	2.937	3.508		1.000	0.938	0.734	0.438
	FHS SOR	14.069	9.594	5.439	4.423		1.000	1.466	2.587	3.181		1.000	0.733	0.647	0.398
	FHS MSOR	13.668	9.308	5.390	4.380		1.000	1.468	2.536	3.121		1.000	0.734	0.634	0.390
	FHS AOR	20.954	11.639	7.796	6.286		1.000	1.800	2.688	3.333		1.000	0.900	0.672	0.417
	FHS MAOR	20.733	11.073	7.766	6.151		1.000	1.872	2.670	3.371		1.000	0.936	0.667	0.421
1.50	FHS GS	62.763	33.514	20.204	15.894		1.000	1.873	3.106	3.949		1.000	0.936	0.777	0.494
	FHS SOR	19.329	10.243	6.546	5.205		1.000	1.887	2.953	3.713		1.000	0.944	0.738	0.464
	FHS MSOR	15.503	10.243	6.272	5.030		1.000	1.514	2.472	3.082		1.000	0.757	0.618	0.385
	FHS AOR	28.397	15.413	9.105	8.368		1.000	1.842	3.119	3.394		1.000	0.921	0.780	0.424
	FHS MAOR	25.672	13.808	8.412	6.566		1.000	1.859	3.052	3.910		1.000	0.930	0.763	0.489
1.75	FHS GS	141.165	75.080	40.019	30.915		1.000	2.085	3.426	3.735		1.000	1.043	0.856	0.467
	FHS SOR	13.392	9.247	5.338	4.379		1.000	2.067	3.522	3.892		1.000	1.033	0.880	0.486
	FHS MSOR	12.844	8.437	5.210	3.669		1.000	2.067	3.522	3.892		1.000	1.033	0.880	0.486
	FHS AOR	20.733	11.073	7.766	6.151		1.000	2.221	4.035	4.302		1.000	1.110	1.009	0.538
	FHS MAOR	20.141	10.392	7.114	5.699		1.000	2.166	3.934	4.203		1.000	1.083	0.984	0.525
2.00	FHS GS	0.673	0.411	0.309	0.299		1.000	1.638	2.180	2.250		1.000	0.819	0.545	0.281
	FHS SOR	0.017	0.016	0.014	0.011		1.000	1.032	1.156	1.536		1.000	0.516	0.289	0.192
	FHS MSOR	0.016	0.015	0.013	0.011		1.000	1.104	1.283	1.521		1.000	0.552	0.321	0.190
	FHS AOR	0.023	0.020	0.016	0.014		1.000	1.130	1.469	1.685		1.000	0.565	0.367	0.211
	FHS MAOR	0.023	0.019	0.015	0.013		1.000	1.208	1.476	1.786		1.000	0.604	0.369	0.223

Table 5.10: Execution time (seconds), speedup and parallel efficiency for fractional QS iterative methods with various relaxation schemes for Problem 3 at $M = 80$.

α	Methods	Execution Time					Speedup				Efficiency			
		T1	T2	T4	T8		S1	S2	S4	S8	E1	E2	E4	E8
1.25	FQS GS	6.918	5.207	3.783	2.711		1.000	1.329	1.829	2.552	1.000	0.664	0.457	0.319
	FQS SOR	4.701	2.646	1.536	1.299		1.000	1.777	3.061	3.619	1.000	0.888	0.765	0.452
	FQS MSOR	4.701	2.646	1.536	1.299		1.000	1.777	3.061	3.619	1.000	0.888	0.765	0.452
	FQS AOR	7.721	5.661	3.934	2.961		1.000	1.364	1.963	2.608	1.000	0.682	0.491	0.326
	FQS MAOR	7.241	5.460	3.934	2.741		1.000	1.326	1.840	2.642	1.000	0.663	0.460	0.330
1.50	FQS GS	14.606	10.159	6.105	4.821		1.000	1.438	2.393	3.030	1.000	0.719	0.598	0.379
	FQS SOR	4.557	2.462	1.383	1.097		1.000	1.851	3.294	4.154	1.000	0.926	0.823	0.519
	FQS MSOR	4.268	2.286	1.237	1.028		1.000	1.868	3.451	4.151	1.000	0.934	0.863	0.519
	FQS AOR	6.663	4.927	3.623	2.696		1.000	1.352	1.839	2.471	1.000	0.676	0.460	0.309
	FQS MAOR	6.374	4.722	3.094	2.312		1.000	1.350	2.060	2.757	1.000	0.675	0.515	0.345
1.75	FQS GS	21.011	12.610	8.351	6.427		1.000	1.666	2.516	3.269	1.000	0.833	0.629	0.409
	FQS SOR	3.937	2.131	1.201	0.998		1.000	1.848	3.279	3.945	1.000	0.924	0.820	0.493
	FQS MSOR	3.714	2.008	1.198	0.968		1.000	1.849	3.100	3.838	1.000	0.925	0.775	0.480
	FQS AOR	9.882	6.619	4.025	3.301		1.000	1.493	2.455	2.994	1.000	0.746	0.614	0.374
	FQS MAOR	9.669	6.602	4.015	3.094		1.000	1.465	2.408	3.126	1.000	0.732	0.602	0.391
2.00	FQS GS	0.098	0.097	0.086	0.083		1.000	1.008	1.147	1.178	1.000	0.504	0.287	0.147
	FQS SOR	0.009	0.007	0.006	0.005		1.000	1.324	1.680	1.954	1.000	0.662	0.420	0.244
	FQS MSOR	0.007	0.005	0.005	0.005		1.000	1.221	1.419	1.481	1.000	0.611	0.355	0.185
	FQS AOR	0.014	0.011	0.011	0.007		1.000	1.212	1.237	1.976	1.000	0.606	0.309	0.247
	FQS MAOR	0.012	0.008	0.008	0.007		1.000	1.554	1.622	1.841	1.000	0.777	0.405	0.230

Table 5.11: Execution time (seconds), speedup and parallel efficiency for fractional EG iterative methods with various relaxation schemes for Problem 3 at $M = 80$.

α	Methods	Execution Time				Speedup				Efficiency			
		T1	T2	T4	T8	S1	S2	S4	S8	E1	E2	E4	E8
1.25	FEG GS	10.367	6.717	4.142	3.321	1.000	1.544	2.503	3.122	1.000	0.772	0.626	0.390
	FEG SOR	6.697	4.927	3.668	2.696	1.000	1.359	1.825	2.484	1.000	0.680	0.456	0.311
	FEG MSOR	6.662	4.862	3.277	2.655	1.000	1.370	2.033	2.509	1.000	0.685	0.508	0.314
	FEG AOR	14.434	9.841	5.450	4.645	1.000	1.467	2.648	3.108	1.000	0.733	0.662	0.388
	FEG MAOR	14.352	9.829	5.450	4.423	1.000	1.460	2.633	3.245	1.000	0.730	0.658	0.406
1.50	FEG GS	11.248	6.748	4.210	3.380	1.000	1.667	2.672	3.328	1.000	0.833	0.668	0.416
	FEG SOR	5.883	4.581	2.922	2.024	1.000	1.284	2.013	2.907	1.000	0.642	0.503	0.363
	FEG MSOR	5.742	4.459	2.894	1.774	1.000	1.288	1.984	3.237	1.000	0.644	0.496	0.405
	FEG AOR	12.599	7.920	4.987	3.624	1.000	1.591	2.526	3.477	1.000	0.795	0.632	0.435
	FEG MAOR	12.433	7.766	4.855	3.516	1.000	1.601	2.561	3.536	1.000	0.800	0.640	0.442
1.75	FEG GS	19.433	10.260	6.562	5.354	1.000	1.894	2.961	3.630	1.000	0.947	0.740	0.454
	FEG SOR	5.742	4.459	2.894	1.774	1.000	1.288	1.984	3.237	1.000	0.644	0.496	0.405
	FEG MSOR	5.632	3.872	2.738	1.755	1.000	1.454	2.057	3.210	1.000	0.727	0.514	0.401
	FEG AOR	12.433	7.766	4.855	3.516	1.000	1.601	2.561	3.536	1.000	0.800	0.640	0.442
	FEG MAOR	11.879	7.302	4.391	3.422	1.000	1.627	2.705	3.471	1.000	0.813	0.676	0.434
2.00	FEG GS	1.087	0.747	0.601	0.530	1.000	1.455	1.808	2.049	1.000	0.728	0.452	0.256
	FEG SOR	0.051	0.035	0.031	0.026	1.000	1.441	1.653	1.952	1.000	0.721	0.413	0.244
	FEG MSOR	0.043	0.033	0.029	0.024	1.000	1.297	1.459	1.766	1.000	0.649	0.365	0.221
	FEG AOR	0.084	0.045	0.035	0.031	1.000	1.859	2.385	2.713	1.000	0.929	0.596	0.339
	FEG MAOR	0.054	0.040	0.032	0.029	1.000	1.362	1.708	1.860	1.000	0.681	0.427	0.233

Table 5.12: Execution time (seconds), speedup and parallel efficiency for fractional EDG iterative methods with various relaxation schemes for Problem 3 at $M = 80$.

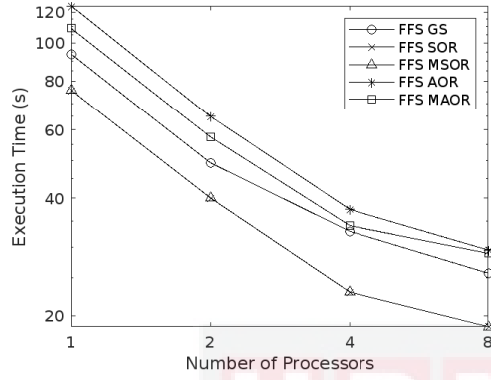
α	Methods	Execution Time					Speedup					Efficiency			
		T1	T2	T4	T8		S1	S2	S4	S8		E1	E2	E4	E8
1.25	FEDG GS	4.221	2.280	1.206	1.000		1.000	1.852	3.499	4.221		1.000	0.926	0.875	0.528
	FEDG SOR	3.529	1.864	1.113	0.882		1.000	1.893	3.170	3.999		1.000	0.947	0.793	0.500
	FEDG MSOR	3.310	1.785	1.065	0.861		1.000	1.854	3.108	3.844		1.000	0.927	0.777	0.480
	FEDG AOR	5.524	3.661	2.667	1.743		1.000	1.509	2.071	3.169		1.000	0.754	0.518	0.396
	FEDG MAOR	5.341	3.343	2.458	1.501		1.000	1.597	2.173	3.559		1.000	0.799	0.543	0.445
1.50	FEDG GS	4.601	2.610	1.422	1.130		1.000	1.763	3.235	4.073		1.000	0.882	0.809	0.509
	FEDG SOR	3.080	1.716	0.937	0.775		1.000	1.795	3.286	3.974		1.000	0.897	0.822	0.497
	FEDG MSOR	2.500	1.377	0.733	0.667		1.000	1.816	3.409	3.748		1.000	0.908	0.852	0.468
	FEDG AOR	5.193	3.185	1.769	1.417		1.000	1.630	2.936	3.664		1.000	0.815	0.734	0.458
	FEDG MAOR	5.016	2.845	1.567	1.356		1.000	1.763	3.201	3.700		1.000	0.882	0.800	0.462
1.75	FEDG GS	6.139	4.656	2.932	2.131		1.000	1.318	2.094	2.881		1.000	0.659	0.523	0.360
	FEDG SOR	2.549	1.711	0.893	0.752		1.000	1.490	2.854	3.387		1.000	0.745	0.713	0.423
	FEDG MSOR	2.153	1.334	0.719	0.643		1.000	1.614	2.993	3.347		1.000	0.807	0.748	0.418
	FEDG AOR	5.193	3.185	1.769	1.417		1.000	1.630	2.936	3.664		1.000	0.815	0.734	0.458
	FEDG MAOR	5.052	2.856	1.601	1.414		1.000	1.769	3.155	3.574		1.000	0.884	0.789	0.447
2.00	FEDG GS	0.345	0.331	0.233	0.195		1.000	1.042	1.478	1.765		1.000	0.521	0.369	0.221
	FEDG SOR	0.020	0.018	0.015	0.011		1.000	1.095	1.352	1.764		1.000	0.548	0.338	0.220
	FEDG MSOR	0.016	0.015	0.013	0.011		1.000	1.099	1.287	1.553		1.000	0.549	0.322	0.194
	FEDG AOR	0.035	0.032	0.025	0.018		1.000	1.115	1.423	1.933		1.000	0.558	0.356	0.242
	FEDG MAOR	0.026	0.024	0.016	0.015		1.000	1.118	1.700	1.787		1.000	0.559	0.425	0.223

Table 5.13: Execution time (seconds), speedup and parallel efficiency for fractional MEG iterative methods with various relaxation schemes for Problem 3 at $M = 80$.

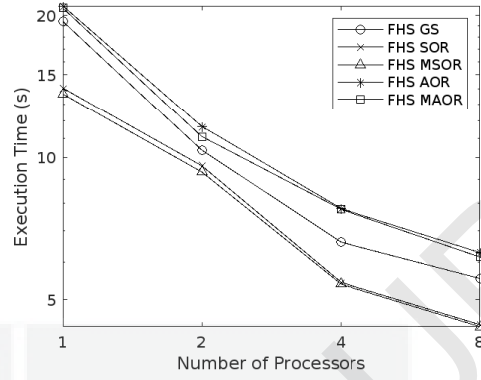
α	Methods	Execution Time				Speedup				Efficiency			
		T1	T2	T4	T8	S1	S2	S4	S8	E1	E2	E4	E8
1.25	FMEG GS	0.920	0.656	0.502	0.408	1.000	1.404	1.832	2.255	1.000	0.702	0.458	0.282
	FMEG SOR	0.755	0.453	0.315	0.242	1.000	1.665	2.399	3.122	1.000	0.833	0.600	0.390
	FMEG MSOR	0.586	0.428	0.311	0.242	1.000	1.369	1.885	2.422	1.000	0.685	0.471	0.303
	FMEG AOR	1.337	0.718	0.557	0.455	1.000	1.861	2.401	2.941	1.000	0.931	0.600	0.368
	FMEG MAOR	0.954	0.698	0.552	0.411	1.000	1.367	1.729	2.320	1.000	0.684	0.432	0.290
1.50	FMEG GS	0.954	0.698	0.554	0.433	1.000	1.367	1.723	2.206	1.000	0.684	0.431	0.276
	FMEG SOR	0.501	0.424	0.308	0.227	1.000	1.181	1.628	2.205	1.000	0.590	0.407	0.276
	FMEG MSOR	0.493	0.374	0.280	0.219	1.000	1.319	1.762	2.247	1.000	0.659	0.440	0.281
	FMEG AOR	1.337	0.718	0.557	0.455	1.000	1.861	2.401	2.941	1.000	0.931	0.600	0.368
	FMEG MAOR	0.954	0.698	0.552	0.411	1.000	1.367	1.729	2.320	1.000	0.684	0.432	0.290
1.75	FMEG GS	3.150	1.759	1.065	0.781	1.000	1.791	2.958	4.034	1.000	0.896	0.739	0.504
	FMEG SOR	0.755	0.453	0.315	0.242	1.000	1.665	2.399	3.122	1.000	0.833	0.600	0.390
	FMEG MSOR	0.755	0.453	0.315	0.242	1.000	1.665	2.399	3.122	1.000	0.833	0.600	0.390
	FMEG AOR	1.506	0.930	0.627	0.562	1.000	1.619	2.401	2.678	1.000	0.809	0.600	0.335
	FMEG MAOR	1.506	0.930	0.627	0.562	1.000	1.619	2.401	2.678	1.000	0.809	0.600	0.335
2.00	FMEG GS	0.052	0.035	0.031	0.028	1.000	1.459	1.649	1.833	1.000	0.730	0.412	0.229
	FMEG SOR	0.012	0.008	0.007	0.006	1.000	1.502	1.716	1.953	1.000	0.751	0.429	0.244
	FMEG MSOR	0.011	0.007	0.006	0.005	1.000	1.473	1.951	2.077	1.000	0.737	0.488	0.260
	FMEG AOR	0.014	0.013	0.012	0.008	1.000	1.036	1.150	1.625	1.000	0.518	0.287	0.203
	FMEG MAOR	0.013	0.011	0.010	0.007	1.000	1.222	1.258	1.883	1.000	0.611	0.314	0.235

Table 5.14: Computational costs of fractional order iterative methods with various relaxation schemes for Problem 3 at $M = 80$.

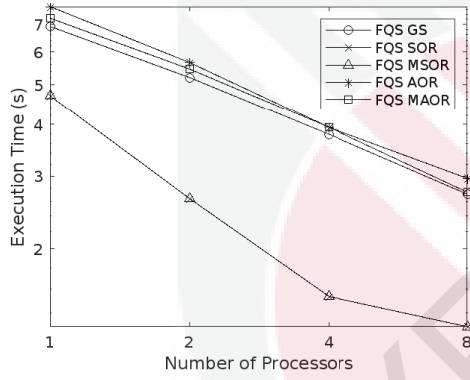
Methods	Computational Cost			
	$\alpha = 1.25$	$\alpha = 1.50$	$\alpha = 1.75$	$\alpha = 2.00$
FFS GS	719,325,178	1,913,278,406	4,533,225,242	354,688,512
FFS SOR	418,920,884	549,563,737	308,792,198	18,723,000
FFS MSOR	418,920,884	526,890,184	302,314,040	17,537,210
FFS AOR	723,955,304	708,352,819	664,665,861	26,959,680
FFS MAOR	720,834,807	677,147,849	652,183,873	25,274,700
FHS GS	156,624,375	385,952,656	852,889,517	108,366,960
FHS SOR	84,575,993	94,260,967	78,616,009	7,260,240
FHS MSOR	83,830,995	92,398,472	77,498,512	6,951,360
FHS AOR	172,712,493	190,723,500	171,653,022	9,924,720
FHS MAOR	171,653,022	189,664,029	166,355,667	9,478,950
FQS GS	40,234,513	91,403,995	185,485,450	27,626,944
FQS SOR	22,096,588	21,530,776	17,428,639	2,506,108
FQS MSOR	22,096,588	21,389,323	17,145,733	2,439,184
FQS AOR	41,157,448	38,784,706	43,925,647	3,509,253
FQS MAOR	40,366,534	36,807,421	42,739,276	3,338,936
FEG GS	62,124,244	66,949,428	99,519,420	338,435,592
FEG SOR	38,942,944	32,661,824	31,405,600	20,025,530
FEG MSOR	38,314,832	31,405,600	30,777,488	19,373,536
FEG AOR	89,400,240	75,990,204	74,500,200	26,676,400
FEG MAOR	86,420,232	74,500,200	73,010,196	25,609,344
FEDG GS	18,074,505	21,685,662	33,368,817	105,961,003
FEDG SOR	13,288,115	10,364,350	10,139,445	7,657,155
FEDG MSOR	12,838,305	9,914,540	9,914,540	7,253,005
FEDG AOR	28,820,969	26,469,765	26,469,765	10,364,864
FEDG MAOR	27,645,367	25,294,163	25,881,964	9,943,800
FMEG GS	6,221,056	8,026,804	12,756,144	24,445,665
FMEG SOR	3,804,872	3,620,728	3,804,872	2,584,838
FMEG MSOR	3,712,800	3,528,656	3,804,872	2,560,735
FMEG AOR	8,244,256	8,244,256	8,454,880	3,547,512
FMEG MAOR	7,823,008	7,823,008	8,454,880	3,344,574



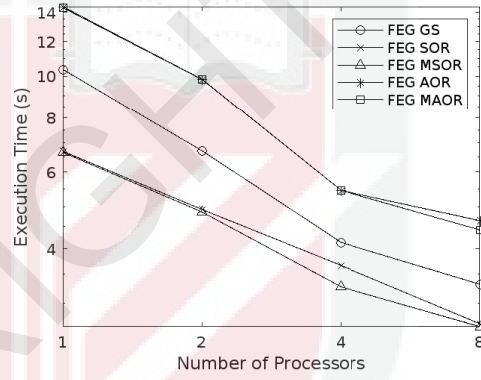
(a)



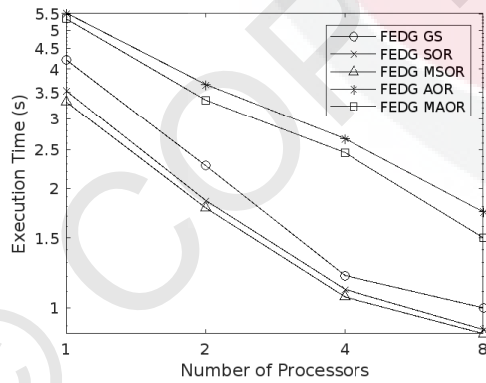
(b)



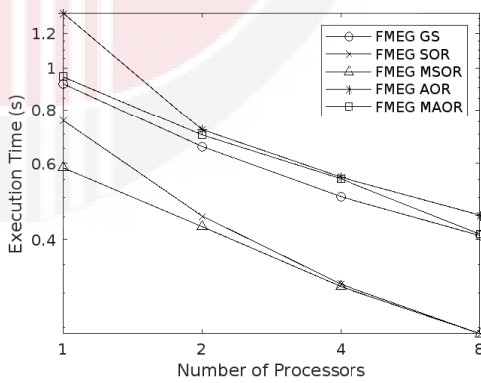
(c)



(d)

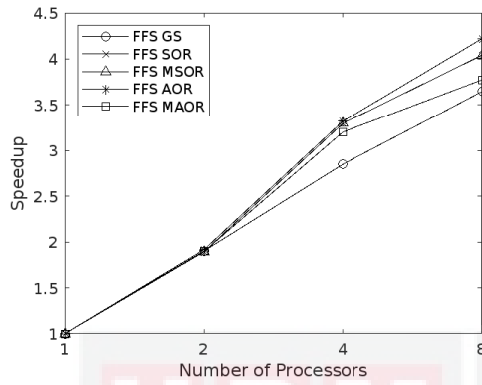


(e)

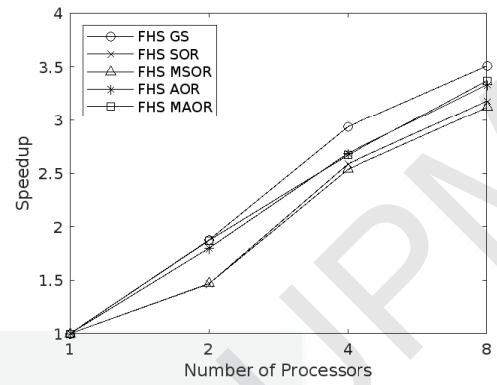


(f)

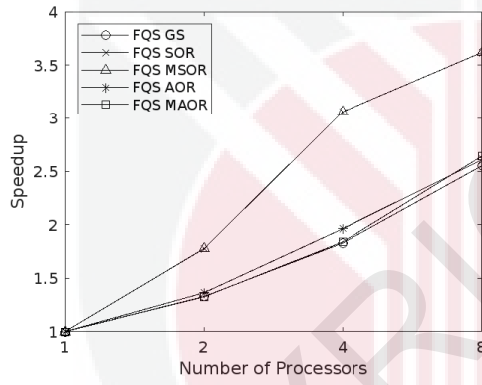
Figure 5.12: The execution time (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 1.25$.



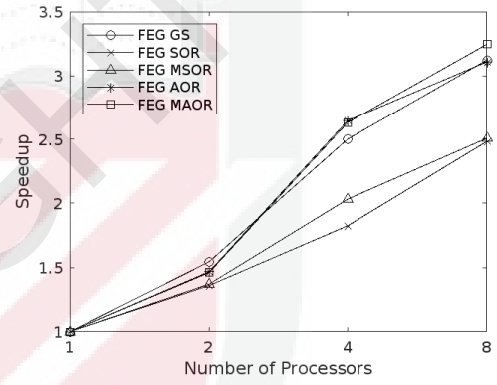
(a)



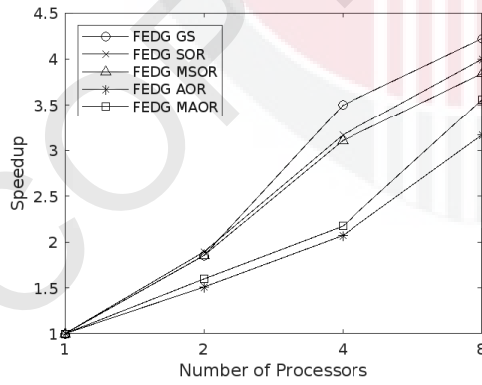
(b)



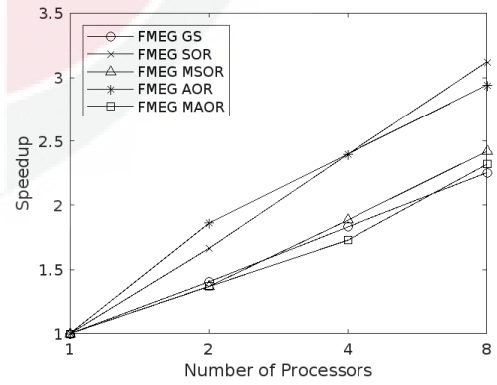
(c)



(d)

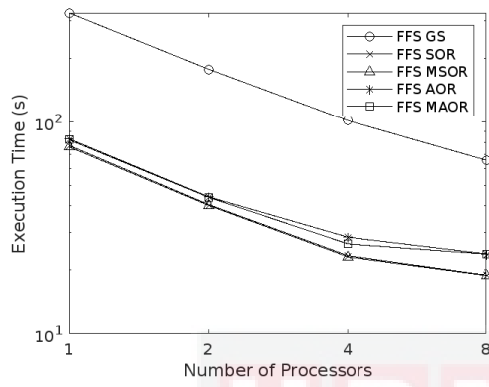


(e)

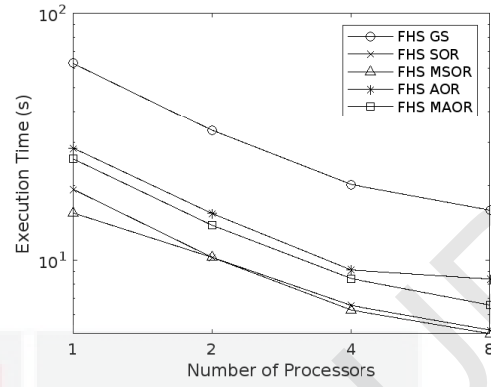


(f)

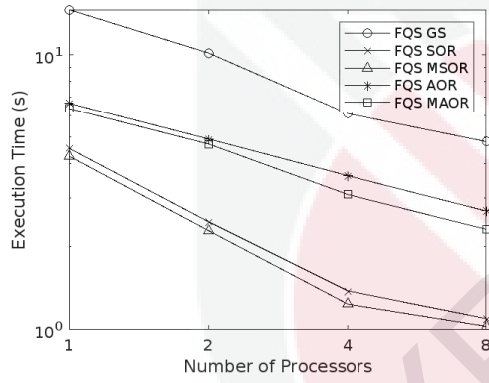
Figure 5.13: The speedup for (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 1.25$.



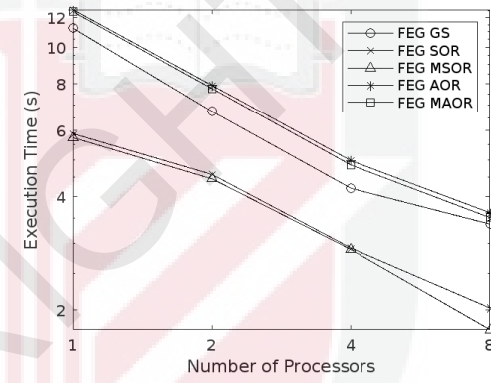
(a)



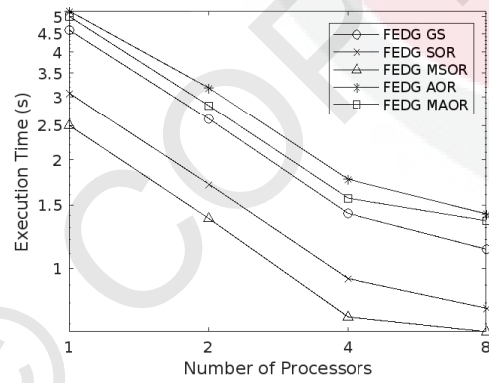
(b)



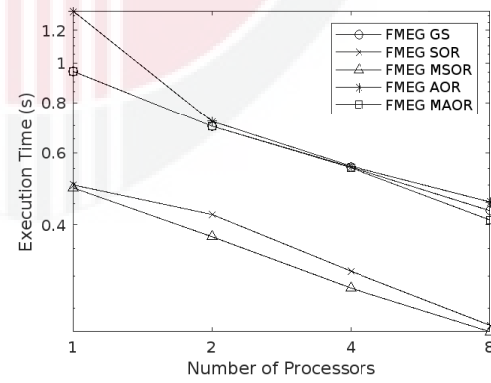
(c)



(d)

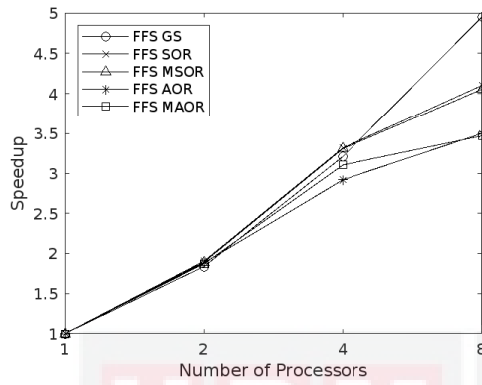


(e)

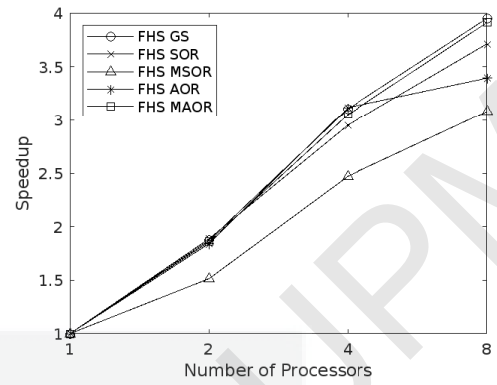


(f)

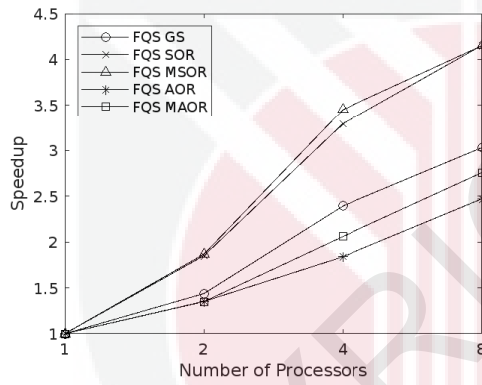
Figure 5.14: The execution time (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 1.50$.



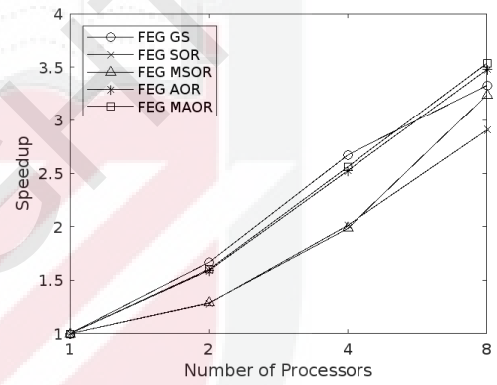
(a)



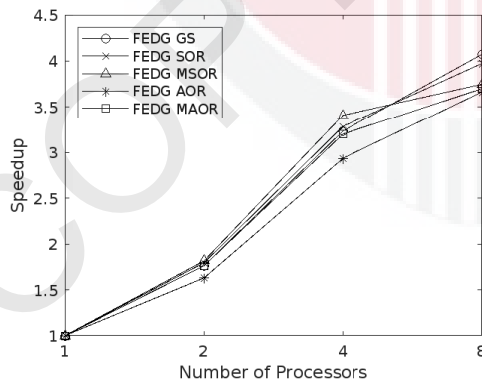
(b)



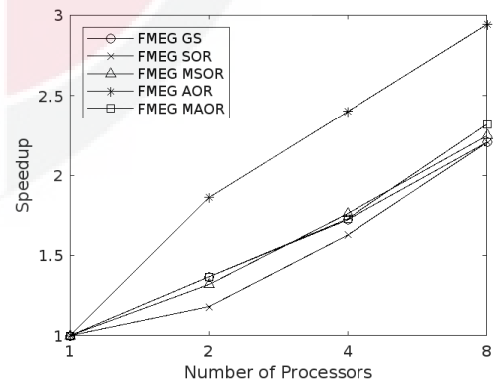
(c)



(d)

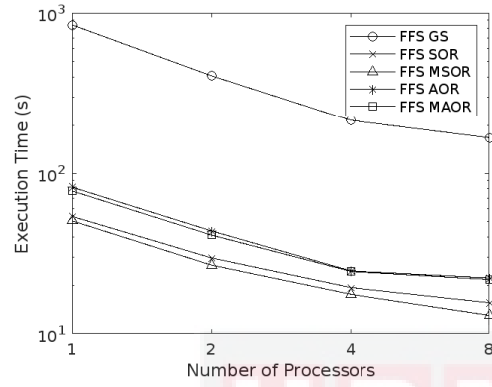


(e)

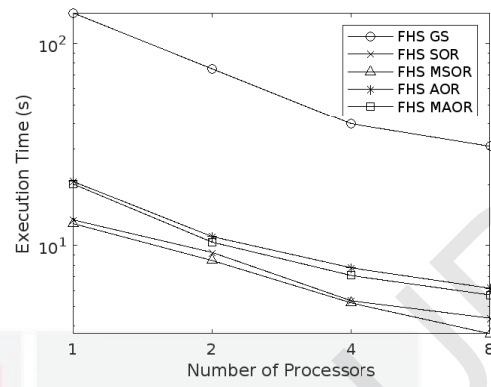


(f)

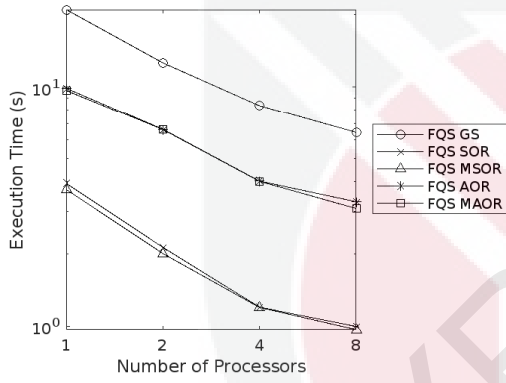
Figure 5.15: The speedup for (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 1.50$.



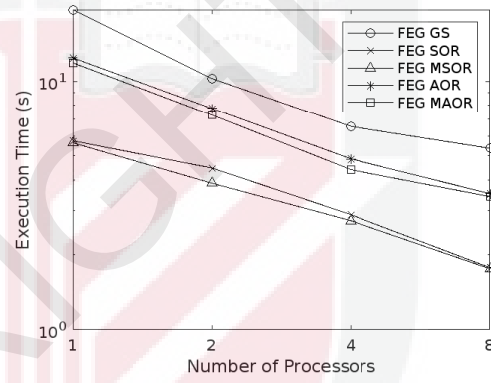
(a)



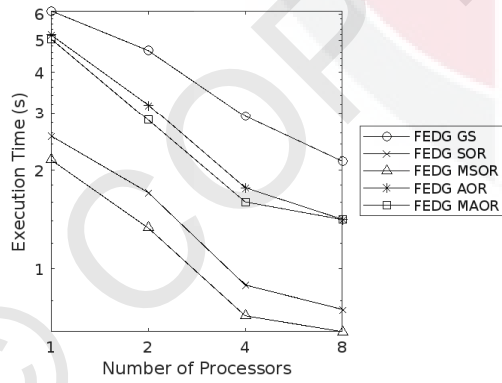
(b)



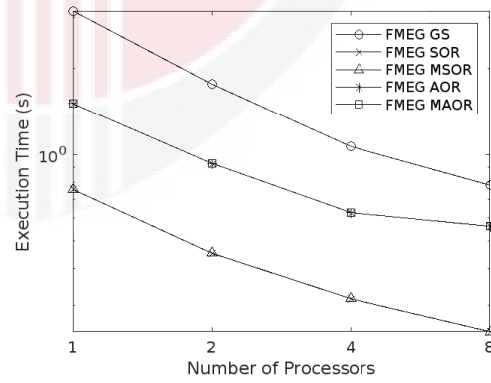
(c)



(d)

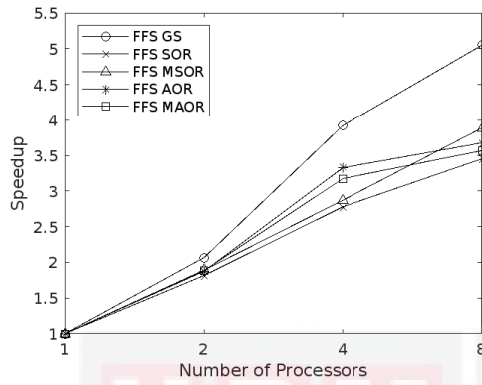


(e)

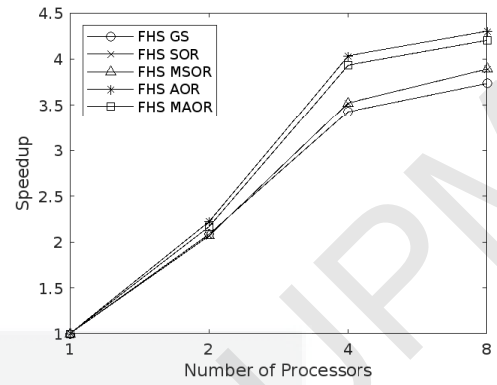


(f)

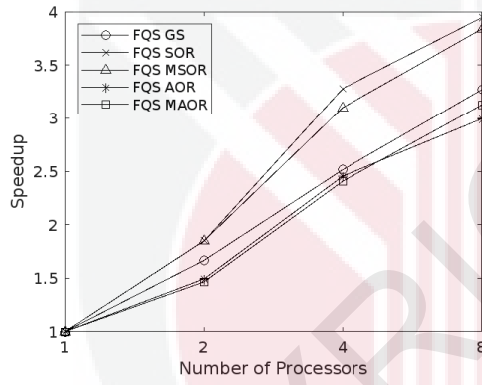
Figure 5.16: The execution time (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 1.75$.



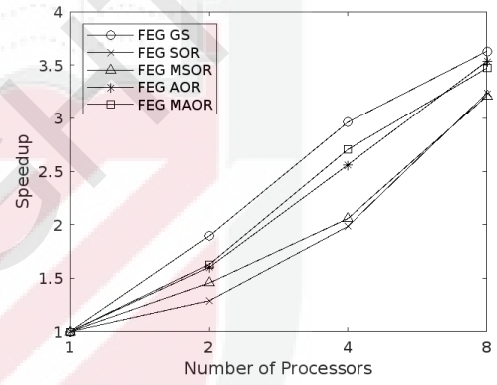
(a)



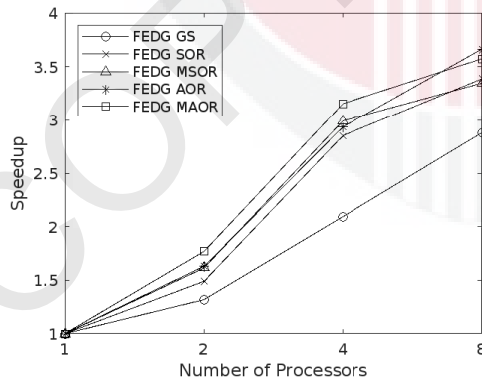
(b)



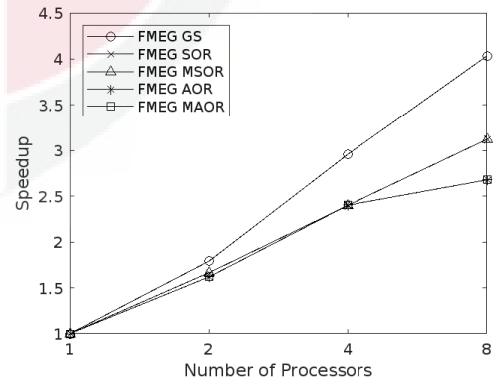
(c)



(d)

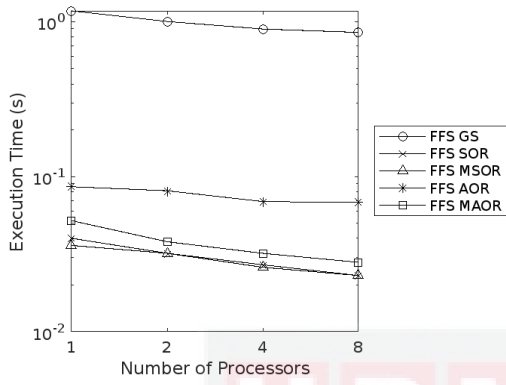


(e)

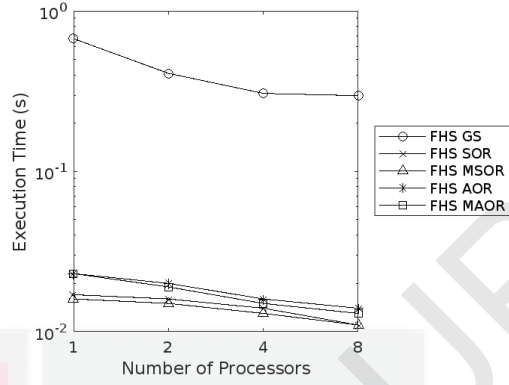


(f)

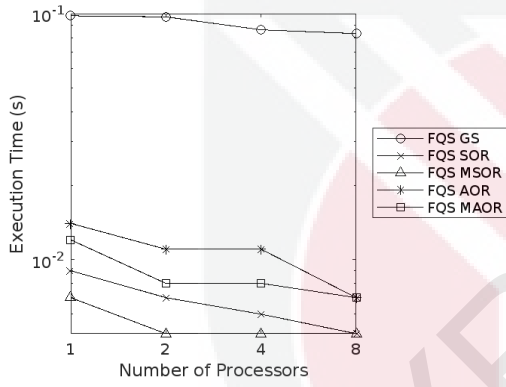
Figure 5.17: The speedup for (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 1.75$.



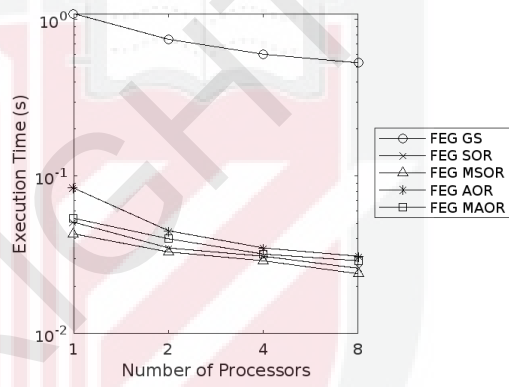
(a)



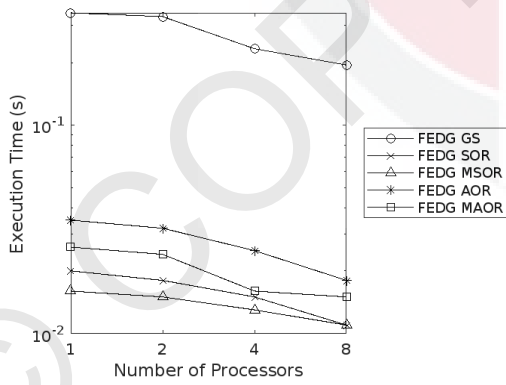
(b)



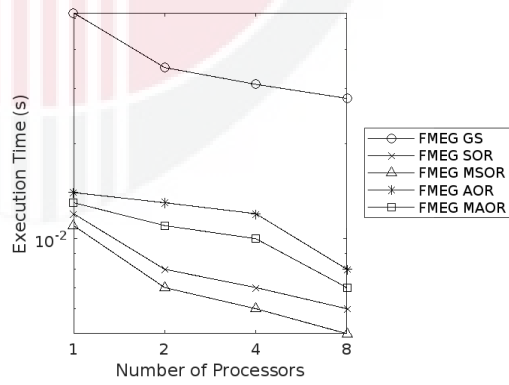
(c)



(d)

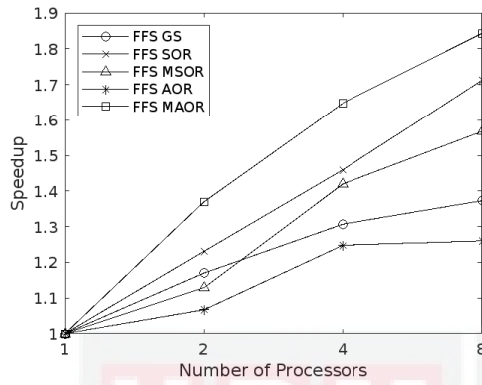


(e)

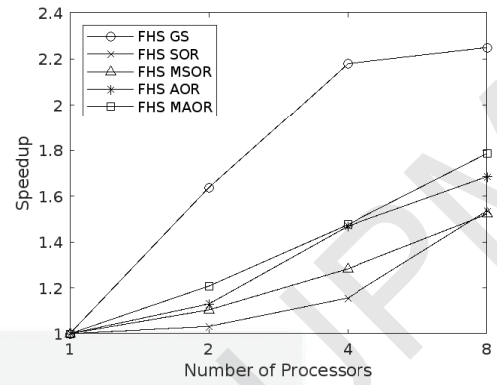


(f)

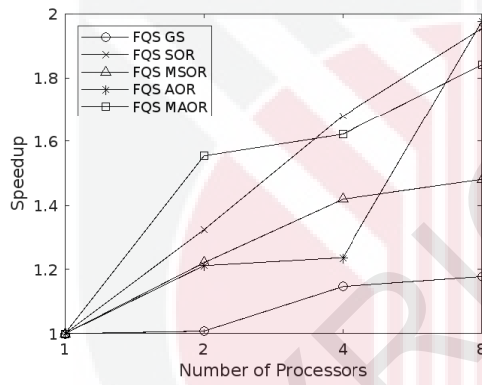
Figure 5.18: The execution time (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 2.00$.



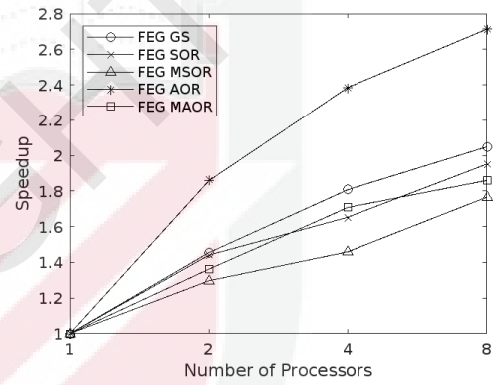
(a)



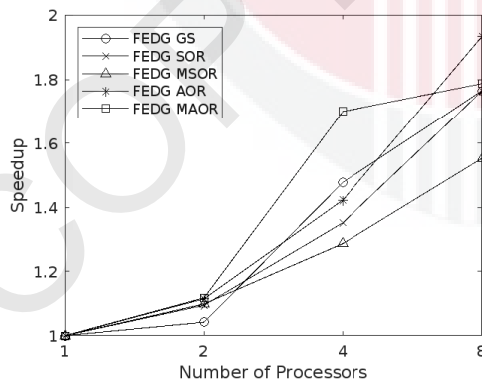
(b)



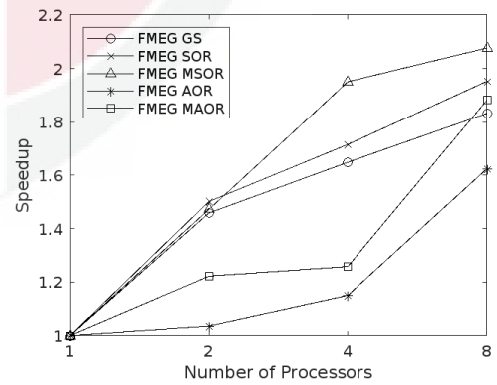
(c)



(d)



(e)



(f)

Figure 5.19: The speedup for (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 3 at $M = 80$ and $\alpha = 2.00$.

Table 5.15: Number of iterations (Ite) and Error for fractional order iterative methods with various relaxation schemes for Problem 4.

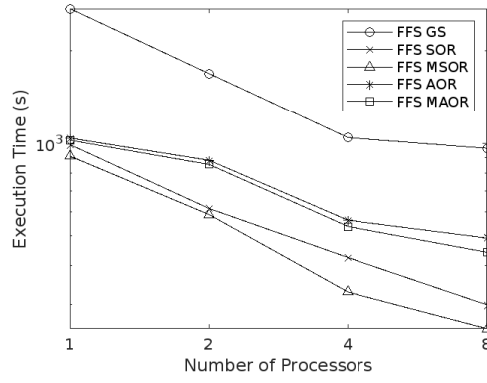
Method	M	Ite	Error	Ite	Error	Ite	Error
		FFS		FHS		FQS	
GS	16	162	4.71E-02	101	4.46E-02	62	4.08E-02
	32	423	2.33E-02	262	2.24E-02	162	2.18E-02
	64	1102	1.16E-02	682	1.13E-02	423	1.12E-02
	128	2865	5.78E-03	1773	5.65E-03	1102	5.69E-03
SOR	16	45	4.71E-02	34	4.46E-02	22	4.08E-02
	32	113	2.33E-02	74	2.24E-02	45	2.18E-02
	64	333	1.16E-02	176	1.13E-02	113	1.12E-02
	128	1015	5.78E-03	550	5.65E-03	333	5.69E-03
MSOR	16	45	4.71E-02	34	4.46E-02	21	4.08E-02
	32	111	2.33E-02	71	2.24E-02	45	2.18E-02
	64	333	1.16E-02	176	1.13E-02	111	1.12E-02
	128	1002	5.78E-03	549	5.65E-03	333	5.69E-03
AOR	16	42	4.71E-02	32	4.46E-02	22	4.08E-02
	32	87	2.33E-02	66	2.24E-02	42	2.18E-02
	64	220	1.16E-02	148	1.13E-02	87	1.12E-02
	128	582	5.78E-03	348	5.65E-03	220	5.69E-03
MAOR	16	40	4.71E-02	31	4.46E-02	21	4.08E-02
	32	84	2.33E-02	63	2.24E-02	40	2.18E-02
	64	216	1.16E-02	142	1.13E-02	84	1.12E-02
	128	578	5.78E-03	338	5.65E-03	216	5.69E-03
		FEG		FEDG		FMEG	
GS	16	53	1.41E-02	43	3.47E-02	31	3.01E-02
	32	79	5.74E-03	67	1.04E-02	53	6.41E-03
	64	103	6.00E-03	93	1.76E-03	79	2.76E-03
	128	115	3.74E-03	109	2.11E-03	103	2.95E-03
SOR	16	23	1.41E-02	19	3.47E-02	15	3.01E-02
	32	34	5.74E-03	28	1.04E-02	23	6.41E-03
	64	47	6.00E-03	41	1.76E-03	34	2.76E-03
	128	60	3.74E-03	54	2.11E-03	47	2.95E-03
MSOR	16	22	1.41E-02	19	3.47E-02	15	3.01E-02
	32	33	5.74E-03	28	1.04E-02	22	6.41E-03
	64	45	6.00E-03	40	1.76E-03	33	2.76E-03
	128	58	3.74E-03	52	2.11E-03	45	2.95E-03
AOR	16	22	1.41E-02	19	3.47E-02	15	3.01E-02
	32	33	5.74E-03	28	1.04E-02	22	6.41E-03
	64	46	6.00E-03	39	1.76E-03	33	2.76E-03
	128	59	3.74E-03	53	2.11E-03	46	2.95E-03
MAOR	16	22	1.41E-02	18	3.47E-02	14	3.01E-02
	32	30	5.74E-03	26	1.04E-02	22	6.41E-03
	64	44	6.00E-03	38	1.76E-03	30	2.76E-03
	128	57	3.74E-03	51	2.11E-03	44	2.95E-03

Table 5.16: Execution time (seconds), speedup and parallel efficiency for all fractional order point iterative methods with various relaxation schemes for Problem 4 at $M = 128$.

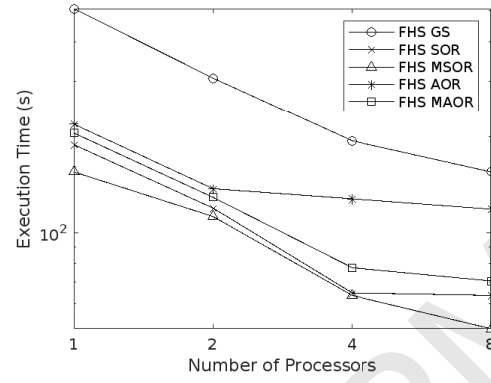
Methods	Execution Time				Speedup				Efficiency			
	T1	T2	T4	T8	S1	S2	S4	S8	E1	E2	E4	E8
FFS GS	2738.160	1687.080	1042.150	961.600	1.000	1.623	2.627	2.848	1.000	0.812	0.657	0.356
FFS SOR	989.042	613.846	425.982	299.404	1.000	1.611	2.322	3.303	1.000	0.806	0.580	0.413
FFS MSOR	907.360	586.720	329.485	250.741	1.000	1.546	2.754	3.619	1.000	0.773	0.688	0.452
FFS AOR	1038.480	879.632	562.527	492.650	1.000	1.181	1.846	2.108	1.000	0.590	0.462	0.263
FFS MAOR	1021.910	852.584	537.000	442.924	1.000	1.199	1.903	2.307	1.000	0.599	0.476	0.288
FHS GS	507.805	307.323	194.400	155.072	1.000	1.652	2.612	3.275	1.000	0.826	0.653	0.409
FHS SOR	188.836	119.472	64.735	63.548	1.000	1.581	2.917	2.972	1.000	0.790	0.729	0.371
FHS MSOR	155.107	112.398	63.584	49.973	1.000	1.380	2.439	3.104	1.000	0.690	0.610	0.388
FHS AOR	219.106	137.406	127.699	118.523	1.000	1.595	1.716	1.849	1.000	0.797	0.429	0.231
FHS MAOR	205.493	129.106	77.653	70.555	1.000	1.592	2.646	2.913	1.000	0.796	0.662	0.364
FQS GS	150.417	98.244	61.458	49.234	1.000	1.531	2.447	3.055	1.000	0.766	0.612	0.382
FQS SOR	53.142	35.970	20.561	15.190	1.000	1.477	2.585	3.499	1.000	0.739	0.646	0.437
FQS MSOR	50.263	30.180	19.808	13.304	1.000	1.665	2.537	3.778	1.000	0.833	0.634	0.472
FQS AOR	87.877	65.185	58.390	48.105	1.000	1.348	1.505	1.827	1.000	0.674	0.376	0.228
FQS MAOR	85.689	64.274	51.732	47.986	1.000	1.333	1.656	1.786	1.000	0.667	0.414	0.223

Table 5.17: Execution time (seconds), speedup and parallel efficiency for all fractional order group iterative methods with various relaxation schemes for Problem 4 at $M = 128$.

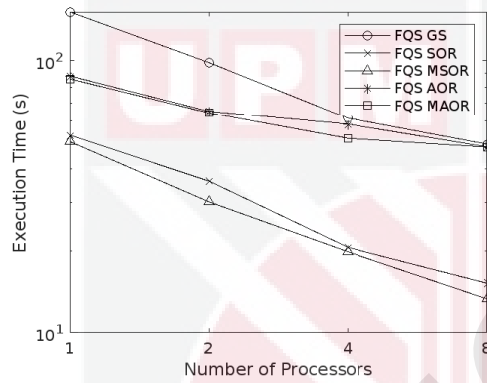
Methods	Execution Time				Speedup				Efficiency			
	T1	T2	T4	T8	S1	S2	S4	S8	E1	E2	E4	E8
FEG GS	75.427	38.740	30.694	28.007	1.000	1.947	2.457	2.693	1.000	0.974	0.614	0.337
FEG SOR	46.464	28.099	18.453	12.154	1.000	1.654	2.518	3.823	1.000	0.827	0.630	0.478
FEG MSOR	41.864	22.060	17.054	11.515	1.000	1.898	2.455	3.636	1.000	0.949	0.614	0.454
FEG AOR	79.871	62.336	43.983	42.025	1.000	1.281	1.816	1.901	1.000	0.641	0.454	0.238
FEG MAOR	75.427	62.336	43.983	42.025	1.000	1.210	1.715	1.795	1.000	0.605	0.429	0.224
FEDG GS	22.886	14.687	11.937	9.684	1.000	1.558	1.917	2.363	1.000	0.779	0.479	0.295
FEDG SOR	20.169	12.488	10.037	8.781	1.000	1.615	2.009	2.297	1.000	0.808	0.502	0.287
FEDG MSOR	19.621	12.009	9.465	5.687	1.000	1.634	2.073	3.450	1.000	0.817	0.518	0.431
FEDG AOR	41.667	20.324	16.490	11.209	1.000	2.050	2.527	3.717	1.000	1.025	0.632	0.465
FEDG MAOR	24.332	18.386	15.255	11.107	1.000	1.323	1.595	2.191	1.000	0.662	0.399	0.274
FMEG GS	4.306	2.421	2.180	1.781	1.000	1.779	1.976	2.418	1.000	0.889	0.494	0.302
FMEG SOR	4.023	2.338	1.371	1.012	1.000	1.721	2.934	3.977	1.000	0.860	0.733	0.497
FMEG MSOR	2.533	2.169	1.286	0.827	1.000	1.168	1.969	3.064	1.000	0.584	0.492	0.383
FMEG AOR	7.722	4.824	3.157	2.090	1.000	1.601	2.446	3.694	1.000	0.800	0.611	0.462
FMEG MAOR	5.728	2.527	2.419	1.793	1.000	2.267	2.368	3.194	1.000	1.134	0.592	0.399



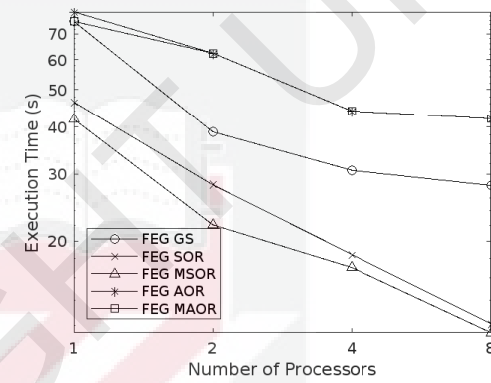
(a)



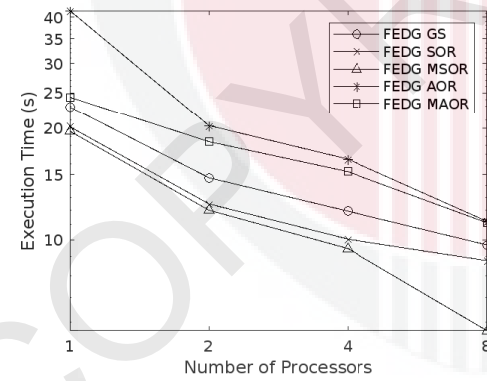
(b)



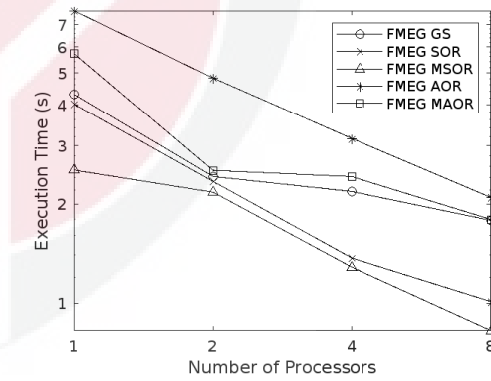
(c)



(d)

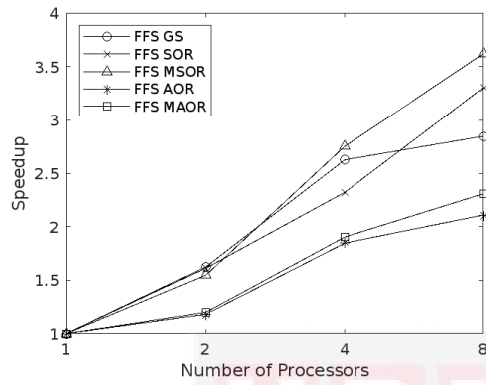


(e)

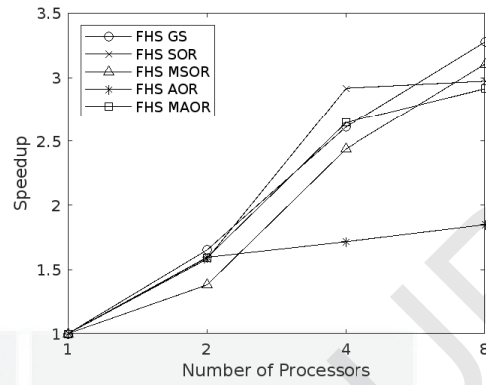


(f)

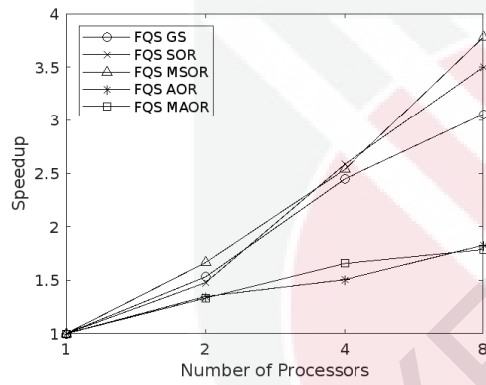
Figure 5.20: The execution time (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 4 at $M = 128$.



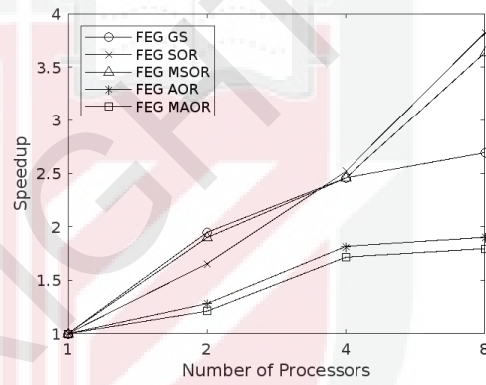
(a)



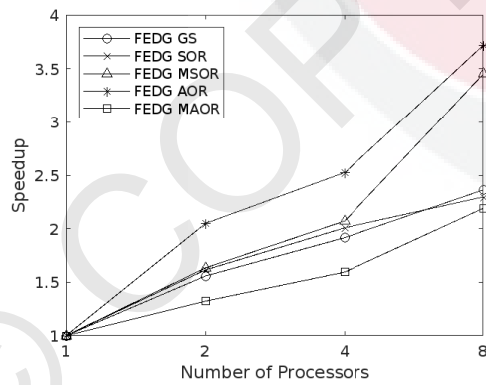
(b)



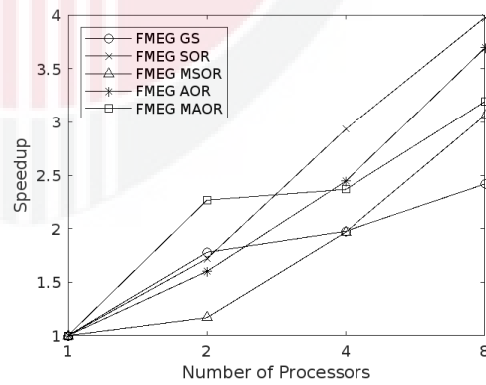
(c)



(d)



(e)

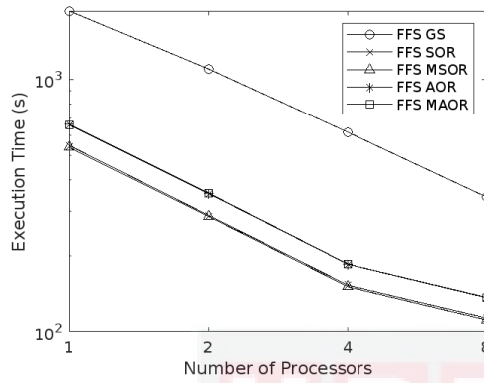


(f)

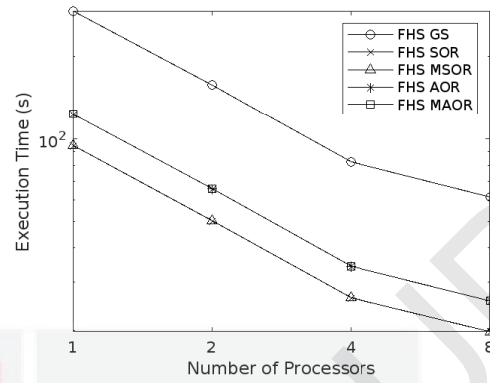
Figure 5.21: The speedup for (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 4 at $M = 128$.

Table 5.18: Number of iterations (Ite) and Error for fractional order iterative methods with various relaxation schemes for Problem 5.

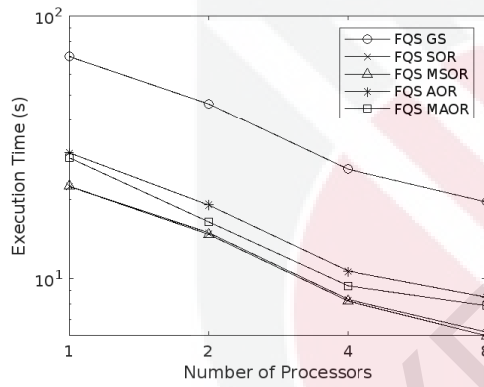
Method	M	Ite	Error	Ite	Error	Ite	Error
		FFS		FHS		FQS	
GS	16	125	6.09E-05	73	1.57E-04	47	1.27E-04
	32	323	1.43E-05	189	7.02E-05	125	2.95E-05
	64	833	3.48E-06	486	3.40E-05	323	7.07E-06
	128	2135	8.58E-07	1241	1.69E-05	833	1.73E-06
SOR	16	39	6.09E-05	26	1.57E-04	19	1.27E-04
	32	89	1.43E-05	54	7.02E-05	39	2.95E-05
	64	267	3.48E-06	130	3.40E-05	89	7.07E-06
	128	751	8.58E-07	396	1.69E-05	267	1.73E-06
MSOR	16	39	6.09E-05	26	1.57E-04	18	1.27E-04
	32	88	1.43E-05	54	7.02E-05	39	2.95E-05
	64	262	3.48E-06	129	3.40E-05	88	7.07E-06
	128	740	8.58E-07	396	1.69E-05	262	1.73E-06
AOR	16	36	6.09E-05	26	1.57E-04	19	1.27E-04
	32	77	1.43E-05	49	7.02E-05	36	2.95E-05
	64	178	3.48E-06	106	3.40E-05	77	7.07E-06
	128	456	8.58E-07	259	1.69E-05	178	1.73E-06
MAOR	16	35	6.09E-05	25	1.57E-04	18	1.27E-04
	32	75	1.43E-05	48	7.02E-05	35	2.95E-05
	64	171	3.48E-06	103	3.40E-05	75	7.07E-06
	128	454	8.58E-07	259	1.69E-05	171	1.73E-06
		FEG		FEDG		FMEG	
GS	16	43	1.18E-04	41	1.53E-04	25	1.30E-04
	32	65	8.18E-05	54	7.47E-05	43	5.62E-05
	64	83	5.22E-05	75	4.47E-05	65	3.97E-05
	128	97	2.92E-05	91	2.68E-05	83	2.57E-05
SOR	16	20	1.18E-04	16	1.53E-04	13	1.30E-04
	32	30	8.18E-05	24	7.47E-05	20	5.62E-05
	64	40	5.22E-05	35	4.47E-05	30	3.97E-05
	128	51	2.92E-05	46	2.68E-05	40	2.57E-05
MSOR	16	19	1.18E-04	16	1.53E-04	13	1.30E-04
	32	29	8.18E-05	24	7.47E-05	19	5.62E-05
	64	38	5.22E-05	34	4.47E-05	29	3.97E-05
	128	49	2.92E-05	44	2.68E-05	38	2.57E-05
AOR	16	20	1.18E-04	15	1.53E-04	13	1.30E-04
	32	29	8.18E-05	24	7.47E-05	20	5.62E-05
	64	39	5.22E-05	35	4.47E-05	29	3.97E-05
	128	50	2.92E-05	45	2.68E-05	39	2.57E-05
MAOR	16	19	1.18E-04	15	1.53E-04	12	1.30E-04
	32	28	8.18E-05	23	7.47E-05	19	5.62E-05
	64	38	5.22E-05	33	4.47E-05	28	3.97E-05
	128	48	2.92E-05	43	2.68E-05	38	2.57E-05



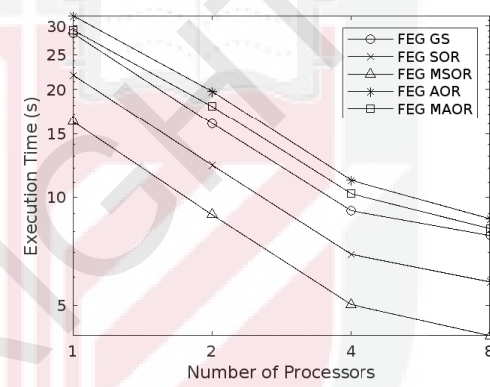
(a)



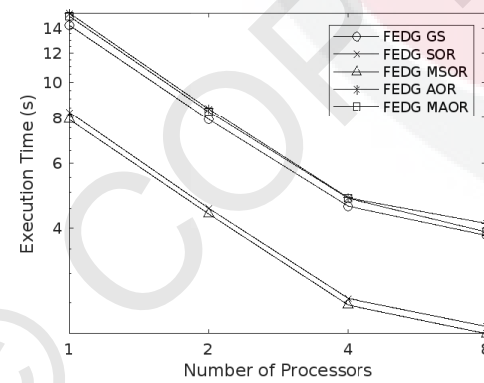
(b)



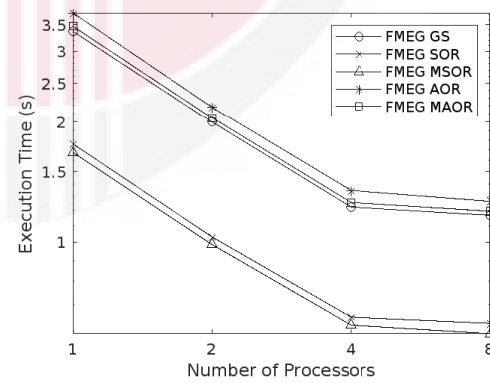
(c)



(d)



(e)



(f)

Figure 5.22: The execution time (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 5 at $M = 128$.

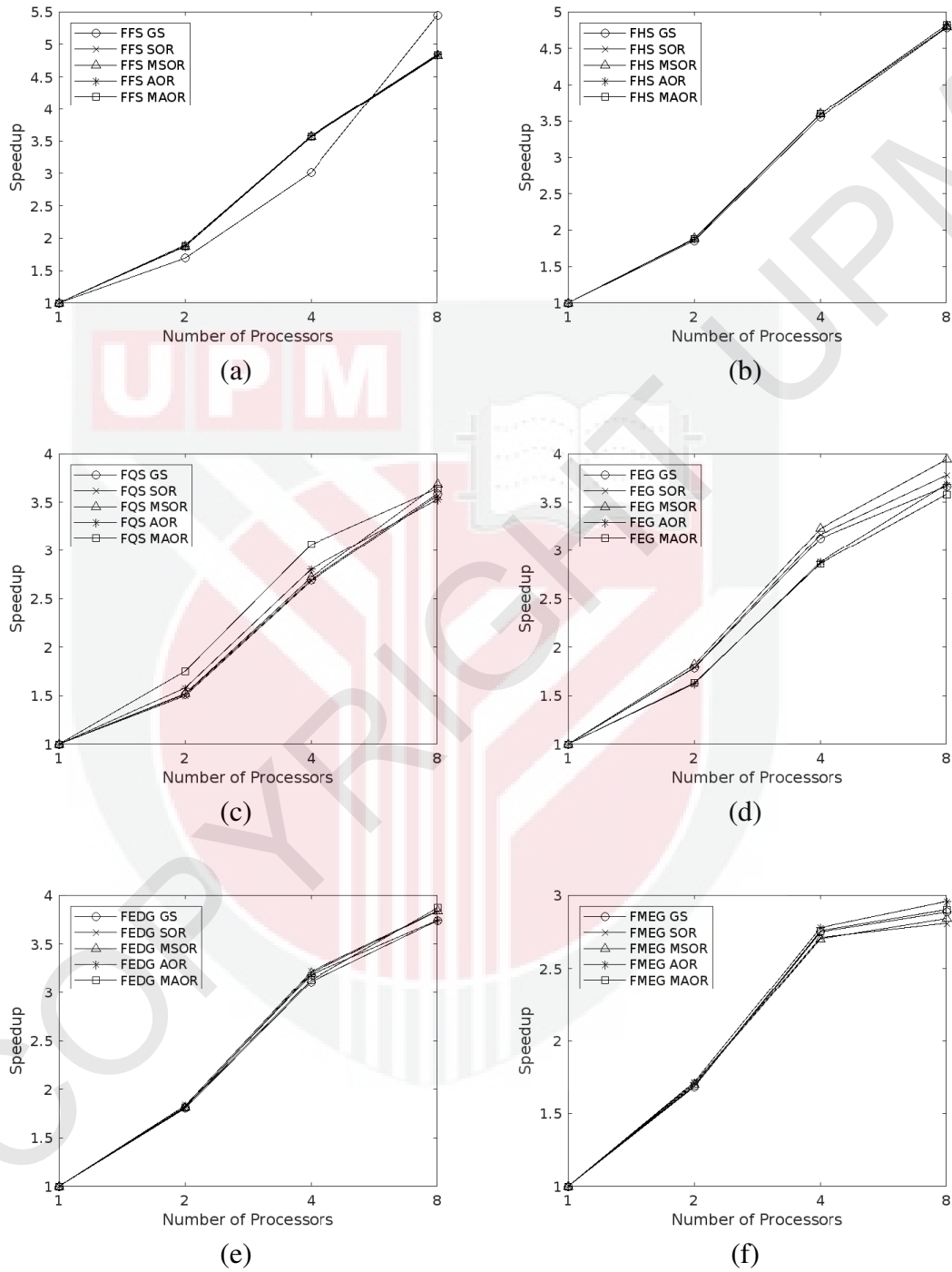


Figure 5.23: The speedup for (in seconds) for (a) FFS; (b) FHS; (c) FQS; (d) FEG; (e) FEDG; and (f) FMEG iterative methods on all relaxation schemes for Problem 5 at $M = 128$.

Table 5.19: Computational costs of fractional order iterative methods with various relaxation schemes for Problems 4 and 5 at $M = 128$.

Methods	Computational Cost	
	Problem 4	Problem 5
FFS GS	12,245,540,025	9,125,384,975
FFS SOR	4,403,781,515	3,258,364,451
FFS MSOR	4,347,378,402	3,210,638,740
FFS AOR	7,397,015,718	5,795,599,944
FFS MAOR	7,346,177,122	5,770,180,646
FHS GS	2,564,538,633	1,795,046,917
FHS SOR	813,319,634	585,603,684
FHS MSOR	811,840,959	585,603,684
FHS AOR	1,491,704,100	1,110,217,437
FHS MAOR	1,448,840,430	1,110,217,437
FQS GS	599,292,862	453,023,305
FQS SOR	186,433,513	149,497,999
FQS MSOR	186,433,513	146,699,854
FQS AOR	352,841,116	285,495,250
FQS MAOR	346,427,224	274,270,939
FEG GS	268,363,540	226,358,812
FEG SOR	143,886,720	122,303,712
FEG MSOR	139,090,496	117,507,488
FEG AOR	341,477,604	289,387,800
FEG MAOR	329,902,092	277,812,288
FEDG GS	88,045,937	73,514,231
FEDG SOR	45,385,542	38,668,926
FEDG MSOR	43,706,388	36,989,772
FEDG AOR	121,613,053	103,263,669
FEDG MAOR	117,025,707	98,676,323
FMEG GS	33,033,348	26,634,068
FMEG SOR	15,861,536	13,510,656
FMEG MSOR	15,189,856	12,838,976
FMEG AOR	36,350,632	30,830,740
FMEG MAOR	34,773,520	30,042,184

Table 5.20: Execution time (seconds), speedup and parallel efficiency for all fractional order point iterative methods with various relaxation schemes for Problem 5 at $M = 128$.

Methods	Execution Time					Speedup				Efficiency			
	T1	T2	T4	T8		S1	S2	S4	S8	E1	E2	E4	E8
FFS GS	1858.680	1098.340	616.411	341.212		1.000	1.692	3.015	5.447	1.000	0.846	0.754	0.681
FFS SOR	546.328	288.486	152.595	113.249		1.000	1.894	3.580	4.824	1.000	0.947	0.895	0.603
FFS MSOR	536.271	285.498	150.386	111.251		1.000	1.878	3.566	4.820	1.000	0.939	0.891	0.603
FFS AOR	662.481	353.467	184.873	136.732		1.000	1.874	3.583	4.845	1.000	0.937	0.896	0.606
FFS MAOR	658.553	351.695	184.619	136.203		1.000	1.873	3.567	4.835	1.000	0.936	0.892	0.604
FHS GS	293.055	157.940	82.486	61.320		1.000	1.855	3.553	4.779	1.000	0.928	0.888	0.597
FHS SOR	94.770	50.308	26.351	19.798		1.000	1.884	3.596	4.787	1.000	0.942	0.899	0.598
FHS MSOR	94.770	50.308	26.351	19.798		1.000	1.884	3.596	4.787	1.000	0.942	0.899	0.598
FHS AOR	123.492	65.877	34.341	25.625		1.000	1.875	3.596	4.819	1.000	0.937	0.899	0.602
FHS MAOR	123.492	65.877	34.341	25.625		1.000	1.875	3.596	4.819	1.000	0.937	0.899	0.602
FQS GS	70.012	46.234	25.977	19.516		1.000	1.514	2.695	3.587	1.000	0.757	0.674	0.448
FQS SOR	22.373	14.921	8.343	6.271		1.000	1.499	2.682	3.568	1.000	0.750	0.670	0.446
FQS MSOR	22.362	14.708	8.220	6.074		1.000	1.520	2.720	3.681	1.000	0.760	0.680	0.460
FQS AOR	29.993	18.998	10.687	8.499		1.000	1.579	2.806	3.529	1.000	0.789	0.702	0.441
FQS MAOR	28.739	16.401	9.389	7.893		1.000	1.752	3.061	3.641	1.000	0.876	0.765	0.455

Table 5.21: Execution time (seconds), speedup and parallel efficiency for all fractional order group iterative methods with various relaxation schemes for Problem 5 at $M = 128$.

Methods	Execution Time				Speedup				Efficiency			
	T1	T2	T4	T8	S1	S2	S4	S8	E1	E2	E4	E8
FEG GS	28.636	16.042	9.162	7.814	1.000	1.785	3.125	3.665	1.000	0.893	0.781	0.458
FEG SOR	21.947	12.259	6.934	5.811	1.000	1.790	3.165	3.777	1.000	0.895	0.791	0.472
FEG MSOR	16.229	8.927	5.029	4.120	1.000	1.818	3.227	3.939	1.000	0.909	0.807	0.492
FEG AOR	32.046	19.746	11.134	8.698	1.000	1.623	2.878	3.684	1.000	0.811	0.720	0.461
FEG MAOR	29.239	17.916	10.218	8.166	1.000	1.632	2.862	3.581	1.000	0.816	0.715	0.448
FEDG GS	14.261	7.910	4.582	3.814	1.000	1.803	3.113	3.739	1.000	0.901	0.778	0.467
FEDG SOR	8.280	4.522	2.572	2.157	1.000	1.831	3.219	3.839	1.000	0.915	0.805	0.480
FEDG MSOR	7.903	4.365	2.467	2.061	1.000	1.811	3.204	3.834	1.000	0.905	0.801	0.479
FEDG AOR	15.386	8.459	4.828	4.112	1.000	1.819	3.187	3.742	1.000	0.909	0.797	0.468
FEDG MAOR	15.080	8.304	4.816	3.897	1.000	1.816	3.131	3.869	1.000	0.908	0.783	0.484
FMEG GS	3.367	2.000	1.224	1.167	1.000	1.683	2.749	2.884	1.000	0.842	0.687	0.361
FMEG SOR	1.761	1.031	0.650	0.627	1.000	1.708	2.711	2.809	1.000	0.854	0.678	0.351
FMEG MSOR	1.677	0.987	0.621	0.591	1.000	1.699	2.702	2.840	1.000	0.849	0.675	0.355
FMEG AOR	3.742	2.181	1.347	1.265	1.000	1.716	2.778	2.957	1.000	0.858	0.695	0.370
FMEG MAOR	3.466	2.051	1.257	1.195	1.000	1.690	2.757	2.901	1.000	0.845	0.689	0.363

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 Conclusion

The early part of this thesis introduced briefly on the Helmholtz equation and fractional Poisson equation along with second, fourth and fractional order discretization on point and group iterative methods. Then, some previously well-known relaxation schemes are presented which are the Gauss-Seidel (GS), Successive Overrelaxation (SOR) and Modified SOR (MSOR) schemes. The thesis will focus on the Modified Accelerated Overrelaxation (MAOR) scheme which is an extension to the Accelerated Overrelaxation (AOR) scheme. This was followed by the fundamentals of parallel computing which focuses on the OpenMP parallel programming. The idea of this thesis is to construct a parallel MAOR iterative method on the different order of discretization with various iterative methods. To assist in the construction, numerous ordering strategies are utilized ensures that the points in the iterative process are independent from each other. This will reduce the execution time of the iterative process while maintaining a similar number of iterations.

First, the thesis implemented the MAOR scheme for solving the Helmholtz equation using second and fourth order iterative methods. The MAOR scheme demonstrated its capability to minimise the spectral radius of the iteration matrix, resulting in fewer iterations for convergence when applied to second and fourth order iterative methods. This advantage is particularly beneficial for the Helmholtz equation, known for its challenges in stability and convergence in iterative solvers. However, the scheme's additional acceleration parameters, while effective for reducing iterations, introduce computational overhead. This results in a higher per-iteration cost compared to simpler methods like SOR and MSOR.

Next, the thesis developed the MAOR scheme for solving the fractional Poisson equation. The extension of the MAOR scheme to fractional Poisson equations highlights its versatility. The iterative framework is suitable for fractional problems, where standard solvers often struggle due to nonlocality and the higher computational complexity of fractional operators. The MAOR scheme retains its advantage in reducing iterations, making it a promising candidate for such applications. However, the computational cost for fractional equations is significantly higher due to the nonlocal terms in fractional derivatives. The MAOR scheme's iterative framework requires further optimisation to handle these costs effectively. Additionally, the accuracy of the scheme may be sensitive to parameter selection in fractional contexts, necessitating advanced strategies for parameter tuning.

Finally, the thesis constructed the parallel algorithms for all the iterative methods. The parallel implementation of the MAOR scheme demonstrated significant reductions in execution time and improvements in speedup and parallel efficiency, particularly for large-scale problems. The ability to distribute computational tasks across multiple processors ensures scalability and practicality for real-world applications.

In a nut shell, the MAOR scheme strikes a balance between convergence speed and computational efficiency, making it a valuable tool for solving large-scale systems. However, its higher computational cost per iteration and complexity are notable drawbacks. While it is not universally the fastest or most accurate method, its strengths in reducing iteration counts and maintaining robust parallel performance make it a competitive choice for certain applications. Careful consideration of problem-specific requirements and computational resources is essential when selecting the MAOR scheme over simpler alternatives like SOR and MSOR. Overall, this thesis has successfully achieved its objective in constructing a parallel MAOR iterative method on the different order of discretization with various iterative methods.

6.2 Future Works

The MAOR scheme has demonstrated its effectiveness for solving stationary equations using second, fourth and fractional discretisations. However, further advancements can broaden its applicability and address complex mathematical and computational challenges. One of the advancements that could significantly enhance the MAOR scheme's utility is by extending it to solve nonlinear systems. Nonlinear problems appear in various fields, including fluid dynamics, elasticity, and reaction-diffusion systems. This adaptation would require the development of iterative frameworks that efficiently linearize the nonlinear terms while maintaining the scheme's convergence properties. The MAOR scheme could also be adapted for equations that evolve over time, such as parabolic or hyperbolic PDEs. This would involve coupling temporal discretisation techniques with the MAOR scheme to ensure stability and accuracy.

Futhermore, the MAOR scheme can be constructed within the finite element framework which offers an opportunity to solve problems on irregular domains or with complex boundary conditions. The finite element methods (FEM) provide flexibility in mesh generation, enabling solutions for geometries that are not well-suited to traditional grid-based discretization. The integration of the MAOR scheme with FEM could lead to more efficient iterative solvers for engineering and scientific applications. Another extension could see the MAOR scheme incorporated into time-fractional derivatives. This could exploit its iterative efficiency while addressing the computational complexity associated with fractional operators. Another advancement for the MAOR scheme is an adaptation to three-dimensional problems. This is crucial for real-world applications such as geophysics, fluid dynamics, and biomedical simulations. Discretizing 3D equations presents challenges, including increased computational demands and memory requirements. However, leveraging the MAOR scheme's potential for iteration reduction and parallel execution could address these challenges effectively.

The proposed future work for the MAOR scheme encompass a range of advancements,

from extending its applicability to nonlinear, timw fractional, and three-dimensional problems to optimising its performance. These developments could establish the MAOR scheme as a versatile and robust tool in computational mathematics, addressing the increasing complexity of modern scientific and engineering problems.



REFERENCES

- Agrawal, O. P. 2002. Solution for a Fractional Diffusion-Wave Equation Defined in a Bounded Domain, Mechanical Engineering and Energy Processes. *Nonlinear Dynamics* 29: 145–155.
- Agrawal, O. P., Defterli, O. and Baleanu, D. 2010. Fractional Optimal Control Problems with Several State and Control Variables. *Journal of Vibration and Control* 16 (13): 1967–1976.
- Akhir, M. K. M., Othman, M., Majid, Z. A., Suleiman, M. and Sulaiman, J. 2012a. An Efficient Quarter-Sweep Modified SOR Iterative Method for Solving Helmholtz Equation. *International Journal of Pure and Applied Mathematics* 80 (5): 751–764.
- Akhir, M. K. M., Othman, M., Sulaiman, J., Majid, Z. A. and Suleiman, M. 2011a. The Four Point-EDGMSOR Iterative Method for Solution of 2D Helmholtz Equations. In *Informatics Engineering and Information Science* (eds. A. Abd Manaf, S. Sahibuddin, R. Ahmad, S. Mohd Daud, and E. El-Qawasmeh), 218–227. Berlin, Heidelberg: Springer Berlin Heidelberg.
- Akhir, M. K. M., Othman, M., Sulaiman, J., Majid, Z. A. and Suleiman, M. 2011b. Half-Sweep Modified Successive OverRelaxation for Solving Two-Dimensional Helmholtz Equations. *Australian Journal of Basic and Applied Sciences* 5 (12): 3033–3039.
- Akhir, M. K. M., Othman, M., Sulaiman, J., Majid, Z. A. and Suleiman, M. 2011c. Numerical Solution of Helmholtz Equation Using a New Four Point EGMSOR Iterative Method. *Applied Mathematical Sciences* 5 (80): 3991–4004.
- Akhir, M. K. M., Othman, M., Suleiman, M., Majid, Z. A., Sulaiman, J., Muthuvalu, M. S. and Aruchuan, E. 2014. QSMSOR Iterative Method for the Solution of 2D Homogeneous Helmholtz Equations. *International Journal of Mathematical Analysis* 8 (51): 2493–2505.
- Akhir, M. K. M., Othman, M., Suleiman, M. and Sulaiman, J. 2012b. A New Octo Modified Explicit Group Iterative Method for the Solution of 2D Elliptic PDEs. *Applied Mathematical Sciences* 6 (31): 1505–1524.
- Akhir, M. K. M., Sulaiman, J., Othman, M., Majid, Z. A., Muthuvalu, M. S. and Aruchuan, E. 2015. Application of Four-Point MEGMSOR Method for the Solution of 2D Helmholtz Equations. *International Journal of Mathematical Analysis* 9 (38): 1847–1862.
- Akhir, M. K. M., Sulaiman, J., Othman, M. and Muthuvalu, M. S. 2016. An Implementation of QSAOR Iterative Method for Non-Homogeneous Helmholtz Equations. *International Journal of Contemporary Mathematical Sciences* 11 (2): 85–96.
- Ali, A. and Ali, N. H. M. 2019a. Explicit Group Iterative Methods for the Solution of Two-Dimensional Time-Fractional Telegraph Equation. In *AIP Conference Proceedings*, 030006. AIP Publishing LLC.

- Ali, A. and Ali, N. H. M. 2019b. On Skewed Grid Point Iterative Method for Solving 2D Hyperbolic Telegraph Fractional Differential Equation. *Advances in Difference Equations* 303.
- Ali, N. H. M. and Fu, N. K. 2007. Modified Explicit Decoupled Group Method in the Solution of 2-D Elliptic PDEs. In *Proceedings of the 12th WSEAS International Conference on Applied Mathematics*, 162–167. Stevens Point, Winsconsin, USA: World Scientific and Engineering Academy and Society (WSEAS).
- Ali, N. H. M. and Pin, F. K. 2012. Modified Explicit Group AOR Methods in the Solution of Elliptic Equations. *Applied Mathematical Sciences* 6 (50): 2465–2480.
- Ali, N. H. M. and Ping, T. W. 2015. New Fourth Order Explicit Group Method in the Solution of the Helmholtz Equation. *International Journal of Mathematical and Computational Sciences* 9 (1): 7–11.
- Ali, N. H. M. and Ping, T. W. 2016. A Comparative Study of High Order Rotated Group Iterative Schemes on Helmholtz Equation. *International Journal of Mathematical and Computational Sciences* 10 (10): 486–491.
- Bagley, R. L. and Torvik, P. J. 1984. On the Appearance of the Fractional Derivative in the Behavior of Real Materials. *Journal of Applied Mechanics* 51: 294–298.
- Bagley, R. L. and Torvik, P. J. 2013. Theoretical Basis for the Application of Fractional Calculus to Viscoelasticity. *Journal of Rheology* 27 (3): 201–210.
- Baharuddin, Sunarto, A. and Dalle, J. 2017. QSAOR Iterative Method for the Solution of Time-Fractional Diffusion Equation. *Journal of Engineering and Applied Sciences* 12 (12): 3220–3224.
- Bailo, R., Carrillo, J. A., Fronzoni, S. and Gómez-Castro, D. 2024. A Finite-Volume Scheme for Fractional Diffusion on Bounded Domains. *European Journal of Applied Mathematics* 1–21.
- Balasim, A. T. and Ali, N. H. M. 2016. Group Iterative Methods for the Solution of Two-Dimensional Time-Fractional Diffusion Equation. In *AIP Conference Proceedings*, 030003. AIP Publishing LLC.
- Balasim, A. T. and Ali, N. H. M. 2017. New Group Iterative Schemes in the Numerical Solution of the Two-Dimensional Time Fractional Advection-Diffusion Equation. *Cogent Mathematics* 4 (1): 1412241.
- Benson, D. A. and Wheatcraft, S. W. 1984. Application of a Fractional Advection Dispersion Equation. *Water Resources Research* 36 (6): 1403–1412.
- Boev, T. and Georgiev, G. 2023. Globally Existing Solutions to the Problem of Dirichlet for the Fractional 3D Poisson Equation. *Fractal and Fractional* 7 (2): 180.
- Borhanifar, A. and Valizadeh, S. 2013. A Fractional Finite Difference Method for Solving the Fractional Poisson Equation Based on the Shifted Grünwald Estimate. *Walailak Journal of Science and Technology* 10 (5): 427–435.

- Chen, Z.-C., Kim, P., Kumagai, T. and Wang, J. 2020. Time Fractional Poisson Equations: Representations and Estimates. *Journal of Functional Analysis* 278 (2): 108311.
- Collatz, L. 1960. *The Numerical Treatment of Differential Equations*. Berlin: Springer.
- Dahalan, A. A., Jusoh, R. M., Saudi, A. and Sulaiman, J. 2020. Path Finding of Indoor Mobile Robot Using Harmonic Potentials via Explicit Decoupled Group Modified Accelerated Over Relaxation Method. *Solid State Science and Technology* 28 (1-2): 42–54.
- Dahalan, A. A. and Saudi, A. 2022. Pathfinding Algorithm Based on Rotated Block AOR Technique in Structured Environment. *AIMS Mathematics* 7 (7): 11529–11550.
- Dahalan, A. A., Saudi, A. and Sulaiman, J. 2019. Autonomous Navigation on Modified AOR Iterative Method in Static Indoor Environment. In *Journal of Physics: Conference Series*, 012020. IOP Publishing.
- Dahalan, A. A., Saudi, A. and Sulaiman, J. 2022. Pathfinding for Mobile Robot Navigation by Exerting the Quarter-Sweep Modified Accelerated Overrelaxation (QS-MAOR) Iterative Approach via the Laplacian Operator. *Modelling and Simulation in Engineering* 2022.
- Eng, J. H., Saudi, A. and Sulaiman, J. 2017. HSAOR Iteration for Poisson Image Blending Problem via Rotated Five-Point Laplacian Operator. *Global Journal of Pure and Applied Mathematics* 13 (9): 5447–5459.
- Eng, J. H., Saudi, A. and Sulaiman, J. 2018. Numerical Evaluation of Quarter-Sweep SOR Iteration for Solving Poisson Image Blending Problem. In *2018 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAET)*, 1–5. IEEE.
- Eng, J. H., Saudi, A. and Sulaiman, J. 2019. Performance Analysis of Solving Poisson Image Blending Problem by Four-Point EGAOR Iterative Method. *Malaysian Journal of Science* 38: 50–61.
- Flynn, M. J. 1972. A Parallel Halfsweep Multigrid Algorithm on the Shared Memory Multiprocessors. *IEEE Transactions on Computers* 21 (9): 948–960.
- Fu, N. K. and Ali, N. H. M. 2008. Performance Analysis of Explicit Group Parallel Algorithms for Distributed Memory Multicomputer. *Parallel Computing* 34: 427–440.
- Goyal, M., Baskonus, H. M. and Prakash, A. 2019. An Efficient Technique for a Time Fractional Model of Lassa Hemorrhagic Fever Spreading in Pregnant Women. *The European Physical Journal Plus* 134 (10): 482.
- Hadjidimos, A. 1978. Accelerated Overrelaxation Method. *Mathematics of Computation* 32: 149–157.

- Hadjidimos, A., Psimarni, A. and Yeyios, A. K. 1992. On the Convergence of the Modified Accelerated Overrelaxation (MAOR) Method. *Applied Numerical Mathematics* 10 (2): 115–127.
- Herrmann, R. 2011. *Fractional Calculus: An Introduction for Physicists*. Hackensack: World Scientific.
- Heydari, M. H., Hooshmandasl, M. R., Maalek, F. M. and Fereidouni, F. 2013. Two Dimensional Legendre Wavelets for Solving Fractional Poisson Equation with Dirichlet Boundary Conditions. *Engineering Analysis with Boundary Elements* 37: 1331–1338.
- Hilfer, R. 2000. *Applications of Fractional Calculus in Physics*. Singapore: World Scientific.
- Hoe, N. Y. and Hasan, M. K. 2013. Investigation of Steady State Problems via Quarter Sweep Schemes. *Sains Malaysiana* 42 (6): 837–844.
- Jajarmi, A. and Baleanu, D. 2018. Suboptimal Control of Fractional-Order Dynamic Systems with Delay Argument. *Journal of Vibration and Control* 24 (12): 2430–2446.
- Khan, M. A. and Ali, N. H. M. 2020. High-Order Compact Scheme for the Two-Dimensional Fractional Rayleigh-Stokes Problem for a Heated Generalized Second-Grade Fluid. *Advances in Difference Equations* 233.
- Khan, M. A., Ali, N. H. M. and Hamid, N. N. A. 2020. A New Fourth-Order Explicit Group Method in the Solution of Two-Dimensional Fractional Rayleigh-Stokes Problem for a Heated Generalized Second-Grade Fluid. *Advances in Difference Equations* 598.
- Kincaid, D. R. and Young, D. M. 1972. The Modified Successive Over Relaxation Method with Fixed Parameters. *Mathematics of Computation* 26: 705–717.
- Kumar, D., Singh, J., Baleanu, D. and Rathore, S. 2018. Analysis of a Fractional Model of the Ambartsumian Equation. *The European Physical Journal Plus* 133: 259.
- Ling, S. T. and Ali, N. H. M. 2013. New High Order Group Iterative Schemes in the Solution of Poisson Equation. *International Journal of Mathematical and Computational Sciences* 7 (12): 1694–1699.
- Ling, S. T. and Ali, N. H. M. 2014. A Comparative Study of a New High Order Iterative Poisson Solver. In *International Conference on Quantitative Sciences and its Applications (ICOQSIA 2014)*, 222–226.
- Ling, S. T. and Ali, N. H. M. 2016. Fourth Order Modified Explicit Decoupled Group Scheme in the Solution of Poisson Equation. In *AIP Conference Proceedings*, 030018. AIP Publishing LLC.
- Ling, W. K., Dahalan, A. A. and Saudi, A. 2020. Mobile Robot Path Navigation in Static Indoor Environment via AOR 9-Point Laplacian Iteration Numerical Technique. *International Journal of Difference Equations* 15 (2): 231–242.

- Lopez, J. C., Fernández, R. B. and León, D. A. S. d. 2018. A Parallel Implementation on CUDA for Solving 2D Poisson's Equation. *Research in Computing Science* 147 (12): 183–191.
- Magin, R. L. 2006. *Fractional Calculus in Bioengineering*. Danbury: Begell House Publishers.
- Mainardi, F. 1997. *Fractals and Fractional Calculus Continuum Mechanics*. Bologna: Springer.
- Mainardi, F. 2010. *Fractional Calculus and Waves in Linear Viscoelasticity: An Introduction to Mathematical Models*. London: Imperial College Press.
- Miller, K. S. and Ross, B. 1993. *An Introduction to the Fractional Calculus and Fractional Differential Equations*. New York: Wiley.
- Muhiddin, F. A., Sulaiman, J. and Sunarto, A. 2019. Four-Point EGSOR Iteration for the Grünwald Implicit Finite Difference Solution of One-Dimensional Time-Fractional Parabolic Equations. In *Journal of Physics: Conference Series*, 012086. IOP Publishing.
- Othman, M. and Abdullah, A. R. 2000a. An Efficient Four Points Modified Explicit Group Poisson Solver. *International Journal of Computer Mathematics* 72 (2): 203–217.
- Othman, M. and Abdullah, A. R. 2000b. An Efficient Parallel Quarter-sweep Point Iterative Algorithm for Solving Poisson Equation on SMP Parallel Computer. *Pertanika Journal of Science & Technology* 8 (2): 161–174.
- Othman, M., Abdullah, A. R. and Evans, D. J. 2006. A Parallel Four Points Modified Explicit Group Algorithm on Shared Memory Multiprocessors. *International Journal of Parallel, Emergent and Distributed Systems (formerly known as Parallel Algorithms and Applications)* 19 (1): 1–9.
- Othman, M. and Sulaiman, J. 2008. A Fast, Parallel Performance of Fourth Order Iterative Algorithm on Shared Memory Multiprocessors (SMP) Architecture. In *Modeling, Simulation and Optimization of Complex Processes* (eds. H. G. Bock, E. Kostina, H. X. Phu, and R. Rannacher), 471–478. Berlin, Heidelberg: Springer Berlin Heidelberg.
- Podlubny, I. 1999. *Fractional Differential Equations*. New York: Academic Press.
- Rakhimov, S. 2010. New Quarter Sweep Based Accelerated Over-Relaxation Iterative Algorithms and their Parallel Implementations in Solving the 2D Poisson Equation. Master's thesis, Universiti Putra Malaysia.
- Saad, N., Sunarto, A. and Saudi, A. 2021. Accelerated Red-Black Strategy for Image Composition Using Laplacian Operator. *International Journal of Computing and Digital Systems* 10 (1): 1085–1095.
- Saad, Y. 2003. *Iterative Methods for Sparse Linear Systems*. Philadelphia: SIAM.

- Samko, S. G., Kilbas, A. A. and Marichev, O. I. 1993. *Fractional Integrals and Derivatives, Theory and Applications*. New York: Gordon and Breach.
- Saudi, A. and Sulaiman, J. 2012a. Laplacian Behaviour-Based Control (LBBC) for the Path Planning of Mobile Robot Via Four Point-EGSOR. *International Journal of Computer and Information Technology* 1 (2): 81–87.
- Saudi, A. and Sulaiman, J. 2012b. Path Planning for Indoor Mobile Robot Using Half-Sweep SOR via Nine-Point Laplacian (HSSOR9L). *Journal of Mathematics* 3 (2): 1–7.
- Saudi, A. and Sulaiman, J. 2012c. Path Planning for Mobile Robot Using 4EGSOR via Nine-Point Laplacian (4EGSOR9L) Iterative Method. *International Journal of Computer Applications* 53 (16): 38–42.
- Saudi, A. and Sulaiman, J. 2012d. Robot Path Planning Using Four Point-Explicit Group Via Nine-Point Laplacian (4EG9L) Iterative Method. *Procedia Engineering* 41: 182–188.
- Saudi, A. and Sulaiman, J. 2016. Path Planning Simulation Using Harmonic Potential Fields Through Four Point-EDGSOR Method via 9-Point Laplacian. *Jurnal Teknologi* 78 (8-2): 12–24.
- Saudi, A. and Sulaiman, J. 2017. Application of Harmonic Functions through Modified SOR (MSOR) Method for Robot Path Planning in Indoor Structured Environment. *International Journal of Imaging and Robotics* 17 (3): 77–90.
- Shah, N. A., El-Zahar, E. R., Aljoufi, M. D. and Chung, J. D. 2021. An Efficient Approach for Solution of Fractional-Order Helmholtz Equations. *Advances in Difference Equations* 14.
- Singh, J., Kumar, D., Baleanu, D. and Rathore, S. 2018. An Efficient Numerical Algorithm for the Fractional Drinfeld-Sokolov-Wilson Equation. *Applied Mathematics and Computation* 335: 12–24.
- Sulaiman, J., Othman, M. and Hasan, M. K. 2010. MEGSOR Iterative Scheme for the Solution of 2D Elliptic PDE's. *International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering* 4 (2): 264–270.
- Sunarto, A. and Sulaiman, J. 2017. Solving Space-Fractional Diffusion Equations by Using HSSOR Method. *Far East Journal of Applied Mathematics* 97 (4): 159–169.
- Sunarto, A. and Sulaiman, J. 2019a. Implementation QSGS Iteration Applied to Fractional Diffusion Equation. In *Journal of Physics: Conference Series*, 012086. IOP Publishing.
- Sunarto, A. and Sulaiman, J. 2019b. Investigation of Fractional Diffusion Equation via QSGS Iterations. In *Journal of Physics: Conference Series*, 012014. IOP Publishing.
- Sunarto, A., Sulaiman, J. and Saudi, A. 2014a. Full-Sweep SOR Iterative Method to Solve Space-Fractional Diffusion Equations. *Australian Journal of Basic and Applied Sciences* 8 (24): 153–158.

- Sunarto, A., Sulaiman, J. and Saudi, A. 2014b. Implicit Finite Difference Solution for Time-Fractional Diffusion Equations Using AOR Method. In *Journal of Physics: Conference Series*, 012032. IOP Publishing.
- Sunarto, A., Sulaiman, J. and Saudi, A. 2015. Numerical Solution for Solving Space-Fractional Diffusion Equations Using Half-Sweep Gauss-Seidel Iterative Method. In *Proceeding of International Conference on Electrical Engineering, Computer Science and Informatics (EECSI 2015)*, 22–26. Palembang, Indonesia.
- Sunarto, A., Sulaiman, J. and Saudi, A. 2016a. Application of the Full-Sweep AOR Iteration Concept for Space-Fractional Diffusion Equation. In *Journal of Physics: Conference Series*, 012019. IOP Publishing.
- Sunarto, A., Sulaiman, J. and Saudi, A. 2016b. Caputo's Implicit Solution of Time-Fractional Diffusion Equation Using Half-Sweep AOR Iteration. *Global Journal of Pure and Applied Mathematics* 12 (4): 3469–3479.
- Sunarto, A., Sulaiman, J. and Saudi, A. 2016c. Implementation of the Half-Sweep AOR Iterative Algorithm for Space-Fractional Diffusion Equations. *Jurnal Teknologi* 78 (6-4): 7–12.
- Sunarto, A., Sulaiman, J. and Saudi, A. 2018. Caputo's Implicit Solution of Space-Fractional Diffusion Equations by QSSOR Iteration. *Advanced Science Letters* 24 (3): 1927–1931.
- Tejado, I., Pérez, E. and Valério, D. 2020. Fractional Derivatives for Economic Growth Modelling of the Group of Twenty: Application to Prediction. *Mathematics* 8 (1): 50.
- Xie, J., Huang, Q., Zhao, F. and Gui, H. 2017. Block Pulse Functions for Solving Fractional Poisson Type Equations with Dirichlet and Neumann Boundary Conditions. *Boundary Value Problems* 32.
- Young, D. M. 1971. *Iterative Solution of Large Linear Systems*. New York: Academic Press.
- Yousif, W. S. and Martins, M. M. 2008. Explicit De-couple Group AOR Method for Solving Elliptic Partial Differential Equations. *Neural, Parallel and Scientific Computations* 16 (4): 531–542.
- Zhou, J., Pu, Y. and Liao, K. 2010. *Fractional Order Calculus Principle and Its Application in Latest Single Analysis and Processing*. Beijing: Beijing Science Press.

BIODATA OF THE STUDENT

The student, **NIK AMIR SYAFIQ NIK MAZLAN**, was born on Monday June 17th 1991, in the city of Kuantan of Pahang State of Malaysia. He received his Bachelor of Sciences in Computational and Theoretical Science from International Islamic University Malaysia and his Master of Science degree in Computational Mathematics, at Universiti Putra Malaysia. He is currently pursuing his Ph. D. at the same university in the same field.



LIST OF PUBLICATIONS

The following are the list of publications that arise from this study.

- Syafiq, N. A.,** Othman, M., Senu, N. and Abdul Hamid N. A. W. 2024. A New Fractional EG Modified AOR (FEGMAOR) Method with Shifted Grünwald Estimate for the Solution of the Fractional Poisson Equation. *International Journal of Computer Mathematics: Computer Systems Theory*. (Submitted)
- Syafiq, N. A.,** Othman, M., Senu, N. and Abdul Hamid N. A. W. 2024. A New Efficient Fractional Group Iterative Methods for Solving the Fractional Poisson Equation. *Numerical Methods for Partial Differential Equations*. (Submitted)
- Syafiq, N. A.,** Othman, M., Senu, N. and Abdul Hamid N. A. W. 2024. A Parallel Fractional Explicit Group Modified AOR Iterative Method for Solving Fractional Poisson Equation with Multi-core Architecture. *Journal of Parallel and Distributed Computing*. 185, 104798
- Syafiq, N. A.,** Othman, M., Senu, N. and Ismail, F. 2022. A Modified Accelerated Overrelaxation (MAOR) Scheme for Solving the Shifted Grünwald Estimation Fractional Poisson Equation. *International Journal of Applied Mathematics*, 35 (5)
- Syafiq, N. A.,** Othman, M., Senu, N. and Ismail, F. 2021. A Preliminary Study on Parallel Second Order Modified AOR Iterative Method for Solving Helmholtz Equation with Multi-core Architecture. *The 17th IMT-GT International Conference on Mathematics, Statistics and their Applications*
- Syafiq, N. A.,** Othman, M., Senu, N. and Ismail, F. 2019. New explicit group modified accelerated overrelaxation (EGMAOR) in the solution of stationary partial differential equations. In *AIP Conference Proceedings* 2266, 040003.
- Syafiq, N. A.,** Othman, M., Senu, N. and Ismail, F. 2019. An experimental study of the modified accelerated overrelaxation (MAOR) scheme on stationary helmholtz equation. In *Journal of Physics: Conference Series* 1366, 012093.

