

DEVELOPMENT OF A NON-OVERLAPPING DOMAIN DECOMPOSITION METHOD FOR PROBLEMS WITH BOUNDARY LAYER

A thesis submitted to the University of Manchester for the degree of
Doctor of Philosophy
in the Faculty of Science and Engineering

2022

Hongru LI
Department of Mechanical, Aerospace and Civil Engineering

Contents

Contents	2
List of figures	5
List of tables	8
List of publications	9
Terms and abbreviations	10
Abstract	16
Declaration of originality	18
Copyright statement	19
Acknowledgements	20
1 Introduction and literature review	22
1.1 Near-wall turbulence challenge	22
1.2 Near-wall domain decomposition	27
1.3 Outline of the thesis	30
2 Derivation of non-local domain decomposition algorithms	32
2.1 Robin-type interface boundary condition	33
2.2 Domain decomposition method	35
2.3 NDD in application to HRN and LRN models	38
2.4 Non-local DR type domain decomposition method	41
2.4.1 Approximated near-wall domain decomposition	42
2.4.2 DR type ENDD algorithm	43
2.5 Non-local RR type domain decomposition method	46
2.5.1 Introduction of Steklov-Poincaré operators	46
2.5.2 RR type ENDD algorithms	48
2.6 Summary	52
3 Convergence rate calculation of NDD algorithms	53
3.1 Convergence rate calculation of RR-SP0	54
3.1.1 Calculation via separation of variables	54
3.1.2 Calculation of RR-SP0: parallel version	58

3.1.3	Calculation via Fourier transform	59
3.1.4	Calculation of symbol of SP operators	63
3.2	Convergence rate calculation of DR algorithm	65
3.3	Generalised convergence rate calculation of RR-SP0	67
3.4	Convergence rate calculation of RR-SP1 and RR-SP2	73
3.4.1	Calculation of RR-SP1	73
3.4.2	Calculation of RR-SP2	79
3.5	Summary	83
4	Convergence analysis of NDD algorithms	84
4.1	Convergence analysis of RR-SP0 without boundary effect	85
4.1.1	Analysis of RR-SP0	86
4.1.2	Poisson's equation test	89
4.1.3	Comparison between parallel and sequential algorithms	90
4.2	Convergence analysis of RR-SP2 without boundary effect	92
4.2.1	Analysis of RR-SP2: approximation 1	92
4.2.2	Analysis of RR-SP2: approximation 2	96
4.3	Convergence analysis of conventional RR algorithm	99
4.4	Boundary effect analysis of RR-SP0	102
4.4.1	Modelling of boundary effect	102
4.4.2	Model study of boundary effect	107
4.4.3	Test case to evaluate boundary effect	116
4.5	Convergence rate calculation of RR-SP1 and RR-SP2 with boundary effect . .	119
4.5.1	Calculation of RR-SP1 with boundary effect	119
4.5.2	Calculation of RR-SP2 with boundary effect	123
4.6	Estimated analysis of boundary effect in RR-SP1 and RR-SP2	124
4.6.1	Key terms of boundary effect	125
4.6.2	Boundary effect evaluation of RR-SP1	125
4.6.3	Boundary effect evaluation of RR-SP2	126
4.7	Summary	128
5	Numerical tests	130
5.1	Discrete scheme and coding language	131
5.2	Poisson's equation	133
5.3	Model equations	134
5.3.1	Model equation 1	134
5.3.2	Model equation 2	144
5.4	Floating interface boundary method	149
5.5	Summary	151
6	Analysis and implementation of GMRES	155
6.1	Introduction of GMRES as a projection method	155
6.2	Derivation of GMRES	157

6.2.1	Arnoldi algorithm	157
6.2.2	Full orthogonalization method	159
6.2.3	Standard GMRES and restarted GMRES	160
6.3	Solution to the least square problem and practical discussion	162
6.3.1	Solving the least square problem	162
6.3.2	Compressed row storage	165
6.4	Convergence analysis	166
6.4.1	General analysis using Chebyshev Polynomial	166
6.4.2	Eigenvalue analysis of GMRES convergence	167
6.5	ILU(0) preconditioning of GMRES	173
6.6	Summary	176
7	Conclusions and recommendations	178
7.1	Summary of the work	178
7.2	Future work	180
7.2.1	Theoretical future work	180
7.2.2	Numerical future work	181
	References	182
	Appendices	191
A	Convergence rate calculation of DN and NN algorithms	192

List of figures

3.1	Convergence rate of DR algorithm with various A , $\theta = 1$, $B = 1$	68
4.1	Profile of \hat{S}_1 applied to Poisson's equation. It is known that $\hat{S}_1(\omega) = \omega \coth(\omega A)$, and the profile is plotted against various ω and A	86
4.2	Profile of \hat{S}_2 applied to Poisson's equation. It is known that $\hat{S}_2(\omega) = \omega \coth(\omega(B - A))$, and the profile is plotted against various ω and A	87
4.3	Energy spectrum of $1, x, x^2$ and x^3	88
4.4	Profile of $\Delta s_1(\omega)\Delta s_2(\omega)$ with various A	89
4.5	Profile of $S_1 I, S_2 I$ applied to Poisson's equation, $A = \frac{7}{8}$	91
4.6	Profile of $\hat{S}_{1\omega\omega}(\omega)$ with various A . The profile of $\hat{S}_{2\omega\omega}(\omega)$ could also be depicted by this figure: $\hat{S}_{2\omega\omega}(\omega) _{A^*} = \hat{S}_{1\omega\omega}(\omega) _{(1-A^*)}$	92
4.7	Profile of $\hat{S}_{1\omega}(\omega)$ with various A . The profile of $\hat{S}_{2\omega\omega}(\omega)$ could also be depicted by this figure: $\hat{S}_{2\omega\omega}(\omega) _{A^*} = \hat{S}_{1\omega\omega}(\omega) _{(1-A^*)}$. Every line has a 'turning point' where the line turns to flat. The point value decreases with increasing A . For $A = 7/8$, the point value is about 2π ; for $A = 1/8$, the point value is about 6π or even larger.	93
4.8	Profile of $\hat{S}_{1\omega\omega}(0)\hat{S}_{2\omega\omega}(0)$. Since $\hat{S}_{1\omega\omega A}(0) = \hat{S}_{2\omega\omega 1-A}(0)$, the profile is symmetric along $A = 4/8$, so the cases for $A > 4/8$ are not plotted.	95
4.9	Convergence rate of RR-SP0 and RR-SP2 with various A	96
4.10	Profile of $\hat{S}_1(\omega) := \omega \coth(\omega A)$ with various A . If ω^* is smaller than $\frac{1}{2}\omega^t$, the mean value ω^* is approximately equal to $\frac{1}{2}\omega$ because $\hat{S}_{1\omega}(\omega)$ rises almost linearly before ω^t . Therefore, $\eta_i \approx 1/2$ in this case.	99
4.11	Profile of $\Delta s_{1\omega\omega}(\omega)\Delta s_{2\omega\omega}(\omega)$. The definition of $\Delta s_{i\omega\omega}(\omega)$ is $\Delta s_{i\omega\omega}(\omega) := \hat{S}_{i\omega\omega}(\omega) - \hat{S}_{i\omega\omega}(0)$. By cutting off the part of $\omega > \omega^t$, the figure could show $\Delta s_{1\omega\omega}(\omega^{**}) \cdot \Delta s_{2\omega\omega}(\omega^{**})$	100
4.12	Profile of $\Delta\sigma_1\Delta\sigma_2$. In Lui's case, $\lambda = \sigma_1 = \sigma_2 = 6.794$. In otherwise case, $\lambda = \sigma_1 = \sigma_2 = S_1^* I$	101
4.13	Convergence history of Lui's algorithm in application to solving a Poisson's equation with three different λ , $A = 1/8$	103
4.14	Convergence history of Lui's algorithm in application to solving a Poisson's equation with three different λ , $A = 3/8$	103
4.15	Convergence history of Lui's algorithm in application to solving a Poisson's equation with three different λ , $A = 7/8$	104
4.16	Profile of $CR_i(\omega)$, which is always non-positive and decreasing with rising ω . .	108

4.17	Fourier transform of polynomials, real part. One model line is plotted for the case of order 2 according to Eq.(4.35). The parameters are: $L_r = 0.8, N_r = 0.75, \omega_0^r = -\pi/3$	109
4.18	Fourier transform of polynomials, imaginary part. One model line is plotted for the case of order 2 according to Eq.(4.36). The parameters are: $L_i = 0.85, N_i = 0.9, \omega_0^i = 5\pi/12, \xi = 0.9$	109
4.19	Fourier transform of $P_i(x)$, real part. The first three lines are plotted according to Eq.(4.24). In addition, one model function for $\alpha = 1, \beta = 1$ case is also plotted according to Eq.(4.37) with the following parameters: $l_r = 1.8, n_r = 1.3, \omega_{0p}^r = -\pi/5$. It fits the original function well.	110
4.20	Fourier transform of $P_i(x)$, imaginary part. The first three lines are plotted according to Eq.(4.25). In addition, one model function for $\alpha = 1, \beta = 1$ case is also plotted according to Eq.(4.38) with the following parameters: $l_i = 1.3, n_i = 1.1, \omega_{0p}^i = \pi/6$. It fits the original function well.	111
4.21	Profile of $y_1 = 1.5(\Phi_1 + \Phi_2)$ and $y_2 = 1.5(\Phi_3 + \Phi_4)$ as a function of ω . The magnitude of both functions are mainly less than $y = \pm\omega$ when $\omega > \pi$. $V_i(\omega)$ is assumed to be $3 \sin(\omega)$	115
4.22	Profile of $y_1 = 1.5(\Phi_1 + \Phi_2)$ and $y_2 = 1.5(\Phi_3 + \Phi_4)$ as a function of ω . The magnitude of both functions are mainly less than $y = \pm\omega$ when $\omega > \pi$. $V_i(\omega)$ is assumed to be $3 \cos(\omega)$	115
4.23	Profile of $S_1 I$ applied to a Poisson's equation, $A = \frac{1}{8}, \frac{3}{8}$ and $\frac{7}{8}$	117
4.24	Profile of $S_2 I$ applied to a Poisson's equation, $A = \frac{1}{8}, \frac{3}{8}$ and $\frac{7}{8}$	117
4.25	Profile of $\hat{P}_i(\omega)$, real part.	118
4.26	Profile of $\hat{P}_i(\omega)$, imaginary part.	118
4.27	Profile of P_1^l and P_1^q in a Poisson test, $A = 1/8$. $P_1^l := S_2 x - S_2 I \cdot x - 0$, $P_1^q = (S_2 x^2 - 2x S_2 x + x^2 S_2 I) - (S_2^* x^2 - x^2 S_2^* I)$. $S_2^* x^2$ is obtained numerically with boundaries in x direction set at far place ($x = \pm 5$).	119
4.28	Profile of P_2^l and P_2^q in a Poisson test, $A = 1/8$. $P_2^l := S_1 x - S_1 I \cdot x - 0$, $P_2^q = (S_1 x^2 - 2x S_1 x + x^2 S_1 I) - (S_1^* x^2 - x^2 S_1^* I)$. $S_1^* x^2$ is obtained numerically with boundaries in x direction set at far place ($x = \pm 5$).	120
4.29	Numerator of the standard convergence rate CR_2 of RR-SP0(1) and RR-SP2. The magnitude of the former is always no less than the latter. The numerator of $CR_2(SP0) := \frac{1}{A} - \hat{S}_1(\omega)$, and the numerator of $CR_2(SP2) := \frac{1}{A} - \hat{S}_1(\omega) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0)$	128
5.1	Convergence history of Lui's algorithm ($\lambda = 6.794$) and (parallel) RR-SP0 in comparison to solve a Poisson's equation. $A = \frac{1}{8}, \frac{7}{8}$	135
5.2	Convergence history of (sequential) RR-SPN (N=0,1,2) and DR algorithms in application to solving a Poisson's equation. $A = \frac{1}{8}$	135
5.3	Convergence history of (sequential) RR-SPN (N=0,1,2) and DR algorithms in application to solving a Poisson's equation. $A = \frac{3}{8}$	136
5.4	Convergence history of (sequential) RR-SPN (N=0,1,2) and DR algorithms in application to solving a Poisson's equation. $A = \frac{7}{8}$	136

5.5	Convergence history of (sequential) RR-SPN (N=0,1,2) and DR algorithms in application to solving a model equation. $\alpha = 1$	139
5.6	Convergence history of (sequential) RR-SPN (N=0,1,2) and DR algorithms in application to solving a model equation. $\alpha = 10^2$	139
5.7	Convergence history of (sequential) RR-SPN (N=0,1,2) and DR algorithms in application to solving a model equation. $\alpha = 10^4$	140
5.8	The profile of $S_i I$ in application to a model equation, $\alpha = 1$	142
5.9	The profile of $S_i I$ in application to a model equation, $\alpha = 10^2$	142
5.10	The profile of $S_i I$ in application to a model equation, $\alpha = 10^4$	143
5.11	The profile of P_1^l and P_1^q in application to the model equation, $\alpha = 10^2$. $P_1^l := S_2 x - S_2 I \cdot x - 0$, $P_1^q := (S_2 x^2 - 2x S_2 x + x^2 S_2 I) - (S_2^* x^2 - x^2 S_2^* I)$. $S_2^* x^2$ is obtained numerically with boundaries in x direction set at far place ($x = \pm 2$). .	143
5.12	The profile of P_2^l and P_2^q in application to the model equation, $\alpha = 10^2$. $P_2^l := S_1 x - S_1 I \cdot x - 0$, $P_2^q := (S_1 x^2 - 2x S_1 x + x^2 S_1 I) - (S_1^* x^2 - x^2 S_1^* I)$. $S_1^* x^2$ is obtained numerically with boundaries in x direction set at far place ($x = \pm 2$). .	144
5.13	The profile of P_1^l and P_1^q in application to the model equation, $\alpha = 10^4$. $P_1^l := S_2 x - S_2 I \cdot x - 0$, $P_1^q := (S_2 x^2 - 2x S_2 x + x^2 S_2 I) - (S_2^* x^2 - x^2 S_2^* I)$. $S_2^* x^2$ is obtained numerically with boundaries in x direction set at far place ($x = \pm 2$). .	145
5.14	The profile of P_2^l and P_2^q in application to the model equation, $\alpha = 10^4$. $P_2^l := S_1 x - S_1 I \cdot x - 0$, $P_2^q := (S_1 x^2 - 2x S_1 x + x^2 S_1 I) - (S_1^* x^2 - x^2 S_1^* I)$. $S_1^* x^2$ is obtained numerically with boundaries in x direction set at far place ($x = \pm 2$). .	146
5.15	The contour plot of the solution to $\alpha = 10^4$ model equation. The solution mainly changes with x	146
5.16	The divergence history of RR-SP0(1)* in application to the model equation, $\alpha = 10^6$. The convergence of V is speculated to be achieved within the first 5 iterations.	147
5.17	Solution profile of the model equation ($Re = 10^6$). $\alpha = 10^2$	152
5.18	Solution profile of the model equation ($Re = 10^6$). $\alpha = 10^4$	152
5.19	Solution profile of the model equation ($Re = 10^6$). $\alpha = 10^6$	153
5.20	Profile of e_1 and e_2 , which are defined in Eqs.(5.11) and (5.12).	153

List of tables

4.1	Convergence iterations of RR-SP0 in application to a Poisson's equation case. .	90
4.2	Convergence iterations of Lui's algorithm in application to a Poisson's equation.	102
4.3	Parameters of the modelled pike functions.	119
5.1	Convergence iterations of (parallel) RR-SP0, RR-SP1 and RR-SP2 in application to a Poisson's equation.	133
5.2	Convergence iterations of RR-SPN,DR and Lui's algorithm in application to solving a Poisson's equation.	134
5.3	Convergence iteration of (sequential) RR-SP0, RR-SP1, RR-SP2 and DR in application to a model equation, $A = 10^{-3}$	138
5.4	Convergence iteration of (sequential) RR-SPN (N=0,1,2) and RR-SPN* in comparison to solve a model equation, $A = 10^{-3}$	138
5.5	Convergence iteration of RR-SP0, RR-SP1, RR-SP2 and DR algorithms in application to a model equation, $A = 5 \times 10^{-4}$	145
5.6	Convergence iteration of a model equation case. Note 'NA' denotes not applicable.	148
5.7	Convergence iteration of RR-SPN and RR-SPN* in application to a model equation, $\alpha = 10^6$. Note 'DIV' denotes divergence.	149

List of publications

Utyuzhnikov, Sergey V., and Hongru Li. "Domain decomposition with nonlocal interface boundary conditions." *Journal of Computational and Applied Mathematics* 421 (2023): 114847.

Terms and abbreviations

Acronyms

Computational Fluid Dynamics	CFD
Navier-Stokes	NS
Direct Numerical Simulation	DNS
Large Eddy Simulation	LES
Reynolds-averaged Navier-Stokes	RANS
Low Reynolds Number	LRN
High Reynolds Number	HRN
Eddy-Viscous Model	EVM
Non-Linear Eddy Viscous Model	NLEVMM
Near-wall Domain Decomposition	NDD
Domain Decomposition	DD
Interface Boundary Condition	IBC
Thin Boundary Layer	TBL
Analytical Wall Function	AWF
Generalized Wall Function	GWF
Robin-type Wall Function	RWF

Scalable Wall Function	SWF
Approximated Near-wall Domain Decomposition	ANDD
Exact Near-wall Domain Decomposition	ENDD
Dirichlet-to-Robin	DR
Dirichlet-to-Neumann	DN
Neumann-to-Neumann	NN
Robin-to-Robin	RR
Steklov-Poincaré	SP
Unsteady Interface Boundary Condition	UIBC
Laminar-Turbulent Transition	LTT
Generalized Minimum Residual Methods	GMRES
Boundary Value Problem	BVP
Parabolized Navier-Stokes	PNS
Partial Differential Equation	PDE
Ordinary Differential Equation	ODE
Zero Fill-in Incomplete LU	ILU(0)
Full Orthogonalization Method	FOM
Classic Gram-Schmidt	CGS
Modified Gram-Schmidt	MGS
Compressed Row Storage	CRS

Roman symbols

S_i	Steklov-Poincaré operator defined in subdomain Ω_i
I	unit function
S_i^*	'boundary-effect-free' Steklov-Poincaré operator defined in subdomain Ω_i
y^*	interface boundary position
\hat{S}_i	symbol of Steklov-Poincaré operator defined in subdomain Ω_i
$\hat{S}_{i\omega}$	derivative of symbol of Steklov-Poincaré operator defined in subdomain Ω_i
$\hat{S}_{i\omega\omega}$	second order derivative of symbol of Steklov-Poincaré operator defined in subdomain Ω_i
\mathcal{F}	Fourier transform operator
u_i^n	function value u in subdomain Ω_i at iteration n
\hat{u}	Fourier transformation of function u
\hat{e}	Fourier transformation of error function e
j	imaginary unit
A	interface boundary position in Poisson's equation
B	upper boundary position of subdomain 2 in Poisson's equation
P_i	pike function defined by Steklov-Poincaré operator applied to unit function

P_i^l	pike function defined by Steklov-Poincaré operator applied to x
P_i^q	pike function defined by Steklov-Poincaré operator applied to x^2
\hat{P}_i	Fourier transformation of pike function
RR-SP0	Robin-to-Robin type algorithm where Steklov-Poincaré operators are constantly approximated
RR-SP1	Robin-to-Robin type algorithm where Steklov-Poincaré operators are linearly approximated
RR-SP2	Robin-to-Robin type algorithm where Steklov-Poincaré operators are quadratically approximated
RR-SP0*	simplified variant of RR-SP0 algorithm without considering boundary effect
RR-SP1*	simplified variant of RR-SP1 algorithm without considering boundary effect
RR-SP2*	simplified variant of RR-SP2 algorithm without considering boundary effect
Re	Reynolds number
Δh_x	spacing step in x direction
Δh_y	spacing step in y direction
DIV	divergence
\mathcal{L}	two-dimensional linear differential operator
\mathcal{L}_y	one-dimensional linear differential operator with respect to y

\mathbf{A}	coefficient matrix of linear system
\mathbf{x}	unknown vector to be solved of linear system
\mathbf{b}	right hand side vector of linear system
\mathbf{x}_m	m -dimensional approximated solution to \mathbf{x}
\mathbf{x}_0	initial guess of \mathbf{x}_m
\mathcal{K}_m	m -dimensional search subspace
\mathcal{L}_m	m -dimensional constraint subspace
\mathbf{r}	residual vector
\mathbf{V}_m	m -dimensional orthogonal basis of \mathcal{K}_m
$\bar{\mathbf{H}}_m$	m -dimensional upper Hessenberg matrix
\mathbf{H}_m	$\bar{\mathbf{H}}_m$ excluding the last row
\mathbf{e}_m	m -th column of $m \times m$ identical matrix
\mathbf{y}_m	reflection of δ in \mathcal{K}_m
\mathbf{q}_i	a vector equal to i -th column of \mathbf{V}_m
\mathbf{F}_j	rotation matrix to modify two certain entries of a given vector
\mathbf{F}	rotation matrix used in Givens rotation
$\bar{\mathbf{R}}_m$	product of Givens rotation applied to upper Hessenberg matrix
\mathbf{R}_m	$\bar{\mathbf{R}}_m$ excluding the last row

Greek symbols

Γ	interface boundary between two subdomains
ω	wave number
$\hat{\gamma}$	Fourier transformation of function γ
θ	relaxation parameter
θ_i	representation of boundary effect on convergence in subdomain Ω_i
Ω	computing domain
δ	Dirac δ function
δ	the difference between estimated solution and initial guess
Λ	diagonal matrix composed of eigenvalues of coefficient matrix
κ	condition number of a matrix
λ_i	i —th eigenvalue of a given matrix, listed in reduced order of magnitude

Abstract

Development of a non-overlapping domain decomposition method for problems with boundary layer

Hongru LI

A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy , 2022

Near-wall turbulence modelling has been a popular research object in computational fluid dynamics over the past few decades. Apart from low Reynolds number models and high Reynolds number models, the method of near-wall domain decomposition has been developed to be an effective approach for solving this issue. The basic idea of near-wall domain decomposition is to transfer the boundary condition from the wall to an interface boundary with no/little knowledge of the off-wall solution. It is important to note that unlike conventional methods, near-wall domain decomposition could be applied to both low Reynolds number models and high Reynolds number models, which makes the approach capable of overcoming the limitations of the two models by maintaining a trade-off between accuracy and efficiency. The approach allows for the distance between the interface boundary and the wall to be varied so as to improve either efficiency and accuracy. Previous research has proved that if the governing equation in the near-wall region is of boundary-layer type and locally one-dimensional, the transfer of the boundary condition could be exact. However, in more generalised cases the local interface boundary condition is found to be deficient in capturing the essential non-local feature of turbulent flows in both time and space. The challenge posed by this problem motivates the development of the near-wall domain decomposition approach with non-local interface boundary conditions.

In this thesis an effective and novel near-wall domain decomposition approach with non-local interface boundary conditions is developed and tested in application to model equations that simulate high-Reynolds-number flow with a boundary layer. The approach is also compared with existing non-local near-wall domain decomposition approaches in the literature and is proven to be more efficient. The supreme convergence of the approach is analysed theoretically in Poisson's equation and the result is further confirmed to be valid in application to the model equations. Both theoretical and numerical analysis shows that the approach has great potential to be applied to near-wall turbulence modelling.

The non-local interface boundary condition of the approach is obtained by approximating the non-local Steklov-Poincaré operators, which are decomposed into a few basic units under local Taylor expansions. The approximation is proved to be effective in retaining the non-local nature of the operators and is easy to implement. The convergence analysis is performed in two steps: first ignore the boundary effect when analysing the product of Steklov-Poincaré operators applied to given functions (standard analysis); next restore the ignored boundary effect. These two steps reflect the influence of the governing equation and boundary conditions respectively. Together they describe the complete non-local nature of Steklov-Poincaré operators in application to a given problem.

As an advanced numerical method to solve large linear systems: the generalised minimum residual method is widely used in the project and its convergence is studied. The convergence of the generalised minimum residual method is found to be determined by the eigenvalue distribution of the coefficient matrix of the linear system. The result is intended to serve as a theoretical foundation to evaluate the comparative computing cost of applying the near-wall domain decomposition approach as opposed to the one-block approach.

Declaration of originality

I hereby confirm that no portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright statement

- i The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.
- ii Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made *only* in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.
- iii The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.
- iv Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see <http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420>), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see <http://www.library.manchester.ac.uk/about/regulations/>) and in The University’s policy on Presentation of Theses.

Acknowledgements

Firstly, I would like to thank my supervisor Dr Sergey Utyuzhnikov for his immense support and patience during the last 4 years of my postgraduate study. He was a great mentor and his honest advice was crucial for my development as an academic.

Secondly, I am grateful to Dr Adel Nasser, Dr Andrea Cioncolini and Prof. Dominique Laurence for helping me during my different stages of PhD journey. Their help was not only useful in helping me improve my annual report quality but also in reminding me of what better research could look like so that I was greater aware of my weakness and tried to strengthen it. I could not have progressed without their support.

Thirdly, I would like to acknowledge the China scholarship council (Grant No. 201908890054) for funding my research. I have always been feeling honoured to be awarded the funding and hence motivated to study hard throughout my PhD career. I wish I did not fail the support by my efforts and could always feel proud of not only receiving but also deserving the funding.

I would also like to mention some of my good friends: Afreen Islam, Chen Wang, Elliot Buss, Fanchao Meng, Kai Liu, Dr Kamil Fedorowicz, Lei Chen, Xiaoxi Pang, Xinyu Bian and Yang Zhou, for spending with me a lot of relaxing and wonderful time in chatting, cooking, eating, drinking, exercising, walking, hiking and gaming, and offering generous help in every aspects of my life when I was in need. The memories with them are so incredible that hardly did I feel lonely nor helpless in the tough and long journey of PhD-I knew I always had hands around. Meanwhile, I can not forget the enjoyable time spent with my school colleagues and flatmates in all kinds of academic and social events, including GTA training and lab/tutorial delivery (special thanks to Dr Dale Smith and Dr Tom Fisher who helped and cheered me up so much in my first year of GTA), coffee mornings, table tennis night, pool game night, Halloween carving, etc.

There were some 'silent' friends keeping accompanying and inspiring me of which I would like to specially acknowledge: my books. I can not forget those summer afternoons sitting on the bench of nearby parks and immersed in the adventures of Harry Potter and his friends, or draining my brain power to catch the steps of Hercule Poirot, or immersing myself in the legendary world of Narnia and seeking Aslan. There were still so many that I do not have chance to introduce but every single one created a wonderful dream for me to explore and I would like even to write another thesis to report my exploration result!

My particular thanks to my girlfriend Peimiao Zou, who was on her PhD journey as well but spared no efforts to support my study as well. Although not living in the same city, she knew my life better than anyone else and played an unique and irreplaceable role in entertaining and

comforting me in many occasions. Awareness of the fact that we were together was a powerful secret weapon I always possessed to overcome the confronted difficulties in my PhD life.

Last but not least, I would like to thank my loving parents, who were always unconditionally helpful and supportive to my PhD study in Manchester. I feel extraordinarily lucky to have such open-minded parents on my back and appreciate every efforts they made to support my study in both physical and emotional ways. What is more important, we never felt lack of words in the regular video calls on weekends and the calls usually lasted hours and ended up with one side feeling too late to continue. The talks included everything in our lives and my parents synchronised well with my progress/dilemma of the PhD study although living apart by thousands of kilo-meters. Actually I never felt I was too far from them.

Chapter 1

Introduction and literature review

1.1 Near-wall turbulence challenge

Turbulence simulation has been a challenging subject in computational fluid dynamics (CFD) field over the past few decades. Especially when it comes to the vicinity of the wall, the problem becomes more complex [1], [2]. Turbulent flow losses its velocity fluctuation at solid-fluid interface due to non-slip boundary condition, which makes the momentum transport dominated by molecular diffusion in the near-wall region, and the flow is locally subjected to laminar flow law due to strong viscous effect. This region is very thin in size and is known as viscous sublayer. Further off the sublayer the flow changes rapidly to turbulent status that prevails over the remainder of the flow. Between the viscous sublayer and the fully turbulent flow there lies a transition zone, and the so-called boundary layer is composed of the above three parts.

The challenge of modelling near-wall turbulence is related to the small thickness (usually around 1% of entire flow [3]) of the zone and extremely high rate of transport properties change (could be two or more order of magnitude higher than elsewhere in the flow [3]). If the mesh density is roughly equal between the near-wall region and off-wall region, the computing time of solving the former could be three to 300 times more than the latter, depending on the flow, turbulence model and CFD solver [3]. In addition, the thickness (δ) of the boundary layer shrinks as Reynolds number (Re) increases ($\delta \sim L/\sqrt{Re}$) [4] and it makes the simulation more challenging in highly turbulent scenario.

The straightforward way to resolve the problem is to discretize Navier-Stokes (NS) equations and solve the linear system in the entire flow domain that naturally includes the near-wall region. The method is so-called Direct Numerical Simulation (DNS) and could provide the most accurate solution among all existing simulation methods. However, the computing expense is very high: nodes number $\sim Re^{9/4}$ in three-dimensional simulation and time steps are proportional to $Re^{3/4}$ [5]. Therefore, DNS is not affordable for most of industrial simulations especially if Re is high. The solution to DNS is usually referred to as benchmark solution by other simulation schemes, for example, [6]–[10]

To reduce computing expense, Large Eddy Simulation (LES) [11] only simulates large-scale flow motions numerically as DNS does but builds simplified models to simulate small-scaled flow behaviour. Since its invention in 1960s, the method has become a popular and powerful

tool to provide affordable simulation result for complex turbulence problems [12], [13]. However, applying LES to near-wall turbulence simulation is still high in computing costs: nodes number and computing time proportional to $Re^{1.8}$ and $Re^{2.4}$ respectively [14], while the ratio is merely $Re^{0.4}$ and $Re^{0.6}$ in main flow [15]. Therefore, LES is restricted to be applied to medium- Re turbulent flows and usually excluded from being considered as effective tool to resolve near-wall turbulence.

Apart from DNS and LES, vast efforts have been made to resolve near-wall turbulence with so-called Reynolds-averaged Navier-Stokes (RANS) models. The method considers the complex turbulent flow in larger time scale thus provide approximated time-averaged solutions to NS equations. Within the framework of RANS approach, one class of models resolve the entire flow domain down to the wall with very fine mesh. Due to strong viscous effect and locally low Reynolds number in the near-wall region, the models adopting the scheme are denoted by low Reynolds number (LRN) models. The governing equations of LRN models are valid throughout the entire turbulence region. One other class of models resolve the near-wall flow by replacing the differential equations across the viscous sublayer with algebraic formulae or other low-cost routes that are 'packaged' in so-called wall functions. The overall characteristics of the momentum transport in the region are represented by wall functions. Due to locally high Reynolds number outside the near-wall region, the models to resolve near-wall flow with the 'package' strategy are denoted by high Reynolds number (HRN) models. Unlike the firm validity in LRN models, the relations in near-wall region are often empirical in HRN models. LRN and HRN models constitute the two basic classes of RANS models.

LRN models range from mixing-length schemes in 1960s and two-equation eddy-viscosity models (EVMs) in 1970s to more delicate non-linear eddy-viscous models (NLEVMs) that enables non-linear connections between turbulent quantities and mean-flow field gradients. Among all, LRN $k - \varepsilon$ model has been developing to one of the most successful and most widely-adopted models in RANS family so far. Launder and Jones proposed the earliest LRN $k - \varepsilon$ model to resolve near-wall turbulence flow [16], where turbulent viscosity is determined from the solution to transport equation for turbulent kinetic energy and energy dissipation rate. In [17], the model was used to predict turbulent shear flow and showed satisfactory agreement with experiment result. Furthermore, the model was applied to simulating flows with high gradients of swirl velocity in the vicinity of spinning wall. The simulation result was found to predict accurately the flow, heat and mass transfer in the vicinity of the rotating wall, and more importantly, provides a successful sample of the generality of the model (known as LS model) for the flow simulation of such kind [18]. Although originally designed to simulate swirling flow, soon the model was widely accepted as classic benchmark LRN $k - \varepsilon$ models to simulate turbulent wall-bounded flow. The comparative study of various near-wall turbulence models stated that LS model is one of the four models that managed to yield satisfactory result in the test of simulating a flat-plate boundary layer [19]. One other model among the four was proposed by [20] (Chieng model), where turbulent shear stress, kinetic energy and dissipation rate were expanded by Taylor series. Chieng model was compared with LS model

in application to turbulent channel flow and boundary-layer flow over flat plate, where it showed comparable result as LS model and even better prediction of peak turbulent kinetic energy. Due to their popularity and wide application, the two classic models represent the so-called standard $k - \varepsilon$ LRN models to a large extent.

In [21], the near-wall region was modelled with one-equation model while the standard $k - \varepsilon$ model was applied to simulating the bulk flow. The 'two-layer approach' was tested in simulating a two-dimensional boundary layer flow and showed comparable result to LRN $k - \varepsilon$ model. On the basis of DNS data for boundary-layer flow [22], a LRN $k - \varepsilon$ model proposed by Rodi and Mansour [6] introduced a new damping function in the eddy-viscosity relation and near-wall source/sink terms in ε equation [7]. The modified model yielded satisfactory approximation for all major quantities in boundary layer flows. Similarly, based on $k - \varepsilon$ model, Durbin proposed an approach to impose kinematic boundary conditions on normal component of turbulent intensity in order to avoid containing damping functions [8]. The model was applied to resolving near-wall shear flow and showed close agreement with DNS data. Other modified $k - \varepsilon$ models to resolve near-wall turbulent flows include [23], [24]. In the former case, eddy viscosity was characterised by a turbulent velocity scale and a turbulent time scale. The time scale was also used to reformulate the dissipation equation. On application level, [25] develops a specific model to simulate the movement and heat transfer of turbulent non-isothermal non-Newtonian fluid through a pipe with cold wall. The fluid turbulence is modelled in the framework of the isotropic two-parameter $k - \varepsilon$ model. The application of several popular LRN models to simulate turbulent flow and heat transfer of highly buoyant horizontal supercritical fluids is reported in [10].

Apart from $k - \varepsilon$, one other important RANS model type is $k - \omega$ [26], [27] that has been under constant development in the past few decades to be applied to a wide range of turbulent flows [28]–[31]. In particular, the model is considered more effective than $k - \varepsilon$ model in solving near-wall flows and has been widely used to resolve flows with adverse pressure gradients [32]. The main problem of the approach is that ω -equation is sensitive to the main flow value of ω in off-wall region. To remedy the problem, [33] firstly suggested complimenting a cross diffusion term in ω equation for reducing its dependence on frees-stream value. Similar strategy of implementing additional cross diffusion term was adopted by [34]–[36] to eliminate the boundary condition sensitivity. Specifically, [34] blended high Reynolds number $k - \varepsilon$ model at the edge of boundary layer and conventional $k - \omega$ model in elsewhere zone. Although it incurred resolution accuracy loss of free shear flows, the improved model demonstrated much less dependent on main flow value and was able to give promising simulation result for boundary layer flows with adverse pressure gradient. The 'blending' method was specially developed in [36] for the requirement in high-lift aerodynamics such as high-lift air-foil flows. On the other hand, to avoid the use of blending models, [37] limited the magnitude of eddy-viscosity by requiring turbulent energy not exceeding turbulent-energy production, and [38] restricted the value of shear-stress in similar way. The modifications were proved equally accurate as the blending models. What is more, [39] reformulated the conventional $k - \omega$ model with the methodology in [30]

that analysed boundary layer and free shear flows using perturbation methods and similarity solutions. The improved model enabled a significant expansion of the applicability including in boundary layer flow simulation where accurate solution could be obtained at the presence of complicated separated flows. In [40], the standard ω model is modified by applying a non-linear (quadratic) stress-strain relation for the Reynolds stress tensor. In particular for resolving the turbulent flow where near-wall anisotropy effects are important, the near-wall turbulence anisotropy treatment is enhanced by a Reynolds number-based formulation. The improved model is successfully applied to two-dimensional and three-dimensional internal flows with boundary layers, curvature, separation, and swirl, showing improved performance than standard $k - \omega$ model. The application of anisotropic turbulence treatment in complex turbomachinery flows is incorporated with the so-called 'Advanced RANS models' [41], and the anisotropic models demonstrate exclusively promising ability in predicting flow behaviours in choking condition.

Comparing with LRN models, HRN models are able to show indisputable advantage in saving computing time. Therefore, the approach is especially popular in commercial CFD software-developing industry. HRN models do not resolve near-wall region directly (thus computationally cheap), instead the region is represented by Dirichlet boundary conditions at interface between the near-wall and free-stream regions, and the boundary conditions are so-called wall functions. The construction of wall functions is crucial to the resolution accuracy of HRN models. The earliest concept and study of wall functions were far back to the pioneer work by [42]–[44], etc. Soon the conventional wall functions [45] that are based on the assumption of log-law of velocity and temperature prevailed in the industry although the log-law is an inadequately formulated empirical model to predict the viscous near-wall region.

Over decades, there have been vast efforts made to improve the log-law-based model. To reduce the dependence of the solution on near-wall size, [46] considered near-wall flow as viscous and logarithmic layers with two velocity scales separately. Similarly strategy was adopted by Amano who developed two new wall functions (namely 'two-layer model' and a 'three-layer model') based on basic $k - \varepsilon$ model [47]. Although the models were applied to separated and reattached flows successfully, the log-law still retained in velocity formulation. Similar study could also be seen in [48] where Shih defined a 'hybrid velocity scale' and non-dimensional length that were used to categorise wall-bounded flow into three regions: viscous sublayer, buffer layer and inertial sublayer. The piece-wise wall function was denoted by 'unified wall function' and was proved particularly effective in dealing with flows with complex geometries. Apart from the 'hybrid' approach, much efforts were made on re-constructing wall functions or modifying log-law to produce more reliable near-wall model. [49], [50] proposed simplified approximation of dissipation rate by regarding kinetic energy and Reynolds stress tensor (\overline{uv}) between viscous sublayer edge and near-wall cell wall as constant. Efforts have also been made to modification of viscous sublayer determination [51] and kinetic energy by fitting experimental data [49]. In [52], Kim introduced a pressure gradient term and a convection term to modify the log-law. The modified model showed

more accurate result in application to boundary layer flow with separation, reattachment and circulating flows, and demonstrated the significance of considering pressure gradient.

Since wall function usually requires the near-wall cell centre lies in logarithmic region, the applicability of HRN models is restricted. To address the issue, [53] proposed so-called scalable wall function (SWF, also known as adaptive wall function) that assumes the cell centre is located at the edge of viscous sublayer if it is actually within the sublayer. The approach mitigates the applicability dilemma of log-law-based HRN models to some extent but the cell size dependence retains. In addition, log-law is not accurate itself thus incurring inevitable errors to the wall functions. To remedy it, the so-called numerical and analytical wall functions are developed.

Numerical wall functions [54]–[56] use two independent mesh schemes in near-wall region and entire flow region, which are denoted by 'wall function mesh' and 'primary mesh' respectively. The former could also be considered as sub-grid of the latter. In principle, RANS equations are solved throughout the entire domain with relatively coarse primary mesh. The wall function mesh are much finer with which one-dimensional parabolic RANS momentum equation is solved in near-wall region to evaluate shear stress and averaged near-wall cell quantities. The two meshes are linked through the common boundary conditions. Unlike log-law-based wall functions, the simplified RANS equation in near-wall region contains convective terms and complete Reynolds stresses, which marks its superiority in computing accuracy over conventional wall functions. In addition, the approach overcomes the weakness of log-law that is only assumed to be applicable for simple-geometric and local-equilibrium flows. In [56], the approach was applied to complex turbulent flows (including plane channel flow, axisymmetric impinging jet and flow near spinning disc) with $k - \varepsilon$ models and showed significant computational savings (one order of magnitude less than LRN models) with better reproduction of LRN predictions than conventional wall functions. The approach was also successfully applied to three-dimensional simulation with body-fitted mesh [54].

Analytical wall functions (AWFs) [57], [58] consider near-wall flow as an analytical problem and solve the equations in the region by integration. Although several assumptions (mainly include: boundary layer type momentum equations in near-wall region and linear turbulent viscosity profile outside viscous sublayer) are essential to facilitate the integration, the analysis is supposed to have greater accuracy than standard wall functions since it includes source terms of RANS equations. The near-wall region is split into two regions: viscous sublayer and turbulent region. Correspondingly, the integration is separately performed in the two regions. The thickness of viscous sublayer is usually empirical thus it incurs the weakness of the method: the computation result could be largely influenced by the relative size of the viscous sublayer and near-wall cell.

Apart from turbulent flow problem, AWFs are also widely applied to heat transfer problems. As a matter of fact, the wall functions were essentially designed for both momentum and energy transport equations. [59] applied several turbulent models including standard and analytical wall functions to channel flows with heat transfer. The result

showed that AWF outperformed standard wall functions in both flow and heat transfer quantity prediction accuracy. The research in [60] considered a modification on the constant-treated eddy viscosity in conventional AWF to improve the solution accuracy in complex flows such as recirculating and impinging flows. Simulations including rough, porous and gas-liquid surface surface/interface were also considered [61]–[64]. Further three-dimensional applications of the approach include the study of turbulent impinging heat transfer where wall-normal convective effect was included in the AWF for energy equation [65]. The utilisation of a new and more generalised AWF (so called ‘buoyancy adapted’ AWF) in application to two/three-dimensional heated cavities modelling is numerically proved to make significant improvement of the HRN $k - \varepsilon$ model in predicting the Nusselt number compared with the log-law based wall function [66]. The improved prediction value is in close agreement with the experimental data.

Kalitzin [67] proposed a look-up-table wall function that is applicable to any RANS models. The idea of the method is to produce a table to refer to for turbulence quantities and friction velocity by solving the near-wall equation numerically and storing the friction velocity as a function of dimensionless parallel velocity. Numerical tests were implemented on flat plate boundary layer flow with four advanced RANS models including $v^2 - f$ and Spalart-Allmaras models. Numerical result showed that the analysis of $v^2 - f$ model resulted in new analytical solutions in viscous sublayer and logarithmic layer, and the analytical solution to Spalart-Allmaras model could be used as wall function directly. The approach was also considered by [68] to reduce the AWF solution dependence on spatial resolution in viscous, buffer and logarithmic layers of near-wall region.

The so-called compound wall function was originally designed for computing complex turbulent flows where the near-wall cell is located in transitional zone between viscous sublayer and logarithmic layer [69]–[71]. In that case, neither wall function in near-wall region nor turbulence model in off-wall region resolves the flow precisely. The idea of [70], [71] was to combine the solution from both AWF [58] and $\overline{v^2} - f$ ($\zeta - f$) turbulence model into a single boundary condition at interface. The aim of the mixed-style wall treatment is to reduce the sensitivity of the result to the near-wall cell size so that the size could impose less influence on the result. The compound wall function was tested in several pulsating/impinging flows with satisfactory result. The industrial applications of the approach could be seen in spark ignition engine heat transfer simulation and unsteady flow simulation over vehicles [72], [73].

1.2 Near-wall domain decomposition

AWF was originally the prototype of so-called near-wall domain decomposition (NDD) method that has been developed over the past decade to an effective tool to model near-wall turbulent flows. The NDD method was firstly proposed in [74] where a linear model equation was studied to imitate the mathematical peculiarities of LRN models including viscous sublayer and transition region. Regarding the near-wall region, the boundary condition is transferred from the wall to a near-wall interface, yielding to a Robin type interface boundary

condition (IBC). If in the inner region the governing equations correspond to a thin boundary layer (TBL) model, then the IBCs are always of Robin type. The Robin type IBC could also be interpreted as generalised wall functions (GWF, sometimes also denoted by Robin-type wall functions RWF) when in application to HRN models [75]. In the case, it is found that the tangential velocity U , temperature T and turbulent kinetic energy k could all be solved by the same method [76]. The GWF considers source terms such as pressure gradient or buoyancy and do not include free parameters. One major advantage of the approach is that the solution is mesh-independent, which means the interface boundary needs not necessarily be related to the nearest wall cell. The application to a turbulent channel flow case (using HRN $k - \varepsilon$ model) confirmed that accurate result could be obtained even if IBC is set at viscous sublayer or somewhere far off the wall [75]. Similar IBC transfer technique was also presented in [77] in application to LRN models, which yielded to the domain decomposition method with both Dirichlet and Neumann boundary conditions considered. The decomposition facilitates the separate implementation of mesh schemes and numerical approaches in inner (near-wall) and outer (off-wall) regions so that both regions could be effectively resolved. In addition, It was proved in [76] that the IBC could also be derived by the theory of Calderón-Ryabenkii's potential [78], [79].

The solution to the inner region could be restored once the outer region is obtained using the Robin-type IBC, and the trade-off between the efficiency and accuracy is largely affected by the location of the interface boundary [80]. If the interface boundary is too close to the wall, the solution in the inner region tends to be fully accurate but large quantity of computation is actually carried out in outer region and the computing time could be relatively high. If the interface boundary is too far from the wall, the inner region is not actually resolved thus much time could be saved but the solution accuracy might not be guaranteed. The former case corresponds to LRN model characteristic and the latter case corresponds to HRN model characteristic.

In [81], a two-dimensional model equation that imitates the major mathematical peculiarities of near-wall high-Reynolds-number flow was considered and the so-called non-local wall functions were proposed to capture the two-dimensional characteristics of the problem on interface. The non-local effects of the problem were represented by a pseudo-differential operator and the non-local wall functions were derived by explicitly approximating the pseudo-differential operator and retaining part of the approximation terms. The non-local wall function inherits the good property of GWF that near-wall and off-wall regions are considered separately and the solution is weakly mesh-dependent. Correspondingly, the non-local wall functions could be developed into two-dimensional (non-local) domain decomposition method directly as GWF in one-dimensional domain decomposition method. A two-dimensional test was implemented between one-dimensional wall function and non-local wall function, and the latter showed clear improvement in prediction of velocity and friction coefficient than the former.

As will be elaborated later on, the one-dimensional domain decomposition based on GWF is denoted by 'approximated near-wall domain decomposition' (ANDD) method and the

two-dimensional domain decomposition based on non-local wall function is denoted by 'exact near-wall domain decomposition' (ENDD) method. The derivation and formulation of ANDD and ENDD are shown in Chapter 2. The notations of the methods are how they appear in publications nowadays and would be used in the rest part of the thesis where necessary. The concepts are hereby explained in order to avoid reader confusion.

In [82], it was stated that ENDD could be used as supplement of ANDD when the latter was unable to accurately resolve complex flows such as separated flow, laminar-turbulent transition flow. In practice, ANDD could solve the problem with low time costs and let ENDD complete the rest of the computation based on ANDD result. The two methods could be shifted fluently based on the same mesh during the same-case computing. In addition, in [83] the method of ANDD was proved to be a well-conditioned preconditioner in solving near-wall turbulence problem because the construction of the preconditioner (inverted ANDD operator) is directly associated with near-wall turbulence physics: the assumption of boundary-layer type equation in ANDD method is mostly valid in near-wall region.

As an important version of ENDD method, Dirichlet-to-Robin (DR) algorithm (will be derived in Chapter 2) proved its superiority over ANDD method [83]. In that study, the DR type ENDD method was applied to simulating a one-dimensional channel flow using LRN $k - \varepsilon$ model, and the result showed that it offered accurate prediction of turbulent kinetic energy and turbulent dissipation when ANDD failed to do so. The test case demonstrated the advantage of ENDD in terms of maintaining resolution accuracy in multidimensional situation. The convergence of DR method was also theoretically proved, showing that its convergence is faster than one other ENDD method of Dirichlet-to-Neumann (DN) type.

Further to DR method, a more advanced ENDD method of Robin-to-Robin (RR) type was proposed [84]. The construction of the RR method is based on so-called Steklov-Poincaré (SP) interface equation and the realisation is implemented by the approximation of Steklov-Poincaré operators, which play a crucial role in retaining the non-local nature of the problem. There were three kinds of approximation methods proposed and comparatively tested in the numerical cases of Poisson's equation as well as a model equation to simulate high-Reynolds-number flow with a boundary layer. Analytical analysis proved that RR method achieves the convergence faster than DR and the convergence rate calculation was also performed for one of the three cases in application to Poisson's equation. The numerical study showed that the algorithm with locally quadratic approximation of SP operators led to the fastest convergence.

In [85], the Robin-type boundary condition transfer was derived for an unsteady model equation that simulates the key terms of unsteady boundary layer equations. The derivation showed that in the case of stationary situation the result led to the same domain decomposition technique as in steady flow study. The numerical test of the model equation demonstrated that the unsteady effect must be included in the boundary condition transfer process to avoid significant resolution accuracy loss if domain decomposition is used to resolve unsteady flow. Furthermore, it is found that the unsteady interface boundary condition (UIBC) must be

non-local in time and should be modified to include a memory term, which was proven to consist of both the interface solution and driving force unsteadiness [86]. The properties of the unsteady interface boundary conditions were studied through the case study of oscillatory and pulsating laminar flows in a channel and a pipe. In addition, the accuracy of the NDD method with UIBC was studied through unsteady case that revealed its association with viscosity, frequency and location of the interface boundary (represented by Womersley number). With unsteady cases in low frequency, the steady NDD method could still result in high accuracy, and lower interface boundary position enabled the accuracy loss compensated for unsteady cases in higher frequency. The approach was applied to three-dimensional unstructured-mesh unsteady RANS equations in [87] with applications including pulsating flow in a channel and past half-cylinder, as well as supersonic flow around circular cylinder. As mentioned earlier, the UIBC contains a memory term that is crucial to capture the unsteady characteristic of the flow, and the numerical test yielded satisfactory prediction of the skin friction with interface boundary location up to $y^{*+} = 300$.

The application of the NDD method was extended to compressible flows in [82], where flows over blunted plate, half cylinder, compression corner and sharp edge were simulated with Spalart-Allmaras model. The test demonstrated the capability of the method applied to supersonic flows, and showed positive result in reconstructing density profile and computing skin friction with high accuracy. What is more, it was the first time that ENDD method was implemented to solving real turbulence problem.

In [88], the ANDD method was modified to be applied to non-equilibrium flows with laminar-turbulent transition (LTT). The modification was implemented by considering different viscosity profiles in laminar, transitional and turbulent regions individually and the intermittent implementation expanded the capabilities of the adopted Spalart-Allmaras model to non-equilibrium turbulent flows with extended transition region. In addition, ENDD was found to be capable of simulating LLT directly but it took more computing time. It was confirmed by the numerical tests (including subsonic, supersonic flows over flat plate, compression corner and planar shock wave impinging on a turbulent boundary layer) that the skin friction obtained by the NDD method almost agreed with one-domain method. The NDD method was also proved effective in predicting accurately the position of separation point and formation of shock wave with taking account of LTT.

1.3 Outline of the thesis

In Chapter 2, the thesis begins with the derivation of the non-local RR-type algorithms that are the main study subjects of the project. Since the non-local algorithms are the further development of existing ANDD methods that have been developed for over a decade, the derivation starts from the prototype of ANDD method: the technique of Robin type IBC transfer, and focuses on the evolution of the method from one-dimensional to two-dimensional. Three variants (algorithms) of RR type method are derived from the NDD method. The application of the IBC transfer to turbulence models is not fully elaborated

because only model equations that are supposed to simulate high-Reynolds-number flow with a boundary layer are considered in the project. The derivation is performed with model equations only.

In Chapter 3, the convergence rate calculation of RR algorithms is performed without considering the boundary effect of the given boundary value problem (BVP). The calculation is implemented by separation of variables and Fourier transform. The same calculation is also performed to a Dirichlet-to-Robin algorithm for comparison. The boundary-effect-free calculation method is denoted by 'standard approach'. In addition, the standard approach is proved applicable for all BVPs that could be solved with separation of variables.

Chapter 4 analyses the standard convergence rate obtained from Chapter 3, and formulates the convergence rate in generalised manner via the symbols of SP operators. The generalised formulation is helpful in evaluating the effect of the interface boundary position on the standard convergence. Next, the boundary effect of the three RR algorithms is analysed. Together with the result of the standard approach in Chapter 3, the real convergence property of the algorithms could be completely evaluated.

Chapter 5 conducts a series of numerical tests on Poisson's equation and model equations. The test on Poisson's equation is to verify the convergence analysis result obtained from Chapters 3 and 4. Then the tests are extensively applied to two model equations that link Poisson's equation and real turbulence modelling. The numerical result is discussed in terms of non-local effect of the equations. At the end of the chapter, a new algorithm based on movable interface boundary position is proposed and tested.

Chapter 6 studies an efficient numerical algorithm to solve linear system known as the generalised minimum residual method (GMRES). It is essential to understand the mechanism and implementation of GMRES because all the linear systems in the project are solved by the algorithm. The derivation of GMRES is introduced first, followed by the convergence analysis of the algorithm, which is closely associated with the computing costs of the NDD algorithms. The aim of GMRES study is to evaluate the time cost of the NDD methods and if possible, to help improve future NDD algorithm design. The implementation discussion of GMRES and the preconditioning technique introduction are also shown in the chapter.

The main result and conclusion of each chapter are summarised at the end of the chapters. Chapter 7 assembles them in a wider framework and highlights the novelty and main progress in terms of developing the non-local NDD method. The deficiency of the work and potential future work are recommended in the end.

Chapter 2

Derivation of non-local domain decomposition algorithms

This chapter introduces the derivation of the non-local NDD method through model equation demonstrations. The derivation does not include either LRN or HRN models applications and specifically focuses on the development of the method from local to non-local evolution in space.

At the beginning, the NDD method that is studied in this project were not actually based on domain decomposition framework. The prototype of NDD method was originated from the so-called Robin-type interface boundary condition transfer that aims at solving boundary-layer-type equations. It is built by transferring the boundary condition on the wall to an intermediate interface boundary condition of Robin type off the wall, so that the near-wall region needs not be computed before the solution in off-wall region is obtained. If the equation is locally one-dimensional (such as boundary-layer-type equation), the transfer is exact. Otherwise the transfer could only be approximated due to the non-local effect, which is the motivation of the development of the non-local NDD method. In particular, applying the boundary condition transfer technique to HRN models (variables including tangential and normal velocity components, temperature and kinetic energy) gives rise to the so-called generalised wall function (Robin-type wall function). It is proved that GWF has advantages over conventional wall functions in terms of interface location flexibility, mesh scheme independence and computing efficiency.

There are more than one approach to realised the one-dimensional interface boundary condition transfer and this chapter demonstrates two methods: integration and superposition. The integration method assumes that the governing equation in near-wall region is simplified to boundary-layer type and superposition method assumes the governing equation to be linear. Note non-linear case derivation results in an iterative NDD method [89], and the method is not elaborated in the thesis. The integration method is implemented by integrating the equation twice in near-wall region with considering the boundary condition at the wall. In superposition method, the boundary value problem in near-wall region is considered as the sum of two auxiliary BVPs defined in the same domain. The derivation results of the two methods are consistent. Comparatively speaking, the superposition method has wider applicability, such as in the kinetic energy equation of LRN $k - \varepsilon$ models. As a result, the

NDD method could be applied to both LRN and HRN models. The NDD method inherits the good properties of GWFs in robustness, efficiency, mesh independence and resolution accuracy, and is capable of extending the applications to LRN models and even LES.

Next, the superposition technique is extended to two-dimensional cases where non-local effect of the BVP is studied and represented by a pair of pseudo-differential operators (Steklov-Poincaré). The reservation of the non-local effect represents the essential difference between the two-dimensional approaches and the one-dimensional approaches. The approximation of the pseudo-differential operators plays the key role in designing the domain decomposition algorithms because the approximation amounts to the reservation of the non-local effect when dealing with the derivative of the problem with respect to y on interface. Technically, the approximation comes with computing cost: the more accurate the approximation, the higher the cost. If the non-local effect is fully computed, the algorithm is called 'exact near-wall domain decomposition' method, otherwise it is called 'approximated near-wall domain decomposition' method. The derivation of ANDD is closely associated with the one-dimensional case derivation because non-local effect of the SP operators are partially/fully ignored in ANDD, which results in the one-dimensional derivation of NDD. In otherwise case where the non-local effect is fully retained, the ENDD method is obtained. There are several variants of ENDD method and they could be classified into 2 categories in terms of IBC types: DR and RR. In practice, ANDD methods cost less time than the standard method (one-block method) but the solution is not fully accurate. To achieve the trade-off between computing efficiency and accuracy, ENDD is used as a supplement of ANDD where necessary.

2.1 Robin-type interface boundary condition

This section demonstrates how to transfer the boundary condition from the wall to some intermediate interface and obtain the interface boundary condition of Robin type.

Consider the following BVP [74]–[77]:

$$(\mu u_y)_y = C. \quad (2.1)$$

which is defined in $\Omega = [0, y_e]$ with Dirichlet boundary conditions:

$$u(0) = u_0, \quad (2.2)$$

$$u(y_e) = u_1. \quad (2.3)$$

As a model equation, Eq.(2.1) represents the boundary-layer-type equation in near-wall region. The model equation is reduced from the generalised LRN model equation [74] by neglecting the convective term. Note the right hand side term is retained, which represents source term in transport equations or pressure gradient.

Firstly assume C is constant, and integrate Eq.(2.1) once from 0 to y :

$$(\mu u_y)_{|y=0} = \mu u_y - Cy.$$

Divide both sides by $\mu(y)$ and integrate from 0 to y again:

$$u(y) = u_0 + (\mu u_y)_{|y=0} \int_0^y \frac{d\xi}{\mu(\xi)} + C \int_0^y \frac{\xi}{\mu(\xi)} d\xi. \quad (2.4)$$

Apparently, at some intermediate point $y = y^*$ ($0 < y^* < y_e$),

$$(\mu u_y)_{|y=0} = (\mu u_y)_{|y=y^*} - Cy^*. \quad (2.5)$$

Substitute it into Eq.(2.4) to remove $(\mu u_y)_{|y=0}$:

$$u(y) = u_0 + u_y(y^*) \int_0^y \frac{\mu(y^*)}{\mu(y')} dy' - \frac{C}{\mu(y^*)} \int_0^y \frac{\mu(y^*)}{\mu(y')} (y^* - y') dy'.$$

Particularly, at $y = y^*$:

$$u(y^*) = u_0 + f_1 u_y(y^*) - f_2 \frac{C}{\mu(y^*)}, \quad (2.6)$$

where

$$f_1 = \int_0^{y^*} \frac{\mu(y^*)}{\mu(y)} dy, \quad (2.7)$$

$$f_2 = \int_0^{y^*} \frac{\mu(y^*)}{\mu(y)} (y^* - y) dy. \quad (2.8)$$

Next, consider when $C(y)$ is not constant, the first integral becomes slightly different:

$$(\mu u_y)_{|y=0} = \mu u_y - \int_0^y C(\xi) d\xi.$$

Similarly to $C = \text{const.}$ case, divide both sides by $\mu(y)$ and integrate from 0 to y once more:

$$u(y) = u_0 + (\mu u_y)_{|y=0} \int_0^y \frac{d\xi}{\mu(\xi)} + \int_0^y \frac{\int_0^{y^*} C(\xi) d\xi}{\mu(y')} dy'. \quad (2.9)$$

To remove $(\mu u_y)_{|y=0}$ from the integral, substitute $(\mu u_y)_{|y=0} = (\mu u_y)_{|y=y^*} - \int_0^{y^*} C(\xi) d\xi$ into above equation and it yields to:

$$\begin{aligned} u(y) = & u_0 + u_y(y^*) \int_0^y \frac{\mu(y^*)}{\mu(y')} dy' - \int_0^{y^*} C(\xi) d\xi \int_0^y \frac{dy'}{\mu(y')} \\ & + \int_0^y \frac{\int_0^{y^*} C(\xi) d\xi}{\mu(y')} dy'. \end{aligned}$$

In particular, at $y = y^*$:

$$u(y^*) = u_0 + f_1 u_y(y^*) - \frac{f_2}{y^* \mu(y^*)} \left(\int_0^{y^*} C dy \right), \quad (2.10)$$

where

$$f_1 = \int_0^{y^*} \frac{\mu(y^*)}{\mu(y)} dy, \quad (2.11)$$

$$f_2 = y^* \int_0^{y^*} \frac{\mu(y^*)}{\mu(y)} \left(1 - \frac{\int_0^y C dy}{\int_0^{y^*} C dy} \right) dy. \quad (2.12)$$

Particularly, if it is only the wall friction $\tau_w = (\mu u_y)|_{y=0}$ required to be computed, the inner region solution does not need to be obtained to compute τ_w . For instance, in Eq.(2.5) (and the same with $C \neq \text{const.}$ case), τ_w is represented as a function of $u_y(y^*)$ that could already be known after outer region is computed:

$$\tau_w = \frac{\mu(y^*)}{f_1} (u^* - u_0) + \left(\frac{f_2}{f_1} - y^* \right) C, \quad C = \text{const.}, \quad (2.13)$$

$$\tau_w = \frac{\mu(y^*)}{f_1} (u^* - u_0) + \left(\frac{f_2}{y^* f_1} - 1 \right) \int_0^{y^*} C(y) dy. \quad C(y) \neq \text{const.}. \quad (2.14)$$

2.2 Domain decomposition method

This section derives the Robin type boundary conditions using superposition method. For the same boundary-layer-type equation, the result is identical to that derived in Section 2.1. As a matter of fact, superposition could be applied to more generalised governing equations than the model equation Eq.(2.1), and the Robin-type interface boundary transfer technique could be developed to the NDD method in generalised one-dimensional linear case. Therefore, the boundary condition transfer derivation through superposition could also be interpreted as the NDD method derivation, as the section title suggests. Due to wider applicability of superposition, the NDD could be applied to LRN model equations and further to two-dimensional NDD method derivation, as will be shown in Section 2.3.

Consider the following one-dimensional linear BVP with Dirichlet boundary condition on the wall [77], [80]:

$$\mathcal{L}_y u = f, \quad (2.15)$$

$$u(0) = u_0, \quad u(y_e) = u_1. \quad (2.16)$$

Note \mathcal{L}_y is a linear differential operator with respect to y . Assume the boundary condition is supposed to be transferred to an intermediate interface y^* , so the entire domain $\Omega := [0, y_e]$ is split into two subdomains: $\Omega_1 := [0, y^*]$ (inner domain) and $\Omega_2 := [y^*, y_e]$ (outer domain). The original function is split into two correspondingly: namely u_1 and u_2 . Now consider two

auxiliary BVPs in inner domain:

$$\begin{aligned}\mathcal{L}_y v &= f, \\ v(0) &= u_0, \quad v(y^*) = 0,\end{aligned}$$

$$\begin{aligned}\mathcal{L}_y w &= 0, \\ w(0) &= 0, \quad w(y^*) = 1.\end{aligned}$$

Clearly, the solution to Eq.(2.15) in Ω_1 could be written as

$$u_1(y) = v + u_1(y^*)w. \quad (2.17)$$

Particularly, take derivative at $y = y^*$,

$$u_{1y}(y^*) = v_y(y^*) + u_1(y^*)w_y(y^*). \quad (2.18)$$

This is the Robin-type boundary condition of BVP1. The solution to v and w should not be difficult to obtain (so are v_y and w_y) as they are one-dimensional. Since at interface $u_1 = u_2$ and $u_{1y} = u_{2y}$, the Robin type boundary condition applies to u_2 too, which reads:

$$u_{2y}(y^*) = v_y(y^*) + u_2(y^*)w_y(y^*). \quad (2.19)$$

Therefore, the BVP in outer region is completed and could be solved separately from inner region. Next, the inner region could be computed with Eq.(2.17). As is shown, the computation is separate in two regions as well as mesh generations thus is regarded as a domain decomposition method.

The result of integration and superposition could be proved consistent. Take Eq.(2.1) as example, so that $\mathcal{L}_y = \frac{d}{dy}(\mu \frac{d}{dy})$ and take C as a constant. Applying integration on this case, the solutions to the corresponding auxiliary BVPs are

$$\begin{aligned}v(y) &= u_0 + \mu(0)v_y(0) \int_0^y \frac{d\xi}{\mu(\xi)} + C \int_0^y \frac{\xi}{\mu(\xi)} d\xi, \\ w(y) &= \mu(0)w_y(0) \int_0^y \frac{d\xi}{\mu(\xi)},\end{aligned}$$

where

$$\begin{aligned}v_y(0) &= -\frac{u_0 + C \int_0^{y^*} \frac{\xi}{\mu(\xi)} d\xi}{\mu(0) \int_0^{y^*} \frac{d\xi}{\mu(\xi)}}, \\ w_y(0) &= \frac{1}{\mu(0) \int_0^{y^*} \frac{d\xi}{\mu(\xi)}}.\end{aligned}$$

Therefore, $v_y(y^*)$ and $w_y(y^*)$ could be obtained:

$$v_y(y^*) = \frac{u_0 + C \int_0^{y^*} (y - y^*) \frac{dy}{\mu(y)}}{-\mu(y^*) \int_0^{y^*} \frac{dy}{\mu(y)}},$$

$$w_y(y^*) = \frac{1}{\mu(y^*) \int_0^{y^*} \frac{dy}{\mu(y)}}.$$

Substitute $v_y(y^*)$ and $w_y(y^*)$ into Eq.(2.18), the result is consistent with the one obtained from integral method, see Eq.(2.6). It could be verified that for $C \neq \text{const.}$ case, the result of the two methods are consistent too.

The computation of v and w could be implemented either by analytical calculation or by numerical approach in complex cases. For example, Thomas algorithm (simplified Gauss elimination) is effective numerical tool to solving one-dimensional problem of such kind. Since the inner and outer regions are separately computed, the two regions could use different mesh schemes to cater for separated needs without compromising to the other. In standard (one-block) method, the interface between the two regions has always to be carefully considered in terms of cell size continuity and smooth junction.

One other advantage of domain decomposition method is that the solution to w could be repeatedly used for similar BVPs in the form of $\mathcal{L}u = f$ with different right hand side f . The inner region computing cost is roughly halved if w is already known. As will be seen in Section 2.3, the merit is also true for two-dimensional cases. In addition, the method could be widely used in solving both LRN and HRN model equations, as has been mentioned previously.

Lastly, let us see two examples of Thomas algorithm to solve the one-dimensional function $v(w)$. Applying central finite difference method to the BVP yields to the following linear system to solve:

$$\begin{bmatrix} a_{11} & 0 & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 \\ 0 & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & 0 & 0 & 0 & a_{n,n} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ \dots \\ b_n \end{bmatrix}.$$

Note the boundary conditions are of Dirichlet type so the function value on both sides (x_1, x_n) is known (and $a_{11} = a_{n,n} = 1$). In this case, Thomas algorithm is made of two steps:

- Forward transformation: for $i = 2$ to $n - 1$, $a_{i,i-1} \rightarrow 0$, $a_{i,i} \rightarrow a_{i,i} - \frac{a_{i,i-1}}{a_{i-1,i-1}} a_{i-1,i}$, $b_i \rightarrow b_i - \frac{a_{i,i-1}}{a_{i-1,i-1}} b_{i-1}$.
- Back substitution: for $i = n - 1, 2$, $x_i = (b_i - a_{i,i+1} x_{i+1}) / a_{i,i}$.

In the forward transformation, the first non-zero entry of each row in the matrix becomes zero so that the coefficient matrix reduces to upper tridiagonal form $\mathbf{U}\mathbf{x} = \mathbf{b}$. The linear system

could be solved at once by back substitution (or forward substitution in this case because the first row has only one non-zero coefficient).

If the boundary conditions are not Dirichlet type, the first and last row of the coefficient matrix are slightly different and Thomas algorithm changes accordingly. In this case, the initial linear system reads:

$$\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 \\ 0 & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & 0 & 0 & a_{n,n-1} & a_{n,n} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ \dots \\ b_n \end{bmatrix}.$$

There are two non-zero entries in the first/last row of the matrix, and normally none of them are equal to one. Then, Thomas algorithm is made of the following two steps:

- Forward transformation:
for $i = 2$ to n , $a_{i,i-1} \rightarrow 0$, $a_{i,i} \rightarrow a_{i,i} - \frac{a_{i,i-1}}{a_{i-1,i-1}} a_{i-1,i}$, $b_i \rightarrow b_i - \frac{a_{i,i-1}}{a_{i-1,i-1}} b_{i-1}$.
- Back substitution: $x_n = b_n/a_{n,n}$; for $i = n-1, 1$, $x_i = (b_i - a_{i,i+1}x_{i+1})/a_{i,i}$.

The forward transformation and back substitution are implemented from row two to n and $n-1$ to one respectively. This is the standard Thomas algorithm that also applies to the specific case with Dirichlet boundary conditions. Since Dirichlet boundary conditions already give the boundary value explicitly, x_1 and x_n do not need to be addressed by the algorithm.

2.3 NDD in application to HRN and LRN models

When the boundary transfer technique is applied to solving HRN models, it yields to generalised (Robin-typed) wall functions. For example, in Section 2.1 although the Robin-type IBC is derived based on a model equation, the equation represents the common form of the momentum, energy and kinetic energy equations in HRN $k - \varepsilon$ model. The equations are also known as parabolised (reduced) Navier-Stokes (PNS) equations. In other words, the boundary condition transfer technique could be used to derive the wall functions for tangential and normal velocity components, temperature and kinetic energy.

The formulation of PNS equations in HRN $k - \varepsilon$ model is shown in [76], Eqs.(7)-(10). The right hand side of the equations are assumed constant except in kinetic energy equation. The resulting Robin-type IBCs correspond to Eqs.(2.6) and (2.10) respectively. Particularly in constant case, f_1 and f_2 could be obtained analytically if μ is approximated in piece-wise linear manner [76]. The sublayer thickness is calculated as the way in [50].

It is demonstrated in [89] that the implementation of Eqs.(2.6) and (2.10) depends on only interface y^* and kinetic energy $k(y^*)$, and the latter could be determined from the solution

to HRN models at interface. Therefore, the interface boundary conditions complete the BVP2 and could be considered as generalised wall functions. They differ from standard wall functions due to the fact that the latter substitute the solution in boundary layer region with Dirichlet boundary conditions, which means the profile of u in this region is already approximately restored and the value on interface boundary y^* is also given. By contrast, Robin type (generalised) wall functions do not restore BVP1 directly but transfer the boundary conditions from wall to some intermediate interface y^* . The transferred boundary conditions enable BVP2 to be solved independently from BVP1 like what conventional wall functions do in solving HRN models. In both wall functions, BVP1 is avoided to be computed.

Unlike conventional wall functions that there is a strong dependence on the location of the nearest cell point to the wall, one of the major advantages of GWFs is that the interface y^* could be located far from wall and the inner region could still use fine mesh without losing stability.

The Robin-type IBC could also be applied to LRN models. Apart from the piece-wise linear approximation of μ ($\mu = \mu_t + \mu_l$) in HRN, the more accurate approximation is nonlinear and dependent on wall friction τ_w that could be computed via the friction at interface boundary [89]:

$$\tau_w = (\mu_t(y^*) + \mu_l)u_y(y^*) - \int_0^{y^*} C dy. \quad (2.20)$$

The above equation is essentially equal to Eqs.(2.13) and (2.14). The non-linear approximation of μ_t could be found in [90], [91], or equivalently Eq.(18) in [89], which is τ_w -dependent. Since both μ_t and τ_w remain unknown, iterative procedures are required. As has been stated that τ_w could be obtained through interface friction, see Eq.(2.20), the inner region does not need to be computed in the iterations. The entire iterative procedures [80] are as follows:

- Step one: initialise μ_t with a preliminary guess.
- Step two: initialise τ_w with a preliminary guess.
- Step three: calculate the Robin type IBC.
- Step four: solve BVP2 with the obtained IBC.
- Step five: update τ_w by Eq.(2.20).
- Step six: update μ_t using new τ_w .
- Step seven: re-calculate the Robin type IBC and repeat the following process from step 3 until convergence.

One specific example of applying the above procedures to LRN $k-\varepsilon$ model is given in [89]. As has been mentioned earlier, the calculation of μ_t is τ_w -dependent, so the iterative procedures should be arranged to obtain it (corresponding to steps five and six in above procedures). Firstly assume ω_w and ε are known, then μ_t could be calculated based on the assumed value

using the method proposed in [90] [91]. Next, the Robin type interface boundary condition for the velocity u could be obtained, see Eq.(2.6), which is essentially equal to

$$u_y(y^*) = \frac{u}{I_1} + \frac{P_x}{\mu(y^*)} \frac{y^* I_1 - \bar{I}_2}{I_1},$$

where I_1 and P_x (pressure gradient) are equal to f_1 and C in Eq.(2.6) respectively, $\bar{I}_2 = \int_0^{y^*} y' \mu(y^*) / \mu dy'$. Clearly \bar{I}_2 corresponds to part of f_2 in Eq.(2.6).

Apart from u , the turbulent kinetic energy and energy dissipation rate ε could be formulated at interface [50]:

$$k(y^*) = \sqrt{\tilde{\varepsilon} \mu_t(y^*) / (C_\mu \rho f_\mu)},$$

$$\varepsilon(y) = \max\left\{\frac{k^{3/2}}{C_l y}, \frac{2\nu k}{y^2}\right\},$$

where $\mu_t = C_\mu \rho f_\mu k^2 / \tilde{\varepsilon}$, $\tilde{\varepsilon}(y) = \varepsilon - 2\nu k / y^2$, $\nu = \mu_l / \rho$, $f_\mu = f_\mu(y^+)$ and $C_l = 2.55$.

The interface boundary condition of other variables other than u , k and ε could be obtained by Eq.(2.10). Finally the outer region could be resolved by the completion of the IBCs as obtained above. Practically, several BVPs regarding corresponding variables are solved with either Dirichlet/Robin type IBCs, and the solution gives more accurate estimation of τ_w and ε that would be fed to drive the new iteration until convergence.

Assume the NDD method is applied to LRN model and the inner region is approximated using Eq.(2.17). Regarding different y^* , there is a trade-off between computational efficiency and accuracy. Consider 2 extreme cases: $y^* \rightarrow 0$ and y^* is far off wall and laminar sub-layer. In the former case, computing accuracy is guaranteed but the computing time could be high (close to LRN) because the inner region is too thin to save time. In the latter case, like HRN, computing time could be expected to drop largely but the computing accuracy would be lost. This is because the inner region contains not only the laminar sub-layer but also large area of main flow that fails to resemble the laminar sub-layer characteristic. In this way the laminar sub-layer is not resolved and the wide 'mixed' inner region could not be approximated accurately by one single model equation. Assume the NDD method is applied to LRN models and the inner region is approximated using Eq.(2.17). Regarding different y^* , there is a trade-off between computational efficiency and accuracy. Consider 2 extreme cases: $y^* \rightarrow 0$ and y^* is far off wall. In the former case, computing accuracy is guaranteed but the computing time could be high (close to LRN) because the inner region is too thin to save time. In the latter case, like HRN, computing time could be expected to drop dramatically but the computing accuracy would be lost. This is because the inner region contains not only the laminar sublayer but also large area of main flow that fails to resemble laminar sublayer characteristic. In this way the laminar sublayer is not resolved and the wide 'mixed' inner region could not be approximated accurately by one single model equation.

The trade-off between the efficiency and accuracy of the NDD method is studied in [92], which shows that the method could reduce the computational time by one order of magnitude

with incurring just 1% of error compared with LRN resolution. Meanwhile, the solution is not sensitive to y^* given y^* is small. A reasonable evaluation on the value of y^* could be obtained by preliminary calculation on coarse mesh or empirical approximation [80].

To summarise, GWF (RWF) and conventional AWF have the same nature and they are derived based on similar assumptions, but GWF allows more complicated wall treatment of the governing equations thus yields to more accurate IBC of Robin type. It shows advantage in less solution dependence on interface boundary position and mesh schemes between the two regions too. Further more, the NDD method could be interpreted as the extension of GWF from HRN model application to LRN model application. Apart from the merits as mentioned above, the method shows comparable accuracy to LRN and efficiency to conventional AWF. The trade-off between accuracy and efficiency is manipulated by the position of interface boundary y^* .

2.4 Non-local DR type domain decomposition method

This section considers the superposition method applied to two-dimensional problem to set up the interface boundary conditions as has been done with one-dimensional case. The challenge is the non-local effect of the problem makes the superposition relationship no longer simply linear (not like Eq.(2.18)). To overcome it, a pair of pseudo-differential operators known as Steklov-Poincaré operators are introduced to represent the non-local effect of the problem and the key point turns to be the approximation of the pseudo-differential operators. Apart from this, the analysis to represent the pseudo-differential operators by spectral approach [93] is also performed. So far, it is yet impossible to fully approximate (compute) the operators without large amount of computation, and two kinds of domain decomposition methods are proposed: approximated and exact. The former retains only the constant term of the operators once it is expanded using spectral approach and as a result part of the non-local characteristic is lost in return for iteration-free computing. It saves computing time but losses solution accuracy. The latter retains all non-local effect of the pseudo operators but requires iterations to complete the calculation. Although capable of giving exact solution, sometimes the method has problem with computing time compared to the standard (one-block) method. The two methods are named 'approximated near-wall domain decomposition' and 'exact near-wall domain decomposition' respectively.

2.4.1 Approximated near-wall domain decomposition

Firstly, formulate a two-dimensional linear BVP defined in $[0, 1] \times [0, y_e]$ with Dirichlet boundary condition on the wall to consider [81]:

$$\begin{aligned}\mathcal{L}u &= f, \\ u(x, 0) &= u_s(x), \\ u(0, y) &= u_w(y), \\ u(1, y) &= u_e(y), \\ u(x, y_e) &= u_n(x).\end{aligned}\tag{2.21}$$

Note \mathcal{L} is a linear differential operator with respect to x and y . Let the domain be split into 2 subdomains: inner region $\Omega_1 := [0, 1] \times [0, y^*]$ and outer region $\Omega_2 := [0, 1] \times [y^*, y_e]$. At the moment it is only the inner region that is concerned because the goal is to decompose the BVP in inner region into 2 auxiliary BVPs defined in the inner region ($[0, 1] \times [0, y^*]$) as well:

$$\begin{aligned}\mathcal{L}V &= f, \\ V(x, 0) &= u_s(x), \\ V(0, y) &= u_w(y), \\ V(1, y) &= u_e(y), \\ V(x, y^*) &= 0.\end{aligned}$$

$$\begin{aligned}\mathcal{L}W &= 0, \\ W(x, 0) &= 0, \\ W(0, y) &= 0, \\ W(1, y) &= 0, \\ W(x, y^*) &= U^*(x).\end{aligned}$$

Note $U^*(x)$ is unknown.

Apparently, the solution to u in inner domain (denoted u_1) is the sum of V and W : $u_1 = V + W$. Since the solution to V is independent from outer domain and interface boundary value, it could be solved with an independent mesh scheme efficiently, as is the same with one-dimensional domain decomposition method. However, the solution to W could not be obtained until U^* is known.

Recall in one-dimensional case, Eq.(2.18), the second term is represented by $u(y^*)w$, and similarly, an auxiliary BVP could be defined in $\Omega_1 := [0, 1] \times [0, y^*]$ to help approximate W in the two-dimensional case:

$$\mathcal{L}\tilde{W} = 0,$$

$$\begin{aligned}
\tilde{W}(x, 0) &= 0, \\
\tilde{W}(0, y) &= 0, \\
\tilde{W}(1, y) &= 0, \\
\tilde{W}(x, y^*) &= 1.
\end{aligned}$$

The solution to W could be approximated as

$$W \approx \tilde{W}U^*(x), \quad (2.22)$$

and

$$u_1(x, y) \approx V(x, y) + \tilde{W}(x, y)U^*(x).$$

Taking derivative with respect to y at $y = y^*$ yields to (approximated) Robin-type boundary condition of BVP1:

$$\frac{\partial u_1(x, y)}{\partial y} \approx \frac{\partial V(x, y)}{\partial y} + \frac{\partial \tilde{W}(x, y)}{\partial y}U^*(x). \quad (2.23)$$

Like in one-dimensional case, the boundary condition could be used to solve the BVP in Ω_2 first (so that the left hand side becomes u_{2y} , since $u_{1y}(x, y^*) = u_{2y}(x, y^*)$), and consequently the solution to u_2 on interface helps complete the BVP set-up in Ω_1 . The calculation does not take iteration but the solution is not fully accurate because the boundary condition is not exactly transferred.

If U^* is a constant, Eq.(2.23) would be exact because $W = \tilde{W}U^*$. Note here the linear operation with \tilde{W} fully captures non-local nature of W but this is a special case (since U^* is constant). In more general cases (U^* is not constant), the term ' $\tilde{W}U^*$ ' merely captures part of the non-local nature of W , as will be shown by more quantified analysis in next subsection (2.4.2).

2.4.2 DR type ENDD algorithm

Consider the same BVP as Eq.(2.21). Clearly, the key to approximating W precisely is to find a good method to represent W in a series of non-local terms.

Consider the following BVPs defined in $[0, 1] \times [0, y^*]$ [81]:

$$\mathcal{L}w_n = 0, \quad (2.24)$$

$$w_n(x, 0) = 0, \quad (2.25)$$

$$w_n(0, y) = 0, \quad (2.26)$$

$$w_n(1, y) = 0, \quad (2.27)$$

$$w_n(x, y^*) = s_n(x), \quad (2.28)$$

where s_n is a set of basis function ($n = 1, 2, \dots$). Then let $W \approx \sum_{n=0}^{n=N} u_n^* w_n$ with u_n^*

satisfying

$$\min_{u_n^*} \|U^* - \sum_0^N u_n^* s_n(x)\|_2. \quad (2.29)$$

Assume $s_0 = 1$ (then $w_0 = \tilde{W}$ in last Subsection 2.4.1) and $N \geq 1$, expand W in terms of w_n and transform:

$$W(x, y) \approx \sum_0^N u_n^* w_n + (U^* - \sum_0^N u_n^* s_n) w_0, \quad (2.30)$$

$$= U^* w_0 + \sum_1^N (w_n - s_n w_0) u_n^*. \quad (2.31)$$

Take derivative and complement V , then it comes to the Robin type boundary condition of the BVP1 at interface:

$$\frac{\partial u_1(x, y^*)}{\partial y} = u_1(x, y^*) \frac{\partial w_0}{\partial y} + \sum_1^N \left(\frac{\partial w_n}{\partial y} - s_n \frac{\partial w_0}{\partial y} \right) u_n^* + \frac{\partial V}{\partial y}. \quad (2.32)$$

Eq.(2.32) shows the 'structure' of u_1 in terms of w_n , which contribute to the non-local nature of u_1 together with V . However, it is difficult to compute u_n^* until U^* is known, see Eq.(2.29). One remedy is to drop the second term and retains only the first and last terms of the right hand side of Eq.(2.32). In other words, it is equal to simplifying the case to $N = 0$ at the cost of computing accuracy lost. This way, only two BVPs need to be computed in advance: w_0 and V , on a separated mesh from Ω_2 , and those two BVPs do capture part of the non-local nature of u_1 . This leads to the same ANDD boundary condition as Eq.(2.23) shows in Subsection 2.4.1.

Recall that Eq.(2.22) is actually inspired by one-dimensional local case, and when the strategy is applied to two-dimensional equation, it naturally fails to represent the entire non-local effect. Eq.(2.32) elaborates the failure in quantity: $s_0 = 1$ only represents the constant component of U^* and $W = U^* w_0$ is exact if U^* is simply constant. Otherwise, there must be other terms of s_n ($n \neq 0$) representing more complex nature of U^* and the corresponding w_n that also contribute to the non-local nature of W could not be represented in Eq.(2.23). Therefore, the more constant-like the function U^* , the more accurate approximation Eq.(2.23) makes. In this project, the inner region Ω_1 mainly simulates the boundary layer zone where the variation mostly occurs in y direction. This is the physical foundation of domain decomposition method applied to solving near-wall turbulence problem.

To obtain exact boundary condition, any simplification to Eq.(2.32) is not allowed but there remain so many unknown terms, therefore iterative process is needed. One possible iteration

scheme could be designed as follows: re-organise Eq.(2.31):

$$\begin{aligned}
W(x, y) &= U^* w_0 + \sum_{n=1}^N (w_n - s_n w_0) u_n^*, \\
&= U^* w_0 + \sum_{n=0}^N u_n^* w_n - (u_0^* w_0 + \sum_{n=1}^N u_n^* s_n w_0), \\
&= \sum_{n=0}^N u_n^* w_n + w_0 (U^* - \sum_{n=0}^N u_n^* s_n).
\end{aligned} \tag{2.33}$$

Complement V and take derivative on both sides, the equation yields to

$$\frac{\partial u_1(x, y^*)}{\partial y} = \sum_{n=0}^N u_n^* \frac{\partial w_n}{\partial y} + \frac{\partial w_0}{\partial y} (U^* - \sum_{n=0}^N u_n^* s_n) + \frac{\partial V}{\partial y}. \tag{2.34}$$

The second term on the right hand side is actually 0, but it is only true when U^* has been obtained. Before that, the second term retains and it offers space to arrange iterations: substitute $\frac{\partial u_1(x, y^*)}{\partial y}$ with $\frac{\partial u_2(x, y^*)}{\partial y}$ on the left and U^* with $u_2(x, y^*)$ on the right, the above equation yields to the following interface boundary condition for BVP2:

$$\frac{\partial u_2(x, y^*)}{\partial y} = \frac{\partial u_1(x, y^*)}{\partial y} + \frac{\partial w_0}{\partial y} (u_2(x, y^*) - u_1(x, y^*)). \tag{2.35}$$

Note that in Eq.(2.34), the term $\sum_{n=0}^N u_n^* s_n$ is considered as a whole and is equal to $u_1(x, y^*)$ (not known yet). The first and last terms on the right hand side are also considered together as $u_{1y}(x, y^*)$ and not explicitly known as well. Not one thing is neglected in the algorithm, so the algorithm is exact. However, the cost is that not one thing is explicitly known (unlike in ANDD case, w_0 and V are known by pre-calculation) and all $w_n(s_n)$ components have to be taken as a whole to be solved in iterations. As is well known, iterative method is much concerned about convergence and the convergence analysis of the ENDD algorithm would be performed in Chapter 3 and 4.

Eq(2.35) is the Robin type boundary condition for solving BVP in Ω_2 . Once the solution in Ω_2 is available, the interface value could be used as Dirichlet type boundary condition for solving BVP in Ω_1 , and a relaxation parameter θ ($0 < \theta \leq 1$) is introduced:

$$u_1 = \theta u_2 + (1 - \theta) u_1.$$

Overall, the algorithm is called Dirichlet-to-Robin exact domain decomposition algorithm. The algorithm starts from solving u_2 and requires iterations:

$$\frac{\partial u_2^{k+1}}{\partial y} = \frac{\partial u_1^k}{\partial y} + \frac{\partial w_0}{\partial y} (u_2^{k+1} - u_1^k) \quad \text{on } \Gamma, \tag{2.36}$$

$$u_1^{k+1} = \theta u_2^{k+1} + (1 - \theta) u_1^k \quad \text{on } \Gamma. \tag{2.37}$$

Note there are more than just DR type ENDD algorithm. If the second term on the right hand

side of Eq.(2.36) is dropped, the Robin type IBC reduces to Neumann type and DR algorithms becomes so-called DN algorithm. Converting the interface interaction from Dirichlet into Neumann type makes the algorithm become Neumann-to-Neumann (NN) type. The study of the two algorithms could be found in [83], [94], and the convergence rate calculation is presented in Appendix A.

2.5 Non-local RR type domain decomposition method

This section introduces an important concept that is frequently used in the project: Steklov-Poincaré operators, and interprets the ENDD algorithm from a different angle using the operators. Mathematically, Steklov-Poincaré operators represent the Dirichlet-to-Neumann map of the BVP on the interface thus capturing the non-local nature of the problem. As will be seen shortly, the operators applying to the interface corresponds to the derivative of W (defined in Subsection 2.4.1) on Γ . In the error equation, the governing equation is homogeneous as well as the boundary conditions on three sides. Therefore, W actually corresponds to the error function and Steklov-Poincaré operators maps the Dirichlet-to-Neumann values on the interface of the error equation in a global scale, which is of great significance to understanding the interfacial behaviour (convergence behaviour) of the error problem. It is the interface value of the error problem that is concerned in the convergence analysis. Then, one other exact domain decomposition method of RR type is introduced. The design of the method (algorithms) is inspired by Steklov-Poincaré interface equation and there are three variants of the algorithm that are closely associated with the approximation of Steklov-Poincaré operators.

2.5.1 Introduction of Steklov-Poincaré operators

Let us start from considering a pair of linear BVPs [94] in domain decomposition framework:

$$\begin{aligned} \mathcal{L}u_1 &= f & \text{in } \Omega_1, \\ u_1 &= 0 & \text{on } \partial\Omega_1 \cap \partial\Omega, \\ u_1 &= u_2 & \text{on } \Gamma, \\ \frac{\partial u_1}{\partial n} &= \frac{\partial u_2}{\partial n} & \text{on } \Gamma, \\ \mathcal{L}u_2 &= f & \text{in } \Omega_2, \\ u_2 &= 0 & \text{on } \partial\Omega_2 \cap \partial\Omega. \end{aligned}$$

Note n denotes upward normal derivative. The above BVPs are the re-formulation of the original BVP defined in an entire domain Ω :

$$\begin{aligned} \mathcal{L}u &= f & \text{in } \Omega, \\ u &= 0 & \text{on } \partial\Omega. \end{aligned}$$

Now, consider one other pair of linear BVPs ($i = 1, 2$):

$$\begin{aligned}\mathcal{L}w_i &= f & \text{in } \Omega_i, \\ w_i &= 0 & \text{on } \partial\Omega_i \cap \partial\Omega \\ w_i &= 0 & \text{on } \Gamma.\end{aligned}$$

It could be stated that

$$w_i = u_i^0 + u_i^*, \quad (2.38)$$

if u_i^0 and u_i^* satisfy the following BVPs:

$$\begin{aligned}\mathcal{L}u_i^0 &= 0 & \text{in } \Omega_i, \\ u_i^0 &= 0 & \text{on } \partial\Omega_i \cap \partial\Omega, \\ u_i^0 &= \lambda & \text{on } \Gamma.\end{aligned}$$

$$\begin{aligned}\mathcal{L}u_i^* &= f & \text{in } \Omega_i, \\ u_i^* &= 0 & \text{on } \partial\Omega_i \cap \partial\Omega, \\ u_i^* &= 0 & \text{on } \Gamma.\end{aligned}$$

Note that for each $i = 1, 2$, u_i^0 is the harmonic extension of λ into Ω_i , and will be denoted by $\mathcal{H}_i\lambda$, and u_i^* will be denoted by \mathcal{G}_if .

Compare u_i and w_i , it follows that

$$w_i = u_i \quad i = 1, 2,$$

if and only if

$$\begin{aligned}\frac{\partial w_1}{\partial n} &= \frac{\partial w_2}{\partial n} & \text{on } \Gamma, \\ \rightarrow \frac{\partial u_1^0}{\partial n} + \frac{\partial u_1^*}{\partial n} &= \frac{\partial u_2^0}{\partial n} + \frac{\partial u_2^*}{\partial n} & \text{on } \Gamma, \\ \rightarrow \frac{\partial u_1^0}{\partial n} - \frac{\partial u_2^0}{\partial n} &= \frac{\partial u_2^*}{\partial n} - \frac{\partial u_1^*}{\partial n} & \text{on } \Gamma.\end{aligned} \quad (2.39)$$

It could be seen that Eq.(2.39) amounts to the requirement that λ satisfies the Steklov-Poincaré interface equation [94]:

$$S\lambda = \chi, \quad (2.40)$$

where

$$\chi := \frac{\partial}{\partial n}\mathcal{G}_2f - \frac{\partial}{\partial n}\mathcal{G}_1f, \quad (2.41)$$

and Steklov-Poincaré operator [95] is defined as

$$S\eta := \frac{\partial}{\partial n}\mathcal{H}_1\eta - \frac{\partial}{\partial n}\mathcal{H}_2\eta. \quad (2.42)$$

In particular, S can be split into $S = S_1 + S_2$:

$$S_i \eta := \frac{\partial}{\partial n^i} \mathcal{H}_i \eta_i, \quad i = 1, 2, \quad (2.43)$$

where n^i denotes the outward normal derivative direction of domain Ω_i .

Note SP operator is defined by Eq.(2.42) as S , but in the thesis, the split form S_1 and S_2 are more frequently used and hence denote the concept of SP operators.

Recall Eq.(2.35) represents the Robin type boundary condition of BVP2, and clearly the derivative $\frac{\partial w_0}{\partial y}$ is equal to $S_1 I$: S_1 applied to the given differential equation with interface value 1. I is the unit function. Therefore, the DR algorithm could also be written as:

$$\frac{\partial u_2^{k+1}}{\partial y} = \frac{\partial u_1^k}{\partial y} + S_1 I(u_2^{k+1} - u_1^k) \quad \text{on } \Gamma, \quad (2.44)$$

$$u_1^{k+1} = \theta u_2^{k+1} + (1 - \theta) u_1^k \quad \text{on } \Gamma. \quad (2.45)$$

2.5.2 RR type ENDD algorithms

Now it is clear that the domain decomposition method could be realised by some iterative procedures for solving Steklov-Poincaré interface equation, see Eq.(2.40). Meanwhile, the following equations hold true:

$$\begin{aligned} \frac{\partial}{\partial n} \mathcal{G}_1 f &= \frac{\partial u_1}{\partial n} - S_1 \eta, \\ \frac{\partial}{\partial n} \mathcal{G}_2 f &= \frac{\partial u_2}{\partial n} + S_2 \eta. \end{aligned}$$

If the exact solution to u_1 and u_2 is given, it is simply true that $\frac{\partial}{\partial n} \mathcal{G}_1 f = \frac{\partial u_1}{\partial n} - S_1 \eta = \frac{\partial u_2}{\partial n} - S_1 \eta$ and $\frac{\partial}{\partial n} \mathcal{G}_2 f = \frac{\partial u_2}{\partial n} + S_2 \eta = \frac{\partial u_1}{\partial n} + S_2 \eta$, but the solution is yet to be found actually. Therefore, consider some parallel iterative process to solve u_1 and u_2 :

$$\begin{aligned} \frac{\partial u_1^{k+1}}{\partial n} + S_2 u_{1|\Gamma}^{k+1} &= \frac{\partial u_2^k}{\partial n} + S_2 u_{2|\Gamma}^k \quad \text{on } \Gamma, \\ \frac{\partial u_2^{k+1}}{\partial n} - S_1 u_{2|\Gamma}^{k+1} &= \frac{\partial u_1^k}{\partial n} - S_1 u_{1|\Gamma}^k \quad \text{on } \Gamma. \end{aligned} \quad (2.46)$$

Nataf proposed the algorithm and proved the convergence to be guaranteed in only 2 iterations [96]. However, the key challenge is how to compute $S_1 \eta$ and $S_2 \eta$ with low cost. Actually, SP operators represent a kind of N2D map [97] of the given problem at interface, and the map carries all the non-local effect of u_i^0 . Therefore, the approximation of SP operators could be interpreted as the reservation of the non-local effect of the problem when dealing with the derivative in y direction.

Over decades, there have been much efforts in developing RR domain decomposition algorithm as an effective tool to solve linear BVP and Eq.(2.46) is one of them. The development of RR algorithm is based on the classic Schwartz Alternating method [98] that

was originally developed as an overlapping DD approach. Then [99], [100] adopt the method into non-overlapping DD method with Robin type IBCs and [101] formulates the IBC to an equivalent form but with less normal derivative included. [102] proposes optimised parameter for the algorithm based on the convergence analysis with Fourier analysis. Similar analysis method is also presented in [96]. The algorithms in [99], [101] are essentially the same and are denoted as conventional RR algorithm in the thesis.

Taylor series expand a function into a series of terms of its derivatives on a single point, see Eq.(2.47). Near the point, the series could approximate the real function value with acceptable error, but as the approximated point moves further off the single point on which the Taylor series are expanded, the approximation accuracy declines. Therefore, Taylor series could be considered as a local approximation of a function. To overcome the local effect, the fixed (base) point must 'shift' along the interface to make good 'global' approximation of the function. For instance, when approximating $f(x)$, select $x_0 = x + \delta$ (where $\delta \rightarrow 0$, could be the space step of mesh) as the base point to expand Taylor series, and always select new x_0 as making approximation of $f(x)$ elsewhere. This way, each $f(x)$ is approximated by Taylor series expanded on the nearest base point x_0 thus the error is minimised. In other words, the 'global' approximation is performed by combining multiple piece-wise local Taylor expansions on ever-shifting x_0 along the entire interface.

$$\begin{aligned} f(x) &= \sum_{n=0}^{n=\infty} \frac{f^{(n)}(x_0)}{n!} (x - x_0)^n, \\ &= f(x_0) + \frac{f'(x_0)}{1!} (x - x_0) + \frac{f''(x_0)}{2!} (x - x_0)^2 + \dots \end{aligned} \quad (2.47)$$

Having obtained approximation of interface value with Taylor series, it is now possible to approximate $S_i \eta$, ($i = 1, 2$) globally. Depending on the number of expansion terms of Taylor series, there are three variants of the SP operator approximation.

Constant approximation:

$$S_i f(x) = S_i f(x_0) = f(x_0) S_i I.$$

Linear approximation:

$$\begin{aligned} S_i f(x) &= S_i [f(x_0) + f'(x_0)(x - x_0)], \\ &= (f(x_0) - x_0 f'(x_0)) S_i I + f'(x_0) S_i x. \end{aligned}$$

Quadratic approximation:

$$\begin{aligned} S_i f(x) &= S_i [f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2} (x - x_0)^2], \\ &= (f(x_0) - x_0 f'(x_0) + \frac{x_0^2 f''(x_0)}{2}) S_i I + (f'(x_0) - x_0 f''(x_0)) S_i x + \\ &\quad \frac{f''(x_0)}{2} S_i x^2. \end{aligned}$$

Following the algorithm as shown in Eq.(2.46) and the three variants of SP operator approximation, the following three algorithms could be formulated [84]:

RR-SP0:

$$\frac{\partial u_1^{k+1}}{\partial y} + S_2 I u_1^{k+1} = \frac{\partial u_2^k}{\partial y} + S_2 I u_2^k \quad \text{on } \Gamma, \quad (2.48)$$

$$-\frac{\partial u_2^{k+1}}{\partial y} + S_1 I u_2^{k+1} = -\frac{\partial u_1^k}{\partial y} + S_1 I u_1^k \quad \text{on } \Gamma. \quad (2.49)$$

RR-SP1:

$$\frac{\partial u_1^{k+1}}{\partial y} + S_2 I u_1^{k+1} + \frac{\partial u_1^{k+1}}{\partial x} T_{1l} = \frac{\partial u_2^k}{\partial y} + S_2 I u_2^k + \frac{\partial u_2^k}{\partial x} T_{1l} \quad \text{on } \Gamma, \quad (2.50)$$

$$-\frac{\partial u_2^{k+1}}{\partial y} + S_1 I u_2^{k+1} + \frac{\partial u_2^{k+1}}{\partial x} T_{2l} = -\frac{\partial u_1^k}{\partial y} + S_1 I u_1^k + \frac{\partial u_1^k}{\partial x} T_{2l} \quad \text{on } \Gamma, \quad (2.51)$$

where

$$T_{1l} = S_2 x - x_0 S_2 I, \quad T_{2l} = S_1 x - x_0 S_1 I. \quad (2.52)$$

RR-SP2:

$$\begin{aligned} \frac{\partial u_1^{k+1}}{\partial y} + S_2 I u_1^{k+1} + \frac{\partial u_1^{k+1}}{\partial x} T_{1l} + \frac{\partial^2 u_1^{k+1}}{2 \partial x^2} T_{1q} = \\ \frac{\partial u_2^k}{\partial y} + S_2 I u_2^k + \frac{\partial u_2^k}{\partial x} T_{1l} + \frac{\partial^2 u_2^k}{2 \partial x^2} T_{1q} \quad \text{on } \Gamma, \end{aligned} \quad (2.53)$$

$$\begin{aligned} -\frac{\partial u_2^{k+1}}{\partial y} + S_1 I u_2^{k+1} + \frac{\partial u_2^{k+1}}{\partial x} T_{2l} + \frac{\partial^2 u_2^{k+1}}{2 \partial x^2} T_{2q} = \\ -\frac{\partial u_1^k}{\partial y} + S_1 I u_1^k + \frac{\partial u_1^k}{\partial x} T_{2l} + \frac{\partial^2 u_1^k}{2 \partial x^2} T_{2q} \quad \text{on } \Gamma, \end{aligned} \quad (2.54)$$

where T_{1l} and T_{2l} are identical to Eq.(2.52) and

$$T_{1q} = S_2 x^2 - 2x S_2 x + x^2 S_2 I, \quad T_{2q} = S_1 x^2 - 2x S_1 x + x^2 S_1 I. \quad (2.55)$$

Practically, the constant approximation requires the computing of $S_i I$ in advance; and linear approximation requires $S_i I$ and $S_i x$; and quadratic approximation requires $S_i I$, $S_i x$ and $S_i x^2$. The pre-computing could take place in separate mesh between subdomains and once complete, the result could be used multiple times for the equation $\mathcal{L}u = f$ with different right hand side f . Apparently, the higher order of Taylor series expansion comes with more accurate approximation of SP operators and less iterations of the algorithm to achieve convergence, but the pre-computation is more computationally expensive. There should be a trade-off between the cost of pre-computation and algorithm efficiency. Empirically, the numerical tests conducted in the project suggests that in quadratic case, the pre-computation of $S_i I$, $S_i x$ and $S_i x^2$ takes such amount of time that the algorithm is supposed to converge in 2 iterations,

otherwise the algorithm can not save time in account of pre-computation cost.

The algorithms introduced above are parallel, which abide by classical domain decomposition principle of parallel computing. Regarding this project in which only 2 subdomains are generated, sequential computing is also worth considering. As will be seen in Chapter 3, the convergence rate of sequential version is actually 2 times faster than the parallel counterpart, which could be confirmed by numerical tests.

The sequential version of RR-SP0, RR-SP1 and RR-SP2 all starts from computing u_2 and reads as follows:

RR-SP0 (sequential):

$$-\frac{\partial u_2^{k+1}}{\partial y} + S_1 I u_2^{k+1} = -\frac{\partial u_1^k}{\partial y} + S_1 I u_1^k \quad \text{on } \Gamma, \quad (2.56)$$

$$\frac{\partial u_1^{k+1}}{\partial y} + S_2 I u_1^{k+1} = \frac{\partial u_2^{k+1}}{\partial y} + S_2 I u_2^{k+1} \quad \text{on } \Gamma. \quad (2.57)$$

RR-SP1 (sequential):

$$-\frac{\partial u_2^{k+1}}{\partial y} + S_1 I u_2^{k+1} + \frac{\partial u_2^{k+1}}{\partial x} T_{2l} = -\frac{\partial u_1^k}{\partial y} + S_1 I u_1^k + \frac{\partial u_1^k}{\partial x} T_{2l} \quad \text{on } \Gamma, \quad (2.58)$$

$$\frac{\partial u_1^{k+1}}{\partial y} + S_2 I u_1^{k+1} + \frac{\partial u_1^{k+1}}{\partial x} T_{1l} = \frac{\partial u_2^{k+1}}{\partial y} + S_2 I u_2^{k+1} + \frac{\partial u_2^{k+1}}{\partial x} T_{1l} \quad \text{on } \Gamma, \quad (2.59)$$

where T_{1l} and T_{2l} coincide with those in Eq.(2.52).

RR-SP2 (sequential):

$$\begin{aligned} -\frac{\partial u_2^{k+1}}{\partial y} + S_1 I u_2^{k+1} + \frac{\partial u_2^{k+1}}{\partial x} T_{2l} + \frac{\partial^2 u_2^{k+1}}{2\partial x^2} T_{2q} = \\ -\frac{\partial u_1^k}{\partial y} + S_1 I u_1^k + \frac{\partial u_1^k}{\partial x} T_{2l} + \frac{\partial^2 u_1^k}{2\partial x^2} T_{2q} \quad \text{on } \Gamma, \end{aligned} \quad (2.60)$$

$$\begin{aligned} \frac{\partial u_1^{k+1}}{\partial y} + S_2 I u_1^{k+1} + \frac{\partial u_1^{k+1}}{\partial x} T_{1l} + \frac{\partial^2 u_1^{k+1}}{2\partial x^2} T_{1q} = \\ \frac{\partial u_2^{k+1}}{\partial y} + S_2 I u_2^{k+1} + \frac{\partial u_2^{k+1}}{\partial x} T_{1l} + \frac{\partial^2 u_2^{k+1}}{2\partial x^2} T_{1q} \quad \text{on } \Gamma, \end{aligned} \quad (2.61)$$

where T_{1l} and T_{2l} coincide with those in Eq.(2.52), and T_{1q} and T_{2q} are identical to Eq.(2.55).

Nataf has proved the convergence of the algorithm as displayed in Eq.(2.46) guaranteed to be achieved in 2 iterations but it could not be applied to RR-SPN ($N = 0, 1, 2$) algorithms directly. The convergence analysis of RR-SPN (both parallel and sequential) will be performed in Chapter 3 and 4.

2.6 Summary

This chapter presents the derivation of the non-local NDD method, which originates from the Robin interface boundary condition transfer for boundary-layer-type equation. When applied to HRN models and LRN models, the method yields to generalised wall function and near-wall domain decomposition method respectively. The derivation follows the development of the method from one-dimensional to two-dimensional evolution, and focuses on the technique of retaining the non-local effect during the boundary condition transfer. Compared with AWF, the NDD method could prove its advantage in terms of accuracy, efficiency, applicability and solution independence on mesh scheme as well as interface boundary position.

SP operators play a key role in retaining the non-local effect of the problem thus the approximation of SP operators becomes especially significant. There are more than one way to implement the approximation and this chapter shows two approaches that yield to two different types of the NDD methods: namely DR and RR. DR has been previously studied in existing literature and is used for the purpose of comparison in the thesis. Regarding RR method, there are three variants (algorithms) of the method: RR-SP0, RR-SP1 and RR-SP2 which are based on the local expansion of Taylor series with different number of terms. Although Taylor expansion is local, the expansion is considered as a whole all along the entire interface with implicit formulation for every expansion node, therefore the local expansion becomes almost global given there is fine node density on the interface. The accurate global approximation of the operators is the key to the retaining of the non-local nature of the given problem when constructing the IBCs between the two subdomains.

Chapter 3

Convergence rate calculation of NDD algorithms

This chapter performs the calculation of the convergence rate of RR-SP0 (both parallel and sequential), RR-SP1 (sequential), RR-SP2 (sequential) and DR (sequential) algorithms. The analysis of the calculation result would be elaborated in Chapter 4.

Fourier method is used to calculate the convergence rate of the algorithms, which are applied to solving Poisson's equation because the solution to Poisson's equation could be obtained analytically. Poisson's equation mimics part of the nature of the RANS model equations whose analytical solution could not be obtained, so the conclusion drawn from Poisson's equation is expected to be analogous to that applied to RANS model equations to some extent.

In Section 3.1, the convergence rate calculation of RR-SP0 is performed using two methods: separation of variables and Fourier transform. Both methods make minor simplification on calculation of $S_i I$. The method of separation of variables assumes $S_i I$ is a constant, which is not true at the vicinity of the boundaries in x direction. The method of Fourier transform assumes unit function I is defined in $(-\infty, \infty)$ but it actually has boundaries, which is $[0, -1]$ in the project. With the two simplification, both methods result in identical calculation result. This is because both simplifications are equal to removing the boundary effect in x direction when calculating $S_i I$, and the boundary conditions at x direction are too far to impose any effect on the solution. In practice, the solution accuracy loss due to the simplification could be ignored when the boundary effect is not strong.

In Section 3.2, the convergence rate calculation of DR algorithm is performed using separation of variables. The calculation result is identical to that in [83], which originally proposed the algorithm and studied it in detail. In this project, DR algorithm is not the main object but mainly an algorithm to compare with RR-SPN ($N = 0, 1, 2$) algorithms.

The calculation of Poisson's equation shows that the convergence rate of RR-SP0 algorithm could be completely expressed by the symbols of SP operators: \hat{S}_i ($i = 1, 2$). In Section 3.3, the conclusion is proved applicable beyond Poisson's equation itself. Poisson's equation has analytical solution and its convergence rate could be obtained when necessary, so studying Poisson's equation alone does not help much on understanding other problems. However, Poisson's equation represents a kind of equations that have variable-separation form solutions

and the conclusions drawn from Poisson's equation also apply to those equations as well. The convergence rate of those equations has the same form/structure as Poisson's equation. What is more, for the BVPs whose solution has limited dependence on x , the Poisson result could be partially applied as well. Therefore, by merely studying the properties of the SP operator symbols in application to given equations, the convergence rate could be obtained directly. The cost of calculating the SP operator symbols is low because if the solution has variable-separation form or weak dependence on x , the problem could be reduced to one-dimensional.

In Section 3.4, the convergence rate of RR-SP1 and RR-SP2 algorithms is calculated by Fourier transform. The calculation is performed based on limited assumptions: I , x and x^2 are all defined in $(-\infty, \infty)$. Meanwhile, the assumptions are made in order: the calculation result of the lower-order algorithm is used as assumption for the calculation of the higher-order algorithm. For example, $S_i I$ is taken as constant in the calculation of RR-SP1 because it has been justified by the calculation result of RR-SP0, which assumes I is defined in $(-\infty, \infty)$. Similarly, $S_i x = S_i^* I \cdot x$ is presumed in the calculation of RR-SP2 because it is the result of the RR-SP1 calculation. The aim of making all those assumptions is to remove any boundary effect whenever it appears so that the convergence rate obtained is 'boundary effect free'. In practice, when the boundary effect is negligible, the result could be used directly. Otherwise, the result could provide a foundation to evaluate the real case where the boundary effect should be held accountable. The influence of the boundary effect in RR-SP0 will be discussed in Chapter 4.

3.1 Convergence rate calculation of RR-SP0

This section considers RR-SP0 applied to Poisson's equation and performs the calculation of the convergence rate by 2 approaches: separation of variables and Fourier transform. They differ in some technical ways and lead to identical result. Note in this section, there are two simplifications made regarding calculating $S_i I$ in the two methods:

1. In separation of variables, $S_i I$ ($i = 1, 2$) is considered as constant, and the constant is denoted by $S_i^* I$.
2. In Fourier transform, the unit function I is defined in $(-\infty, \infty)$.

The two simplifications both amount to removing the boundary effect in x direction.

3.1.1 Calculation via separation of variables

Consider a Poisson's equation defined in a square domain $\Omega : \{0 \leq x \leq 1, 0 \leq y \leq B\}$. An artificial interface boundary Γ (at $y = A$) splits the domain into two: Ω_1 and Ω_2 . The following iterative equations formulate the boundary conditions of the two subdomains, as is

displayed in Chapter 2 as RR-SP0 (sequential). The algorithm starts from u_2 .

$$\begin{aligned} u_{1y}^{n+1} + S_2^* I \cdot u_1^{n+1} &= u_{2y}^{n+1} + S_2^* I \cdot u_2^{n+1}, \\ -u_{2y}^{n+1} + S_1^* I \cdot u_2^{n+1} &= -u_{1y}^n + S_1^* I \cdot u_1^n. \end{aligned}$$

The corresponding IBCs for the error problem are:

$$\begin{aligned} e_{1y}^{n+1} + S_2^* I \cdot e_1^{n+1} &= e_{2y}^{n+1} + S_2^* I \cdot e_2^{n+1}, \\ -e_{2y}^{n+1} + S_1^* I \cdot e_2^{n+1} &= -e_{1y}^n + S_1^* I \cdot e_1^n. \end{aligned}$$

Let on interface:

$$\gamma_1^{n+1} := e_{2y}^{n+1}(x, A) + S_2^* I \cdot e_2^{n+1}(x, A), \quad (3.1)$$

$$\gamma_2^{n+1} := -e_{1y}^n(x, A) + S_1^* I \cdot e_1^n(x, A), \quad (3.2)$$

and the error problem could be written as Laplace equation with boundary conditions:

$$-\nabla^2 e_1^{n+1} = 0, \quad \text{on } \Omega_1, \quad (3.3)$$

$$e_1^{n+1} = 0, \quad \text{on } \partial\Omega_1/\Gamma, \quad (3.4)$$

$$e_{1y}^{n+1} + S_2^* I \cdot e_1^{n+1} = \gamma_1^{n+1}, \quad \text{on } \Gamma, \quad (3.5)$$

$$-\nabla^2 e_2^{n+1} = 0, \quad \text{on } \Omega_2, \quad (3.6)$$

$$e_2^{n+1} = 0, \quad \text{on } \partial\Omega_2/\Gamma, \quad (3.7)$$

$$-e_{2y}^{n+1} + S_1^* I \cdot e_2^{n+1} = \gamma_2^{n+1}, \quad \text{on } \Gamma. \quad (3.8)$$

Laplace equation could be solved using separation with variables that reduces two-dimensional partial differential equation (PDE) to 2 ordinary differential equations (ODEs). The solution to the PDE has the form

$$e_i(x, y) = X_i(x)Y_i(y).$$

Substitute the general form into Laplace equation yields to 2 ODEs with respect to x and y respectively. Note λ is the separation constant:

$$\frac{1}{X_i} \frac{\partial^2 X_i}{\partial x^2} = \lambda, \quad (3.9)$$

$$-\frac{1}{Y_i} \frac{\partial^2 Y_i}{\partial y^2} = \lambda. \quad (3.10)$$

First, apply the homogeneous boundary conditions with respect to x to solve the ODEs about x (Eq.(3.9)). The homogeneous boundary conditions could be separated from original boundary conditions:

$$X_i(0) = 0, \quad X_i(1) = 0. \quad (3.11)$$

The ODEs as shown in Eq.(3.9) have the general solution:

$$X_i(x) = C_i \sin(k\pi x),$$

and the separation constant is also determined: $\lambda = k^2\pi^2$, $k = 1, 2, \dots$

Second, consider the ODE with respect to y , as shown in Eq.(3.10):

$$-\frac{1}{Y_i} \frac{\partial^2 Y_i}{\partial y^2} = k^2\pi^2.$$

The ODE has the general form

$$Y_i(y) = A'_i \sinh(k\pi y) + B'_i \cosh(k\pi y).$$

Therefore, the solution to $e_i(x, y)$ has the form:

$$e_i(x, y) = \sum_{k=1}^{\infty} \sin(k\pi x) (A_i \sinh(k\pi y) + B_i \cosh(k\pi y)).$$

So far, the solution has been formulated as Fourier sine series. The Fourier coefficient remains unsolved and denote it by \hat{e}_i^{n+1} :

$$\hat{e}_i^{n+1}(k, y) := A_i^{n+1} \sinh(k\pi y) + B_i^{n+1} \cosh(k\pi y). \quad (3.12)$$

In this step, \hat{e}_i starts to have superscript $(n + 1)$ because the solution is subjected to IBCs that vary with iterations. Correspondingly, A_i and B_i have superscript too. To determine the coefficient (in other words, A_i^{n+1} and B_i^{n+1}), one needs to consider the yet-unused boundary conditions on interface Γ and those on $y = 0$ and $y = B$. Since the solution e_i has Fourier series form, γ_i should be expanded to Fourier sine series too and only the Fourier coefficients are considered:

$$\gamma_i^{n+1}(x, y) = \sum_{k=1}^{\infty} \hat{\gamma}_i^{n+1}(k, y) \sin(k\pi x),$$

and the boundary conditions with respect to y read

$$\text{BVP1} : \begin{cases} \hat{e}_1^{n+1}(x, 0) = 0, \\ \hat{e}_{1y}^{n+1}(k, A) + S_2^* I \cdot \hat{e}_1^{n+1}(k, A) = \hat{\gamma}_1^{n+1}(k, A). \end{cases} \quad (3.13)$$

$$\text{BVP2} : \begin{cases} \hat{e}_2^{n+1}(x, 0) = 0, \\ -\hat{e}_{2y}^{n+1}(k, A - B) + S_1^* I \cdot \hat{e}_2^{n+1}(k, A - B) = \hat{\gamma}_2^{n+1}(k, A - B). \end{cases} \quad (3.14)$$

Note there is a variable substitute $\bar{y} = y - B$ when considering \hat{e}_2 . The final solution could be restored later on.

Apply above boundary conditions to Eq.(3.12), \hat{e}_i could be obtained:

$$\begin{aligned}\hat{e}_1^{n+1}(k, y) &= \frac{\hat{\gamma}_1^{n+1}(k, A) \sinh(k\pi y)}{k\pi \cosh(k\pi A) + S_2^* I \sinh(k\pi A)}, \\ \hat{e}_2^{n+1}(k, y) &= \frac{\hat{\gamma}_2^{n+1}(k, A) \sinh(k\pi(y - B))}{-k\pi \cosh(k\pi(A - B)) + S_1^* I \sinh(k\pi(A - B))}.\end{aligned}$$

Particularly, the solution at interface is

$$\hat{e}_1^{n+1}(k, A) = \frac{\hat{\gamma}_1^{n+1}(k, A)}{k\pi \coth(k\pi A) + S_2^* I}, \quad (3.15)$$

$$\hat{e}_2^{n+1}(k, A) = \frac{\hat{\gamma}_2^{n+1}(k, A)}{-k\pi \coth(k\pi(A - B)) + S_1^* I}. \quad (3.16)$$

Next, expand $\hat{\gamma}_i^{n+1}$ and according to Eqs.(3.1)&(3.2), consider only the Fourier coefficients:

$$\begin{aligned}\hat{\gamma}_1^{n+1}(k, A) &:= \hat{e}_{2y}^{n+1} + S_2^* I \cdot \hat{e}_2^{n+1}, \\ &= (k\pi \coth(k\pi(A - B)) + S_2^* I) \hat{e}_2^{n+1}(k, A), \\ \hat{\gamma}_2^{n+1}(k, A) &:= -\hat{e}_{1y}^n + S_1^* I \cdot \hat{e}_1^n, \\ &= (-k\pi \coth(k\pi A) + S_1^* I) \hat{e}_1^n(k, A).\end{aligned}$$

Finally, the convergence rate is obtained:

$$\begin{aligned}&\hat{e}_1^{n+1}(k, A) \\ &= \frac{\hat{\gamma}_1^{n+1}(A)}{k\pi \coth(k\pi A) + S_2^* I}, \\ &= \frac{k\pi \coth(k\pi(A - B)) + S_2^* I}{k\pi \coth(k\pi A) + S_2^* I} \hat{e}_2^{n+1}(k, A), \\ &= \frac{k\pi \coth(k\pi(A - B)) + S_2^* I}{k\pi \coth(k\pi A) + S_2^* I} \frac{\hat{\gamma}_2^{n+1}(A)}{-k\pi \coth(k\pi(A - B)) + S_1^* I}, \\ &= \frac{k\pi \coth(k\pi(A - B)) + S_2^* I}{k\pi \coth(k\pi A) + S_2^* I} \frac{-k\pi \coth(k\pi A) + S_1^* I}{-k\pi \coth(k\pi(A - B)) + S_1^* I} \hat{e}_1^n(A).\end{aligned} \quad (3.17)$$

$$\begin{aligned}&\hat{e}_2^{n+1}(k, A) \\ &= \frac{\hat{\gamma}_2^{n+1}(A)}{-k\pi \coth(k\pi(A - B)) + S_1^* I}, \\ &= \frac{-k\pi \coth(k\pi A) + S_1^* I}{-k\pi \coth(k\pi(A - B)) + S_1^* I} \hat{e}_1^n(k, A), \\ &= \frac{-k\pi \coth(k\pi A) + S_1^* I}{-k\pi \coth(k\pi(A - B)) + S_1^* I} \frac{\hat{\gamma}_1^n(A)}{k\pi \coth(k\pi A) + S_2^* I}, \\ &= \frac{-k\pi \coth(k\pi A) + S_1^* I}{-k\pi \coth(k\pi(A - B)) + S_1^* I} \frac{k\pi \coth(k\pi(A - B)) + S_2^* I}{k\pi \coth(k\pi A) + S_2^* I} \hat{e}_2^n(k, A).\end{aligned} \quad (3.18)$$

Therefore, on interface: ($k = 1, 2, \dots$)

$$\|\hat{e}_i^{n+1}\| = \left| \frac{\frac{1}{A} - k\pi \coth(k\pi A)}{\frac{1}{A} + k\pi \coth(k\pi(B-A))} \frac{\frac{1}{B-A} - k\pi \coth(k\pi(B-A))}{\frac{1}{B-A} + k\pi \coth(k\pi A)} \right| \|\hat{e}_i^n\|, \quad (3.19)$$

as it is known that $S_1^* I = \frac{1}{A}$ and $S_2^* I = \frac{1}{B-A}$.

Note the solution to the following two one-dimensional BVPs defines $S_i^* I$: $S_1^* I := \frac{du_1(A)}{dy}$ and $S_2^* I := -\frac{du_2(A)}{dy}$.

$$\begin{aligned} \frac{d^2 u_1}{dy^2} &= 0, \\ u_1(0) &= 0, \quad u_1(A) = 1. \end{aligned}$$

$$\begin{aligned} \frac{d^2 u_2}{dy^2} &= 0, \\ u_2(A) &= 1, \quad u_2(B) = 0. \end{aligned}$$

3.1.2 Calculation of RR-SP0: parallel version

This subsection considers the parallel version of RR-SP0 algorithm in terms of convergence rate. The calculation would be performed in the same way as with sequential case in Subsection 3.1.1, and only the calculation that differs from sequential case would be displayed.

Recall the parallel IBC of RR-SP0 reads:

$$\begin{aligned} u_{1y}^{n+1} + S_2^* I \cdot u_1^{n+1} &= u_{2y}^n + S_2^* I \cdot u_2^n, \\ -u_{2y}^{n+1} + S_1^* I \cdot u_2^{n+1} &= -u_{1y}^n + S_1^* I \cdot u_1^n. \end{aligned}$$

The corresponding error BVPs to consider are identical to those as shown in Eqs.(3.3)-(3.8), but the definition of γ_1^{n+1} is slightly different:

$$\gamma_1^{n+1} := e_{2y}^n(x, A) + S_2^* I \cdot e_2^n(x, A), \quad (3.20)$$

$$\gamma_2^{n+1} := -e_{1y}^n(x, A) + S_1^* I \cdot e_1^n(x, A). \quad (3.21)$$

The above formulated BVPs could be solved with separation of variables as performed in Subsection 3.1.1. Much of the calculation is the same and thus skipped to the following intermediate stage:

$$\begin{aligned} \hat{e}_1^{n+1}(k, y) &= \frac{\hat{\gamma}_1^{n+1}(A) \sinh(k\pi y)}{k\pi \cosh(k\pi A) + S_2^* I \sinh(k\pi A)}, \\ \hat{e}_2^{n+1}(k, y) &= \frac{\hat{\gamma}_2^{n+1}(A) \sinh(k\pi(y-B))}{-k\pi \cosh(k\pi(A-B)) + S_1^* I \sinh(k\pi(A-B))}, \end{aligned}$$

where $\hat{\gamma}_i$ is the Fourier coefficient of γ_i :

$$\begin{aligned}\hat{\gamma}_1^{n+1} &:= \hat{e}_{2y}^n + S_2^* I \cdot \hat{e}_2^n(k, A), \\ \hat{\gamma}_2^{n+1} &:= -\hat{e}_{1y}^n + S_1^* I \cdot \hat{e}_1^n(k, A).\end{aligned}$$

Particularly, the solution at interface is

$$\hat{e}_1^{n+1}(k, A) = \frac{\hat{\gamma}_1^{n+1}(A)}{k\pi \coth(k\pi A) + S_2^* I}, \quad (3.22)$$

$$\hat{e}_2^{n+1}(k, A) = \frac{\hat{\gamma}_2^{n+1}(A)}{-k\pi \coth(k\pi(A - B)) + S_1^* I}. \quad (3.23)$$

Transform Eq.(3.22):

$$\begin{aligned}\hat{e}_1^{n+1}(k, A) &= \frac{\hat{\gamma}_1^{n+1}(A)}{k\pi \coth(k\pi A) + S_2^* I}, \\ &= \frac{k\pi \coth(k\pi(A - B)) + S_2^* I}{k\pi \coth(k\pi A) + S_2^* I} \hat{e}_2^n(k, A),\end{aligned} \quad (3.24)$$

$$\begin{aligned}&= \frac{k\pi \coth(k\pi(A - B)) + S_2^* I}{k\pi \coth(k\pi A) + S_2^* I} \frac{\hat{\gamma}_2^n(A)}{-k\pi \coth(k\pi(A - B)) + S_1^* I}, \\ &= \frac{k\pi \coth(k\pi(A - B)) + S_2^* I}{k\pi \coth(k\pi A) + S_2^* I} \frac{-k\pi \coth(k\pi A) + S_1^* I}{-k\pi \coth(k\pi(A - B)) + S_1^* I} \hat{e}_1^{n-1}(A).\end{aligned} \quad (3.25)$$

Transform Eq.(3.23):

$$\begin{aligned}\hat{e}_2^{n+1}(k, A) &= \frac{\hat{\gamma}_2^{n+1}(A)}{-k\pi \coth(k\pi(A - B)) + S_1^* I}, \\ &= \frac{-k\pi \coth(k\pi A) + S_1^* I}{-k\pi \coth(k\pi(A - B)) + S_1^* I} \hat{e}_1^n(k, A), \\ &= \frac{-k\pi \coth(k\pi A) + S_1^* I}{-k\pi \coth(k\pi(A - B)) + S_1^* I} \frac{\hat{\gamma}_1^n(A)}{k\pi \coth(k\pi A) + S_2^* I}, \\ &= \frac{-k\pi \coth(k\pi A) + S_1^* I}{-k\pi \coth(k\pi(A - B)) + S_1^* I} \frac{k\pi \coth(k\pi(A - B)) + S_2^* I}{k\pi \coth(k\pi A) + S_2^* I} \hat{e}_2^{n-1}(k, A).\end{aligned} \quad (3.26)$$

Therefore, the convergence rate is ($k = 1, 2, \dots$)

$$\|\hat{e}_i^{n+1}\| = \left| \frac{\frac{1}{A} - k\pi \coth(k\pi A)}{\frac{1}{A} + k\pi \coth(k\pi(B - A))} \frac{\frac{1}{B-A} - k\pi \coth(k\pi(B - A))}{\frac{1}{B-A} + k\pi \coth(k\pi A)} \right| \|\hat{e}_i^{n-1}\|. \quad (3.27)$$

3.1.3 Calculation via Fourier transform

This section makes the convergence analysis via Fourier transform. The final result is equal to that of separation of variables.

Consider a Poisson's equation again and the same boundary conditions. This time let us start from the error equation directly but note $S_i^* I$ is no more considered as constant. This makes the calculation complicated but it shall be simplified in another way.

$$-\nabla^2 e_1^{n+1} = 0, \quad \text{on } \Omega_1, \quad (3.28)$$

$$e_1^{n+1} = 0, \quad \text{on } \partial\Omega_1/\Gamma, \quad (3.29)$$

$$e_{1y}^{n+1} + S_2 I \cdot e_1^{n+1} = \gamma_1^{n+1}, \quad \text{on } \Gamma, \quad (3.30)$$

$$-\nabla^2 e_2^{n+1} = 0, \quad \text{on } \Omega_2, \quad (3.31)$$

$$e_2^{n+1} = 0, \quad \text{on } \partial\Omega_2/\Gamma, \quad (3.32)$$

$$-e_{2y}^{n+1} + S_1 I \cdot e_2^{n+1} = \gamma_2^{n+1}, \quad \text{on } \Gamma, \quad (3.33)$$

where $\gamma_1^{n+1} := e_{2y}^{n+1}(x, A) + S_2 I \cdot e_2^{n+1}(x, A)$ and $\gamma_2^{n+1} := -e_{1y}^n(x, A) + S_1 \cdot e_1^n(x, A)$.

Now apply Fourier transform with respect to x on the error equations as well as the boundary conditions, then the original two-dimensional BVPs are reduced to one-dimensional BVPs.

BVP1:

$$\frac{\partial^2 \hat{e}_1^{n+1}}{\partial y^2} - \omega^2 \hat{e}_1^{n+1} = 0, \quad (3.34)$$

$$\hat{e}_1^{n+1}(0) = 0, \quad (3.35)$$

$$\hat{e}_{1y}^{n+1}(A) + \frac{1}{2\pi} \mathcal{F}(S_2 I) * \hat{e}_1^{n+1}(A) = \hat{\gamma}_1^{n+1}(A). \quad (3.36)$$

BVP2:

$$\frac{\partial^2 \hat{e}_2^{n+1}}{\partial y^2} - \omega^2 \hat{e}_2^{n+1} = 0, \quad (3.37)$$

$$\hat{e}_2^{n+1}(0) = 0, \quad (3.38)$$

$$-\hat{e}_{2y}^{n+1}(A - B) + \frac{1}{2\pi} \mathcal{F}(S_1 I) * \hat{e}_2^{n+1}(A - B) = \hat{\gamma}_2^{n+1}(A). \quad (3.39)$$

Note in the formulation of BVP2, y -coordinate shifts by $y' = y - B$. The final result would be restored based on y afterwards. Meanwhile, convolution is defined as $f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau)g(x - \tau)d\tau$. $\mathcal{F}f(x)$ means applying Fourier transform on $f(x)$, such as $\mathcal{F}e_i^n(x, y) = \hat{e}_i^n(\omega, y)$. Expanding the Fourier transform of γ_i reads

$$\mathcal{F}\gamma_1^{n+1} := \hat{\gamma}_1^{n+1}(A) := \hat{e}_{2y}^n(\omega, A) + \mathcal{F}(S_2 I \cdot e_2^n(\omega, A)), \quad (3.40)$$

$$\mathcal{F}\gamma_2^{n+1} := \hat{\gamma}_2^{n+1}(A) := -\hat{e}_{1y}^n(\omega, A) + \mathcal{F}(S_1 \cdot e_1^n(\omega, A)). \quad (3.41)$$

The BVPs have general solution:

$$\hat{e}_1(\omega, y) = C_1(\omega) \sinh(\omega y) + D_1(\omega) \cosh(\omega y),$$

$$\hat{e}_2(\omega, y) = C_2(\omega) \sinh(\omega y) + D_2(\omega) \cosh(\omega y).$$

To determine the unknown coefficients of the general solution, the boundary conditions need to be considered. Since the solution to the problem has variable-separation form, the boundary conditions could be considered here independently from x . Firstly, applying the homogeneous boundary conditions Eqs.(3.35), (3.38) eliminate D_1, D_2 so that the general solution retains only $\sinh(\omega y)$ term: $\hat{e}_i(\omega, y) = C_i \sinh(\omega y)$. Then, apply the boundary conditions Eqs.(3.36), (3.39) to the general solution to determine C_1 and C_2 . Since the calculation involves convolution that is complex to compute, the following simplification is made: assume unit function 'I' is defined in the entire real domain (where it should have been in $[0, 1]$), then the Fourier transform of $S_i I$ results in $\hat{S}_i 2\pi\delta$, where \hat{S}_i is the symbol (the symbol will be introduced and the calculation will be performed in Subsection 3.1.4.) of operator S_i and δ is the Dirac delta function. As will be shown next, Dirac delta function δ could simplify the calculation to some extent. Meanwhile, it is worth noting that the integral calculation containing δ as shown below should not be interpreted as classic integral of conventional functions. Instead, the calculation is performed under the concept of generalised function (distribution).

$$\begin{aligned}
& \hat{e}_{1y}^{n+1}(A) + \frac{1}{2\pi} \mathcal{F}(S_2 I) * \hat{e}_1^{n+1}(A) = \hat{\gamma}_1^{n+1}(A), \\
& \rightarrow C_1(\omega) \omega \cosh(\omega A) + \int_{-\infty}^{\infty} \hat{S}_2(\tau) \delta(\tau) C_1(\omega - \tau) \sinh((\omega - \tau)A) d\tau = \hat{\gamma}_1^{n+1}(A), \\
& \rightarrow C_1(\omega) (\omega \cosh(\omega A) + \hat{S}_2(0) \sinh(\omega A)) = \hat{\gamma}_1^{n+1}(A), \\
& \rightarrow C_1(\omega) = \frac{\hat{\gamma}_1^{n+1}(A)}{\omega \cosh(\omega A) + \hat{S}_2(0) \sinh(\omega A)}.
\end{aligned} \tag{3.42}$$

$$\begin{aligned}
& -\hat{e}_{2y}^{n+1}(A') + \frac{1}{2\pi} \mathcal{F}(S_1 I) * \hat{e}_2^{n+1}(A') = \hat{\gamma}_2^{n+1}(A'), \\
& \rightarrow -C_2(\omega) \omega \cosh(\omega A') + \int_{-\infty}^{\infty} \hat{S}_1(\tau) \delta(\tau) D_1(\omega - \tau) \sinh((\omega - \tau)A') d\tau = \hat{\gamma}_2^{n+1}(A'), \\
& \rightarrow C_2(\omega) (-\omega \cosh(\omega A') + \hat{S}_1(0) \sinh(\omega A')) = \hat{\gamma}_2^{n+1}(A'), \\
& \rightarrow C_2(\omega) = \frac{\hat{\gamma}_2^{n+1}(A')}{-\omega \cosh(\omega A') + \hat{S}_1(0) \sinh(\omega A')}.
\end{aligned} \tag{3.43}$$

Since the coordinate shifts by $y' = y - B$ in above calculation, the 'new' interface is defined as $A' = A - B$.

Having known the coefficients C_i , the solution to the 2 BVPs could be obtained. Note at this step the coordinate has been shifted back to y rather than y' in \hat{e}_2^{n+1} case:

$$\hat{e}_1^{n+1}(\omega, y) = \frac{\hat{\gamma}_1^{n+1}(A) \sinh(\omega y)}{\omega \cosh(\omega A) + \hat{S}_2(0) \sinh(\omega A)}, \tag{3.44}$$

$$\hat{e}_2^{n+1}(\omega, y) = \frac{\hat{\gamma}_2^{n+1}(A) \sinh(\omega(y - B))}{-\omega \cosh(\omega(A - B)) + \hat{S}_1(0) \sinh(\omega(A - B))}. \tag{3.45}$$

Particularly, on $y = A$,

$$\hat{e}_1^{n+1}(\omega, A) = \frac{\hat{\gamma}_1^{n+1}(A)}{\omega \coth(\omega A) + \hat{S}_2(0)}, \quad (3.46)$$

$$\hat{e}_2^{n+1}(\omega, A) = \frac{\hat{\gamma}_2^{n+1}(A)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0)}. \quad (3.47)$$

It is obvious that $\hat{e}_i^{n+1}(A)$ is a function of $\hat{\gamma}_i^{n+1}(A)$, ($i = 1, 2$). The convergence rate $\frac{\hat{e}_i^{n+1}}{\hat{e}_i^n}$ is equal to $\frac{\hat{\gamma}_i^{n+1}}{\hat{\gamma}_i^n}$. Next, consider $\hat{\gamma}_1^{n+1}, \hat{\gamma}_2^{n+1}$:

$$\begin{aligned} \hat{\gamma}_1^{n+1}(A) &= \hat{e}_{2y}^n(\omega, A) + \mathcal{F}(S_2 I \cdot e_2^n(\omega, A)), \\ \rightarrow \hat{\gamma}_1^{n+1}(A) &= \omega \coth(\omega(A - B)) \hat{e}_2^n(\omega, A) + \frac{1}{2\pi} \mathcal{F}(S_2 I) * \hat{e}_2^n(\omega, A), \\ \rightarrow \hat{\gamma}_1^{n+1}(A) &= \omega \coth(\omega(A - B)) \hat{e}_2^n(\omega, A) + \hat{S}_2(0) \hat{e}_2^n(\omega, A), \\ \rightarrow \hat{\gamma}_1^{n+1}(A) &= (\omega \coth(\omega(A - B)) + \hat{S}_2(0)) \frac{\hat{\gamma}_2^n(A)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0)}, \\ \rightarrow \hat{\gamma}_1^{n+1}(A) &= \frac{\omega \coth(\omega(A - B)) + \hat{S}_2(0)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0)} [-\hat{e}_{1y}^n(\omega, A) + \mathcal{F}(S_1 \cdot e_1^n(\omega, A))], \\ \rightarrow \hat{\gamma}_1^{n+1}(A) &= \frac{\omega \coth(\omega(A - B)) + \hat{S}_2(0)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0)} [-\omega \coth(\omega A) \hat{e}_1^n(\omega, A) + \hat{S}_1(0) \hat{e}_1^n(\omega, A)], \\ \rightarrow \hat{\gamma}_1^{n+1}(A) &= \frac{\omega \coth(\omega(A - B)) + \hat{S}_2(0)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0)} \frac{-\omega \coth(\omega A) + \hat{S}_1(0)}{\omega \coth(\omega A) + \hat{S}_2(0)} \hat{\gamma}_1^n(A). \end{aligned} \quad (3.48)$$

$$\begin{aligned} \hat{\gamma}_2^{n+1}(A) &= -\hat{e}_{1y}^n(\omega, A) + \mathcal{F}(S_1 I \cdot e_1^n(\omega, A)), \\ \rightarrow \hat{\gamma}_2^{n+1}(A) &= -\omega \coth(\omega A) \hat{e}_1^n(\omega, A) + \frac{1}{2\pi} \mathcal{F}(S_1 I) * \hat{e}_1^n(\omega, A), \\ \rightarrow \hat{\gamma}_2^{n+1}(A) &= -\omega \coth(\omega A) \hat{e}_1^n(\omega, A) + \hat{S}_1(0) \hat{e}_1^n(\omega, A), \\ \rightarrow \hat{\gamma}_2^{n+1}(A) &= (-\omega \coth(\omega A) + \hat{S}_1(0)) \frac{\hat{\gamma}_1^n(A)}{\omega \coth(\omega A) + \hat{S}_2(0)}, \\ \rightarrow \hat{\gamma}_2^{n+1}(A) &= \frac{-\omega \coth(\omega A) + \hat{S}_1(0)}{\omega \coth(\omega A) + \hat{S}_2(0)} [\hat{e}_{2y}^n(\omega, A) + \mathcal{F}(S_2 \cdot e_2^n(\omega, A))], \\ \rightarrow \hat{\gamma}_2^{n+1}(A) &= \frac{-\omega \coth(\omega A) + \hat{S}_1(0)}{\omega \coth(\omega A) + \hat{S}_2(0)} [\omega \coth(\omega(A - B)) \hat{e}_2^n(\omega, A) + \hat{S}_2(0) \hat{e}_2^n(\omega, A)], \\ \rightarrow \hat{\gamma}_2^{n+1}(A) &= \frac{-\omega \coth(\omega A) + \hat{S}_1(0)}{\omega \coth(\omega A) + \hat{S}_2(0)} \frac{\omega \coth(\omega(A - B)) + \hat{S}_2(0)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0)} \hat{\gamma}_2^n(A). \end{aligned} \quad (3.49)$$

The above transformation of $\hat{\gamma}_1(A)$ and $\hat{\gamma}_2(A)$ shows that

$$\hat{e}_i^{n+1} = \frac{\hat{S}_1(0) - \omega \coth(\omega A)}{\hat{S}_1(0) + \omega \coth(\omega(B - A))} \frac{\hat{S}_2(0) - \omega \coth(\omega(B - A))}{\hat{S}_2(0) + \omega \coth(\omega A)} \hat{e}_i^n.$$

Given $\hat{S}_1(0) = \frac{1}{A}$ and $\hat{S}_2(0) = \frac{1}{B-A}$, the convergence rate is:

$$\|\hat{e}_i^{n+1}\| = \left| \frac{\frac{1}{A} - \omega \coth(\omega A)}{\frac{1}{A} + \omega \coth(\omega(B-A))} \cdot \frac{\frac{1}{B-A} - \omega \coth(\omega(B-A))}{\frac{1}{B-A} + \omega \coth(\omega A)} \right| \|\hat{e}_i^n\|. \quad (3.50)$$

Lastly, perform Fourier transform with respect to y on the original problem with the homogeneous boundary conditions: $e_1^{n+1}(0, y) = 0$ and $e_1^{n+1}(1, y) = 0$. The two-dimensional problem is then reduced to the following one-dimensional BVPs regarding x :

$$\begin{aligned} \frac{\partial^2 \hat{e}_i^{n+1}}{\partial x^2} - \omega^2 \hat{e}_i^{n+1} &= 0, \\ \hat{e}_i^{n+1}(0) &= 0, \quad \hat{e}_i^{n+1}(1) = 0. \end{aligned}$$

The solution to above BVPs is

$$\hat{e}_i^{n+1}(x, \omega) = \sin(\omega x). \quad \omega = k\pi, k = 1, 2, \dots \quad (3.51)$$

The above calculation regulates $\omega := k\pi$, ($k = 1, 2, \dots$).

It should be emphasized again that because Laplace equation has variable-separation-form solution, Fourier transform could be independently applied twice with respect to x and y on the original Laplace equations, and the boundary conditions could also be applied separately to the one-dimensional BVPs derived from Fourier transform. In otherwise case, inverse Fourier transform is needed to restore the solution (with unknown functions) and all boundary conditions should be considered together to determine the specific solution to the given BVP.

In this case, two pairs of one-dimensional BVPs (with respect to x and y separately) are considered independently. The convergence rate is determined by the solution to the BVPs regarding y , see Eq.(3.50). Next, the BVPs regarding x regulates the value of ω apart from determining the solution w.r.t. x . Therefore, the convergence rate should be written in the same form as Eq.(3.50) but restricting ω to particular values:

$$\|\hat{e}_i^{n+1}\| = \left| \frac{\frac{1}{A} - k\pi \coth(k\pi A)}{\frac{1}{A} + k\pi \coth(k\pi(B-A))} \cdot \frac{\frac{1}{B-A} - k\pi \coth(k\pi(B-A))}{\frac{1}{B-A} + k\pi \coth(k\pi A)} \right| \|\hat{e}_i^n\|. \quad (3.52)$$

($k = 1, 2, \dots$)

3.1.4 Calculation of symbol of SP operators

The formulae to calculate the symbol $a(x, \omega)$ of a pseudo-differential operator T is as follows [103]:

$$a(x, \omega) := e^{-2\pi j x \omega} T(e^{2\pi j x \omega}). \quad (3.53)$$

In this case, the pseudo-operator (S_i) operates on y direction at interface $y = A$. First consider the symbol of S_1 . According to the definition of S_1 and its application on Poisson's equation

in Ω_1 , the following BVP for calculating $S_1(e^{2\pi jx\omega})$ could be formulated:

$$\begin{aligned} -\nabla^2 e_1^{n+1} &= 0, & \text{on } \Omega_1, \\ e_1^{n+1} &= 0, & \text{on } \partial\Omega_1/(y = A), \\ e_1^{n+1} &= e^{2\pi jx\omega}, & \text{on } y = A. \end{aligned}$$

To solve it, apply Fourier transform w.r.t. x on the BVP defined in Ω_1 to get a reduced one-dimensional BVP:

$$\begin{aligned} \frac{\partial^2 \hat{e}_1^{n+1}}{\partial y^2} - \omega^2 \hat{e}_1^{n+1} &= 0, \\ \hat{e}_1^{n+1}(0) &= 0, \quad \hat{e}_1^{n+1}(A) = 1. \end{aligned} \tag{3.54}$$

Note again, it is because the solution to the original problem has variable-separation form, the above 2 boundary conditions could be applied immediately to obtain the solution to the one-dimensional BVP, which is:

$$\hat{e}_1^{n+1}(\omega, y) = \frac{\sinh(\omega y)}{\sinh(\omega A)}.$$

Furthermore, apply inverse Fourier transform on \hat{e}_1^{n+1} to restore interface solution and take derivative. Note there is only one harmonic to consider so the integral vanishes:

$$e_1^{n+1}(x, y) = \int \hat{e}_1^{n+1}(\omega, y) e^{2\pi jx\omega} d\omega = \hat{e}_1^{n+1}(\omega, y) e^{2\pi jx\omega}.$$

$$S_1(e^{2\pi jx\omega}) := e_{1y}^{n+1}(x, A) = e^{2\pi jx\omega} \omega \coth(\omega A). \tag{3.55}$$

Therefore, the symbol (\hat{S}_1) of the operator is:

$$\hat{S}_1(\omega) := e^{-2\pi jx\omega} S_1(e^{2\pi jx\omega}) = \omega \coth(\omega A). \tag{3.56}$$

Next, consider the symbol of S_2 . The calculation is almost the same as that with \hat{S}_1 . Apply Fourier transform w.r.t. x on BVP2 defined in Ω_2 to get a reduced one-dimensional BVP:

$$\begin{aligned} \frac{\partial^2 \hat{e}_2^{n+1}}{\partial y^2} - \omega^2 \hat{e}_2^{n+1} &= 0, \\ \hat{e}_2^{n+1}(0) &= 0, \quad \hat{e}_2^{n+1}(A - B) = 1. \end{aligned} \tag{3.57}$$

The solution is:

$$\hat{e}_2^{n+1}(\omega, y) = \frac{\sinh(\omega(y - B))}{\sinh(\omega(A - B))}.$$

Furthermore, restore the interface solution,

$$e_2^{n+1}(x, y) = \int \hat{e}_2^{n+1}(\omega, y) e^{2\pi jx\omega} d\omega = \hat{e}_2^{n+1}(\omega, y) e^{2\pi jx\omega}.$$

$$S_2(e^{2\pi j x \omega}) := -e_{2y}^{n+1}(x, A) = e^{2\pi j x \omega} \omega \coth(\omega(B - A)). \quad (3.58)$$

Therefore, the symbol (\hat{S}_2) of the operator is

$$\hat{S}_2(\omega) := e^{-2\pi j x \omega} S_2(e^{2\pi j x \omega}) = \omega \coth(\omega(B - A)). \quad (3.59)$$

Naturally, all ω should be restricted: $\omega = k\pi$, $k = 1, 2, \dots$ because the boundary conditions are defined at $x = 0, 1$ in this case, see Eq.(3.51). However, in Subsection 3.1.3, there is $\hat{S}_i(0)$ showing up in the calculation and this is because the simplification made when dealing with $S_i(I)$. Recall it is assumed that the unit function I is defined in $(-\infty, \infty)$ rather than where it should have been: $[0, 1]$. In the latter case, there are boundaries on both sides of the domain in x directions and two homogeneous boundary conditions that restrict the value of $\omega := k\pi$, $k = 1, 2, \dots$. In other words, it is the existence of the boundaries at $x = 0$ and $x = 1$ that requires ω taking particular values. If such boundaries become too far to exist anymore, there would be no homogeneous boundary conditions at $x = 0$ and $x = 1$ too, hence the restriction on ω disappears. As a matter of fact, in above calculation, the domain length is set 1 in x -direction, so the separation constant is $\lambda = k^2\pi^2$. If the length is assumed L , the constant λ would be $\lambda = (k\pi/L)^2$. The bigger L , the smaller interval between neighbour $\omega := k\pi/L$. In the case that L is infinity, the interval tends to 0 and ω becomes continuous. This is the scenario that happens when I is defined in $(-\infty, \infty)$, see Eq.(3.50). In addition, it could also be seen from the definition: $\mathcal{F}[S_i I] := \hat{S}_i(\omega) \mathcal{F}I$. The unit function (defined in entire real domain) is composed of only 1 harmonic ($\omega = 0$), so only $\hat{S}_i(0)$ exists. That is the consequent of assuming I is defined in real domain and how $\hat{S}_i(0)$ arises in the calculation.

To sum up, $\hat{S}_i(0)$ is the result of the simplification that I is assumed to be defined throughout the entire real domain, but it does not violate the principle of the convergence rate calculation. The calculation result based on the simplification is identical to the result that is based on the simplification of $S_i I$ being a constant. The effect of the simplification will be discussed in Chapter 4.

Compare $\hat{S}_i(\omega)$ with the general form of the symbol $a(x, \omega)$, it is seen that \hat{S}_i is not a function of x but only ω . This is still because the solution could be separated in terms of x and y . The separation gives rise to the solution to the reduced one-dimensional BVP w.r.t. y directly, as stated above, see Eq.(3.54), and the solution is free from containing any x . In otherwise case, the solution to the one-dimensional BVP could not be determined until all boundary conditions are considered together, in which case inverse Fourier transform is applied to restore the solution and x is brought in.

3.2 Convergence rate calculation of DR algorithm

This section performs the convergence rate calculation of DR algorithm applied to Poisson's equation too. The separation of variables is used again. Since the calculation is highly similar

to previous calculation, some intermediate process would not be repeatedly elaborated.

Consider the following BVPs:

$$\begin{aligned}
-\nabla^2 e_1^{n+1} &= 0, & \text{on } \Omega_1, \\
e_1^{n+1} &= 0, & \text{on } \partial\Omega_1/\Gamma, \\
e_1^{n+1} &= \theta e_2^{n+1} + (1 - \theta)e_1^n, & \text{on } \Gamma, \\
-\nabla^2 e_2^{n+1} &= 0, & \text{on } \Omega_2, \\
e_2^{n+1} &= 0, & \text{on } \partial\Omega_2/\Gamma, \\
e_{2y}^{n+1} &= e_{1y}^n + S_1^* I(e_2^{n+1} - e_1^n), & \text{on } \Gamma,
\end{aligned}$$

where $0 < \theta < 1$.

As has been mentioned earlier, the solution has variable-separation form: $e_i(x, y) = X_i(x)Y_i(y)$. The BVPs regarding x are the same as those defined in RR-SP0, see Eqs.(3.9) and (3.11). Here the solution is directly given: $X_i(x) = \sin(k\pi x)$, $k = 1, 2, \dots$

Next, consider $Y_i(y)$, which is the Fourier coefficient of the solution to the Laplace equation. Denote the Fourier coefficient by $\hat{e}_i(k, y) := Y_i(k, y)$. The general solution to $Y_i(y)$ is directly given:

$$\hat{e}_i^{n+1}(y) := Y_i^{n+1}(y) = A_i^{n+1} \sinh(k\pi y) + B_i^{n+1} \cosh(k\pi y).$$

To determine A_i^{n+1} and B_i^{n+1} , apply the rest 2 boundary conditions (at $y = 0$ and $y = A$) to the general solution:

$$\text{BVP1} : \begin{cases} \hat{e}_1^{n+1}(x, 0) = 0, \\ \hat{e}_{1|\Gamma}^{n+1} = \theta \hat{e}_{2|\Gamma}^{n+1} + (1 - \theta) \hat{e}_{1|\Gamma}^n. \end{cases} \quad (3.60)$$

$$\text{BVP2} : \begin{cases} \hat{e}_2^{n+1}(x, 0) = 0, \\ \hat{e}_{2y|\Gamma}^{n+1} = \hat{e}_{1y|\Gamma}^n + S_1^* I(\hat{e}_{2|\Gamma}^{n+1} - \hat{e}_{1|\Gamma}^n). \end{cases} \quad (3.61)$$

Note there is a variable substitute $\bar{y} = y - B$ when considering \hat{e}_2 . The final solution would be restored.

The two homogeneous boundary conditions enables the elimination of $\sinh(k\pi y)$ term in the general solution, so only $\cosh(k\pi y)$ term retains: $\hat{e}_i^{n+1} = A_i^{n+1}(k) \cosh(k\pi y)$. It is clear that the convergence rate could be obtained by calculating the ratio between $A_i^{n+1}(k)$ and $A_i^n(k)$. Next, transform the IBC as shown in Eq.(3.60):

$$\begin{aligned}
A_1^{n+1} \sinh(k\pi A) &= \theta A_2^{n+1} \sinh(k\pi(A - B)) + (1 - \theta) A_1^n \sinh(k\pi A), \\
\rightarrow A_1^{n+1} &= \theta \frac{\sinh(k\pi(A - B))}{\sinh(k\pi A)} A_2^{n+1} + (1 - \theta) A_1^n, \\
\rightarrow A_2^{n+1} &= \frac{\sinh(k\pi A)}{\theta \sinh(k\pi(A - B))} (A_1^{n+1} + (\theta - 1) A_1^n).
\end{aligned} \quad (3.62)$$

Transform the IBC as shown in Eq.(3.61):

$$\begin{aligned}
& k\pi \cosh(k\pi(A - B))A_2^{n+1} \\
& = k\pi \cosh(k\pi A)A_1^n + S_1^* I(\sinh(k\pi(A - B))A_2^{n+1} - \sinh(k\pi A)A_1^n), \\
\rightarrow A_2^{n+1} & = \frac{k\pi \cosh(k\pi A) - S_1^* I \sinh(k\pi A)}{k\pi \cosh(k\pi(A - B)) - S_1^* I \sinh(k\pi(A - B))} A_1^n.
\end{aligned} \tag{3.63}$$

Combining Eqs.(3.62) and (3.63), it could be obtained that

$$A_1^{n+1} = (1 - \theta\Omega)A_1^n, \tag{3.64}$$

where

$$\Omega := \frac{k\pi \coth(k\pi A) + k\pi \coth(k\pi(B - A))}{1/A + k\pi \coth(k\pi(B - A))}. \tag{3.65}$$

From Eq.(3.63):

$$\begin{aligned}
\frac{A_2^{n+1}}{A_1^n} & = \frac{A_2^{n+2}}{A_1^{n+1}}, \\
\rightarrow A_2^{n+1} & = (1 - \theta\Omega)A_2^n \quad (\text{from Eq.(3.64)}).
\end{aligned}$$

Therefore, the convergence rate is obtained:

$$\|\hat{e}_i^{n+1}(k, A)\| = |1 - \theta\Omega| \|\hat{e}_i^n(k, A)\|, \tag{3.66}$$

where Ω is given in Eq.(3.65).

The convergence rate is plotted in Fig.3.1 with various A .

3.3 Generalised convergence rate calculation of RR-SP0

This section extends the convergence rate calculation from Poisson's equation to those equations that could be solved with separation of variables. It would be demonstrated that the calculation process resembles that of Poisson's equation so the calculation about Poisson's equation is representative. The convergence rate is determined by the symbols/eigenvalues of SP operators applied to given equations. Due to separation of variables, either symbols or eigenvalues of SP operators are not difficult to obtain, after which the convergence rate of RR-SP0 algorithm could be obtained directly for the given problem. The convergence analysis result obtained above applies to BVPs whose governing equation is interface-boundary-type as well, which is the case for the model equation at vicinity of the wall in Chapter 5. Note the following calculation is still performed using Fourier transform with the assumption that unit function I is defined in $(-\infty, \infty)$.

Consider a BVP that could be solved with separation of variables defined in a square domain

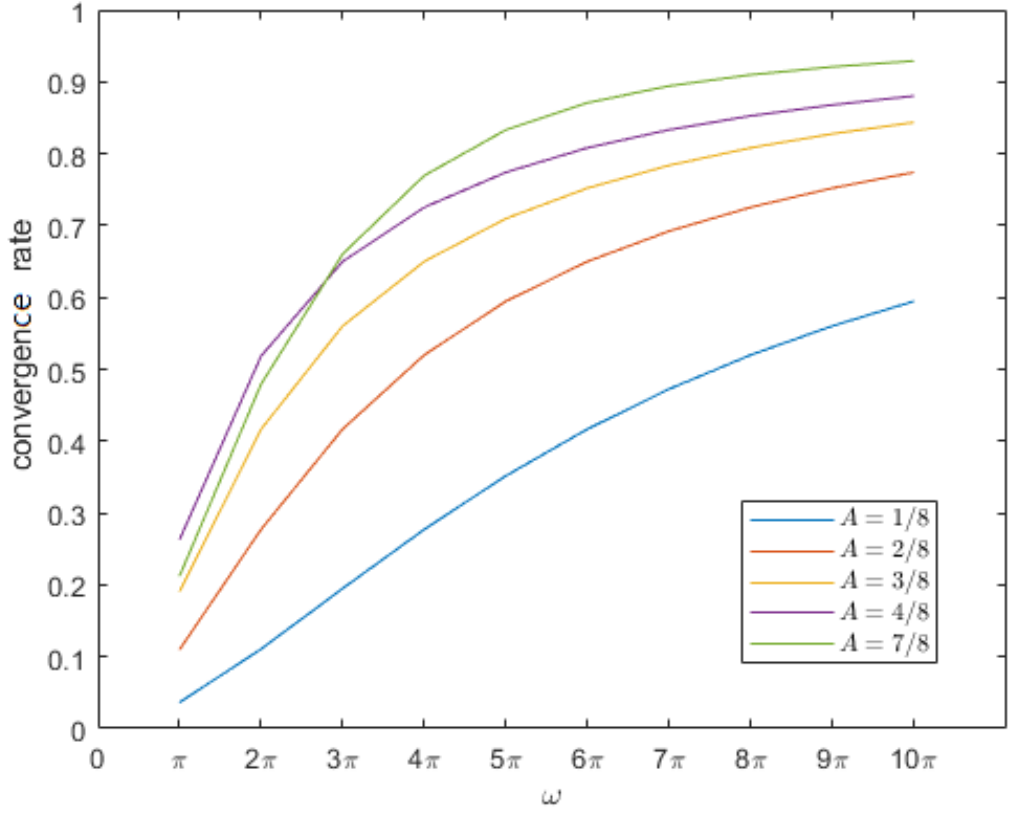


Figure 3.1. Convergence rate of DR algorithm with various A , $\theta = 1$, $B = 1$.

$\Omega : \{1 < x < 1, 0 < y < B\}$. The corresponding error BVPs for this problem are

$$\begin{aligned}
 \mathcal{L}e_1^{n+1} &= 0, & \text{on } \Omega_1, \\
 e_1^{n+1} &= 0, & \text{on } \partial\Omega_1/\Gamma, \\
 e_{1y}^{n+1} + S_2^* I \cdot e_1^{n+1} &= \gamma_1^{n+1}, & \text{on } \Gamma, \\
 \mathcal{L}e_2^{n+1} &= 0, & \text{on } \Omega_2, \\
 e_2^{n+1} &= 0, & \text{on } \partial\Omega_2/\Gamma, \\
 -e_{2y}^{n+1} + S_1^* I \cdot e_2^{n+1} &= \gamma_2^{n+1}, & \text{on } \Gamma.
 \end{aligned}$$

where γ_1^{n+1} and γ_2^{n+1} share the same definition as previous section, see Eqs.(3.1), (3.2).

Apply Fourier transform with respect to x to the above BVPs. The transform has been performed on Laplace equations in previous section, and here on unknown governing equations:

BVP1:

$$\begin{aligned}
 \mathcal{F}(\mathcal{L}e_1^{n+1}) &= 0, \\
 \hat{e}_1^{n+1}(0) &= 0, \\
 \hat{e}_{1y}^{n+1}(A) + \frac{1}{2\pi} \mathcal{F}(S_2 I) * \hat{e}_1^{n+1}(A) &= \hat{\gamma}_1^{n+1}(A).
 \end{aligned}$$

BVP2:

$$\begin{aligned}\mathcal{F}(\mathcal{L}e_2^{n+1}) &= 0, \\ \hat{e}_2^{n+1}(0) &= 0, \\ -\hat{e}_{2y}^{n+1}(A - B) + \frac{1}{2\pi}\mathcal{F}(S_1 I) * \hat{e}_2^{n+1}(A - B) &= \hat{\gamma}_2^{n+1}(A),\end{aligned}$$

where $\hat{\gamma}_1^{n+1}(A)$ and $\hat{\gamma}_2^{n+1}(A)$ are defined in Eqs.(3.40) and (3.41).

Because the operator \mathcal{L} is not known, it is not possible to solve the above BVPs. However, one could assume the general form of the solution as follows and apply the general solution to the IBCs:

$$\begin{aligned}\hat{e}_1^n(\omega, y) &= C_1^n(\omega)\xi_1(\omega, y), \\ \hat{e}_2^n(\omega, y) &= C_2^n(\omega)\xi_2(\omega, y).\end{aligned}$$

Particularly, in Poisson's equation case it is known that

$$\begin{aligned}C_1^n(\omega)\xi_1(\omega, y) &= c_1^n(\omega)\sinh(\omega y) + d_1^n(\omega)\cosh(\omega y), \\ C_2^n(\omega)\xi_2(\omega, y) &= c_2^n(\omega)\sinh(\omega(y - B)) + d_2^n(\omega)\cosh(\omega(y - B)).\end{aligned}$$

The selection of $\xi_i(\omega, y)$ is not unique. For instance, in Poisson's equation case, $\xi_1 := \cosh(\omega, y)$ and $C_1^n = c_1^n(\omega)\tanh(\omega y) + d_1^n(\omega)$, and alternatively ξ_1 could be $\sinh(\omega, y)$ as well.

In the following general calculation, the expression of ξ_i and C_i^n remains unknown.

Substitute the general solution to the interface boundary conditions:

$$\begin{aligned}\hat{e}_{1y}^{n+1}(A) + \frac{1}{2\pi}\mathcal{F}(S_2 I) * \hat{e}_1^{n+1}(A) &= \hat{\gamma}_1^{n+1}(A), \\ \rightarrow C_1^{n+1}(\omega)\xi_{1y}(\omega, A) + \hat{S}_2(0)C_1^{n+1}(\omega)\xi_1(\omega, A) &= \hat{\gamma}_1^{n+1}(A), \\ \rightarrow C_1^{n+1}(\omega) &= \frac{\hat{\gamma}_1^{n+1}(A)}{\xi_{1y}(\omega, A) + \hat{S}_2(0)\xi_1(\omega, A)}. \\ \\ -\hat{e}_{2y}^{n+1}(A) + \frac{1}{2\pi}\mathcal{F}(S_1 I) * \hat{e}_2^{n+1}(A) &= \hat{\gamma}_2^{n+1}(A), \\ \rightarrow -C_2^{n+1}(\omega)\xi_{2y}(\omega, A) + \hat{S}_1(0)C_2^{n+1}(\omega)\xi_2(\omega, A) &= \hat{\gamma}_2^{n+1}(A), \\ \rightarrow C_2^{n+1}(\omega) &= \frac{\hat{\gamma}_2^{n+1}(A)}{-\xi_{2y}(\omega, A) + \hat{S}_1(0)\xi_2(\omega, A)}.\end{aligned}$$

The solution to the problem on $y = A$ is $\hat{e}_i^n(\omega, A) = C_i^n(\omega)\xi_i(\omega, A)$:

$$\hat{e}_1^{n+1}(\omega, A) = \frac{\hat{\gamma}_1^{n+1}(\omega, A)}{\kappa_1(\omega) + \hat{S}_2(0)}, \quad (3.67)$$

$$\hat{e}_2^{n+1}(\omega, A) = \frac{\hat{\gamma}_2^{n+1}(\omega, A)}{-\kappa_2(\omega) + \hat{S}_1(0)}, \quad (3.68)$$

where it is defined that

$$\kappa_1(\omega) := \frac{\xi_{1y}(\omega, A)}{\xi_1(\omega, A)}, \quad (3.69)$$

$$\kappa_2(\omega) := \frac{\xi_{2y}(\omega, A)}{\xi_2(\omega, A)}. \quad (3.70)$$

Next, expand $\hat{\gamma}_1^{n+1}, \hat{\gamma}_2^{n+1}$ and transform:

$$\begin{aligned} \hat{e}_1^{n+1}(\omega, A) &= \frac{\hat{\gamma}_1^{n+1}(\omega, A)}{\kappa_1(\omega) + \hat{S}_2(0)}, \\ \rightarrow \hat{e}_1^{n+1}(\omega, A) &= \frac{\hat{e}_{2y}^{n+1} + \frac{1}{2\pi} \mathcal{F}(S_2 I) * \hat{e}_2^{n+1}}{\kappa_1(\omega) + \hat{S}_2(0)}, \\ \rightarrow \hat{e}_1^{n+1}(\omega, A) &= \frac{C_2^{n+1}(\omega) \kappa_2(\omega) \xi_2(\omega, A) + \hat{S}_2(0) C_2^{n+1}(\omega) \xi_2(\omega, A)}{\kappa_1(\omega) + \hat{S}_2(0)}, \\ \rightarrow \hat{e}_1^{n+1}(\omega, A) &= \hat{e}_2^{n+1}(\omega, A) \frac{\kappa_2(\omega) + \hat{S}_2(0)}{\kappa_1(\omega) + \hat{S}_2(0)}, \\ \rightarrow \hat{e}_1^{n+1}(\omega, A) &= \frac{\hat{\gamma}_2^{n+1}}{-\kappa_2(\omega) + \hat{S}_1(0)} \frac{\kappa_2(\omega) + \hat{S}_2(0)}{\kappa_1(\omega) + \hat{S}_2(0)}, \\ \rightarrow \hat{e}_1^{n+1}(\omega, A) &= \frac{-\hat{e}_{1y}^n + \frac{1}{2\pi} \mathcal{F}(S_1 I) * \hat{e}_1^n}{-\kappa_2(\omega) + \hat{S}_1(0)} \frac{\kappa_2(\omega) + \hat{S}_2(0)}{\kappa_1(\omega) + \hat{S}_2(0)}, \\ \rightarrow \hat{e}_1^{n+1}(\omega, A) &= \frac{-C_1^n(\omega) \kappa_1(\omega) \xi_1(\omega, A) + \hat{S}_1(0) C_1^n(\omega) \xi_1(\omega, A)}{-\kappa_2(\omega) + \hat{S}_1(0)} \frac{\kappa_2(\omega) + \hat{S}_2(0)}{\kappa_1(\omega) + \hat{S}_2(0)}, \\ \rightarrow \hat{e}_1^{n+1}(\omega, A) &= \frac{-\kappa_1(\omega) + \hat{S}_1(0)}{-\kappa_2(\omega) + \hat{S}_1(0)} \frac{\kappa_2(\omega) + \hat{S}_2(0)}{\kappa_1(\omega) + \hat{S}_2(0)} \hat{e}_1^n(\omega, A). \end{aligned} \quad (3.71)$$

$$\begin{aligned}
\hat{e}_2^{n+1}(\omega, A) &= \frac{\hat{\gamma}_2^{n+1}(\omega, A)}{-\kappa_2(\omega) + \hat{S}_1(0)}, \\
\rightarrow \hat{e}_2^{n+1}(\omega, A) &= \frac{-\hat{e}_{1y}^n + \hat{S}_1(0)\hat{e}_1^n}{-\kappa_2(\omega) + \hat{S}_1(0)}, \\
\rightarrow \hat{e}_2^{n+1}(\omega, A) &= \frac{-C_1^n(\omega)\kappa_1(\omega)\xi_1(\omega, A) + \hat{S}_1(0)C_1^n(\omega)\xi_1(\omega, A)}{-\kappa_2(\omega) + \hat{S}_1(0)}, \\
\rightarrow \hat{e}_2^{n+1}(\omega, A) &= \hat{e}_1^n(\omega, A) \frac{\kappa_1(\omega) + \hat{S}_1(0)}{-\kappa_2(\omega) + \hat{S}_1(0)}, \\
\rightarrow \hat{e}_2^{n+1}(\omega, A) &= \frac{\hat{\gamma}_1^n}{\kappa_1(\omega) + \hat{S}_2(0)} \frac{-\kappa_1(\omega) + \hat{S}_1(0)}{-\kappa_2(\omega) + \hat{S}_1(0)}, \\
\rightarrow \hat{e}_2^{n+1}(\omega, A) &= \frac{\hat{e}_{2y}^n + \hat{S}_2(0)\hat{e}_2^n}{\kappa_1(\omega) + \hat{S}_2(0)} \frac{-\kappa_1(\omega) + \hat{S}_1(0)}{-\kappa_2(\omega) + \hat{S}_1(0)}, \\
\rightarrow \hat{e}_2^{n+1}(\omega, A) &= \frac{C_2^n(\omega)\kappa_2(\omega)\xi_2(\omega, A) + \hat{S}_2(0)C_2^n(\omega)\xi_2(\omega, A)}{\kappa_1(\omega) + \hat{S}_2(0)} \frac{-\kappa_1(\omega) + \hat{S}_1(0)}{-\kappa_2(\omega) + \hat{S}_1(0)}, \\
\rightarrow \hat{e}_2^{n+1}(\omega, A) &= \frac{\kappa_2(\omega) + \hat{S}_2(0)}{\kappa_1(\omega) + \hat{S}_2(0)} \frac{-\kappa_1(\omega) + \hat{S}_1(0)}{-\kappa_2(\omega) + \hat{S}_1(0)} \hat{e}_2^n(\omega, A).
\end{aligned} \tag{3.72}$$

Therefore, the convergence rate is

$$\|\hat{e}_i^{n+1}(\omega, A)\| = \left| \frac{\kappa_2(\omega) + \hat{S}_2(0)}{\kappa_1(\omega) + \hat{S}_2(0)} \cdot \frac{-\kappa_1(\omega) + \hat{S}_1(0)}{-\kappa_2(\omega) + \hat{S}_1(0)} \right| \|\hat{e}_i^n(\omega, A)\|. \tag{3.73}$$

This is the generalised calculation of the convergence rate for equations that could be solved by separation of variables. Like previously stated, the selection of ω might be restricted and it could be determined by considering the Fourier transform with respect to x on the original equation and boundary conditions.

To calculate the convergence rate of a particular problem, it is important to know κ_i and $\hat{S}_i(0)$. From the definition of κ_i , it could be seen that κ_i is equal to the eigenvalue of SP operator applied to the given problem:

$$\begin{aligned}
\kappa_1(\omega) &:= \frac{\xi_{1y}(\omega, A)}{\xi_1(\omega, A)} = \frac{C_1^n \xi_{1y}(\omega, A)}{C_1^n \xi_1(\omega, A)} = \frac{\hat{e}_{1y}^n(\omega, A)}{\hat{e}_1^n(\omega, A)} = \lambda_1(\omega), \\
\kappa_2(\omega) &:= \frac{\xi_{2y}(\omega, A)}{\xi_2(\omega, A)} = \frac{C_2^n \xi_{2y}(\omega, A)}{C_2^n \xi_2(\omega, A)} = \frac{\hat{e}_{2y}^n(\omega, A)}{\hat{e}_2^n(\omega, A)} = \lambda_2(\omega),
\end{aligned}$$

where $\lambda_i(\omega)$ is the eigenvalue of S_i [84].

Recall the formula to calculate the symbol of pseudo-differential operator T is as Eq(3.53) shown

$$a(x, \omega) := e^{-2\pi j x \omega} T(e^{2\pi j x \omega}).$$

Substitute T with S_i and transform above equation:

$$\begin{aligned} S_i(e^{2\pi j x \omega}) &= \hat{S}_i(x, \omega) e^{2\pi j x \omega}, \\ \rightarrow S_i f(x) &= \int_{-\infty}^{\infty} \hat{S}_i(x, \omega) e^{2\pi j x \omega} \hat{f}(\omega) d\omega. \end{aligned}$$

Note $\hat{f}(\omega)$ is the Fourier transform of $f(x)$. The definition of eigenvalue of S_i applied to $f(x)$ gives:

$$\begin{aligned} \lambda_1(\omega) &:= \frac{\mathcal{F}(S_1 f(x))}{\mathcal{F}f(x)}, \\ &= \frac{\hat{S}_1(x, \omega) \hat{f}(\omega)}{\hat{f}(\omega)}, \\ &= \hat{S}_1(x, \omega). \end{aligned} \tag{3.74}$$

$$\begin{aligned} \lambda_2(\omega) &:= -\frac{\mathcal{F}(S_2 f(x))}{\mathcal{F}f(x)}, \\ &= -\frac{\hat{S}_2(x, \omega) \hat{f}(\omega)}{\hat{f}(\omega)}, \\ &= -\hat{S}_2(x, \omega). \end{aligned} \tag{3.75}$$

As demonstrated earlier, the symbol of pseudo-differential operator is a function of both wave number ω and x in general, but for BVPs that could be solved with separation of variables, the symbol is not related to x . Therefore, it is proved that the eigenvalue and the symbol of SP operator are equivalent:

$$\lambda_1(\omega) = \hat{S}_1(\omega), \quad \lambda_2(\omega) = -\hat{S}_2(\omega).$$

It shows that the convergence rate is purely determined by the symbols of SP operator. The generalised convergence rate Eq.(3.73) could be transformed to Eq.(3.76), see below, although restrictions on ω may vary from case to case. Particularly, in the case of Poisson's equation, the eigenvalues of SP operator is $\kappa_1(\omega) = \omega \coth(\omega A)$ and $\kappa_2(\omega) = \omega \coth(\omega(B - A))$. Substituting them into Eq.(3.73) yields to the same result as Eq.(3.19) or (3.50).

$$\|\hat{e}_1^{n+1}\| = \left| \frac{\hat{S}_1(0) - \hat{S}_1(\omega)}{\hat{S}_1(0) + \hat{S}_2(\omega)} \cdot \frac{\hat{S}_2(0) - \hat{S}_2(\omega)}{\hat{S}_2(0) + \hat{S}_1(\omega)} \right| \|\hat{e}_1^n\|. \tag{3.76}$$

The symbol calculation of SP operator is not expensive for BVPs that has variable-separation form solution because the calculation could be reduced to one-dimensional, as is demonstrated in Subsection 3.1.4. Then the convergence rate could be obtained. In this project, the problem in inner region usually represents boundary layer or laminar sublayer, which means the solution has limited dependence on x , so the symbol calculation could be almost reduced to one-dimensional too. Meanwhile, the solution that is weakly dependent on x could also be interpreted as having variable-separation form solution: the x -term solution

is constant. Therefore, the analysis result obtained in this section could be partially applied to the model equations of high-Reynolds-number flow with a boundary layer (as will be shown in Chapter 5), depending on how strong the solution is dependent on x .

3.4 Convergence rate calculation of RR-SP1 and RR-SP2

This section considers RR-SP1 and RR-SP2 applied to Poisson's equation and performs the calculation of the convergence rate by Fourier transform. Like in RR-SP0, some assumptions are made to help simplify the calculation. The assumptions are made in order: the calculation result of RR-SP0 is used in the calculation of RR-SP1, and the calculation result of RR-SP1 is also used in the calculation of RR-SP2. Each algorithm calculation uses the result obtained from lower-order algorithm and lays a foundation for the calculation of the higher-order algorithm.

In RR-SP0, it has been justified to consider $S_i I$ as a constant. So it will be used in the calculation of RR-SP1 directly. What is more, as introduced in Chapter 2, the implementation of RR-SP1 requires the knowledge of $S_i x$ in advance, so it is necessary to consider the Fourier transform to it: $\mathcal{F}[S_i x]$. The strict calculation is complex so the same simplification strategy as RR-SP0 is considered: assume function x is defined in $(-\infty, \infty)$ apart from I . The simplification is intended to remove any boundary effect incurred by boundary conditions in x direction, and provide analogous result to the real case where the boundary effect really exists. The calculation result shows that in ideal scenario (no boundary effect), RR-SP1 makes no difference to RR-SP0.

In RR-SP1, it is justified to consider $S_i x = S_i^* I \cdot x$, which will be used by RR-SP2 calculation directly. What is more, $S_i x^2$ is needed to implement the algorithm so $\mathcal{F}[S_i x^2]$ is considered with the assumption that x^2 is defined in $(-\infty, \infty)$. The simplification is aimed to remove the boundary effect and provide an analogous result too. The calculation result shows that RR-SP2 has similar convergence rate structure to RR-SP0(1) and only differs by an extra term, which makes the convergence different from RR-SP0(1).

Similar to previous analysis, the calculation result of RR-SP1 and RR-SP2 is beyond Poisson's equation and could be applied to the problems that have variable-separation form solution.

3.4.1 Calculation of RR-SP1

Consider applying RR-SP1 to solving a Poisson's equation. The corresponding error equation with boundary conditions could be formulated:

$$-\nabla^2 e_1^{n+1} = 0, \quad \text{on } \Omega_1, \quad (3.77)$$

$$e_1^{n+1} = 0, \quad \text{on } \partial\Omega_1/\Gamma,$$

$$e_{1y}^{n+1} + S_2 I \cdot e_1^{n+1} + e_{1x}^{n+1} \cdot T_{1l} = \gamma_1^{n+1}, \quad \text{on } \Gamma, \quad (3.78)$$

$$-\nabla^2 e_2^{n+1} = 0, \quad \text{on } \Omega_2, \quad (3.79)$$

$$\begin{aligned}
e_2^{n+1} &= 0, \quad \text{on } \partial\Omega_2/\Gamma, \\
-e_{2y}^{n+1} + S_1 I \cdot e_2^{n+1} + e_{2x}^{n+1} \cdot T_{2l} &= \gamma_2^{n+1}, \quad \text{on } \Gamma,
\end{aligned} \tag{3.80}$$

where

$$\begin{aligned}
\gamma_1^{n+1} &:= e_{2y}^{n+1}(x, A) + S_2 I \cdot e_2^{n+1}(x, A) + e_{2x}^{n+1} \cdot T_{1l}, \\
\gamma_2^{n+1} &:= -e_{1y}^n(x, A) + S_1 I \cdot e_1^n(x, A) + e_{1x}^n \cdot T_{2l}, \\
T_{1l} &:= S_2 x - S_2 I \cdot x, \\
T_{2l} &:= S_1 x - S_1 I \cdot x.
\end{aligned} \tag{3.81}$$

$$T_{2l} := S_1 x - S_1 I \cdot x. \tag{3.82}$$

Apply Fourier transform with respect to x on above BVPs, and it yields to the following two one-dimensional BVPs:

BVP1:

$$\begin{aligned}
\frac{\partial^2 \hat{e}_1^{n+1}}{\partial y^2} - \omega^2 \hat{e}_1^{n+1} &= 0, \\
\hat{e}_1^{n+1}(0) &= 0, \\
\hat{e}_{1y}^{n+1}(A) + \frac{1}{2\pi} \mathcal{F}[S_2 I] * \hat{e}_1^{n+1}(A) + \frac{j\omega}{2\pi} \hat{e}_1^{n+1} * \mathcal{F}[T_{1l}] &= \hat{\gamma}_1^{n+1}(A).
\end{aligned} \tag{3.83}$$

BVP2:

$$\begin{aligned}
\frac{\partial^2 \hat{e}_2^{n+1}}{\partial y^2} - \omega^2 \hat{e}_2^{n+1} &= 0, \\
\hat{e}_2^{n+1}(0) &= 0, \\
-\hat{e}_{2y}^{n+1}(A - B) + \frac{1}{2\pi} \mathcal{F}[S_1 I] * \hat{e}_2^{n+1}(A - B) + \frac{j\omega}{2\pi} \hat{e}_2^{n+1} * \mathcal{F}[T_{2l}] &= \hat{\gamma}_2^{n+1}(A).
\end{aligned} \tag{3.84}$$

Note in the formulation of BVP2, y -coordinate shifts by $y' = y - B$. The final result would be restored based on y later on.

The right-hand-side $\hat{\gamma}_1$ and $\hat{\gamma}_2$ are defined as

$$\begin{aligned}
\hat{\gamma}_1^{n+1}(A) &:= \hat{e}_{2y}^{n+1}(\omega, A) + \mathcal{F}[S_2 I \cdot e_2^{n+1}(\omega, A)] + \mathcal{F}[e_{2x}^{n+1} \cdot T_{1l}], \\
\hat{\gamma}_2^{n+1}(A) &:= -\hat{e}_{1y}^n(\omega, A) + \mathcal{F}[S_1 I \cdot e_1^n(\omega, A)] + \mathcal{F}[e_{1x}^n \cdot T_{2l}].
\end{aligned}$$

Recall the previous calculation of RR-SP0 by separation of variables assumes $S_i I$ is constant (denoted by $S_i^* I$), and the calculation result is equal to that by Fourier transform with the simplification that I is defined in $(-\infty, \infty)$. It shows that the two simplifications are equivalent. Therefore, assume $S_i I$ is constant here and the definition of γ_i and γ_2 could be rearranged as

$$\begin{aligned}
\hat{\gamma}_1^{n+1}(A) &:= \hat{e}_{2y}^{n+1}(\omega, A) + S_2^* I \cdot \mathcal{F}[e_2^{n+1}(\omega, A)] + \mathcal{F}[e_{2x}^{n+1} \cdot T_{1l}], \\
\hat{\gamma}_2^{n+1}(A) &:= -\hat{e}_{1y}^n(\omega, A) + S_1^* I \cdot \mathcal{F}[e_1^n(\omega, A)] + \mathcal{F}[e_{1x}^n \cdot T_{2l}].
\end{aligned}$$

In addition, it is known that the constant S_2^*I is equal to the symbol of S_i of 0-wave number:

$$S_i^*I = \hat{S}_i(0).$$

It could be verified by Poisson's equation, such as $S_1^*I = \hat{S}_1(0) = 1/A$ and $S_2^*I = \hat{S}_2(0) = 1/(B - A)$. Also, it could be proved:

$$\begin{aligned}\hat{S}_i(\omega) &:= e^{-2\pi jx\omega} S_i(e^{2\pi jx\omega}), \\ \rightarrow \hat{S}_i(0) &= S_i(1) = S_i^*(1) = S_i^*I.\end{aligned}$$

Now go back to the calculation of BVPs of \hat{e}_i , which have the general solution:

$$\hat{e}_1(\omega, y) = C_1(\omega) \sinh(\omega y) + D_1(\omega) \cosh(\omega y), \quad (3.85)$$

$$\hat{e}_2(\omega, y) = C_2(\omega) \sinh(\omega y) + D_2(\omega) \cosh(\omega y). \quad (3.86)$$

The homogeneous boundary condition at $y = 0$ and $y = B$ enables the elimination of D_1 and D_2 respectively. Then, apply the two interface boundary conditions to the general solution:

$$\hat{e}_{1y}^{n+1}(A) + \frac{1}{2\pi} \mathcal{F}[S_2I] * \hat{e}_1^{n+1}(A) + \frac{j\omega}{2\pi} \hat{e}_1^{n+1} * \mathcal{F}[T_{1l}] = \hat{\gamma}_1^{n+1}(A). \quad (3.87)$$

$$- \hat{e}_{2y}^{n+1}(A') + \frac{1}{2\pi} \mathcal{F}(S_1I) * \hat{e}_2^{n+1}(A') + \frac{j\omega}{2\pi} \hat{e}_2^{n+1} * \mathcal{F}[T_{2l}] = \hat{\gamma}_2^{n+1}(A'). \quad (3.88)$$

Since the coordinate shifts by $y' = y - B$ in Eq.(3.88), the 'new' interface is defined as $A' = A - B$.

The left-hand-side of Eq.(3.87) is composed of three terms:

1. $\hat{e}_{1y}^{n+1}(A)$,
2. $\frac{1}{2\pi} \mathcal{F}[S_2I] * \hat{e}_1^{n+1}(A)$,
3. $\frac{j\omega}{2\pi} \hat{e}_1^{n+1} * \mathcal{F}[T_{1l}]$.

The first 2 terms have been known from previous calculation of RR-SP0:

$$\begin{aligned}\hat{e}_{1y}^{n+1}(A) &= C_1(\omega) \omega \cosh(\omega A), \\ \frac{1}{2\pi} \mathcal{F}[S_2I] * \hat{e}_1^{n+1}(A) &= \hat{S}_2(0) C_1(\omega) \sinh(\omega A).\end{aligned}$$

Alternatively, having assumed S_2I is constant (\hat{S}_2^*I), the second term could be calculated directly:

$$\frac{1}{2\pi} \mathcal{F}[S_2I] * \hat{e}_1^{n+1}(A) = S_2^*I \cdot \hat{e}_1^{n+1}(A) = \hat{S}_2(0) C_1(\omega) \sinh(\omega A).$$

Next, consider the third term:

$$\begin{aligned}
& \frac{j\omega}{2\pi} \hat{e}_1^{n+1} * \mathcal{F}[T_{1l}] \\
&= \frac{j\omega}{2\pi} \hat{e}_1^{n+1} * \mathcal{F}[S_2 x - S_2^* I \cdot x], \\
&= \frac{j\omega}{2\pi} \int_{-\infty}^{\infty} C_1(\tau) \sinh(\tau A) [\hat{S}_2(\omega - \tau) 2\pi j \delta'(\omega - \tau) - \hat{S}_2(0) 2\pi j \delta'(\omega - \tau)] d\tau, \\
&= -\omega \int_{-\infty}^{\infty} C_1(\tau) \sinh(\tau A) [\hat{S}_2(\omega - \tau) \delta'(\omega - \tau) - \hat{S}_2(0) \delta'(\omega - \tau)] d\tau, \\
&= -\omega \int_{-\infty}^{\infty} C_1(\tau) \sinh(\tau A) (\hat{S}_2(\omega - \tau) - \hat{S}_2(0)) \delta'(\omega - \tau) d\tau, \\
&= \omega [C_1(\tau) \sinh(\tau A) ((\hat{S}_2(\omega - \tau) - \hat{S}_2(0)))']_{\tau=\omega}, \\
&= \omega C_1(\omega) \sinh(\omega A) \hat{S}_{2\omega}(0).
\end{aligned} \tag{3.89}$$

Note $\hat{S}_{2\omega}$ is the derivative of \hat{S}_2 with respect to ω .

The left-hand-side of Eq.(3.88) is composed of three terms:

1. $-\hat{e}_{2y}^{n+1}(A')$,
2. $\frac{1}{2\pi} \mathcal{F}[S_1 I] * \hat{e}_2^{n+1}(A')$,
3. $\frac{j\omega}{2\pi} \hat{e}_1^{n+1} * \mathcal{F}[T_{1l}]$.

The first 2 terms have been known from previous calculation of RR-SP0:

$$\begin{aligned}
& -\hat{e}_{2y}^{n+1}(A') = -C_2(\omega) \omega \cosh(\omega(A - B)), \\
& \frac{1}{2\pi} \mathcal{F}[S_1 I] * \hat{e}_2^{n+1}(A') = \hat{S}_1(0) C_2(\omega) \sinh(\omega(A - B)).
\end{aligned}$$

Next, consider the third term:

$$\begin{aligned}
& \frac{j\omega}{2\pi} \hat{e}_2^{n+1} * \mathcal{F}[T_{2l}] \\
&= \frac{j\omega}{2\pi} \hat{e}_2^{n+1} * \mathcal{F}[S_1 x - S_1^* I \cdot x], \\
&= \frac{j\omega}{2\pi} \int_{-\infty}^{\infty} C_2(\tau) \sinh(\tau A') [\hat{S}_1(\omega - \tau) 2\pi j \delta'(\omega - \tau) - \hat{S}_1(0) 2\pi j \delta'(\omega - \tau)] d\tau, \\
&= -\omega \int_{-\infty}^{\infty} C_2(\tau) \sinh(\tau A') [\hat{S}_1(\omega - \tau) \delta'(\omega - \tau) - \hat{S}_1(0) \delta'(\omega - \tau)] d\tau, \\
&= -\omega \int_{-\infty}^{\infty} C_2(\tau) \sinh(\tau A') (\hat{S}_1(\omega - \tau) - \hat{S}_1(0)) \delta'(\omega - \tau) d\tau, \\
&= \omega [C_2(\tau) \sinh(\tau A') ((\hat{S}_1(\omega - \tau) - \hat{S}_1(0)))']_{\tau=\omega}, \\
&= \omega C_2(\omega) \sinh(\omega(A - B)) \hat{S}_{1\omega}(0).
\end{aligned} \tag{3.90}$$

Note $\hat{S}_{1\omega}$ is the derivative of \hat{S}_1 with respect to ω .

Therefore, re-arranging Eq.(3.87) and (3.88) yields to:

$$C_1(\omega) = \frac{\hat{\gamma}_1^{n+1}(A)}{\omega \cosh(\omega A) + \hat{S}_2(0) \sinh(\omega A) + \sinh(\omega A) \omega \hat{S}_{2\omega}(0)},$$

$$C_2(\omega) = \frac{\hat{\gamma}_2^{n+1}(A')}{-\omega \cosh(\omega A') + \hat{S}_1(0) \sinh(\omega A') + \sinh(\omega A') \omega \hat{S}_{1\omega}(0)}.$$

The solution to the BVPs in both regions could be obtained:

$$\hat{e}_1^{n+1}(\omega, y) = \frac{\hat{\gamma}_1^{n+1}(A) \sinh(\omega y)}{\omega \cosh(\omega A) + \hat{S}_2(0) \sinh(\omega A) + \sinh(\omega A) \omega \hat{S}_{2\omega}(0)},$$

$$\hat{e}_2^{n+1}(\omega, y) = \frac{\hat{\gamma}_2^{n+1}(A) \sinh(\omega(y - B))}{-\omega \cosh(\omega(A - B)) + \hat{S}_1(0) \sinh(\omega(A - B)) + \sinh(\omega(A - B)) \omega \hat{S}_{1\omega}(0)}.$$

Particularly, at $y = A$, the solution is:

$$\hat{e}_1^{n+1}(\omega, A) = \frac{\hat{\gamma}_1^{n+1}(A)}{\omega \coth(\omega A) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0)}, \quad (3.91)$$

$$\hat{e}_2^{n+1}(\omega, A) = \frac{\hat{\gamma}_2^{n+1}(A)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0)}. \quad (3.92)$$

Obviously, $\hat{e}_i^{n+1}(\omega, A)$ is the function of $\hat{\gamma}_i^{n+1}(A)$. So the transformation of $\hat{\gamma}_i$ represents the evolution of \hat{e}_i^{n+1} . The calculation resembles that of RR-SP0, see Eqs.(3.48) and (3.49), so some intermediate process that has been demonstrated previously is skipped.

$$\begin{aligned} \hat{\gamma}_1^{n+1}(A) &:= \hat{e}_{2y}^{n+1}(\omega, A) + \mathcal{F}(S_2 I \cdot e_2^{n+1}(\omega, A)) + \mathcal{F}[e_{2x}^{n+1}(\omega, A) \cdot T_{1l}], \\ &= \omega \coth(\omega(A - B)) \hat{e}_2^{n+1}(\omega, A) + \hat{S}_2(0) \hat{e}_2^{n+1}(\omega, A) + \omega \hat{e}_2^{n+1}(\omega, A) \hat{S}_{2\omega}(0), \\ &= [\omega \coth(\omega(A - B)) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0)] \frac{\hat{\gamma}_2^{n+1}(A)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0)}, \\ &= \frac{\omega \coth(\omega(A - B)) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0)} \hat{\gamma}_2^{n+1}(A), \\ &:= \Phi_{1l}(\omega) \hat{\gamma}_2^{n+1}(A), \\ &= \Phi_{1l}(\omega) [-\hat{e}_{1y}^n(\omega, A) + S_1^* I \cdot \mathcal{F}[e_1^n(\omega, A)] + \mathcal{F}[e_{1x}^n \cdot T_{2l}]], \\ &= \Phi_{1l}(\omega) [-\omega \coth(\omega A) \hat{e}_1^n(\omega, A) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0)] \hat{e}_1^n(\omega, A), \\ &= \Phi_{1l}(\omega) \frac{-\omega \coth(\omega A) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0)}{\omega \coth(\omega A) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0)} \hat{\gamma}_1^n(A). \end{aligned} \quad (3.93)$$

$$\begin{aligned}
\hat{\gamma}_2^{n+1}(A) &:= -\hat{e}_{1y}^n(\omega, A) + \mathcal{F}(S_1 I \cdot e_1^n(\omega, A)) + \mathcal{F}[e_{1x}^n(\omega, A) \cdot T_{2l}], \\
&= -\omega \coth(\omega A) \hat{e}_1^n(\omega, A) + \hat{S}_1(0) \hat{e}_1^n(\omega, A) + \omega \hat{S}_{1\omega}(0) \hat{e}_1^n(\omega, A), \\
&= (-\omega \coth(\omega A) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0)) \frac{\hat{\gamma}_1^n(A)}{\omega \coth(\omega A) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0)}, \\
&= \frac{-\omega \coth(\omega A) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0)}{\omega \coth(\omega A) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0)} \hat{\gamma}_1^n(A), \\
&:= \Phi_{2l}(\omega) \hat{\gamma}_1^n(A), \\
&= \Phi_{2l}(\omega) [\hat{e}_{2y}^n(\omega, A) + S_2^* I \cdot \mathcal{F}[e_2^n(\omega, A)] + \mathcal{F}[e_{2x}^n \cdot T_{1l}]], \\
&= \Phi_{2l}(\omega) [\omega \coth(\omega(A - B)) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0)] \hat{e}_2^n(\omega, A), \\
&= \Phi_{2l}(\omega) \frac{\omega \coth(\omega(A - B)) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0)} \hat{\gamma}_2^n(A).
\end{aligned} \tag{3.94}$$

It has been known that $\hat{S}_1(\omega) = \omega \coth(\omega A)$ and $\hat{S}_2(\omega) = \omega \coth(\omega(B - A))$, then the convergence rate is

$$\begin{aligned}
\|\hat{e}_i^{n+1}\| &= |\Phi_{1l}(\omega) \Phi_{2l}(\omega)| \|\hat{e}_i^n\|, \\
&= \left| \frac{-\hat{S}_2(\omega) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0)}{\hat{S}_2(\omega) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0)} \cdot \frac{-\hat{S}_1(\omega) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0)}{\hat{S}_1(\omega) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0)} \right| \|\hat{e}_i^n\|, \\
&= \left| \frac{\hat{S}_1(0) - \hat{S}_1(\omega) + \omega \hat{S}_{1\omega}(0)}{\hat{S}_1(0) + \hat{S}_2(\omega) + \omega \hat{S}_{1\omega}(0)} \cdot \frac{\hat{S}_2(0) - \hat{S}_2(\omega) + \omega \hat{S}_{2\omega}(0)}{\hat{S}_2(0) + \hat{S}_1(\omega) + \omega \hat{S}_{2\omega}(0)} \right| \|\hat{e}_i^n\|.
\end{aligned} \tag{3.95}$$

Note like in previous case of RR-SP0, there is a restriction on the value of $\omega := k\pi$, ($k = 1, 2, \dots$)

The convergence rate of RR-SP1 differs from that of RR-SP0 only by an extra term at the end of numerator and denominator separately: $\omega \hat{S}_{i\omega}$. Next Consider the value of $\hat{S}_{i\omega}(\omega)$:

$$\begin{aligned}
\hat{S}_{1\omega} &= \coth(A\omega) + A\omega(1 - \coth^2(A\omega)), \\
\hat{S}_{2\omega} &= \coth((B - A)\omega) + (B - A)\omega(1 - \coth^2((B - A)\omega)).
\end{aligned}$$

It could be calculated that $\hat{S}_{i\omega}(0) = 0$. It means the extra term is 0 and the convergence rate of RR-SP1 is identical to that of RR-SP0. Further more, comparing the algorithms of RR-SP0 and RR-SP1, to let them be equal, T_{1l} and T_{2l} must be equal to 0:

$$T_{1l} := S_2 x - x S_2 I = 0, \tag{3.96}$$

$$T_{2l} := S_1 x - x S_1 I = 0. \tag{3.97}$$

Together with the assumption in RR-SP0 that $S_i I$ is constant, the above equation yields to

$$S_2 x = S_2^* I \cdot x, \tag{3.98}$$

$$S_1 x = S_1^* I \cdot x. \tag{3.99}$$

The above two equations are the result of the assumption that I and x are defined in $(-\infty, \infty)$, which will be used in the convergence rate calculation of RR-SP2.

3.4.2 Calculation of RR-SP2

Consider applying RR-SP2 to solving a Poisson's equation. The corresponding error equation with boundary conditions could be formulated:

$$\begin{aligned}
-\nabla^2 e_1^{n+1} &= 0, & \text{on } \Omega_1, \\
e_1^{n+1} &= 0, & \text{on } \partial\Omega_1/\Gamma, \\
e_{1y}^{n+1} + S_2 I \cdot e_1^{n+1} + e_{1x}^{n+1} \cdot T_{1l} + \frac{1}{2} e_{1xx}^{n+1} \cdot T_{1q} &= \gamma_1^{n+1}, & \text{on } \Gamma, \\
-\nabla^2 e_2^{n+1} &= 0, & \text{on } \Omega_2, \\
e_2^{n+1} &= 0, & \text{on } \partial\Omega_2/\Gamma, \\
-e_{2y}^{n+1} + S_1 I \cdot e_2^{n+1} + e_{2x}^{n+1} \cdot T_{2l} + \frac{1}{2} e_{2xx}^{n+1} \cdot T_{2q} &= \gamma_2^{n+1}, & \text{on } \Gamma,
\end{aligned}$$

where

$$\begin{aligned}
\gamma_1^{n+1} &:= e_{2y}^{n+1}(x, A) + S_2 I \cdot e_2^{n+1}(x, A) + e_{2x}^{n+1} \cdot T_{1l} + \frac{1}{2} e_{2xx}^{n+1} \cdot T_{1q}, \\
\gamma_2^{n+1} &:= -e_{1y}^n(x, A) + S_1 I \cdot e_1^n(x, A) + e_{1x}^n \cdot T_{2l} + \frac{1}{2} e_{1xx}^n \cdot T_{2q}, \\
T_{1l} &:= S_2 x - S_2 I \cdot x, \\
T_{2l} &:= S_1 x - S_1 I \cdot x, \\
T_{1q} &:= S_2 x^2 - 2x \cdot S_2 x + x^2 \cdot S_2 I, \\
T_{2q} &:= S_1 x^2 - 2x \cdot S_1 x + x^2 \cdot S_1 I.
\end{aligned}$$

The above BVP resembles that of RR-SP1 and the calculation would be the same for most part of the problem. To avoid repeated calculation, let us separate the overlapping part of the interface boundary conditions from the unique part and re-write the interface boundary conditions:

$$\begin{aligned}
A_{1l}^{n+1}(x, y) + \frac{1}{2} e_{1xx}^{n+1} \cdot T_{1q} &= \gamma_1^{n+1}, & \text{on } \Gamma, \\
A_{2l}^{n+1}(x, y) + \frac{1}{2} e_{2xx}^{n+1} \cdot T_{2q} &= \gamma_2^{n+1}, & \text{on } \Gamma, \\
\gamma_1^{n+1} &:= A_{1r}^{n+1}(x, y) + \frac{1}{2} e_{2xx}^n \cdot T_{1q}, \\
\gamma_2^{n+1} &:= A_{2r}^n(x, y) + \frac{1}{2} e_{1xx}^n \cdot T_{2q},
\end{aligned}$$

where

$$\begin{aligned}
A_{1l}^{n+1} &:= e_{1y}^{n+1} + S_2 I \cdot e_1^{n+1} + e_{1x}^{n+1} \cdot T_{1l}, \\
A_{2l}^{n+1} &:= -e_{2y}^{n+1} + S_1 I \cdot e_2^{n+1} + e_{2x}^{n+1} \cdot T_{2l},
\end{aligned}$$

$$A_{1r}^{n+1} := e_{2y}^{n+1}(x, A) + S_2 I \cdot e_2^{n+1}(x, A) + e_{2x}^{n+1} \cdot T_{1l},$$

$$A_{2r}^n := -e_{1y}^n(x, A) + S_1 I \cdot e_1^n(x, A) + e_{1x}^n \cdot T_{2l}.$$

The calculation of A_{il} and A_{ir} has been performed in RR-SP1 thus would be skipped in this subsection.

Apply Fourier transform with respect to x on above BVPs, it yields to the following two one-dimensional BVPs:

BVP1:

$$\frac{\partial^2 \hat{e}_1^{n+1}}{\partial y^2} - \omega^2 \hat{e}_1^{n+1} = 0, \quad (3.100)$$

$$\hat{e}_1^{n+1}(0) = 0, \quad (3.101)$$

$$\mathcal{F}[A_{1l}^{n+1}] - \frac{\omega^2}{4\pi} \hat{e}_1^{n+1} * \mathcal{F}[T_{1q}] = \hat{\gamma}_1^{n+1}(A). \quad (3.102)$$

BVP2:

$$\frac{\partial^2 \hat{e}_2^{n+1}}{\partial y^2} - \omega^2 \hat{e}_2^{n+1} = 0, \quad (3.103)$$

$$\hat{e}_2^{n+1}(0) = 0, \quad (3.104)$$

$$\mathcal{F}[A_{2l}^{n+1}] - \frac{\omega^2}{4\pi} \hat{e}_2^{n+1} * \mathcal{F}[T_{2q}] = \hat{\gamma}_2^{n+1}(A). \quad (3.105)$$

Note in the formulation of BVP2, y -coordinate shifts by $y' = y - B$. The final result would be restored based on y later on.

The right-hand-side $\hat{\gamma}_1$ and $\hat{\gamma}_2$ are defined as

$$\hat{\gamma}_1^{n+1}(A) := \mathcal{F}[A_{1r}^{n+1}] - \frac{\omega^2}{4\pi} \hat{e}_2^n * \mathcal{F}[T_{1q}],$$

$$\hat{\gamma}_2^{n+1}(A) := \mathcal{F}[A_{2r}^n] - \frac{\omega^2}{4\pi} \hat{e}_1^n * \mathcal{F}[T_{2q}].$$

The problem has the general solution shown as Eq.(3.85), and the homogeneous boundary conditions eliminate D_1 and D_2 . Next, apply the general solution to the two interface

boundary conditions. $\mathcal{F}[A_{il}]$ term does not need to calculate.

$$\begin{aligned}
& -\frac{\omega^2}{4\pi}\hat{e}_1^{n+1} * \mathcal{F}[T_{1q}] \\
& = -\frac{\omega^2}{4\pi}\hat{e}_1^{n+1} * \mathcal{F}[S_2x^2 - 2x \cdot S_2x + x^2 \cdot S_2I], \\
& = -\frac{\omega^2}{4\pi}\hat{e}_1^{n+1} * \mathcal{F}[S_2x^2 - x^2 \cdot S_2I], \quad (S_2x = S_2^*I \cdot x,) \\
& = -\frac{\omega^2}{4\pi}\hat{e}_1^{n+1} * [(\hat{S}_2(\omega) - \hat{S}_2(0))(-2\pi\delta^{(2)}(\omega))], \quad (S_2^*I = \hat{S}_2(0),) \\
& = \frac{1}{2} \int_{-\infty}^{\infty} \tau^2 C_1(\tau) \sinh(\tau A) (\hat{S}_2(\omega - \tau) - \hat{S}_2(0)) \delta^{(2)}(\omega - \tau) d\tau, \\
& = \frac{1}{2} [\tau^2 C_1(\tau) \sinh(\tau A) (\hat{S}_2(\omega - \tau) - \hat{S}_2(0))]''_{\tau=\omega}, \\
& = \frac{1}{2} \omega^2 C_1(\omega) \sinh(\omega A) \hat{S}_{2\omega\omega}(0) - [\omega^2 C_1(\omega) \sinh(\omega A)]' \hat{S}_{2\omega}(0), \\
& = \frac{1}{2} \omega^2 C_1(\omega) \sinh(\omega A) \hat{S}_{2\omega\omega}(0). \quad (\hat{S}_{2\omega}(0) = 0.)
\end{aligned} \tag{3.106}$$

$$\begin{aligned}
& -\frac{\omega^2}{4\pi}\hat{e}_2^{n+1} * \mathcal{F}[T_{2q}] \\
& = -\frac{\omega^2}{4\pi}\hat{e}_2^{n+1} * \mathcal{F}[S_1x^2 - 2x \cdot S_1x + x^2 \cdot S_1I], \\
& = -\frac{\omega^2}{4\pi}\hat{e}_2^{n+1} * \mathcal{F}[S_1x^2 - x^2 \cdot S_1I], \quad (S_1x = S_1^*I \cdot x,) \\
& = -\frac{\omega^2}{4\pi}\hat{e}_2^{n+1} * [(\hat{S}_1(\omega) - \hat{S}_1(0))(-2\pi\delta^{(2)}(\omega))], \quad (S_1^*I = \hat{S}_1(0),) \\
& = \frac{1}{2} \int_{-\infty}^{\infty} \tau^2 C_2(\tau) \sinh(\tau A') (\hat{S}_1(\omega - \tau) - \hat{S}_1(0)) \delta^{(2)}(\omega - \tau) d\tau, \quad (A' = A - B,) \\
& = \frac{1}{2} [\tau^2 C_2(\tau) \sinh(\tau A') (\hat{S}_1(\omega - \tau) - \hat{S}_1(0))]''_{\tau=\omega}, \\
& = \frac{1}{2} \omega^2 C_2(\omega) \sinh(\omega A') \hat{S}_{1\omega\omega}(0) - [\omega^2 C_2(\omega) \sinh(\omega A')] \hat{S}_{1\omega}(0), \\
& = \frac{1}{2} \omega^2 C_2(\omega) \sinh(\omega A') \hat{S}_{1\omega\omega}(0). \quad (\hat{S}_{1\omega}(0) = 0.)
\end{aligned} \tag{3.107}$$

Note $\hat{S}_{i\omega\omega}$ is the second-order derivative of \hat{S}_i with respect to ω .

Therefore, the coefficients C_1 and C_2 could be obtained:

$$\begin{aligned}
C_1(\omega) &= \frac{\hat{\gamma}_1^{n+1}(A)}{[\omega \coth(\omega A) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2 \hat{S}_{2\omega\omega}(0)] \sinh(\omega A)}, \\
C_2(\omega) &= \frac{\hat{\gamma}_2^{n+1}(A')}{[-\omega \coth(\omega A') + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0)] \sinh(\omega A')}.
\end{aligned}$$

The solution to the BVPs in both regions could be obtained:

$$\hat{e}_1^{n+1}(\omega, y) = \frac{\hat{\gamma}_1^{n+1}(A) \sinh(\omega y)}{[\omega \coth(\omega A) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2 \hat{S}_{2\omega\omega}(0)] \sinh(\omega A)},$$

$$\hat{e}_2^{n+1}(\omega, y) = \frac{\hat{\gamma}_2^{n+1}(A) \sinh(\omega(y - B))}{[-\omega \coth(\omega(A - B)) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0)] \sinh(\omega(A - B))}.$$

Particularly, at $y = A$, the solution is

$$\hat{e}_1^{n+1}(\omega, A) = \frac{\hat{\gamma}_1^{n+1}(A)}{\omega \coth(\omega A) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2 \hat{S}_{2\omega\omega}(0)}, \quad (3.108)$$

$$\hat{e}_2^{n+1}(\omega, A) = \frac{\hat{\gamma}_2^{n+1}(A)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0)}. \quad (3.109)$$

Clearly, $\hat{e}_i^{n+1}(\omega, A)$ is a function of $\hat{\gamma}_i^{n+1}(A)$. So the evolution of $\hat{\gamma}_i^{n+1}$ is equal to the evolution of \hat{e}_i^{n+1} :

$$\begin{aligned} \hat{\gamma}_1^{n+1}(A) &:= \mathcal{F}[A_{1r}^{n+1}] - \frac{\omega^2}{4\pi} \hat{e}_2^{n+1}(A) * \mathcal{F}[T_{1q}], \\ &= \mathcal{F}[A_{1r}^{n+1}] + \frac{1}{2}\omega^2 \hat{e}_2^{n+1}(A) \hat{S}_{2\omega\omega}(0), \quad (\text{analogous to Eq.(3.106),}) \\ &= \frac{\omega \coth(\omega(A - B)) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2 \hat{S}_{2\omega\omega}(0)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0)} \hat{\gamma}_2^{n+1}(A), \\ &= \Phi_{1q}(\omega) \hat{\gamma}_2^{n+1}(A), \quad (3.110) \\ &= \Phi_{1q}(\omega) \cdot [\mathcal{F}[A_{2r}^n] - \frac{\omega^2}{4\pi} \hat{e}_1^n * \mathcal{F}[T_{2q}]], \\ &= \Phi_{1q}(\omega) \cdot [\mathcal{F}[A_{2r}^n] + \frac{1}{2}\hat{e}_1^n \omega^2 \hat{S}_{1\omega\omega}(0)], \quad (\text{analogous to Eq.(3.107),}) \\ &= \Phi_{1q}(\omega) \cdot \frac{-\omega \coth(\omega A)(\omega, A) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0)}{\omega \coth(\omega A) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2 \hat{S}_{2\omega\omega}(0)} \hat{\gamma}_1^n. \end{aligned}$$

$$\begin{aligned} \hat{\gamma}_2^{n+1}(A) &:= \mathcal{F}[A_{2r}^n] - \frac{\omega^2}{4\pi} \hat{e}_1^n * \mathcal{F}[T_{2q}], \\ &= \mathcal{F}[A_{2r}^n] + \frac{1}{2}\hat{e}_1^n \omega^2 \hat{S}_{1\omega\omega}(0), \quad (\text{analogous to Eq.(3.107),}) \\ &= \frac{-\omega \coth(\omega A)(\omega, A) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0)}{\omega \coth(\omega A) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2 \hat{S}_{2\omega\omega}(0)} \hat{\gamma}_1^n, \\ &= \Phi_{2q}(\omega) \hat{\gamma}_1^n, \\ &= \Phi_{2q}(\omega) \cdot [\mathcal{F}[A_{1r}^n] - \frac{\omega^2}{4\pi} \hat{e}_2^n(A) * \mathcal{F}[T_{1q}]], \\ &= \Phi_{2q}(\omega) \cdot [\mathcal{F}[A_{1r}^n] + \frac{1}{2}\omega^2 \hat{e}_2^n(A) \hat{S}_{2\omega\omega}(0)], \quad (\text{analogous to Eq.(3.106),}) \\ &= \Phi_{2q}(\omega) \cdot \frac{\omega \coth(\omega(A - B)) + \hat{S}_2(0) + \omega \hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2 \hat{S}_{2\omega\omega}(0)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0) + \omega \hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0)} \hat{\gamma}_2^n(A). \end{aligned} \quad (3.111)$$

It has been known that $\hat{S}_1(\omega) = \omega \coth(\omega A)$ and $\hat{S}_2(\omega) = \omega \coth(\omega(B - A))$, then the

convergence rate is

$$\begin{aligned}
& \|\hat{e}_i^{n+1}\| \\
&= |\Phi_{1q}(\omega)\Phi_{2q}(\omega)| \|\hat{e}_i^n\|, \\
&= \left| \frac{\hat{S}_2(0) - \hat{S}_2(\omega) + \omega\hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)}{\hat{S}_2(\omega) + \hat{S}_1(0) + \omega\hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)} \cdot \frac{\hat{S}_1(0) - \hat{S}_1(\omega) + \omega\hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)}{\hat{S}_1(\omega) + \hat{S}_2(0) + \omega\hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)} \right| \|\hat{e}_i^n\| \\
&= \left| \frac{\hat{S}_1(0) - \hat{S}_1(\omega) + \omega\hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)}{\hat{S}_1(0) + \hat{S}_2(\omega) + \omega\hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)} \cdot \frac{\hat{S}_2(0) - \hat{S}_2(\omega) + \omega\hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)}{\hat{S}_2(0) + \hat{S}_1(\omega) + \omega\hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)} \right| \|\hat{e}_i^n\|
\end{aligned} \tag{3.112}$$

Note such as in previous case of RR-SP0 and RR-SP1, there is a restriction on the value of $\omega := k\pi$, ($k = 1, 2, \dots$)

Comparing with RR-SP1, the convergence rate differs by an extra term at end of the numerator and denominator respectively: $\frac{1}{2}\omega^2\hat{S}_{i\omega\omega}(0)$ The expression of $\hat{S}_{i\omega\omega}$ is

$$\begin{aligned}
\hat{S}_{1\omega\omega}(\omega) &= 2A^2\omega \coth(\omega A)(\coth^2(\omega A) - 1) + 2A(1 - \coth^2(\omega A)), \\
\hat{S}_{2\omega\omega}(\omega) &= 2(B - A)^2\omega \coth(\omega(B - A))(\coth^2(\omega(B - A)) - 1) + \\
&\quad 2(B - A)(1 - \coth^2(\omega(B - A))).
\end{aligned}$$

The value of $\hat{S}_{1\omega\omega}(0)$ is not 0 so the convergence of the algorithm differs from that of RR-SP0. The influence will be discussed in next Chapter 4.

3.5 Summary

This chapter performs the convergence rate calculation of RR-SP0, RR-SP1 and RR-SP2 applied to Poisson's equation via the methods of separation of variables and Fourier transform. Since the application of SP operators yields to two-dimensional problems that could be largely influenced by the boundary conditions with respect to x , the calculation result is actually subjected to the boundary conditions and the influence is called boundary effect. To simplify the calculation, the boundary effect is not considered in the chapter and the calculation method is so-called standard approach. The exclusion of the boundary effect enables $S_i I$ to be reduced to one-dimensional (constant) thus the calculation is largely simplified. The convergence rate of RR-SP0 (also apply to RR-SP1 and RR-SP2) between the parallel and the sequential versions is also compared and it shows that the sequential algorithm could converge twice as fast as the parallel version. Next, based on the calculation result of RR-SP0, the case of RR-SP1 is calculated with additional assumption on $S_i x$. The calculation of RR-SP2 (with additional assumption of $S_i x^2$) is performed on the foundation of RR-SP0 and RR-SP1 calculation results. The calculation result shows that RR-SP0 and RR-SP1 have identical convergence rate and RR-SP2 is lower than the both two. In addition, the calculation result is proved valid for any two-dimensional BVP that has variable-separation form solution.

Chapter 4

Convergence analysis of NDD algorithms

This chapter analyses the convergence property of RR algorithms in application to solving Poisson's equation with various A . The analysis is based on the convergence rate calculation performed in Chapter 3. Then, the influence of the boundary effect of the algorithms is discussed. The motivation of studying the boundary effect is to understand how different A affects the strength of the boundary effect and explain the difference between S_i^* and S_i cases in numerical test.

In Section 4.1, the convergence analysis about RR-SP0(1) is made. The analysis finds that the convergence rate is associated with the increment of SP operator symbols: $\Delta s_i(\omega) := \hat{S}_i(\omega) - \hat{S}_i(0)$, and the multiplication $\Delta s_1 \Delta s_2$ together determines the convergence of RR-SP0 with given A . In addition, the multiplication is symmetric in terms of A so the convergence of $A = 1/8$ is the same as $A = 7/8$. The theory is verified by a numerical test on Poisson's equation. At the end of the section, the parallel and sequential versions of the algorithm are compared.

In Section 4.2, the convergence analysis about RR-SP2 is made. Unlike RR-SP0, the analysis is not fully strict but with limited approximation. Two approximation methods are presented. The first approximation assumes $\hat{S}_{i\omega}(\omega)$ (that is the derivative of $\hat{S}_i(\omega)$) is linear before a turning point and goes flat after the point. As a result of the linearity of $\hat{S}_{i\omega}(\omega)$, its derivative $\hat{S}_{i\omega\omega}(\omega)$ is presumed to be constant when it is actually not. The second approximation applies the first presumption only and analyses $\hat{S}_{i\omega}(\omega)$ based on its real profile. In theory, the second approximation should be slightly more accurate but the two methods actually lead to the same result about the convergence of RR-SP2. According to the two approximations, the convergence of RR-SP2 is determined by: $\hat{S}_{1\omega}(0)\hat{S}_{2\omega}(0)$ and $\Delta s_{1\omega\omega}(\omega^{**})\Delta s_{2\omega\omega}(\omega^{**})$ separately. Note $\Delta s_{i\omega\omega}(\omega) := \hat{S}_{i\omega\omega}(\omega) - \hat{S}_{i\omega\omega}(0)$. In practice, the first approximation is easier to implement because it requires the knowledge of $\hat{S}_{i\omega}(0)$ only while the second requires to know $\hat{S}_{i\omega}(\omega)$ across a range of ω .

In Section 4.3, a conventional Robin-to-Robin algorithm is introduced and analysed. The algorithm is proposed and developed in [99], [101], [102] and it has similar structure to RR-SP0. The study of [102] shows that the convergence of the algorithm could be improved if a free parameter is optimised. As the representation of the conventional RR algorithm, the optimised algorithm is studied in terms of the theoretical convergence rate, and the same numerical test as done in [102] is conducted as well. Both theoretical and numerical study

prove that RR-SP0 has superior convergence property than the conventional algorithm.

Section 4.4 discusses the so-called boundary effect in RR-SP0 calculation. The boundary effect is the result of the existence of the boundaries on both x ends, and has been ignored in previous calculation and analysis. However, the numerical result shows that in some cases it could incur non-negligible error if $S_i I$ is considered as $S_i^* I$. The aim of the study is to understand by how much the boundary effect could affect the convergence and how it is associated with A . An exponential function model is adopted to represent the boundary effect and the convergence calculation is performed again with the boundary effect held accountable. To simplify the calculation, some estimation is made to maintain the main character of the boundary effect and the result shows that the strength of the boundary effect mainly depends on the profile of the 'pike function', which is the reflection/representation of the boundary effect in $S_i I$ profile. Graphically, if the function is high and wide, the boundary effect imposes strong boost on the convergence and otherwise not. Meanwhile, an empirical law is proposed to estimate the boundary effect strength in a quantitative manner, and a Poisson's equation test is implemented to verify the validity of the law. The test result is positive.

Section 4.5 and 4.6 performs the convergence rate calculation and analysis of RR-SP1 and RR-SP2 with considering the boundary effect in Poisson's equation. The calculation method is the same as in Section 4.4: find a 'pike functions' to represent the boundary effect in S_i profile and analyse how the pike functions affect the standard convergence. In the case of RR-SP0, the pike function is denoted by $P_i(x)$ and it is defined in $S_i I$ profile. In this section, similar pike functions are defined in $S_i x$ and $S_i x^2$ respectively. Unlike RR-SP0 case, the pike functions are not studied in fully quantitative manner because the model study as shown in Section 4.4 is not applied in the section. The study of the pike functions shows that the boundary effect helps reduce the standard convergence rate in both RR-SP1 and RR-SP2 algorithms: they both have positive effect on the convergence. In addition, considering the boundary effect, RR-SP1 is supposed to converge faster than RR-SP0, but RR-SP2 could perform very similar convergence to RR-SP1.

4.1 Convergence analysis of RR-SP0 without boundary effect

This section makes convergence analysis based on the convergence rate of RR-SP0 applied to Poisson's equation, which is obtained from Chapter 3. There are two versions of RR-SP0: sequential and parallel, and only the former is discussed. The parallel algorithm should resemble the sequential algorithm in every respect except taking twice number of iterations to converge. It is found that the convergence rate is determined by the increment of the two symbols of SP operators: $\Delta_{S_i}(\omega) := \hat{S}_i(\omega) - \hat{S}_i(0)$. The analysis shows that the algorithm has low convergence rate even if in high- ω phase. Meanwhile, the interface boundary location has influence on the convergence rate, and $A = 1/8$ (or $7/8$) is found optimal. To verify the analysis result, a numerical test implemented on Poisson's equation is conducted and the test result is found to coincide well with the analysis. In addition, the convergence analysis of DR could be found in [83] so no elaboration would be provided in the thesis.

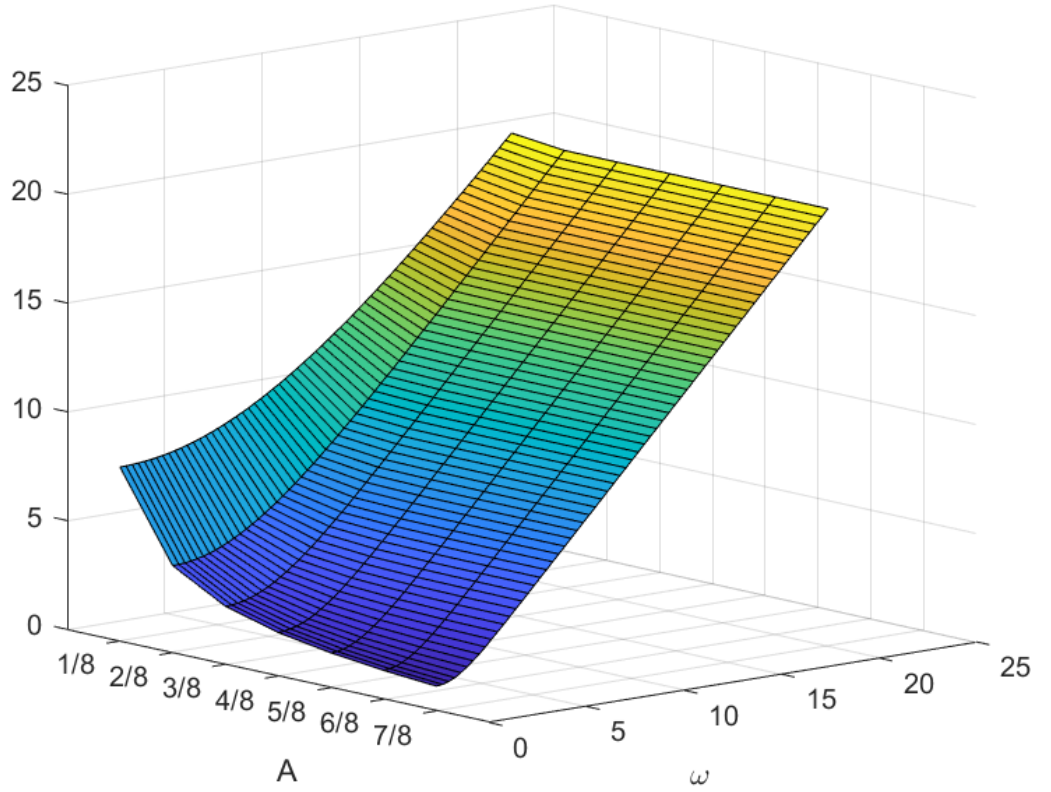


Figure 4.1. Profile of \hat{S}_1 applied to Poisson's equation. It is known that $\hat{S}_1(\omega) = \omega \coth(\omega A)$, and the profile is plotted against various ω and A .

4.1.1 Analysis of RR-SP0

In Chapter 3, it has been calculated that $\hat{S}_1(\omega) = \omega \coth(\omega A)$ and $\hat{S}_2(\omega) = \omega \coth(\omega(B-A))$. Particularly, $\hat{S}_1(0) = 1/A$ and $\hat{S}_2(0) = 1/(B-A)$. Therefore the convergence rate could also be written as

$$\|\hat{e}_1^{n+1}\| = \left| \frac{\hat{S}_1(0) - \hat{S}_1(\omega)}{\hat{S}_1(0) + \hat{S}_2(\omega)} \cdot \frac{\hat{S}_2(0) - \hat{S}_2(\omega)}{\hat{S}_2(0) + \hat{S}_1(\omega)} \right| \|\hat{e}_1^n\|, \quad (4.1)$$

$(\omega = k\pi, k = 1, 2, \dots).$

The symbol value $\hat{S}_i(\omega) := \omega \coth(\omega x)$ is always positive and increases with rising ω , see Fig.4.1 and 4.2, so the convergence rate is always less than 1, which means that there would be no divergence happening. Meanwhile, it is important to know which harmonic needs to be considered and which to ignore in terms of energy distribution, because it is not necessary to consider ω up to ∞ . In this project, the initial guess of the solution is always zero, so the initial error at interface is the solution to the problem itself. Assume the solution is a polynomial function of up to order three (at least the solution could be fitted by polynomial function of such kind) so that the polynomial function is composed of four basic functions: $1, x, x^2$ and x^3 , all defined in $(0 \leq x \leq 1)$. Next, consider the energy spectrum of each basic function.

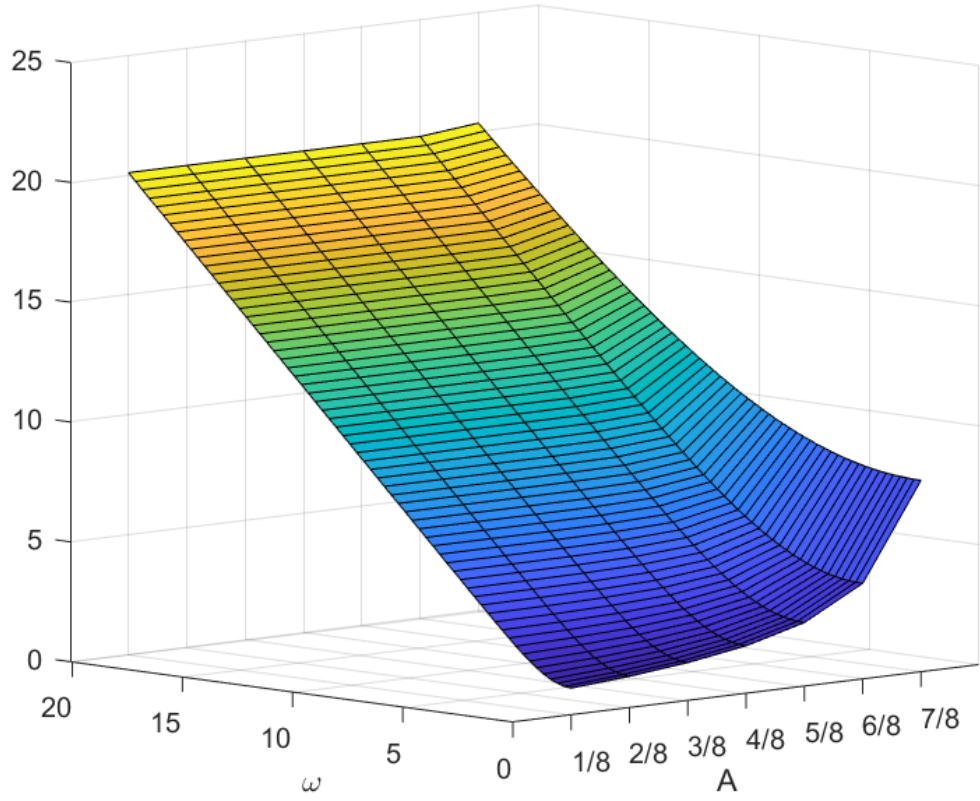


Figure 4.2. Profile of \hat{S}_2 applied to Poisson's equation. It is known that $\hat{S}_2(\omega) = \omega \coth(\omega(B - A))$, and the profile is plotted against various ω and A .

Take Fourier transform on each basic function:

$$\mathcal{F}(1) = \frac{\sin(\omega)}{\omega} + j \frac{\cos(\omega) - 1}{\omega}, \quad (4.2)$$

$$\mathcal{F}(x) = \frac{\sin(\omega)}{\omega} + \frac{\cos(\omega) - 1}{\omega^2} + j \left(\frac{\cos(\omega)}{\omega} - \frac{\sin(\omega)}{\omega^2} \right), \quad (4.3)$$

$$\mathcal{F}(x^2) = \frac{\sin(\omega)}{\omega} + \frac{2 \cos(\omega)}{\omega^2} - \frac{2 \sin(\omega)}{\omega^3} + j \left(\frac{\cos(\omega)}{\omega} - \frac{2 \sin(\omega)}{\omega^2} - \frac{2(\cos(\omega) - 1)}{\omega^3} \right), \quad (4.4)$$

$$\mathcal{F}(x^3) = \frac{\sin(\omega)}{\omega} + \frac{3 \cos(\omega)}{\omega^2} - \frac{6 \sin(\omega)}{\omega^3} - \frac{6(\cos(\omega) - 1)}{\omega^4} + j \left(\frac{\cos(\omega)}{\omega} - \frac{3 \sin(\omega)}{\omega^2} - \frac{6 \cos(\omega)}{\omega^3} + \frac{6 \sin(\omega)}{\omega^4} \right). \quad (4.5)$$

Note j is the imaginary unit. The energy spectrum could be calculated by $\sqrt{a^2 + b^2}$, given the Fourier coefficient is expressed in the form of $a + bj$. The energy spectrum of each basic function is illustrated in Fig.4.3. It can be seen that energy is mostly distributed in low- ω zone. Therefore, for each basic function, it is sufficient to consider only those low-wave-number harmonics, which roughly dominate the convergence nature of the polynomial functions. The standard of 'low' or 'high' wave number may vary, depending on how much energy is neglected. In this project, wave number larger than 6π is cut off.

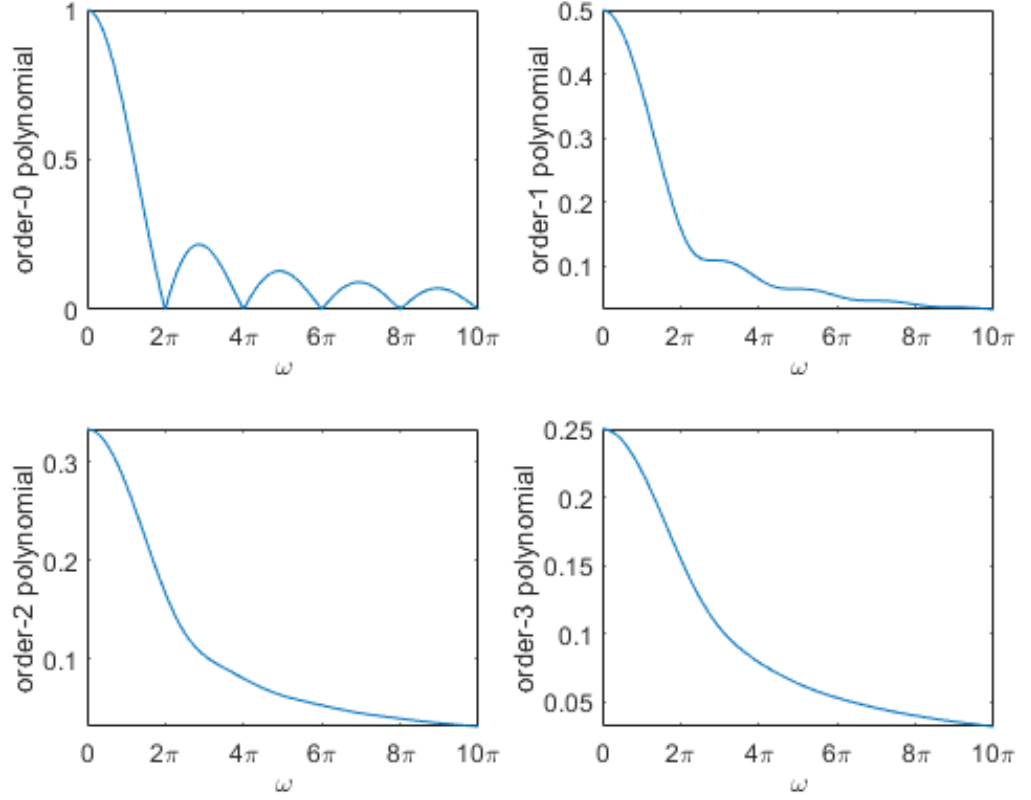


Figure 4.3. Energy spectrum of 1, x , x^2 and x^3 .

Let us define

$$\Delta s_i(\omega) := \hat{S}_i(\omega) - \hat{S}_i(0), \quad (4.6)$$

then the convergence rate Eq.(4.1) could be transformed to

$$\begin{aligned} \|\hat{e}_1^{n+1}\| &= \left| \frac{\Delta s_1(\omega)}{\hat{S}(0) + \Delta s_2(\omega)} \cdot \frac{\Delta s_2(\omega)}{\hat{S}(0) + \Delta s_1(\omega)} \right| \|\hat{e}_1^n\|, \\ &= \left| 1 - \frac{\hat{S}^2(0) + \hat{S}(0)(\Delta s_1(\omega)\Delta s_2(\omega))}{\hat{S}^2(0) + \hat{S}(0)(\Delta s_1(\omega)\Delta s_2(\omega)) + \Delta s_1(\omega)\Delta s_2(\omega)} \right| \|\hat{e}_1^n\|, \end{aligned} \quad (4.7)$$

where $\hat{S}(0) = \hat{S}_1(0) + \hat{S}_2(0)$.

It is clear that the convergence rate depends on the multiplication of $\Delta s_1(\omega)$ and $\Delta s_2(\omega)$: the smaller the multiplication the lower (closer to 0) the convergence rate. When ω is low, the multiplication is negligible so the convergence rate is almost to 0. In high- ω phase, the multiplication dominates and $\hat{S}^2(0)$ is negligible so the convergence rate is roughly equal to $\frac{1}{1+\hat{S}(0)}$. Unlike DR algorithm that converges very slowly for high- ω harmonics, RR-SP0 algorithm performs relatively low convergence rate even if ω is high.

Meanwhile, with different interface boundary location A the multiplication differs and the overall convergence rate could also be influenced correspondingly. Fig.4.4 illustrates the value of $\Delta s_1(\omega)\Delta s_2(\omega)$ with various A , and it shows that $A = 1/8$ case has the smallest multiplication value, while the other three lines ($A = 2/8, 3/8, 4/8$) are very

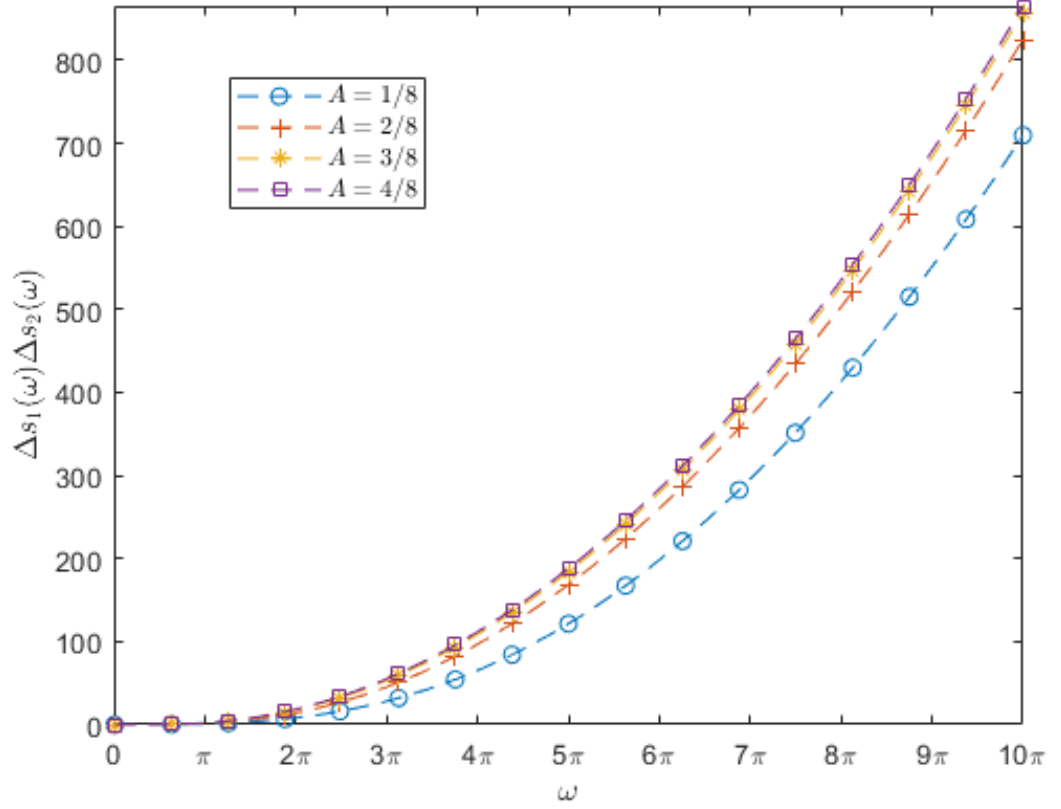


Figure 4.4. Profile of $\Delta s_1(\omega)\Delta s_2(\omega)$ with various A .

close. Consequently, it could be expected that $A = 1/8$ case (optimal case) has the fastest convergence and the other three perform roughly equally.

4.1.2 Poisson's equation test

To verify the analysis result regarding optimal A , let us consider a Poisson's equation that was solved in earlier work [102] (studied here for comparative purpose as will be seen in Chapter 5) as sample case:

$$-\nabla^2 u = f.$$

It is defined in a $[0, 1] \times [0, 1]$ square domain and the boundary conditions are exact. According to [102], it is given that $f = 20(y^2(1 - y) + (3y - 1)x(1 - x))$, and the exact solution is $u = 10x(1 - x)^2y^2(1 - y)$. The mesh is uniform with 65 nodes in each direction. The interface boundary is evenly distributed at $y = A = 1/8, 2/8 \dots 7/8$ respectively. The convergence criterion is

$$\max(\|\frac{e_1^{n+1} - e_1^n}{e_1^{n+1}}\|_\infty, \|\frac{e_2^{n+1} - e_2^n}{e_2^{n+1}}\|_\infty) < 10^{-3} \quad \text{on } \Gamma. \quad (4.8)$$

The Poisson's equation is solved by RR-SP0 (parallel). The parallel algorithm is applied because it takes twice iterations as sequential algorithm to converge, which makes it more sensitive to capture the iteration difference among cases. In the test, $S_i I$ is considered as

constant ($S_i^* I$) and function of x separately. The numerical test result is shown in Table 4.1.

Table 4.1. Convergence iterations of RR-SP0 in application to a Poisson's equation case.

A	1/8	2/8	3/8	4/8	5/8	6/8	7/8
$S_i I$	8	9	9	9	8	8	7
$S_i^* I$	12	16	17	18	17	15	11

The second row of Table 4.1 shows the convergence iterations of RR-SP0 when $S_i I$ is constant, which follows the simplification in the convergence rate calculation. It shows that $A = 1/8$ converges in 12 iterations, while the other three ($A = 2/8, 3/8, 4/8$) perform very close iterations, around 17. The test result fully agrees with the calculation result as shown in Fig.4.4.

In addition, the convergence iteration number of $A = 7/8$ is nearly equal to $A = 1/8$, and so are $A = (2/8, 6/8)$, $A = (3/8, 5/8)$ cases. As a matter of fact, it could be interpreted as a reflection of 'symmetric' property of \hat{S}_1 and \hat{S}_2 in terms of A . From the definition of \hat{S}_i , it could be seen that $\hat{S}_{1|A=A^*}(\omega_0) = \hat{S}_{2|A=1-A^*}(\omega_0)$, as a result $\Delta_{s_1|A=A^*} \Delta_{s_2|A=A^*} = \Delta_{s_2|A=1-A^*} \Delta_{s_1|A=1-A^*}$. In other words, the profile of \hat{S}_1 (as well as Δ_{s_1}) coincides with the mirror profile of \hat{S}_2 (Δ_{s_2}). The axis of the mirror reflection is $A = 4/8$.

The cases of $S_i I$ not being a constant show less iterations than its counterpart, which could be roughly explained by non-local effect of $S_i I$: the two-dimensional operator retains non-local effect but the one-dimensional operator hardly does. From Fig.4.5 it could be seen that the non-local effect mostly happens at the vicinity of the two walls, so it could be stated that the non-local effect of $S_i I$ is almost equal to the boundary effect. What is more, Table 4.3 shows that the iteration numbers of $S_i I$ cases are nearly equal regardless of A , which means that A could also affect the strength of the boundary effect. It could be speculated that $A = 4/8$ results in the strongest boundary effect while $A = 1/8$ (as well as $7/8$) is merely mildly influenced by the boundary effect. It will be discussed in Section 4.4 about how A affecting the boundary effect strength.

To summarise, the convergence of RR-SP0 applied to Poisson's equation depends on the multiplication of SP operator symbols increment: $\Delta_{s_1}(\omega) \Delta_{s_2}(\omega)$. Lower multiplication results in smaller convergence rate and faster convergence. In addition, the two symbol increments are symmetric in terms of interface boundary location A , so the convergence rate is always equivalent for a pair of symmetric A : A^* and $1 - A^*$.

4.1.3 Comparison between parallel and sequential algorithms

Comparing with sequential version, the parallel algorithm takes twice number of iterations in achieving the convergence, because the parallel algorithm takes two iterations to reduce the error to a level to which sequential algorithm takes only 1 iteration. The reason could be seen from the algorithms themselves: in the parallel case, e_1^{n+1} uses the result of e_2^n from last iteration when more accurate data of e_2^{n+1} has been already updated in current iteration. The

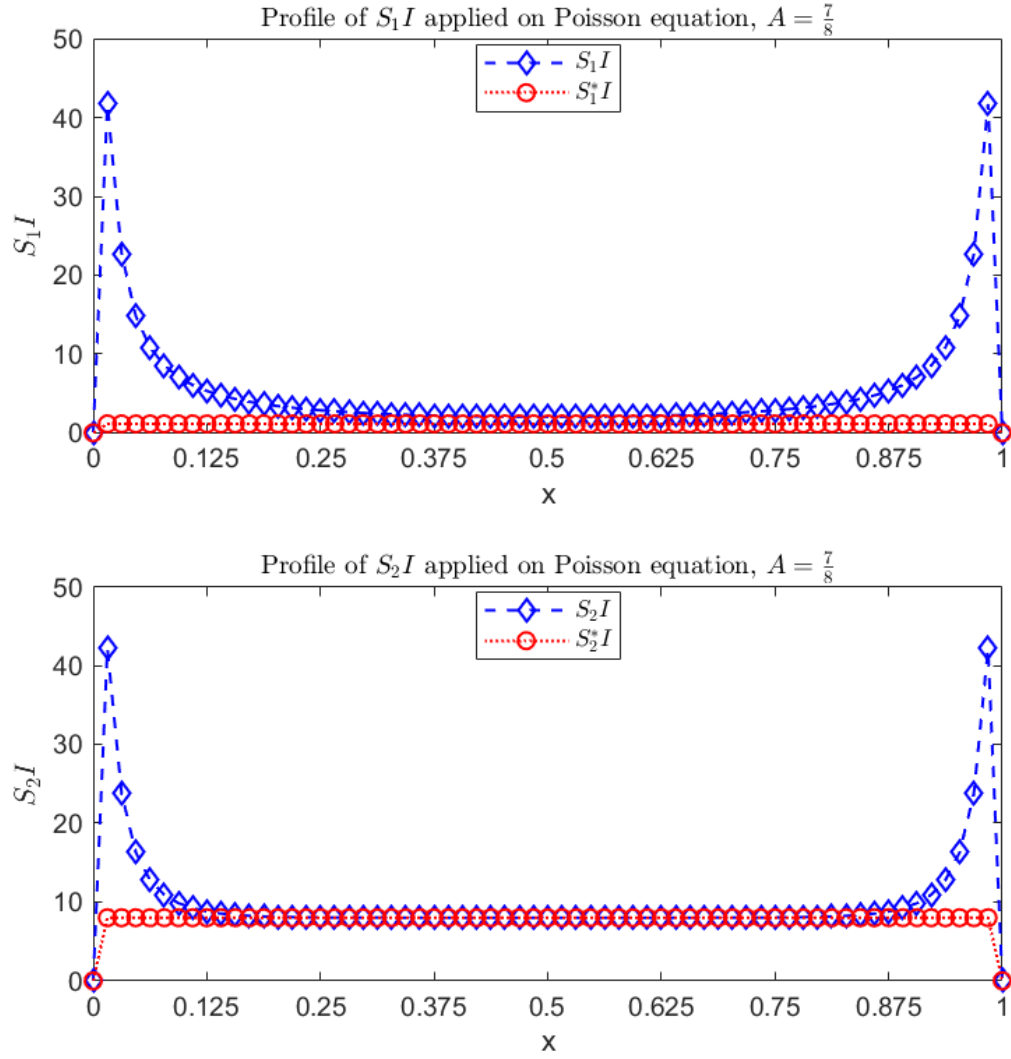


Figure 4.5. Profile of S_1I , S_2I applied to Poisson's equation, $A = \frac{7}{8}$.

newly-generated data is not used until next iteration. On the other hand, the parallel version has the advantage of computing both subdomains at one time (assume two CPUs work at the same time), which means the computing time of the parallel case is roughly 50% of sequential case in each iteration. Therefore, the overall computing time of the two versions is roughly equal.

In this project, only two subdomains are generated in domain decomposition method. If there were more than two subdomains, the parallel computing should have taken less time because in each iteration the computing workload is split by N (N is number of subdomains, $N \gg 2$) CPUs working at the same time. This is the typical way of parallel computing working in conventional domain decomposition methods. In the thesis, ENDD algorithms in application to resolving near-wall turbulence do not take much advantage of parallel computing because only two subdomains are generated, and usually the inner subdomain contains finer mesh and undertakes heavier computation. The numerical tests in this project suggest that inner subdomain usually takes as twice computing time as outer subdomain does, which means two CPUs could not synchronise if they were running in parallel. Therefore, sequential algorithm

costs less computing time in this project.

The comparison result between the parallel and the sequential algorithms applies to RR-SP1 and RR-SP2 as well, although the convergence rate calculation between the parallel and the sequential versions of the two algorithms is not performed.

4.2 Convergence analysis of RR-SP2 without boundary effect

4.2.1 Analysis of RR-SP2: approximation 1

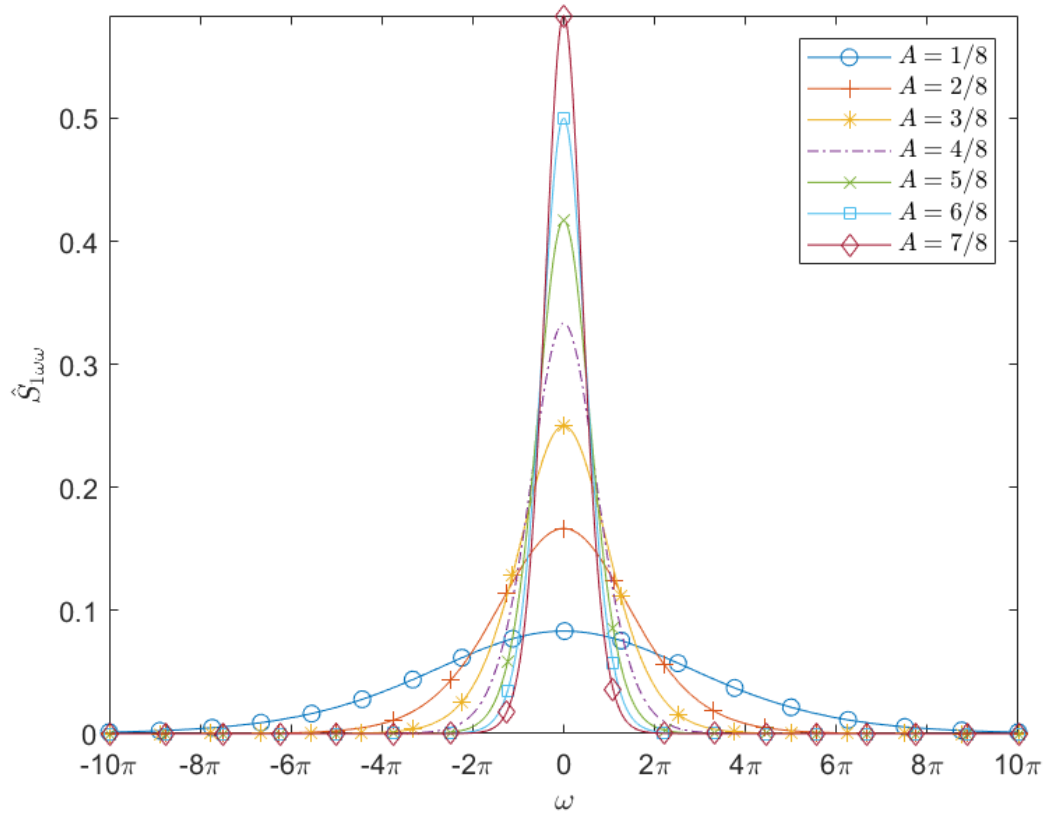


Figure 4.6. Profile of $\hat{S}_{1\omega\omega}(\omega)$ with various A . The profile of $\hat{S}_{2\omega\omega}(\omega)$ could also be depicted by this figure: $\hat{S}_{2\omega\omega}(\omega)|_{A^*} = \hat{S}_{1\omega\omega}(\omega)|_{(1-A^*)}$.

The convergence rate of RR-SP2 has similar structure in formulation to RR-SP0, and the only difference is the extra term in numerator and denominator: $\frac{1}{2}\omega^2\hat{S}_{i\omega\omega}(0)$. The middle term $\omega\hat{S}_{i\omega}(0)$ is 0 thus neglected:

$$\begin{aligned} & \|\hat{e}_i^{n+1}(A)\| \\ &= \left| \frac{\hat{S}_1(0) - \hat{S}_1(\omega) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)}{\hat{S}_1(0) + \hat{S}_2(\omega) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)} \cdot \frac{\hat{S}_2(0) - \hat{S}_2(\omega) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)}{\hat{S}_2(0) + \hat{S}_1(\omega) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)} \right| \|\hat{e}_i^n(A)\|. \end{aligned}$$

Comparing with the convergence rate of RR-SP0, the influence of the extra term depends on the value of $\hat{S}_{i\omega\omega}(0)$, which is shown in Fig.4.6. The figure shows that the value of $\hat{S}_{1\omega\omega}(0)$ increases with rising A . In addition, the profile of $\hat{S}_{2\omega\omega}(\omega)$ is symmetric to $\hat{S}_{1\omega\omega}(\omega)$ along

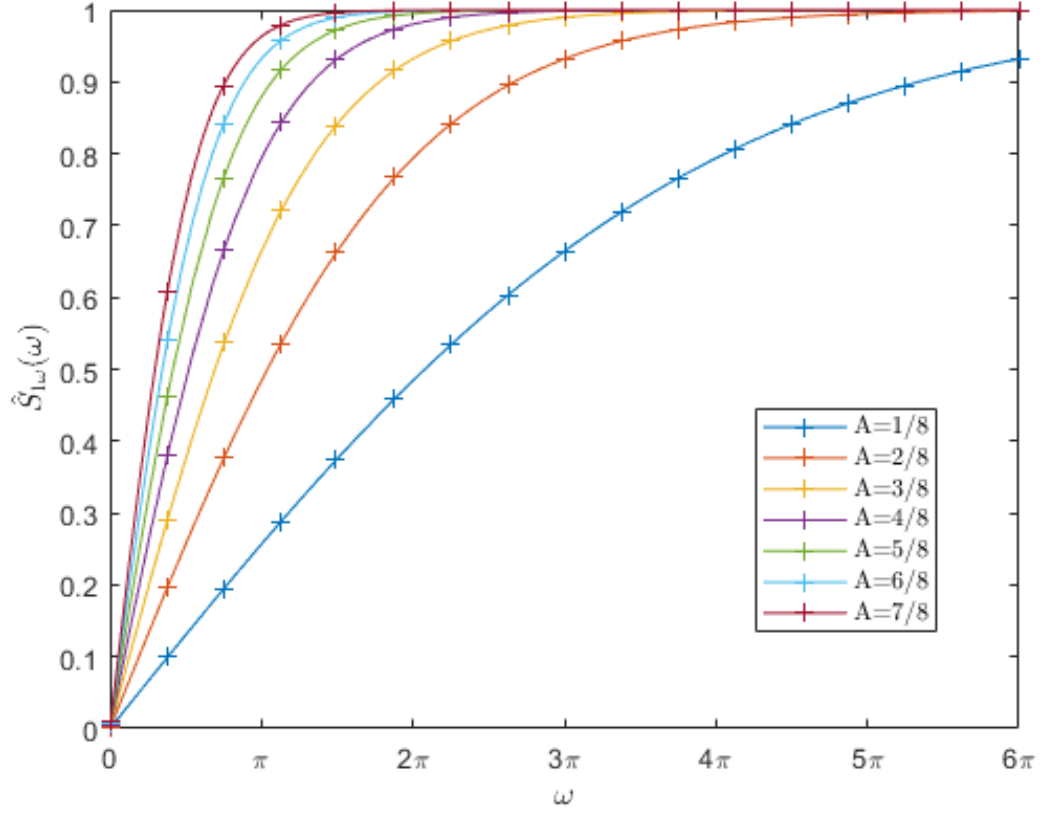


Figure 4.7. Profile of $\hat{S}_{1\omega}(\omega)$ with various A . The profile of $\hat{S}_{2\omega\omega}(\omega)$ could also be depicted by this figure: $\hat{S}_{2\omega\omega}(\omega)|_{A^*} = \hat{S}_{1\omega\omega}(\omega)|_{(1-A^*)}$. Every line has a 'turning point' where the line turns to flat. The point value decreases with increasing A . For $A = 7/8$, the point value is about 2π ; for $A = 1/8$, the point value is about 6π or even larger.

the axis $A = 4/8$ so the former could also be depicted by the figure.

To analyse the convergence rate, let us consider the numerator first:

$$\hat{S}_i(0) - \hat{S}_i(\omega) + \frac{1}{2}\omega^2\hat{S}_{i\omega\omega}(0).$$

Divide above expression by ω two times:

$$\begin{aligned} & \frac{\hat{S}_i(0) - \hat{S}_i(\omega)}{\omega} + \frac{1}{2}\omega\hat{S}_{i\omega\omega}(0), \\ & \rightarrow -\hat{S}_{i\omega}(\omega^*) + \frac{1}{2}\omega\hat{S}_{i\omega\omega}(0), \quad 0 \leq \omega^* \leq \omega, \quad \eta_i\omega = \omega^*, \\ & \rightarrow \eta_i \frac{\hat{S}_{i\omega}(0) - \hat{S}_{i\omega}(\omega^*)}{\eta_i\omega} + \frac{1}{2}\hat{S}_{i\omega\omega}(0), \quad (\hat{S}_{i\omega}(0) = 0,) \\ & \rightarrow -\eta_i\hat{S}_{i\omega\omega}(\omega^{**}) + \frac{1}{2}\hat{S}_{i\omega\omega}(0), \quad 0 \leq \omega^{**} \leq \omega^*. \end{aligned} \tag{4.9}$$

Obviously, $0 < \eta_i < 1$. The magnitude of the numerator depends on the value of $\hat{S}_{i\omega\omega}(0)$ and $\hat{S}_{i\omega\omega}(\omega^{**})$. The former has been given in Fig.4.6 but it is difficult to determine exactly the value of the latter. Here is the estimation:

Fig.4.7 shows the profile of $\hat{S}_{1\omega}(\omega)$ with various A . Due to the symmetry, $\hat{S}_{2\omega}(\omega)$ could also

be depicted by the figure. Therefore, considering only the case of $\hat{S}_{1\omega\omega}(\omega)$ with A ranging from $1/8$ to $7/8$ is enough. The figure shows every line has a 'turning point' at some ω , before which the line rises quickly (nearly linearly) and afterwards almost goes flat. The turning point (denote by ω^t) increases with smaller A . Meanwhile, $\omega^* < \omega^t$. It is obviously true if $\omega < \omega^t$. Here is the explanation for the case when $\omega > \omega^t$:

$$\begin{aligned}
\hat{S}_{i\omega}(\omega^*) &= \frac{\hat{S}_i(\omega) - \hat{S}_i(0)}{\omega}, \\
&= \frac{\int_0^\omega \hat{S}_{i\omega}(\xi) d\xi}{\omega}, \\
&\approx \frac{\int_0^{\omega^t} l\xi d\xi + \int_{\omega^t}^\omega l\omega^t d\xi}{\omega}, \quad (\text{before } \omega^t, \text{ assume } \hat{S}_{i\omega}(\omega) = l\omega \text{ where } l = \text{const.}) \\
&= \frac{l\omega^t(\omega - \frac{\omega^t}{2})}{\omega} < l\omega^t = \hat{S}_{i\omega}(\omega^t).
\end{aligned} \tag{4.10}$$

Since $\hat{S}_{i\omega}(0) < \hat{S}_{i\omega}(\omega^*) < \hat{S}_{i\omega}(\omega^t)$ and $\hat{S}_{i\omega}$ is continuous, it could be concluded that $0 < \omega^* < \omega^t$. In other words, ω^* always falls on the (almost) straight line of $\hat{S}_{i\omega}$. When $\omega \gg \omega^t$, $\omega^* \approx \omega^t$.

Additionally, $\omega^{**} \leq \omega^*$ so ω^{**} is on the straight line too. According to Fig.4.7, $\hat{S}_{1\omega\omega}(\omega^{**}) \approx \hat{S}_{1\omega\omega}(\omega^*) \approx \hat{S}_{1\omega\omega}(0)$. Then Eq.(4.9) becomes

$$(\frac{1}{2} - \eta_1)\hat{S}_{1\omega\omega}(0).$$

Next, consider the convergence rate as a whole:

$$\begin{aligned}
&\left| \frac{\hat{S}_1(0) - \hat{S}_1(\omega) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)}{\hat{S}_1(0) + \hat{S}_2(\omega) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)} \cdot \frac{\hat{S}_2(0) - \hat{S}_2(\omega) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)}{\hat{S}_2(0) + \hat{S}_1(\omega) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)} \right| \\
&\approx \left| \frac{(\frac{1}{2} - \eta_1)\omega^2\hat{S}_{1\omega\omega}(0)}{\hat{S}_1(0) + \hat{S}_2(\omega) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)} \cdot \frac{(\frac{1}{2} - \eta_2)\omega^2\hat{S}_{2\omega\omega}(0)}{\hat{S}_2(0) + \hat{S}_1(\omega) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)} \right|, \tag{4.11} \\
&= \left| 1 - \frac{G(\omega) + [\frac{1}{2}(\eta_1 + \eta_2) - \eta_1\eta_2]\omega^4\hat{S}_{1\omega\omega}(0)\hat{S}_{2\omega\omega}(0)}{G(\omega) + \frac{1}{4}\omega^4\hat{S}_{1\omega\omega}(0)\hat{S}_{2\omega\omega}(0)} \right|,
\end{aligned}$$

where

$$\begin{aligned}
G(\omega) &:= (\hat{S}_1(0) + \hat{S}_2(\omega))(\hat{S}_2(0) + \hat{S}_1(\omega)) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)(\hat{S}_1(0) + \hat{S}_2(\omega)) + \\
&\quad \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)(\hat{S}_2(0) + \hat{S}_1(\omega)).
\end{aligned}$$

It is clear that the convergence rate depends on the multiplication $\hat{S}_{1\omega\omega}(0)\hat{S}_{2\omega\omega}(0)$: the smaller the multiplication, the lower the convergence rate, because when the multiplication is small enough to be negligible to $G(\omega)$, the value tends to zero. Otherwise $G(\omega)$ could not dominate the value and the entire value could be largely influenced by $\hat{S}_{1\omega\omega}(0)\hat{S}_{2\omega\omega}(0)$.

According to symmetry, the multiplication could be transformed to:

$$\hat{S}_{1\omega\omega}(0)|_A \hat{S}_{2\omega\omega}(0)|_A = \hat{S}_{1\omega\omega}(0)|_A \hat{S}_{1\omega\omega}(0)|_{(1-A)}. \quad (4.12)$$

Fig.4.6 shows that a 'well-behaved' A is always paired with 'poorly-behaved' A . For example, $A = 1/8$ and $7/8$. The former has the least $\hat{S}_{1\omega\omega}(0)$ value but the latter has the largest. By calculating Eq.(4.12), it enables the comparison of the convergence rate of RR-SP2 among various A . The calculation result is illustrated in Fig.4.8, which shows that $A = 1/8$ case has the smallest multiplication value and $4/8$ case has the largest. Therefore, the former could be expected to perform the fastest convergence while the latter performs the lowest. The other 2 cases should be in the middle place and follow the order of magnitude of A itself. Note $A = 7/8$ case is identical to $A = 1/8$ and so are $A = 5/8, 6/8$, so the cases of $A > 4/8$ are not plotted. For example, according to Eq.(4.12), $\hat{S}_{1\omega\omega}(0)|_{1/8} \hat{S}_{2\omega\omega}(0)|_{1/8} = \hat{S}_{2\omega\omega}(0)|_{7/8} \hat{S}_{1\omega\omega}(0)|_{7/8}$, which is equal to the case of $A = 7/8$: $\hat{S}_{1\omega\omega}(0)|_{7/8} \hat{S}_{2\omega\omega}(0)|_{7/8}$.

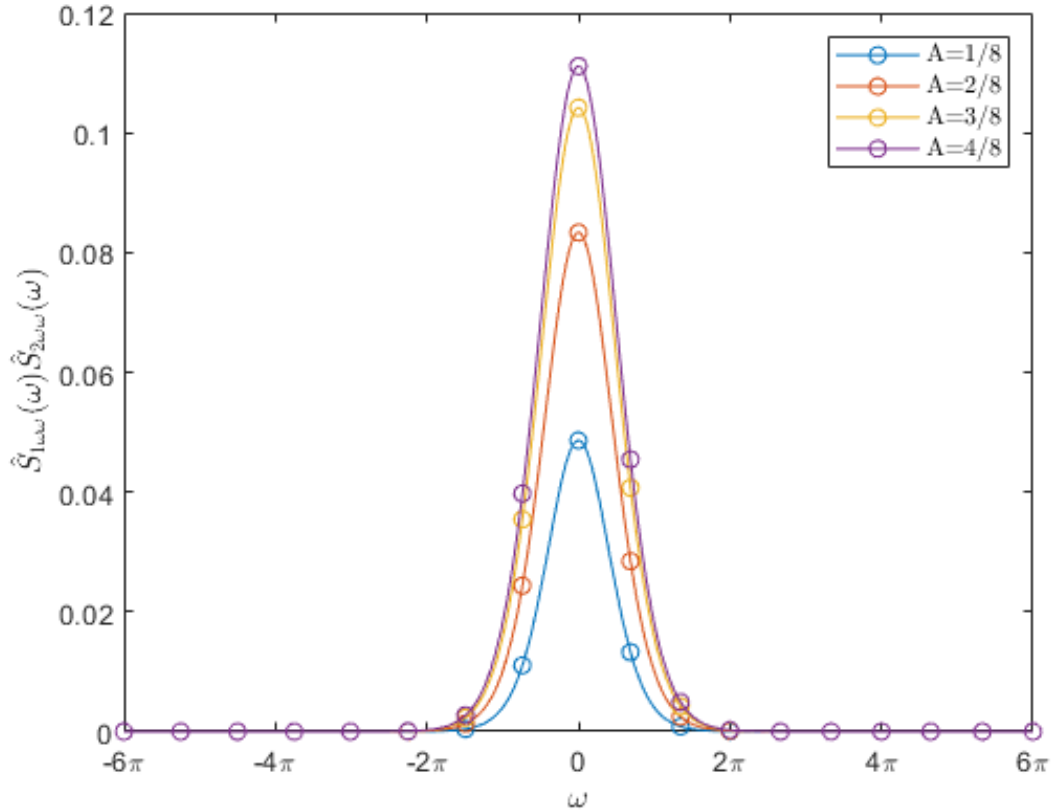


Figure 4.8. Profile of $\hat{S}_{1\omega\omega}(0)\hat{S}_{2\omega\omega}(0)$. Since $\hat{S}_{1\omega\omega}|_A(0) = \hat{S}_{2\omega\omega}|_{1-A}(0)$, the profile is symmetric along $A = 4/8$, so the cases for $A > 4/8$ are not plotted.

Fig.4.9 compares the convergence rate between RR-SP0 and RR-SP2 with A ranging from $1/8$ to $4/8$. The cases of A larger than $4/8$ could be found in the figure by its 'twin' pair: $1 - A$ thus neglected. The figure shows that $A = 1/8$ has the lowest convergence rate and the case of $A = 2/8, 3/8$ and $4/8$ follows in the order of rising A . Meanwhile, the lines of $A = 3/8$ and $4/8$ cases are very close and the two cases have close lines in Fig.4.8 as well. It indicates that $\hat{S}_{1\omega\omega}(0)\hat{S}_{2\omega\omega}(0)$ has good approximation of the convergence rate. In addition, it could be

seen that SP2 has clear advantage over SP0 when A is small. As A approaches the symmetry line ($A = 4/8$), the advantage of SP2 becomes less. Overall, SP2 shows lower convergence rate than SP0 regardless of A .

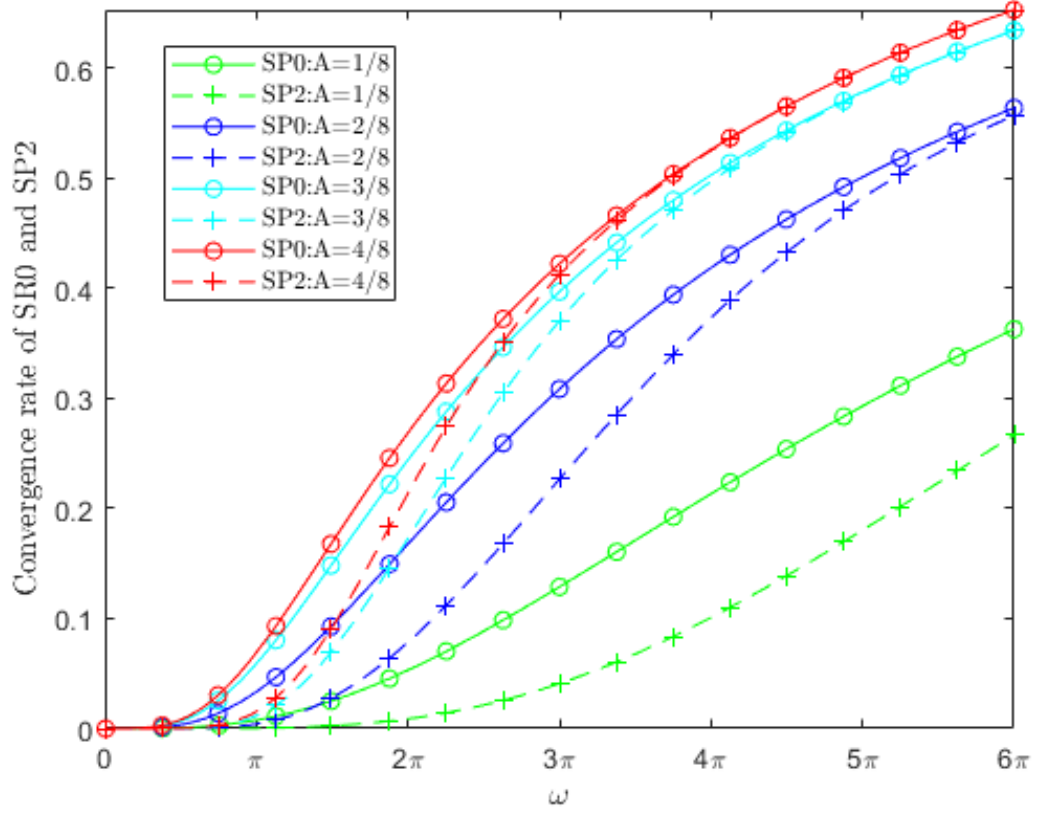


Figure 4.9. Convergence rate of RR-SP0 and RR-SP2 with various A .

4.2.2 Analysis of RR-SP2: approximation 2

The approximation in Subsection 4.2.1 assumes that the profile of $\hat{S}_{i\omega}(\omega)$ is linear before ω^t and flat afterwards. The presumption leads to $\hat{S}_{i\omega\omega}(\omega^{**}) \approx \hat{S}_{i\omega\omega}(0)$, which could incur non-negligible error especially when A is large, see Fig.4.6 ($\hat{S}_{i\omega\omega}(\omega^{**}) \neq \hat{S}_{i\omega\omega}(0)$). This subsection introduces another approximation of the convergence rate of RR-SP2. The approximation still presumes that $\hat{S}_{i\omega}(\omega)$ is linear before ω^t and flat afterwards but will not consider $\hat{S}_{i\omega\omega}$ as a constant.

According to Eq.(4.9), the convergence rate could be expressed as

$$\begin{aligned}
& \left| \frac{\hat{S}_1(0) - \hat{S}_1(\omega) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0)}{\hat{S}_1(0) + \hat{S}_2(\omega) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0)} \cdot \frac{\hat{S}_2(0) - \hat{S}_2(\omega) + \frac{1}{2}\omega^2 \hat{S}_{2\omega\omega}(0)}{\hat{S}_2(0) + \hat{S}_1(\omega) + \frac{1}{2}\omega^2 \hat{S}_{2\omega\omega}(0)} \right| \\
&= \left| \frac{\hat{S}_1(0) - \hat{S}_1(\omega) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0)}{\hat{S}_1(0) - \hat{S}_1(\omega) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0) + \hat{S}(\omega)} \cdot \frac{\hat{S}_2(0) - \hat{S}_2(\omega) + \frac{1}{2}\omega^2 \hat{S}_{2\omega\omega}(0)}{\hat{S}_2(0) - \hat{S}_2(\omega) + \frac{1}{2}\omega^2 \hat{S}_{2\omega\omega}(0) + \hat{S}(\omega)} \right|, \\
&= \left| \frac{-\eta_1 \omega^2 \hat{S}_{1\omega\omega}(\omega^{**}) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0)}{-\eta_1 \omega^2 \hat{S}_{1\omega\omega}(\omega^{**}) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0) + \hat{S}(\omega)} \cdot \frac{-\eta_2 \omega^2 \hat{S}_{2\omega\omega}(\omega^{**}) + \frac{1}{2}\omega^2 \hat{S}_{2\omega\omega}(0)}{-\eta_2 \omega^2 \hat{S}_{1\omega\omega}(\omega^{**}) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0) + \hat{S}(\omega)} \right|, \\
&= \left| \frac{-\frac{1}{2}\omega^2 \Delta s_{1\omega\omega}(\omega^{**}) + (\frac{1}{2} - \eta_1) \omega^2 \hat{S}_{1\omega\omega}(\omega^{**})}{-\frac{1}{2}\omega^2 \Delta s_{1\omega\omega}(\omega^{**}) + (\frac{1}{2} - \eta_1) \omega^2 \hat{S}_{1\omega\omega}(\omega^{**}) + \hat{S}(\omega)} \right| \cdot \\
&\quad \left| \frac{-\frac{1}{2}\omega^2 \Delta s_{2\omega\omega}(\omega^{**}) + (\frac{1}{2} - \eta_2) \omega^2 \hat{S}_{2\omega\omega}(\omega^{**})}{-\frac{1}{2}\omega^2 \Delta s_{2\omega\omega}(\omega^{**}) + (\frac{1}{2} - \eta_2) \omega^2 \hat{S}_{2\omega\omega}(\omega^{**}) + \hat{S}(\omega)} \right|.
\end{aligned} \tag{4.13}$$

Note $\hat{S}(\omega) = \hat{S}_1(\omega) + \hat{S}_2(\omega)$ and $\Delta s_{i\omega\omega}(\omega) := \hat{S}_{i\omega\omega}(\omega) - \hat{S}_{i\omega\omega}(0)$.

Next consider the value of η_i . Let us assume $\omega^* := \eta_i \omega$ is either less or greater than $\frac{1}{2}\omega^t$ (certainly less than ω^t , see Eq.(4.10)). The turning point ω^t could be found from the profile of $\hat{S}_1(\omega)$ as well, see Fig.4.10. The figure is similar to Fig.4.1 in terms of showing $\hat{S}_1(\omega)$ profile but concentrates more on displaying the two-dimensional curvature of the lines. The turning point is roughly where the lines start rising linearly, and before it the rise is acting along changeable tangent.

1. If ω^* is less than $\frac{1}{2}\omega^t$, ω should be smaller than ω^t because $\omega \approx 2\omega^*$ and $\eta_i \approx 1/2$. Here is the elaboration:

$$\begin{aligned}
-\hat{S}_{1\omega}(\omega^*) &= \frac{\hat{S}_1(0) - \hat{S}_1(\omega)}{\omega}, \\
&= -\frac{\int_0^\omega \hat{S}_{1\omega}(\xi) d\xi}{\omega}, \\
&\approx -\frac{\hat{S}_{1\omega}(\frac{\omega}{2})\omega}{\omega}, \quad (\hat{S}_{1\omega} \text{ is almost linear before } \omega^t,) \\
&= -\hat{S}_{1\omega}(\frac{\omega}{2}).
\end{aligned}$$

Therefore, $\eta_i \approx 1/2$ and Eq.(4.13) becomes

$$\left| \frac{-\frac{1}{2}\omega^2 \Delta s_{1\omega\omega}(\omega^{**})}{-\frac{1}{2}\omega^2 \Delta s_{1\omega\omega}(\omega^{**}) + \hat{S}(\omega)} \cdot \frac{-\frac{1}{2}\omega^2 \Delta s_{2\omega\omega}(\omega^{**})}{-\frac{1}{2}\omega^2 \Delta s_{2\omega\omega}(\omega^{**}) + \hat{S}(\omega)} \right|. \tag{4.14}$$

2. If ω^* is larger than $\frac{1}{2}\omega^t$, ω must be larger than ω^t . Assume $\omega = k\omega^t$ ($k > 1$), it could be

proved that $\eta_i = \frac{1}{k} - \frac{1}{2k^2}$. Here is the elaboration:

$$\begin{aligned}\hat{S}_{i\omega}(\omega^*) &= \frac{l\omega^t(\omega - \frac{\omega^t}{2})}{\omega}, \quad \text{from Eq.(4.10),} \\ \rightarrow \omega^* &= \frac{\omega^t(\omega - \frac{\omega^t}{2})}{\omega}, \\ \rightarrow \eta_i &:= \frac{\omega^*}{\omega} = \frac{\omega^t(\omega - \frac{\omega^t}{2})}{\omega^2}, \quad (\omega = k\omega^t,) \\ \rightarrow \eta_i &= \frac{1}{k} - \frac{1}{2k^2}.\end{aligned}$$

Both Fig.4.7 and Fig.4.10 show that for those small-valued A , ω^t is relative large and k could be quite small (no more than 2), so η_i is still close to $1/2$ and $(\frac{1}{2} - \eta_i)$ is small enough to neglect. For those high-valued A , k is large thus $(\frac{1}{2} - \eta_i)$ could not be neglected but on the other hand $\hat{S}_{i\omega\omega}(\omega^{**})$ could be very small, see Fig.4.6. It could be estimated that ω^{**} is close to (slightly smaller) ω^* (due to quasi-linearity of $\hat{S}_{i\omega}$ before ω^t) so when ω^* is larger than $\frac{1}{2}\omega^t$, so is ω^{**} . Take $A = 7/8$ case for example, ω^t is about 2π . At $\omega = 0$ the peak value is reached but at merely $\omega = \frac{\omega^t}{2} = \pi$, $\hat{S}_{i\omega\omega}(\pi)$ drops down to nearly 0. It is common for high-valued A cases that they all drop sharply down to very small number after $\frac{1}{2}\omega^t$ so in this case $\hat{S}_{i\omega\omega}(\omega^{**})$ is negligible.

The above estimation shows that in any case, the term $(\frac{1}{2} - \eta_i)\omega^2\hat{S}_{i\omega\omega}(\omega^{**})$ in Eq.(4.13) could be neglected and the equation could be reduced to Eq.(4.14), which is equal to

$$\left| 1 - \frac{\hat{S}^2(\omega) - \frac{1}{2}\omega^2\hat{S}(\omega)(\Delta s_{1\omega\omega}(\omega^{**})\Delta s_{2\omega\omega}(\omega^{**}))}{\hat{S}^2(\omega) - \frac{1}{2}\omega^2\hat{S}(\omega)(\Delta s_{1\omega\omega}(\omega^{**})\Delta s_{2\omega\omega}(\omega^{**})) + \frac{1}{4}\omega^4\Delta s_{1\omega\omega}(\omega^{**})\Delta s_{2\omega\omega}(\omega^{**})} \right|.$$

It is clear that the smaller the multiplication $\Delta s_{1\omega\omega}(\omega^{**})\Delta s_{2\omega\omega}(\omega^{**})$, the smaller the convergence rate. As mentioned earlier, $0 < \omega^{**} < \omega^t$, so by cutting off the extra part ($\omega > \omega^t$) of the profile of $\Delta s_{1\omega\omega}(\omega)$, the profile of $\Delta s_{1\omega\omega}(\omega^{**})$ could be obtained.

Fig.4.11 illustrates the multiplication of $\Delta s_{1\omega\omega}(\omega^{**})\Delta s_{2\omega\omega}(\omega^{**})$ (extra part is not cut off but the trend remains unchanged), which shows consistent result as Fig.4.8. The case of $A = 1/8$ has the smallest multiplication and it is followed by $A = 2/8, 3/8$ and $4/8$ respectively. The $3/8$ and $4/8$ lines are close, as is verified by Fig.4.9. It means that $\Delta s_{1\omega\omega}(\omega)\Delta s_{2\omega\omega}(\omega)$ also offers good approximation of the convergence rate of RR-SP2. Compared with the approximation by $\hat{S}_{1\omega\omega}(0)\hat{S}_{2\omega\omega}(0)$, this approximation should be more accurate because it does not presume $\hat{S}_{i\omega\omega}$ as constant (and it is really not). The result shows that both methods are effective but the latter might be easier to use because it requires the knowledge of $\hat{S}_{i\omega\omega}(\omega)$ at $\omega = 0$ only.

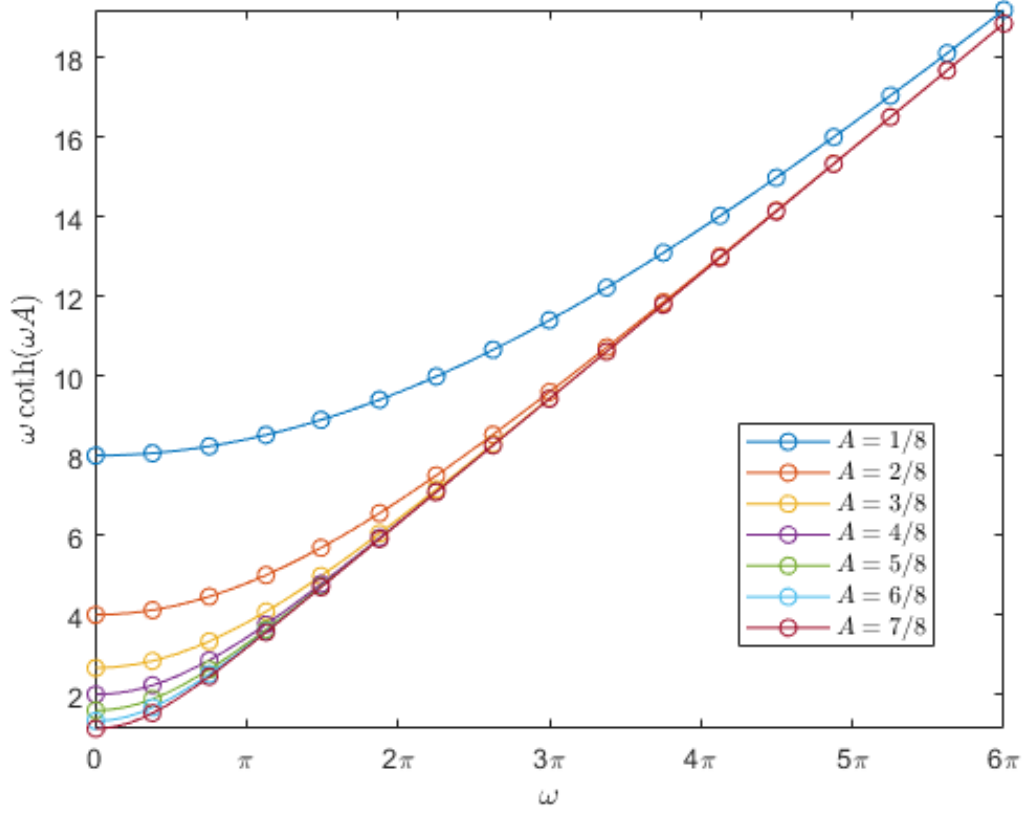


Figure 4.10. Profile of $\hat{S}_1(\omega) := \omega \coth(\omega A)$ with various A . If ω^* is smaller than $\frac{1}{2}\omega^t$, the mean value ω^* is approximately equal to $\frac{1}{2}\omega$ because $\hat{S}_{1\omega}(\omega)$ rises almost linearly before ω^t . Therefore, $\eta_i \approx 1/2$ in this case.

4.3 Convergence analysis of conventional RR algorithm

Consider a BVP defined in $\Omega := \{0 \leq x \leq 1, 0 \leq y \leq B\}$ to be solved by a Robin-Robin algorithm:

$$\begin{aligned}
 \mathcal{L}u_1^{n+1} &= f_1, & \text{on } \Omega_1 \\
 u_{1y}^{n+1} + \sigma_1 u_1^{n+1} &= u_{2y}^n + \sigma_1 u_2^n, & \text{on } \Gamma, \\
 u_1^{n+1} &= 0, & \text{on } \partial\Omega_1 \setminus \Gamma, \\
 \mathcal{L}u_2^{n+1} &= f_2, & \text{on } \Omega_2 \\
 u_{2y}^{n+1} - \sigma_2 u_2^{n+1} &= u_{1y}^n - \sigma_2 u_1^n, & \text{on } \Gamma, \\
 u_2^{n+1} &= 0, & \text{on } \partial\Omega_2 \setminus \Gamma.
 \end{aligned} \tag{4.15}$$

Note σ_1 and σ_2 are constants. Particularly, if let $\sigma_1 := S_2^* I$ and $\sigma_2 := S_1^* I$, the algorithm is equal to RR-SP0. According to the convergence rate calculation performed in Chapter 3, the convergence rate of the algorithm in application to solving Poisson's equation could be given directly:

$$\|\hat{e}_i^{n+1}\| = \left| \frac{\sigma_2 - k\pi \coth(k\pi A)}{\sigma_2 + k\pi \coth(k\pi(B-A))} \frac{\sigma_1 - k\pi \coth(k\pi(B-A))}{\sigma_1 + k\pi \coth(k\pi A)} \right| \|\hat{e}_i^n\|, \tag{4.16}$$

($k = 1, 2, \dots$)

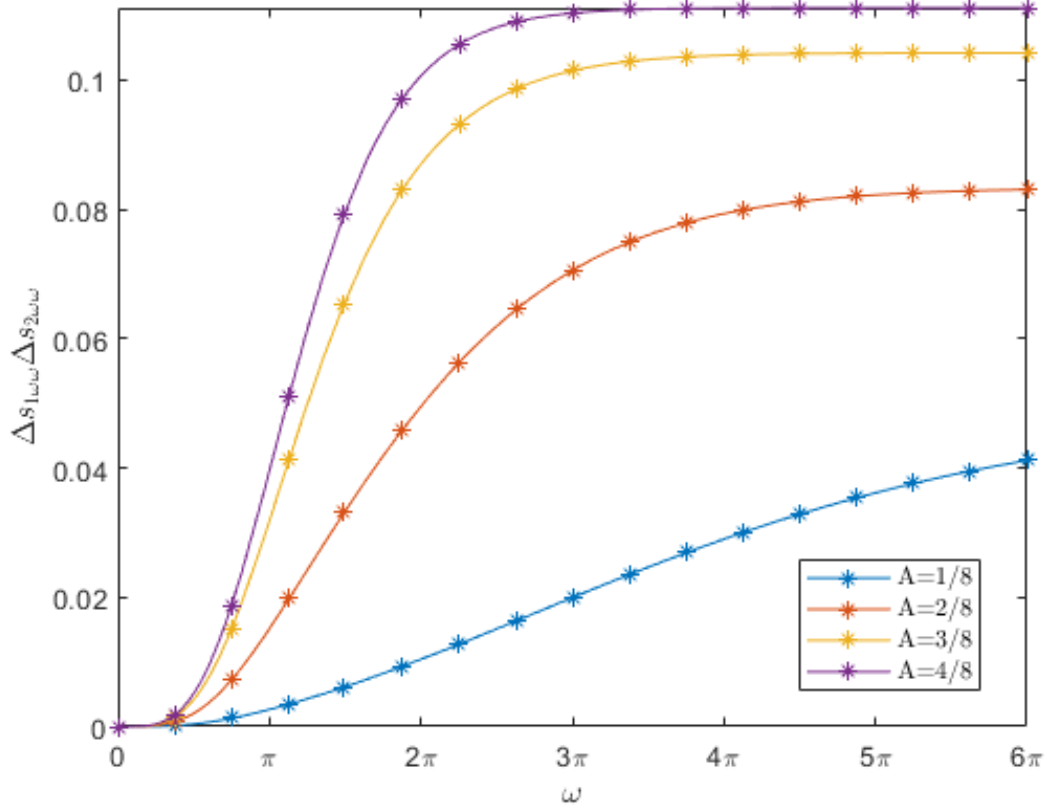


Figure 4.11. Profile of $\Delta s_{1\omega\omega}(\omega)\Delta s_{2\omega\omega}(\omega)$. The definition of $\Delta s_{i\omega\omega}(\omega)$ is $\Delta s_{i\omega\omega}(\omega) := \hat{S}_{i\omega\omega}(\omega) - \hat{S}_{i\omega\omega}(0)$. By cutting off the part of $\omega > \omega^t$, the figure could show $\Delta s_{1\omega\omega}(\omega^{**}) \cdot \Delta s_{2\omega\omega}(\omega^{**})$

If let $\lambda := \sigma_1 = \sigma_2$, the algorithm becomes what Lui proposed in [102]:

$$\begin{aligned} \mathcal{L}u_i^{n+1} &= f_i, \quad \text{on } \Omega_i, \\ \frac{\partial u_i^{n+1}}{\partial n_i} + \lambda u_i^{n+1} &= g_i^{n+1}, \quad \text{on } \Gamma, \\ u_i^{n+1} &= 0 \quad \text{on } \partial\Omega_i \setminus \Gamma, \end{aligned} \quad (4.17)$$

where

$$g_1^{n+1} = 2\lambda u_{2|\Gamma}^n - g_2^n, \quad g_2^{n+1} = 2\lambda u_{1|\Gamma}^n - g_1^n. \quad (4.18)$$

Following Eq.(4.16), the convergence rate is as follows:

$$\|\hat{e}_i^{n+1}\| = \left| \frac{\lambda - k\pi \coth(k\pi A)}{\lambda + k\pi \coth(k\pi(B-A))} \frac{\lambda - k\pi \coth(k\pi(B-A))}{\lambda + k\pi \coth(k\pi A)} \right| \|\hat{e}_i^n\|, \quad (k=1, 2, \dots). \quad (4.19)$$

Lui stated in his paper [102] that the optimal $\lambda = 6.794$ could achieve the fastest convergence regardless of interface boundary position A . However, 6.794 is not always the best choice. Let $\lambda = S_1^* I$, the convergence is proved faster than 6.794 by numerical test when $A = 1/8$. Recall in Section 4.1, it is found that the convergence rate is determined by $\Delta s_1(\omega)\Delta s_2(\omega)$. Similar to the definition of Δs_i , the convergence rate of the algorithm as shown in Eq.(4.15)

is determined by:

$$\begin{aligned}\Delta\sigma_1(\omega)\Delta\sigma_2(\omega) &:= (k\pi \coth(k\pi A) - \sigma_2)(k\pi \coth(k\pi(B-A)) - \sigma_1), \\ &= (\hat{S}_1(\omega) - \sigma_2)(\hat{S}_2(\omega) - \sigma_1), \quad (\omega := k\pi, k = 1, 2, \dots).\end{aligned}\tag{4.20}$$

By calculating the value of $\Delta\sigma_1(\omega)\Delta\sigma_2(\omega)$ with various A , the convergence rate could be compared. Fig.4.12 shows the comparison between $\lambda = 6.794$ case and $\lambda = S_1^*I$ case against several selections of A . The figure shows that when $A = 1/8$, S_1^*I case line is slightly below 6.794 case line, which means the convergence of the former could be slightly faster than the latter. As for $A = 3/8$ and $A = 7/8$ cases, the lines of S_1^*I case are much above those of 6.794 case, which suggests that the convergence of 6.794 case would be better than its counterpart, especially for $A = 7/8$ case.

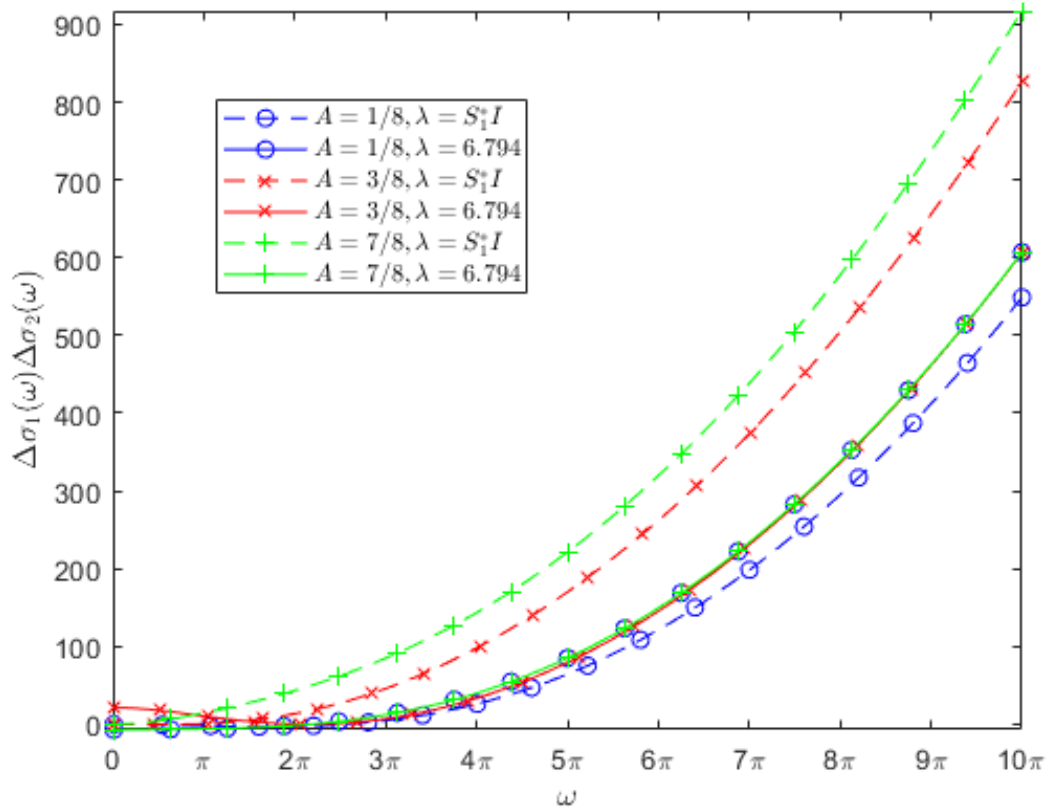


Figure 4.12. Profile of $\Delta\sigma_1\Delta\sigma_2$. In Lui's case, $\lambda = \sigma_1 = \sigma_2 = 6.794$. In otherwise case, $\lambda = \sigma_1 = \sigma_2 = S_1^*I$

A numerical test is performed on the two algorithms to solve a Poisson's equation with $A = 1/8, 3/8$ and $7/8$. The test case is the same as Subsection 4.1.2 except for the convergence criterion, which follows what Lui used in his paper:

$$\max(\|e_1^n\|_\infty, \|e_2^n\|_\infty) < 10^{-4}.\tag{4.21}$$

The numerical test result is summarised in Table 4.2, which shows consistent result with Fig.4.12. Firstly, $\lambda = S_1^*I$ is better than 6.794 in $A = 1/8$ case and worse in other 2 cases. Secondly, given $\lambda = S_1^*I$, $A = 1/8$ case has the fastest convergence, and is followed by $3/8$

and $7/8$ respectively. In addition, the difference between $\lambda = S_1^*I$ and S_1I is related to the boundary effect, which will be elaborated in Section 4.4. The convergence history of the test is displayed in Fig.4.13 to 4.15.

Table 4.2. Convergence iterations of Lui's algorithm in application to a Poisson's equation.

A	$1/8$	$3/8$	$7/8$
$\lambda = 6.794$	5	8	6
$\lambda = S_iI$	4	8	15
$\lambda = S_i^*I$	4	10	22

If let $\lambda = S_2^*I$ or S_2I , the algorithm could be expected to perform equally well in $A = 7/8$ case (and better than $\lambda = 6.794$ case) as $\lambda = S_1^*I$ or S_1I in $A = 1/8$ case because the algorithm is symmetric in terms of A . Here is the elaboration: recall Eq.(4.20) determines the convergence rate:

$$\begin{aligned}
\Delta\sigma_1(\omega)\Delta\sigma_2(\omega) &= (\hat{S}_1(\omega) - \sigma_2)(\hat{S}_2(\omega) - \sigma_1), \\
&= (\hat{S}_1(\omega) - S_1^*I)(\hat{S}_2(\omega) - S_1^*I), \quad (\lambda = \sigma_l = \sigma_r = S_1^*I,) \\
&= (\hat{S}_1^{\frac{1}{8}}(\omega) - \frac{1}{A})(\hat{S}_2^{\frac{1}{8}}(\omega) - \frac{1}{A}), \quad (A = \frac{1}{8},) \\
&= (\hat{S}_2^{\frac{7}{8}}(\omega) - \frac{1}{B-A})(\hat{S}_1^{\frac{7}{8}}(\omega) - \frac{1}{B-A}), \quad (A = \frac{7}{8},) \\
&= (\hat{S}_2^{\frac{7}{8}}(\omega) - S_2^*I)(\hat{S}_1^{\frac{7}{8}}(\omega) - S_2^*I), \quad (A = \frac{7}{8}).
\end{aligned} \tag{4.22}$$

Note the superscript (n) of $\hat{S}_i^n(\omega)$ means $\hat{S}_i(\omega)$ at $A = n$.

It shows that when $A = 1/8$, the convergence rate of $\lambda = S_1^*I$ is equal to that of $\lambda = S_2^*I$ when $A = 7/8$. Compared with RR-SP0, letting $\sigma_l = \sigma_r = S_i^*I$ ($i = 1, 2$) could be expected to result in slower convergence because single $\sigma_{l(r)}$ could not be close to both S_1^*I and S_2^*I at the same time but RR-SP0 could do with 2 different σ . This proves the advantage of RR-SP0 over Lui's single-value $\sigma(\lambda)$ algorithm (conventional RR algorithm).

4.4 Boundary effect analysis of RR-SP0

This section discusses the influence of the so-called 'boundary effect' on the convergence rate of RR-SP0 algorithm. The boundary effect could boost the convergence compared with the case of S_iI treated as constant, but the boost intensity varies with A . This section quantifies the boundary effect in an estimated way and propose an easy empirical method to evaluate the boundary effect.

4.4.1 Modelling of boundary effect

The boundary effect arises from the existence of the boundary on x direction. It makes the unit function I terminate at boundary and the value of S_iI differ from what it should have been

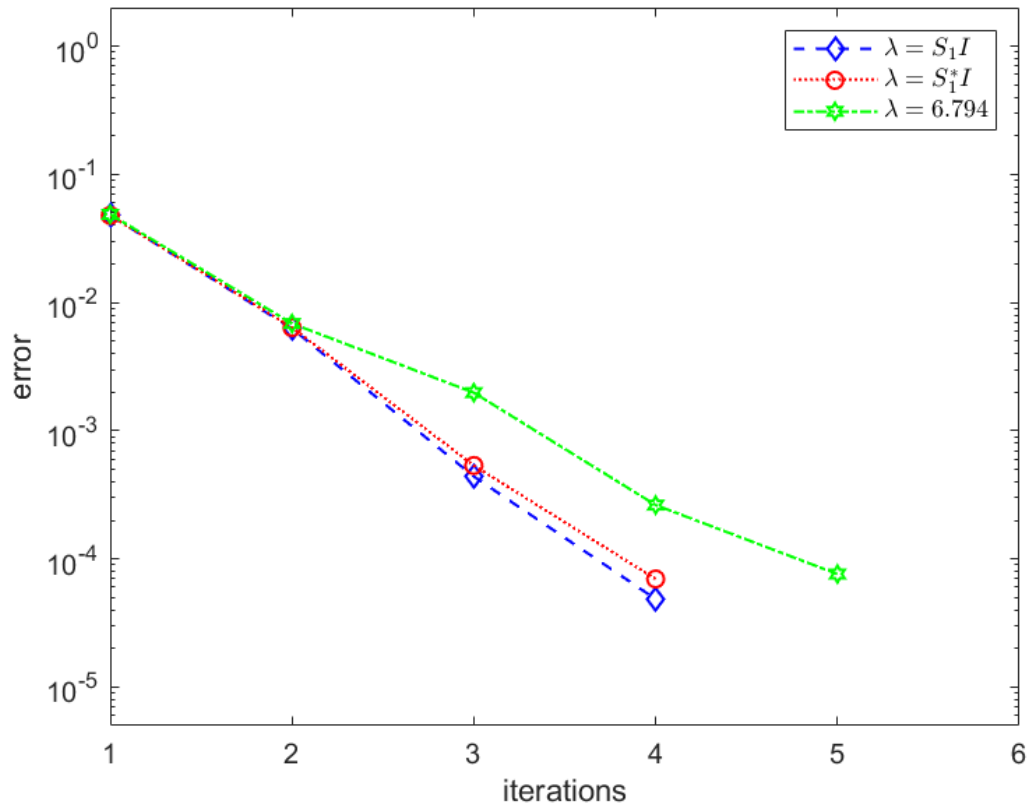


Figure 4.13. Convergence history of Lui's algorithm in application to solving a Poisson's equation with three different λ , $A = 1/8$.

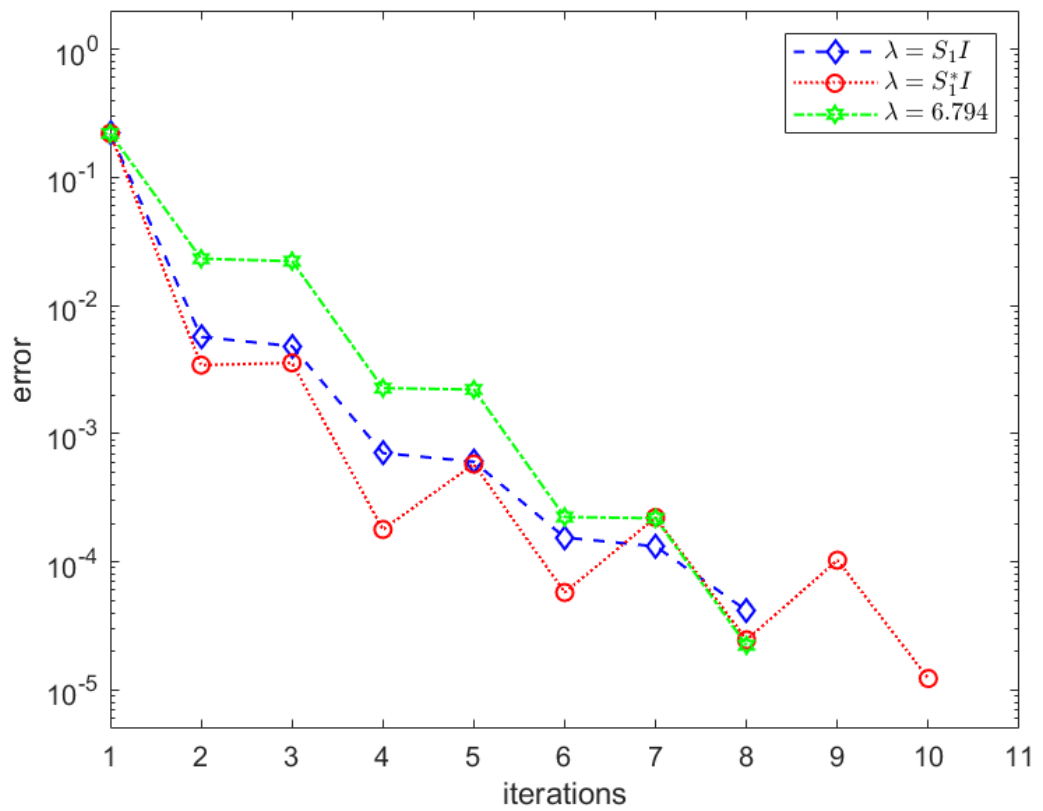


Figure 4.14. Convergence history of Lui's algorithm in application to solving a Poisson's equation with three different λ , $A = 3/8$.

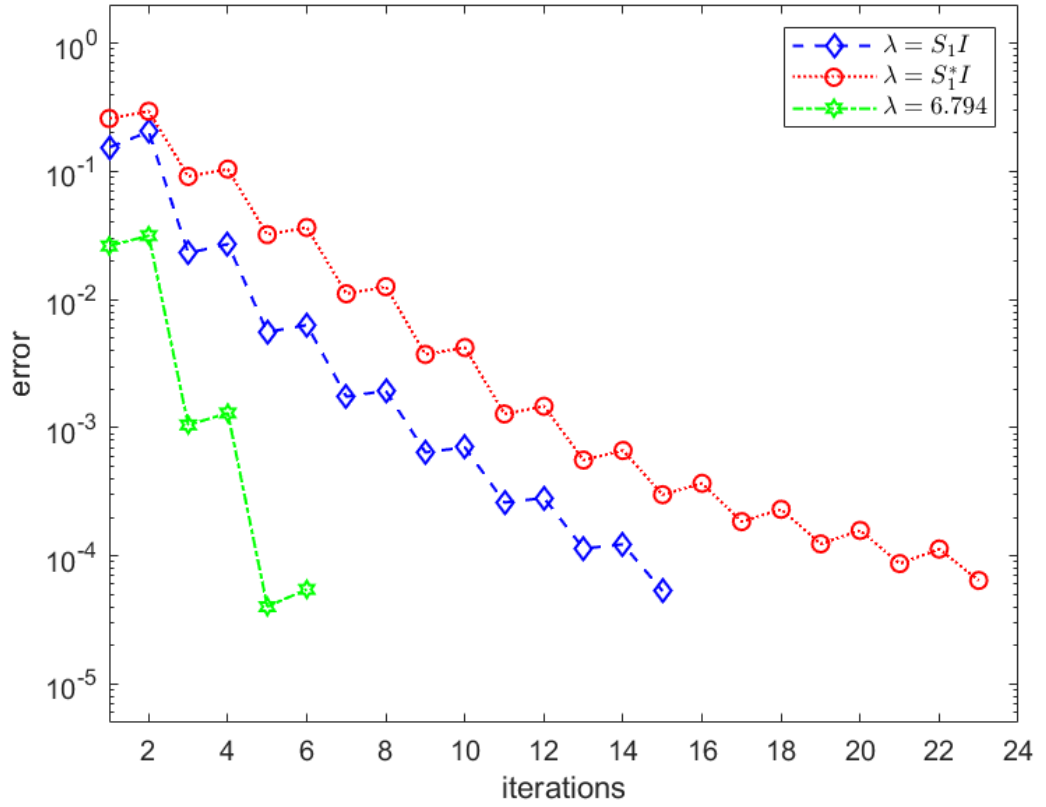


Figure 4.15. Convergence history of Lui's algorithm in application to solving a Poisson's equation with three different λ , $A = 7/8$.

if I had no boundary. The difference resulted by the boundary effect is illustrated in Fig.4.5, where two 'pikes' could be found at both ends of the figures. In otherwise case, the profile of $S_i I$ is a constant. In previous convergence rate calculation, $S_i I$ is taken as a constant because the 'pike function' is difficult to quantify. However, the method might be over-simplified in predicting the convergence, as can be seen from Table 4.1 that the result between $S_i I$ and $S_i^* I$ could vary much especially for middle-valued A cases. The idea to fix the simplification is to fit the 'pike function' with a model function and consider $S_i I$ as the sum of the constant function and the 'pike function':

$$S_i I(x) = S_i^* I + P_i(x),$$

where $P_i(x)$ represents the 'pike function'.

The following exponential function is used to model $P_i(x)$:

$$P_i(x) := \alpha e^{-\beta(x-\sigma_l)} + \alpha e^{\beta(x-\sigma_r)}, \quad (4.23)$$

where $x \in [\sigma_l, \sigma_r]$ and $0 < \sigma_l, \sigma_r < 1$.

The free parameters $(\alpha, \beta, \sigma_l, \sigma_r)$ of the model equation could be determined by fitting $S_i I(x)$ profile. The plotting of $S_i I(x)$ requires the calculation of $S_i I$ in advance, which is part of the pre-calculation of implementing RR-SP0 algorithm. Therefore, there is no extra computing

cost incurred.

Take Fourier transform on $S_i I$: (j is imaginary unit)

$$\mathcal{F}[S_i I] = \hat{S}_i(0) + f_r(\omega) + f_i(\omega)j,$$

where f_r and f_i are the real and imaginary part of Fourier transform of $P_i(x)$ respectively, and they could be obtained analytically:

$$f_r(\omega) := \frac{\alpha}{\beta^2 + \omega^2} [\beta(\cos \sigma_l \omega + \cos \sigma_r \omega) + \omega(\sin \sigma_r \omega - \sin \sigma_l \omega)], \quad (4.24)$$

$$f_i(\omega) := \frac{\alpha}{\beta^2 + \omega^2} [-\beta(\sin \sigma_l \omega + \sin \sigma_r \omega) + \omega(\cos \sigma_r \omega - \cos \sigma_l \omega)]. \quad (4.25)$$

Now consider the (parallel) convergence rate calculation of RR-SP0 again. The standard approach has been performed in Chapter 3 and let us skip to Eqs.(3.42) and (3.43) directly:

$$\begin{aligned} & \hat{e}_{1y}^{n+1}(A) + \mathcal{F}[(S_2^* I + P_2(x)) \cdot e_1^{n+1}(A)] = \hat{\gamma}_1^{n+1}(A), \\ & \rightarrow \hat{e}_{1y}^{n+1}(A) + \hat{S}_2(0) \hat{e}_1^{n+1}(A) + \frac{1}{2\pi} \hat{P}_2(\omega) * \hat{e}_1^{n+1}(A) = \hat{\gamma}_1^{n+1}(A), \\ & \rightarrow C_1(\omega)(\omega \cosh(\omega A) + \hat{S}_2(0) \sinh(\omega A)) + \frac{1}{2\pi} \hat{P}_2(\omega) * \hat{e}_1^{n+1}(A) = \hat{\gamma}_1^{n+1}(A), \\ & \rightarrow C_1(\omega) = \frac{\hat{\gamma}_1^{n+1}(A) - \frac{1}{2\pi} \hat{P}_2(\omega) * \hat{e}_1^{n+1}(A)}{\omega \cosh(\omega A) + \hat{S}_2(0) \sinh(\omega A)}. \\ & -\hat{e}_{2y}^{n+1}(A') + \mathcal{F}[(S_1^* I + P_1(x)) \cdot e_2^{n+1}(A')] = \hat{\gamma}_2^{n+1}(A'), \\ & \rightarrow -\hat{e}_{2y}^{n+1}(A') + \hat{S}_1(0) \hat{e}_2^{n+1}(A') + \frac{1}{2\pi} \hat{P}_1(\omega) * \hat{e}_2^{n+1}(A') = \hat{\gamma}_2^{n+1}(A'), \\ & \rightarrow C_2(\omega)(-\omega \cosh(\omega A') + \hat{S}_1(0) \sinh(\omega A')) + \frac{1}{2\pi} \hat{P}_1(\omega) * \hat{e}_2^{n+1}(A') = \hat{\gamma}_2^{n+1}(A'), \\ & \rightarrow C_2(\omega) = \frac{\hat{\gamma}_2^{n+1}(A') - \frac{1}{2\pi} \hat{P}_1(\omega) * \hat{e}_2^{n+1}(A')}{-\omega \cosh(\omega A') + \hat{S}_1(0) \sinh(\omega A')}. \end{aligned}$$

The solutions to the BVPs are

$$\begin{aligned} \hat{e}_1^{n+1}(\omega, y) &= \frac{(\hat{\gamma}_1^{n+1}(A) - \frac{1}{2\pi} \hat{P}_2(\omega) * \hat{e}_1^{n+1}(A)) \sinh(\omega y)}{\omega \cosh(\omega A) + \hat{S}_2(0) \sinh(\omega A)}, \\ \hat{e}_2^{n+1}(\omega, y) &= \frac{(\hat{\gamma}_2^{n+1}(A) - \frac{1}{2\pi} \hat{P}_1(\omega) * \hat{e}_2^{n+1}(A)) \sinh(\omega(y - B))}{-\omega \cosh(\omega(A - B)) + \hat{S}_1(0) \sinh(\omega(A - B))}. \end{aligned}$$

Particularly, on $y = A$ the (implicit) solution reads

$$\hat{e}_1^{n+1}(\omega, A) = \frac{\hat{\gamma}_1^{n+1}(A) - \frac{1}{2\pi} \hat{P}_2(\omega) * \hat{e}_1^{n+1}(A)}{\omega \coth(\omega A) + \hat{S}_2(0)}, \quad (4.26)$$

$$\hat{e}_2^{n+1}(\omega, A) = \frac{\hat{\gamma}_2^{n+1}(A) - \frac{1}{2\pi} \hat{P}_1(\omega) * \hat{e}_2^{n+1}(A)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0)}. \quad (4.27)$$

Next, expand $\hat{\gamma}_i^{n+1}(A)$:

$$\begin{aligned}\hat{\gamma}_1^{n+1}(A) &:= \hat{e}_{2y}^n(A) + \hat{S}_2(0)\hat{e}_2^n(A) + \frac{1}{2\pi}\hat{P}_2(\omega) * \hat{e}_2^n(A), \\ &= (\omega \coth(\omega(A - B)) + \hat{S}_2(0))\hat{e}_2^n(A) + \frac{1}{2\pi}\hat{P}_2(\omega) * \hat{e}_2^n(A).\end{aligned}$$

$$\begin{aligned}\hat{\gamma}_2^{n+1}(A) &:= -\hat{e}_{1y}^n(A) + \hat{S}_1(0)\hat{e}_1^n(A) + \frac{1}{2\pi}\hat{P}_1(\omega) * \hat{e}_1^n(A), \\ &= (-\omega \coth(\omega A) + \hat{S}_1(0))\hat{e}_1^n(A) + \frac{1}{2\pi}\hat{P}_1(\omega) * \hat{e}_1^n(A).\end{aligned}$$

Substitute the expanded $\hat{\gamma}_i^{n+1}(A)$ into the solution at $y = A$, the solution could be re-arranged as:

$$\begin{aligned}\hat{e}_1^{n+1}(A) + \frac{\frac{1}{2\pi}\hat{P}_2(\omega) * \hat{e}_1^{n+1}(A)}{\omega \coth(\omega A) + \hat{S}_2(0)} &= \frac{\omega \coth(\omega(A - B)) + \hat{S}_2(0)}{\omega \coth(\omega A) + \hat{S}_2(0)}\hat{e}_2^n(A) + \\ &\quad \frac{\frac{1}{2\pi}\hat{P}_2(\omega) * \hat{e}_2^n(A)}{\omega \coth(\omega A) + \hat{S}_2(0)},\end{aligned}\tag{4.28}$$

$$\begin{aligned}\hat{e}_2^{n+1}(A) + \frac{\frac{1}{2\pi}\hat{P}_1(\omega) * \hat{e}_2^{n+1}(A)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0)} &= \frac{-\omega \coth(\omega A) + \hat{S}_1(0)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0)}\hat{e}_1^n(A) + \\ &\quad \frac{\frac{1}{2\pi}\hat{P}_1(\omega) * \hat{e}_1^n(A)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0)}.\end{aligned}\tag{4.29}$$

The above equations are composed of two terms on each side separately, without the second of which the equations are identical to the standard approach performed in Chapter 3:

$$\hat{e}_1^{n+1}(A) = \frac{\omega \coth(\omega(A - B)) + \hat{S}_2(0)}{\omega \coth(\omega A) + \hat{S}_2(0)}\hat{e}_2^n(A) := CR_1(\omega) \cdot \hat{e}_2^n(A),\tag{4.30}$$

$$\hat{e}_2^{n+1}(A) = \frac{-\omega \coth(\omega A) + \hat{S}_1(0)}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0)}\hat{e}_1^n(A) := CR_2(\omega) \cdot \hat{e}_1^n(A).\tag{4.31}$$

It is the second term on both sides of Eqs.(4.28) and (4.29) that changes the convergence rate. It is complicated to calculate the exact convolution, so some approximations need to be made. First consider the convolution in the left extra term:

$$\begin{aligned}\hat{P}_i(\omega) * \hat{e}_j^{n+1}(A) &= \hat{P}_i(\omega) * (\theta_j(\omega)CR_j(\omega)\hat{e}_i^n(A)), \quad i \neq j, \\ &= X_j(\omega) \cdot \hat{P}_i(\omega) * \hat{e}_i^n(A).\end{aligned}$$

Note it is assumed that $\hat{e}_i^{n+1}(A) = \theta_i(\omega)CR_1(\omega)\hat{e}_j^n(A)$, where the boundary effect is represented by a complex function $\theta_i(\omega)$. If $|\theta_i| < 1$, the boundary effect boosts the convergence (as numerical result shows, see Table 4.1), otherwise the boundary effect makes the convergence slower. In addition, it is assumed that $|X_i(\omega)| < 1$ because certainly $|\hat{e}_i^{n+1}(A)| < |\hat{e}_j^n(A)|$, and as a result the convolution of the former with $\hat{P}_i(\omega)$ is smaller

than the convolution of the latter. $X_i(\omega)$ is the nonlinear reflection of $\theta_i(\omega)CR_1(\omega)$ out of the convolution calculation and remains unknown.

Next, move the second term of the left side in Eqs.(4.28) and (4.29) to the right side and re-arrange:

$$\begin{aligned}\hat{e}_1^{n+1}(A) &= \frac{\omega \coth(\omega(A-B)) + \hat{S}_2(0)}{\omega \coth(\omega A) + \hat{S}_2(0)} \hat{e}_2^n(A) + \frac{\frac{1-X_1(\omega)}{2\pi} \hat{P}_2(\omega) * \hat{e}_2^n(A)}{\omega \coth(\omega A) + \hat{S}_2(0)}, \\ \rightarrow \frac{\hat{e}_1^{n+1}(A)}{\hat{e}_2^n(A)} &= \frac{\omega \coth(\omega(A-B)) + \hat{S}_2(0)}{\omega \coth(\omega A) + \hat{S}_2(0)} + \frac{\frac{1-X_1(\omega)}{2\pi} \frac{\hat{P}_2(\omega) * \hat{e}_2^n(A)}{\hat{e}_2^n(A)}}{\omega \coth(\omega A) + \hat{S}_2(0)}.\end{aligned}\quad (4.32)$$

$$\begin{aligned}\hat{e}_2^{n+1}(A) &= \frac{-\omega \coth(\omega A) + \hat{S}_1(0)}{-\omega \coth(\omega(A-B)) + \hat{S}_1(0)} \hat{e}_1^n(A) + \frac{\frac{1-X_2(\omega)}{2\pi} \hat{P}_1(\omega) * \hat{e}_1^n(A)}{-\omega \coth(\omega(A-B)) + \hat{S}_1(0)}, \\ \rightarrow \frac{\hat{e}_2^{n+1}(A)}{\hat{e}_1^n(A)} &= \frac{-\omega \coth(\omega A) + \hat{S}_1(0)}{-\omega \coth(\omega(A-B)) + \hat{S}_1(0)} + \frac{\frac{1-X_2(\omega)}{2\pi} \frac{\hat{P}_1(\omega) * \hat{e}_1^n(A)}{\hat{e}_1^n(A)}}{-\omega \coth(\omega(A-B)) + \hat{S}_1(0)}.\end{aligned}\quad (4.33)$$

The first term on the right side represents the standard convergence rate ($CR_i(\omega)$) obtained from simplified calculation in Chapter 3, and the second terms represents the boundary effect. Comparing the two terms, it could be found that the two terms have identical denominators. Meanwhile, the real part of $X_i(\omega)$ should be negative (since $CR_i(\omega)$ is negative, see Fig.4.16) and $|X_i| < 1$, so it could be estimated that $0 < |1 - X_i(\omega)| < 2$ thus $|\frac{1-X_i}{2\pi}| < 1$. Therefore, the effect is actually dominated by

$$\frac{\hat{P}_i(\omega) * \hat{e}_i^n(A)}{\hat{e}_i^n(A)}.\quad (4.34)$$

4.4.2 Model study of boundary effect

Recall in the convergence analysis of RR-SP0, it is assumed that the interface value $e_i^n(x)$ is made up of polynomials up to order three, and Eqs.(4.2)-(4.5) show the Fourier transform of the polynomials. The profile of the Fourier transform is illustrated in Fig.4.17 and 4.18. Meanwhile, the Fourier transform of $P_i(x)$ is given by Eqs.(4.24) and (4.25), and is illustrated in Figs.4.19 and 4.20.

Figs.4.17 and 4.18 show that the Fourier transform of polynomials of different orders is highly similar, especially when ω is high (thus Fourier transform of even higher order polynomials is also similar) and they could be fitted by two models:

$$g_r := \mathcal{F}[e_i^n(x, A)]_r \approx \frac{L_r}{\omega^{N_r} + 1} \cos(\omega + \omega_0^r); \quad (4.35)$$

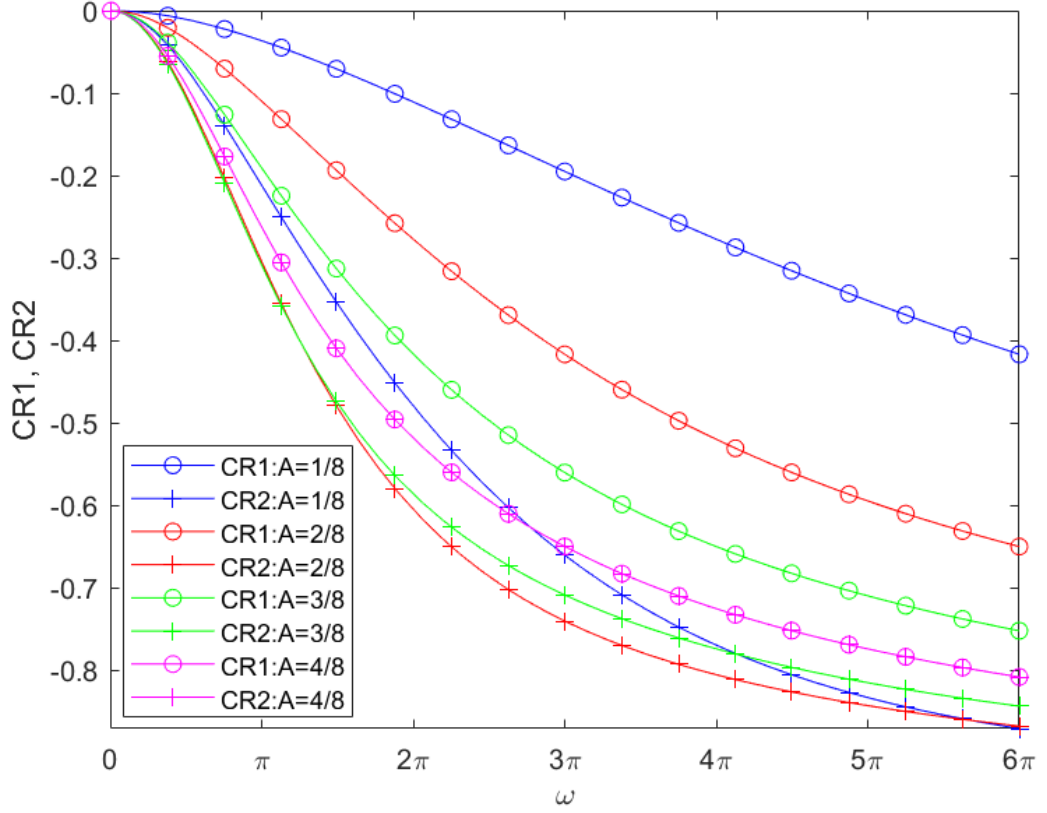


Figure 4.16. Profile of $CR_i(\omega)$, which is always non-positive and decreasing with rising ω .

$$g_i(\omega) := \begin{cases} \mathcal{F}[e_i^n(x, A)]_i \approx \frac{L_i}{\omega^{N_i+1}} (\cos(\xi\omega + \omega_0^i) - 1), & \text{if order} = 0, \\ \mathcal{F}[e_i^n(x, A)]_i \approx \frac{L_i}{\omega^{N_i+1}} \cos(\xi\omega + \omega_0^i). & \text{if order} \neq 0. \end{cases} \quad (4.36)$$

The free parameters are $L_r, L_i, N_r, N_i, \xi, \omega_0^r$ and ω_0^i . $0 < L_r, L_i, 0 < N_r, N_i, 0 < \xi$ and $\pi/2 > |\omega_0^r|, |\omega_0^i|$. By changing them the model could be adjusted to fit the real lines of different order. Actually, the models are simplified version of Eqs.(4.2)-(4.5). As can be seen, the Fourier transform is either sine or cosine function, and the function magnitude declines by the order of ω^{-N} where N depends on the order of polynomials. The combination of sine and cosine means that the model equations should have an initial angle ω_0 when the models are constructed by merely cosine function.

Next consider the modelling of Eqs.(4.24) and (4.25). The original functions are a mixture of *sine* and *cosine* functions and the magnitude declines approximately by order of $\frac{\alpha\beta}{\beta^2+\omega^2}$ or $\frac{\alpha\omega}{\beta^2+\omega^2}$. The figures show that the functions could be fitted by the following models:

$$\begin{aligned} f_r(\omega) &\approx \frac{\alpha\beta}{\beta^2 + \omega^2} \cos(\omega + \omega_{0p}^r), \\ &\approx \frac{\alpha}{\beta} \frac{l_r}{1 + \omega^{n_r}} \cos(\omega + \omega_{0p}^r). \end{aligned} \quad (4.37)$$

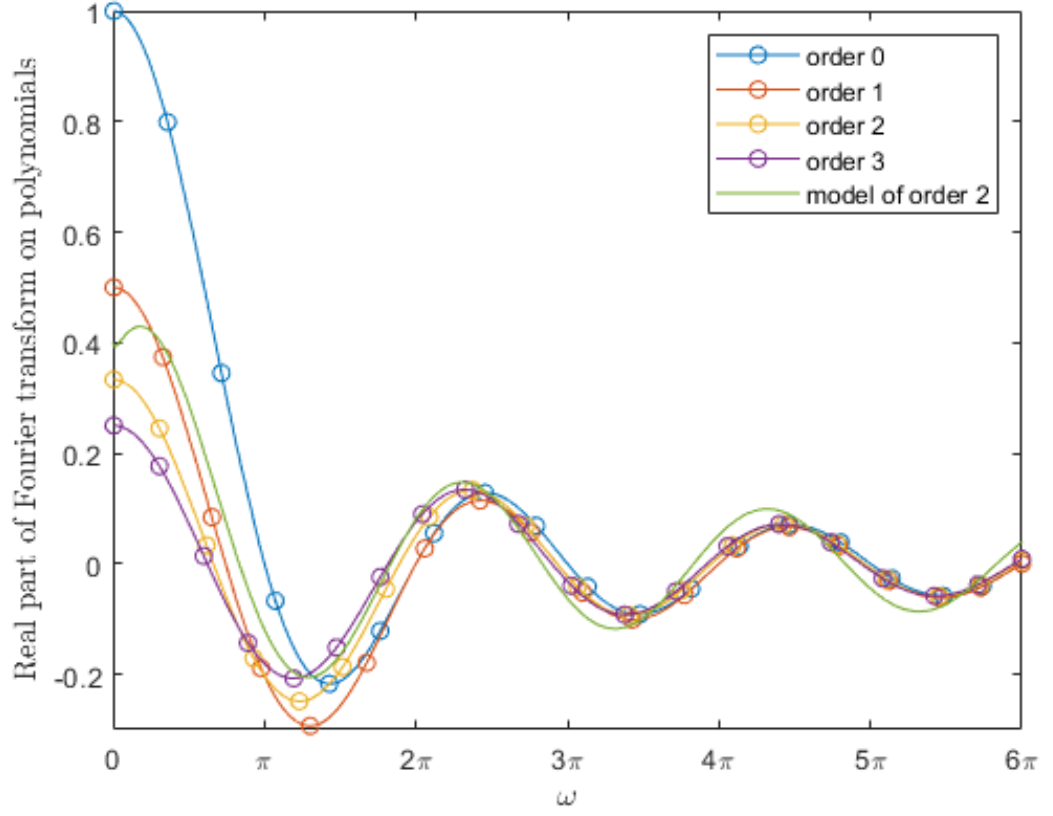


Figure 4.17. Fourier transform of polynomials, real part. One model line is plotted for the case of order 2 according to Eq.(4.35). The parameters are: $L_r = 0.8$, $N_r = 0.75$, $\omega_0^r = -\pi/3$.

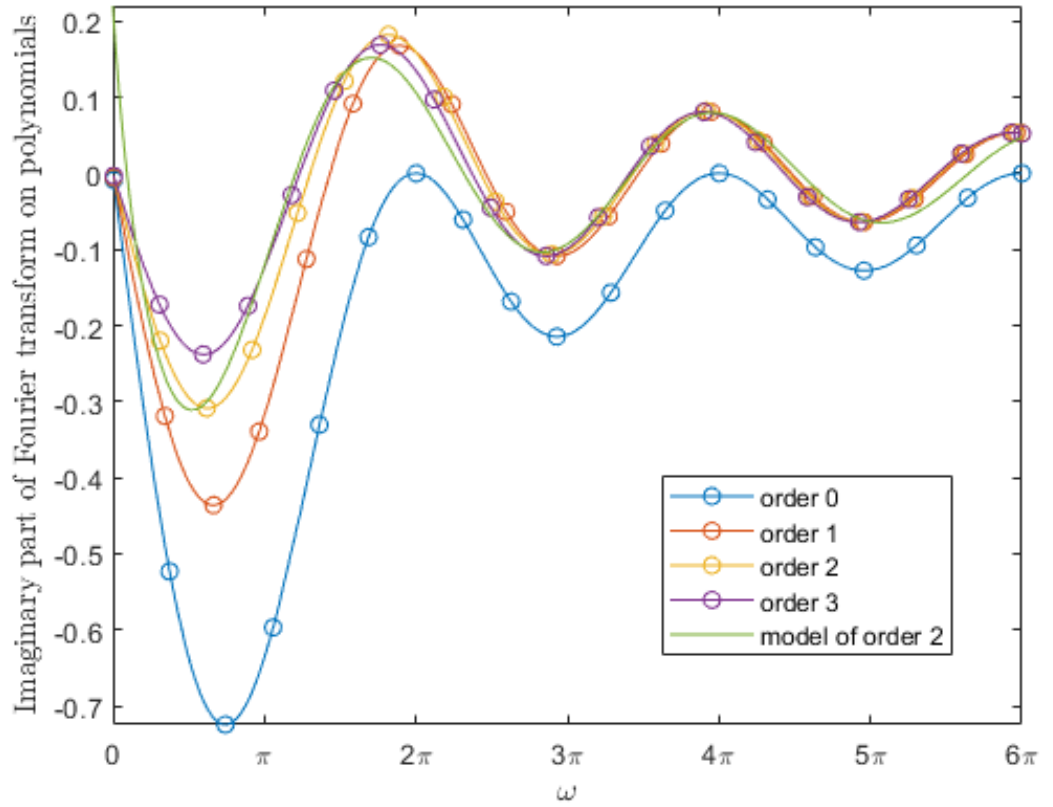


Figure 4.18. Fourier transform of polynomials, imaginary part. One model line is plotted for the case of order 2 according to Eq.(4.36). The parameters are: $L_i = 0.85$, $N_i = 0.9$, $\omega_0^i = 5\pi/12$, $\xi = 0.9$.

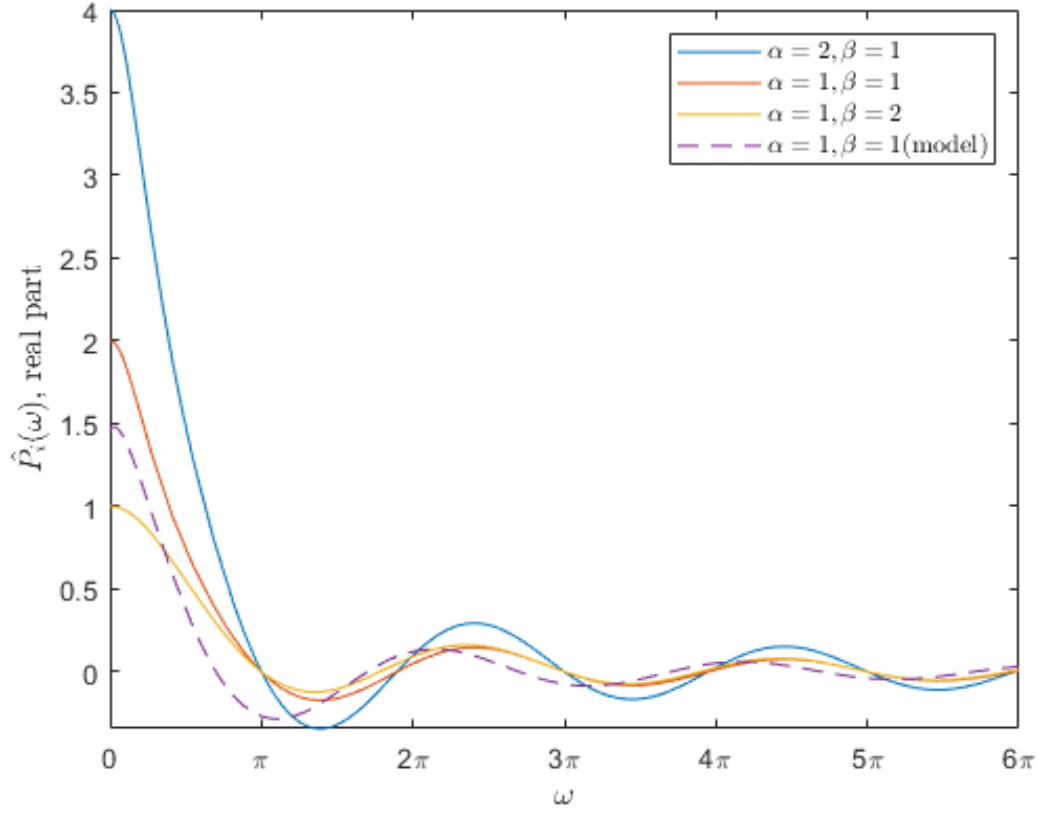


Figure 4.19. Fourier transform of $P_i(x)$, real part. The first three lines are plotted according to Eq.(4.24). In addition, one model function for $\alpha = 1, \beta = 1$ case is also plotted according to Eq.(4.37) with the following parameters: $l_r = 1.8, n_r = 1.3, \omega_{0p}^r = -\pi/5$. It fits the original function well.

$$\begin{aligned} f_i(\omega) &\approx \frac{\alpha\omega}{\beta^2 + \omega^2}(\cos(\omega + \omega_{0p}^i) - 1), \\ &\approx \frac{\alpha}{\beta} \frac{l_i}{1 + \omega^{n_i}}(\cos(\omega + \omega_{0p}^i) - 1). \end{aligned} \quad (4.38)$$

The free parameters of above models are $l_r, l_i, n_r, n_i, \omega_{0p}^r$ and ω_{0p}^i . $0 < l_r, l_i, 0 < n_r, n_i$ and $\pi/2 > |\omega_{0p}^r|, |\omega_{0p}^i|$. The merit of the models is that α and β are separated from ω , which would bring some convenience for the following calculation. In addition, one profile of the model is plotted against the original profile in Figs.4.19 and 4.20 for $(\alpha = 1, \beta = 1)$ case. The figures show that the modelled function fits well the original function.

Having obtained the model functions of $\hat{P}_i(\omega)$ and $\hat{e}_i^n(A)$, let us consider the calculation of Eq.(4.34). Denote $\hat{P}_i(\omega) = f_r(\omega) + f_i(\omega)j$ and $\hat{e}_i^n(A) = g_r(\omega) + g_i(\omega)j$. Note that $\hat{P}_i(\omega)$ (so are f_r, f_i) and $\hat{e}_i^n(A)$ (so are g_r, g_i) are both assumed to be defined in $(0, \infty)$.

$$\begin{aligned} \frac{\hat{P}_i(\omega) * \hat{e}_i^n(A)}{\hat{e}_i^n(A)} &= \frac{g_r(f_r * g_r - f_i * g_i)}{g_r^2 + g_i^2} + \frac{g_i(-f_i * g_r - f_r * g_i)}{g_r^2 + g_i^2} + \\ &\quad \left(\frac{g_i(f_i * g_i - f_r * g_r)}{g_r^2 + g_i^2} + \frac{g_r(f_i * g_r + f_r * g_i)}{g_r^2 + g_i^2} \right) j. \end{aligned} \quad (4.39)$$

The exact calculation is complex but since f_r, f_i, g_r and g_i are simple and uniform in expression, the calculation is similar in every part and the overall calculation could be largely

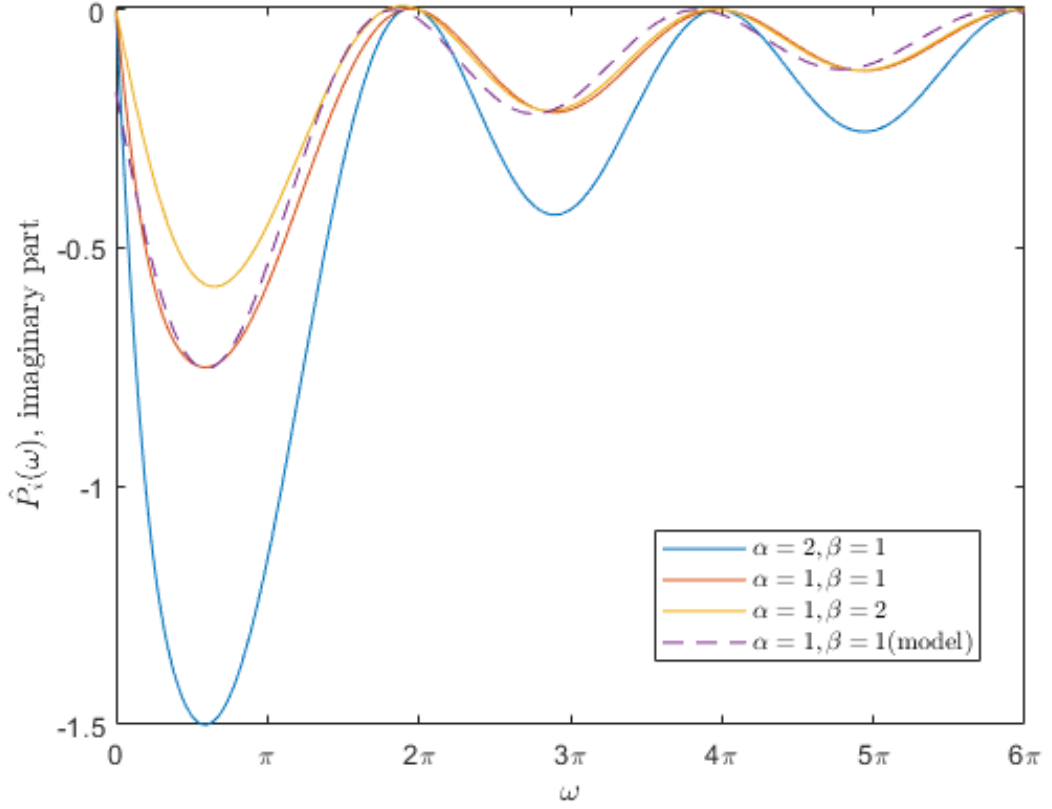


Figure 4.20. Fourier transform of $P_i(x)$, imaginary part. The first three lines are plotted according to Eq.(4.25). In addition, one model function for $\alpha = 1, \beta = 1$ case is also plotted according to Eq.(4.38) with the following parameters: $l_i = 1.3, n_i = 1.1, \omega_{0p}^i = \pi/6$. It fits the original function well.

simplified. In addition, because α and β are separated from ω , they could be moved out from convolution calculation. A sample is given below.

$$\begin{aligned} f_r(\omega) * g_r(\omega) &= \int_0^\omega \frac{\alpha}{\beta} \frac{l_r}{1 + \tau^{n_r}} \cos(\tau + \omega_{0p}^r) \frac{L_r}{(\omega - \tau)^{N_r} + 1} \cos(\xi(\omega - \tau) + \omega_0^r) d\tau \\ &= \frac{\alpha\omega}{\beta} [fg]_{rr}(\omega^*), \quad (0 \leq \omega^* \leq \omega), \end{aligned}$$

where $[fg]_{rr}(x) := \frac{l_r}{1+x^{n_r}} \cos(x + \omega_{0p}^r) \frac{L_r}{(\omega-x)^{N_r}+1} \cos(\xi(\omega-x) + \omega_0^r)$.

Instead of the mean-value estimation, $f_r(\omega) * g_r(\omega)$ could be estimated in a more precise way:

$$\begin{aligned} f_r(\omega) * g_r(\omega) &= \frac{\alpha}{\beta} \int_0^\omega \frac{l_r}{1 + \tau^{n_r}} \cos(\tau + \omega_{0p}^r) \frac{L_r}{(\omega - \tau)^{N_r} + 1} \cos(\omega - \tau + \omega_0^r) d\tau, \\ &< \frac{\alpha l_r}{\beta} \frac{L_r}{1 + \tau^{N_r}} \int_0^\omega \cos(\tau + \omega_{0p}^r) \cos(\omega - \tau + \omega_0^r) d\tau, \\ &= \frac{\alpha l_r}{\beta} \frac{L_r}{1 + \tau^{N_r}} \left(\frac{\sin(\omega_{0p}^r - \omega_0^r + \omega)}{4} + \frac{\sin(\omega_0^r - \omega_{0p}^r + \omega)}{4} + \right. \\ &\quad \left. \frac{\omega \cos(\omega_{0p}^r + \omega_0^r + \omega)}{2} \right), \\ &:= \frac{\alpha}{\beta} \frac{L_r l_r}{1 + \tau^{N_r}} [FG]_{rr}(\omega). \end{aligned}$$

Similar estimation could be made about $f_i * g_i$:

$$\begin{aligned}
f_i(\omega) * g_i(\omega) &= \frac{\alpha}{\beta} \int_0^\omega \frac{l_i}{1 + \tau^{N_i}} \cos(\tau + \omega_{0p}^i) \frac{L_i}{(\omega - \tau)^{N_i} + 1} (\cos(\xi(\omega - \tau) + \omega_0^i) - 1) d\tau, \\
&< \frac{\alpha l_i}{\beta} \frac{L_i}{1 + \tau^{N_i}} \int_0^\omega \cos(\tau + \omega_{0p}^i) (\cos(\xi(\omega - \tau) + \omega_0^i) - 1) d\tau, \\
&= \frac{\alpha l_i}{\beta} \frac{L_i}{1 + \tau^{N_i}} \left(\frac{\sin(\omega_{0p}^i - \omega_0^i + \xi\omega)}{2(\xi - 1)} + \frac{\sin(\omega_{0p}^i - \omega_0^i + \omega)}{2(\xi - 1)} - \right. \\
&\quad \left. \sin(\omega_0^i + \xi\omega) + \sin(\omega_0^i) + \frac{\sin(\omega_{0p}^i + \omega_0^i + \xi\omega)}{2(\xi - 1)} - \frac{\sin(\omega_{0p}^i + \omega_0^i + \omega)}{2(\xi - 1)} \right), \\
&:= \frac{\alpha}{\beta} \frac{L_i l_i}{1 + \tau^{N_i}} [FG]_{ii}(\omega).
\end{aligned}$$

Therefore, the first term on the right-hand-side of Eq.(4.39) could be estimated:

$$\begin{aligned}
&\frac{g_r(f_r * g_r - f_i * g_i)}{g_r^2 + g_i^2} \\
&= \frac{\alpha}{\beta} \frac{g_r \left(\frac{L_r l_r}{1 + \tau^{N_r}} [FG]_{rr}(\omega) + \frac{L_i l_i}{1 + \tau^{N_i}} [FG]_{ii}(\omega) \right)}{g_r^2 + g_i^2} \\
&\approx \frac{\alpha l}{\beta} \frac{\cos(\omega + \omega_0^r) ([FG]_{rr}(\omega) - [FG]_{ii}(\omega))}{\cos(\omega + \omega_0^r)^2 + \cos(\xi\omega + \omega_0^i)^2} \\
&= \frac{\alpha l}{\beta} \Phi_1(\omega), \quad l \approx (l_r + l_i)/2,
\end{aligned} \tag{4.40}$$

where

$$\begin{aligned}
\Phi_1(\omega) &= [\cos(\omega_0^r + \omega) \left(\frac{\sin(\omega_{0p}^r - \omega_0^r + \omega)}{4} + \frac{\sin(\omega_0^r - \omega_{0p}^r + \omega)}{4} - \frac{\sin(\omega_{0p}^i - \omega_0^i + \omega)}{2(\xi + 1)} - \right. \\
&\quad \left. \frac{\sin(\omega_0^i - \omega_{0p}^i + \xi\omega)}{2(\xi + 1)} + \frac{\sin(\omega_0^i + \xi\omega) - \sin(\omega_0^i)}{\xi} + \frac{\omega \cos(\omega_{0p}^r + \omega_0^r + \omega)}{2} - \right. \\
&\quad \left. \frac{\sin(\omega_{0p}^i + \omega_0^i + \xi\omega)}{2(\xi - 1)} + \frac{\sin(\omega_{0p}^i + \omega_0^i + \omega)}{2(\xi - 1)} \right)] / (\cos(\omega_0^r + \omega)^2 + \cos(\omega_0^i + \xi\omega)^2).
\end{aligned}$$

When ω is not too small (for instance, $\omega > \pi$) and $\xi \approx 1$, $\Phi_1(\omega)$ could be estimated as follows:

$$\Phi_1(\omega) \approx \frac{\cos(\omega_0^r + \omega) (V_1 + \frac{\omega \cos(\omega_{0p}^r + \omega_0^r + \omega)}{2})}{\cos(\omega_0^r + \omega)^2 + \cos(\omega_0^i + \omega)^2}, \tag{4.41}$$

where V_1 represents a function whose magnitude is no more than three. The magnitude of V_1 should be far less than $\frac{\omega \cos(\omega_{0p}^r + \omega_0^r + \omega)}{2}$ when ω is not too small. The expression of V_1 is:

$$\begin{aligned}
V_1(\omega) &:= \frac{\sin(\omega_{0p}^r - \omega_0^r + \omega)}{4} + \frac{\sin(\omega_0^r - \omega_{0p}^r + \omega)}{4} - \frac{\sin(\omega_{0p}^i - \omega_0^i + \omega)}{4} - \\
&\quad \frac{\sin(\omega_0^i - \omega_{0p}^i + \omega)}{4} + \sin(\omega_0^i + \omega) - \sin(\omega_0^i).
\end{aligned}$$

The same estimation could be performed on the other three terms of right-hand-side of Eq.(4.39), which should all yield to similar result to above one. The calculation process hereby

is skipped and the results are given directly:

$$\begin{aligned} & \frac{g_i(f_i * g_r + f_r * g_i)}{g_r^2 - g_i^2} \\ & \approx \frac{\alpha l}{\beta} \Phi_2(\omega), \quad l \approx (l_r + l_i)/2, \end{aligned} \quad (4.42)$$

$$\begin{aligned} & \frac{g_i(f_i * g_i - f_r * g_r)}{g_r^2 + g_i^2} \\ & \approx \frac{\alpha l}{\beta} \Phi_3(\omega), \quad l \approx (l_r + l_i)/2, \end{aligned} \quad (4.43)$$

$$\begin{aligned} & \frac{g_r(f_i * g_r + f_r * g_i)}{g_r^2 + g_i^2} \\ & \approx \frac{\alpha l}{\beta} \Phi_4(\omega), \quad l \approx (l_r + l_i)/2, \end{aligned} \quad (4.44)$$

where $\Phi_i(\omega)$ could be approximated as follows if ω is not too small and $\xi \approx 1$:

$$\Phi_2(\omega) \approx \frac{-\cos(\omega_0^i + \omega)(V_2 + \frac{\omega \cos(\omega_{0p}^i + \omega_0^r + \omega)}{2})}{\cos(\omega_0^r + \omega)^2 + \cos(\omega_0^i + \omega)^2}, \quad (4.45)$$

$$\Phi_3(\omega) \approx \frac{-\cos(\omega_0^i + \omega)(V_3 + \frac{\omega \cos(\omega_{0p}^r + \omega_0^r + \omega)}{2})}{\cos(\omega_0^r + \omega)^2 + \cos(\omega_0^i + \omega)^2}, \quad (4.46)$$

$$\Phi_4(\omega) \approx \frac{\cos(\omega_0^r + \omega)(V_4 + \frac{\omega \cos(\omega_{0p}^i + \omega_0^r + \omega)}{2})}{\cos(\omega_0^r + \omega)^2 + \cos(\omega_0^i + \omega)^2}. \quad (4.47)$$

Note $V_i(\omega)$, ($i = 2, 3, 4$) are functions whose magnitude is no more than three. The expression of V_i ($i = 2, 3, 4$) is similar to V_1 and trivial thus would not be displayed.

Therefore,

$$\frac{\hat{P}_i(\omega) * \hat{e}_i^n(A)}{\hat{e}_i^n(A)} = \frac{\alpha l}{\beta} ((\Phi_1 + \Phi_2) + (\Phi_3 + \Phi_4)j). \quad (4.48)$$

In Figs.4.17-4.20, two model lines for order-two polynomial and a pike function ($\alpha = 1, \beta = 1$) are given respectively. The parameters used are summarised in the figure captions. Using the model lines parameters, $1.5(\Phi_1 + \Phi_2)$ and $1.5(\Phi_3 + \Phi_4)$ could be computed and the result is illustrated in Figure 4.21. The function of $V_i(\omega)$ is assumed to be $3 \sin(\omega)$ in the plotting. The figure shows that both $1.5|\Phi_1 + \Phi_2|$ and $1.5|\Phi_3 + \Phi_4|$ are mostly no greater than $y = |\omega|$ if $\omega > \pi$.

The parameters of $\omega_0^r, \omega_0^i, \omega_{0p}^r$ and ω_{0p}^i only shift the plot by some angle but could not change the magnitude. In addition, the profile of $\Phi_i(\omega)$ is actually largely influenced by $V_i(\omega)$. In Fig.4.21, $V_i = 3 \sin(\omega)$. If $V_i = 3 \cos(\omega)$, the profile is as shown in Fig.4.22. The profile

of $(\Phi_3 + \Phi_4)$ especially change much. It is hard to know the exact expression of $V_i(\omega)$, but whatever its real expression, it is important to know that the 1.5 times of the $(\Phi_{1(3)} + \Phi_{2(4)})$ remains less than $y = |\omega|$.

Recall in Eqs.(4.32) and (4.33), the boundary effect term $\frac{1-X_i(\omega)}{2\pi} \frac{\hat{P}_j(\omega)\hat{e}_j^n(A)}{\hat{e}_j^n(A)}$ is supposed to reduce the standard convergence rate CR_i (defined in Eq.(4.30)). As mentioned before, $0 < |1 - X_i| < 2$, the boundary effect term is roughly equal to (but less than) $\frac{\alpha l}{\beta \pi} ((\Phi_1 + \Phi_2) + (\Phi_3 + \Phi_4)j)$. Meanwhile, the numerator of CR_i : $\hat{S}_1(0) - \omega \coth(\omega A)$ or $\hat{S}_2(0) - \omega \coth(\omega(B - A))$ is nearly equal to $-\omega$ because $\coth(\omega A)$ (or $\coth(\omega(B - A))$) is approximately equal to 1 if ω is not too small. Therefore, the boundary effect intensity depends on the value of $|\frac{\alpha l}{\beta \pi}|$. To have appropriate convergence improvement, at least the value should be close to 1.5 or even higher (because it was assumed $\frac{|1-X_i|}{2} = 1$ but it is actually less than 1). By contrast, If $|\frac{\alpha l}{\beta \pi}|$ is far less than 1.5 or even close to 0, the boundary effect could hardly implement any impact on the convergence.

Although X_i is not known more than its magnitude approximation, it could be deduced that $(\Phi_3 + \Phi_4)$ is the real part of the boundary effect that offsets the numerator (roughly equal to $-\omega$) of CR_i , and $(\Phi_1 + \Phi_2)$ is the imaginary part of the boundary effect. The assumption could be validated by both Figs.4.21 and 4.22: at $\omega = \pi, 2\pi, \dots$, $(\Phi_3 + \Phi_4)$ is roughly equal to ω and $(\Phi_1 + \Phi_2)$ is roughly equal to zero in magnitude. Therefore, the former mostly offsets the numerator of CR_i in real part and the latter adds very minor value in imaginary part. Overall, the convergence rate is reduced by the boundary effect and it complies with $|X_i| < 1$. In otherwise case ($(\Phi_1 + \Phi_2)$ is the real part and $(\Phi_3 + \Phi_4)$ is the imaginary part), the overall convergence rate would be increased that leads to the wrong conclusion that $|X_i| > 1$.

To summarise, the procedures of evaluating the boundary effect are the following:

1. First, a pike function $P_i(x)$ needs to be fitted by an exponential-function model, see Eq.(4.23). There are four parameters to determine: α, β, σ_l and σ_r .
2. Once the four parameters are determined, the Fourier transform of $P_i(x)$ could be obtained, see Eqs.(4.24) and (4.25).
3. Then plot the figures of $\mathcal{F}P_i$ according to Eqs.(4.24) and (4.25) to estimate l_r and l_i . l_r is approximately equal to the largest wave magnitude (occurs at around $\omega = 0$) divided by (α/β) . l_i is approximately equal to the largest wave magnitude (occurs at around $\omega = \pi/2$) divided by (α/β) and multiplied by $(1 + \frac{\pi^N}{2}) \approx 2.5$ where N is not known and taken as 1. The wave profile should enable an estimation on whether N is larger or less than 1 so $(1 + \frac{\pi^N}{2})$ could be estimated more precisely. Note the exact model formula (see Eqs.(4.37) and (4.38)) for $\mathcal{F}P_i$ needs not be known because only l_r and l_i are concerned. The two parameter could be estimated from figures directly.
4. Lastly, take $l = (l_r + l_i)/2$ and calculate $(\alpha l)/(\beta \pi)$.

If $(\alpha l)/(\beta \pi)$ is close to 1.5 or slightly larger, the boundary effect could boost the convergence well; if the value is small (close to 0), the boundary effect has limited effect on the

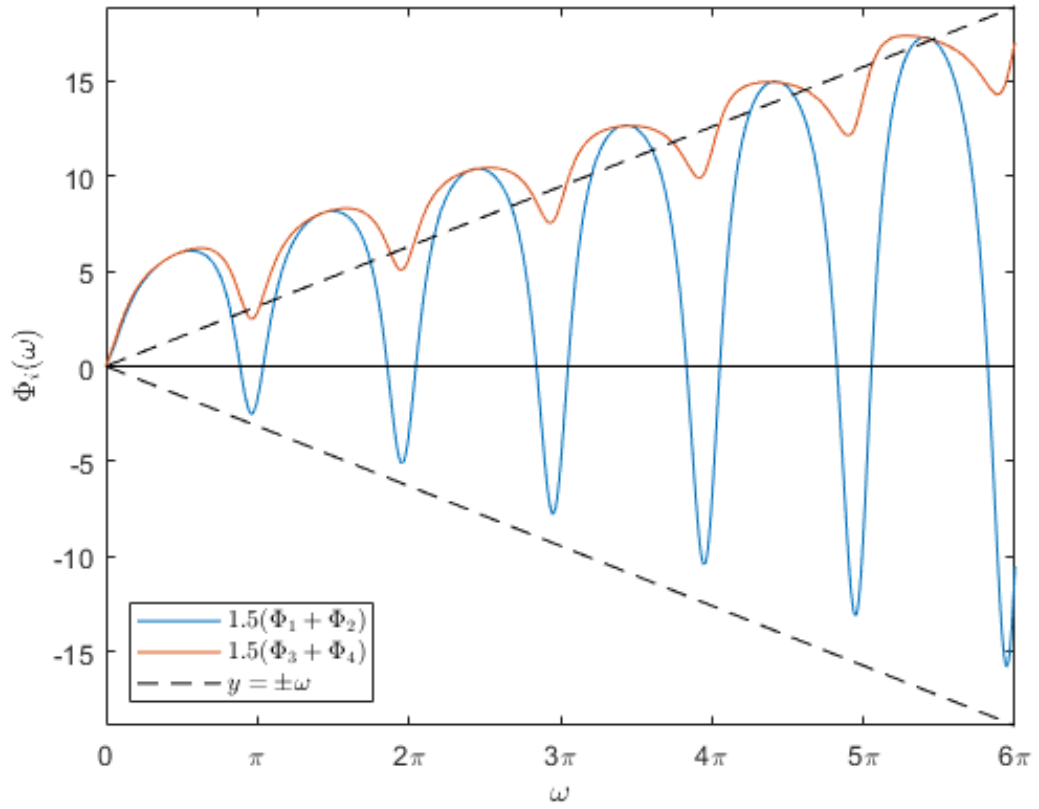


Figure 4.21. Profile of $y_1 = 1.5(\Phi_1 + \Phi_2)$ and $y_2 = 1.5(\Phi_3 + \Phi_4)$ as a function of ω . The magnitude of both functions are mainly less than $y = \pm\omega$ when $\omega > \pi$. $V_i(\omega)$ is assumed to be $3 \sin(\omega)$.

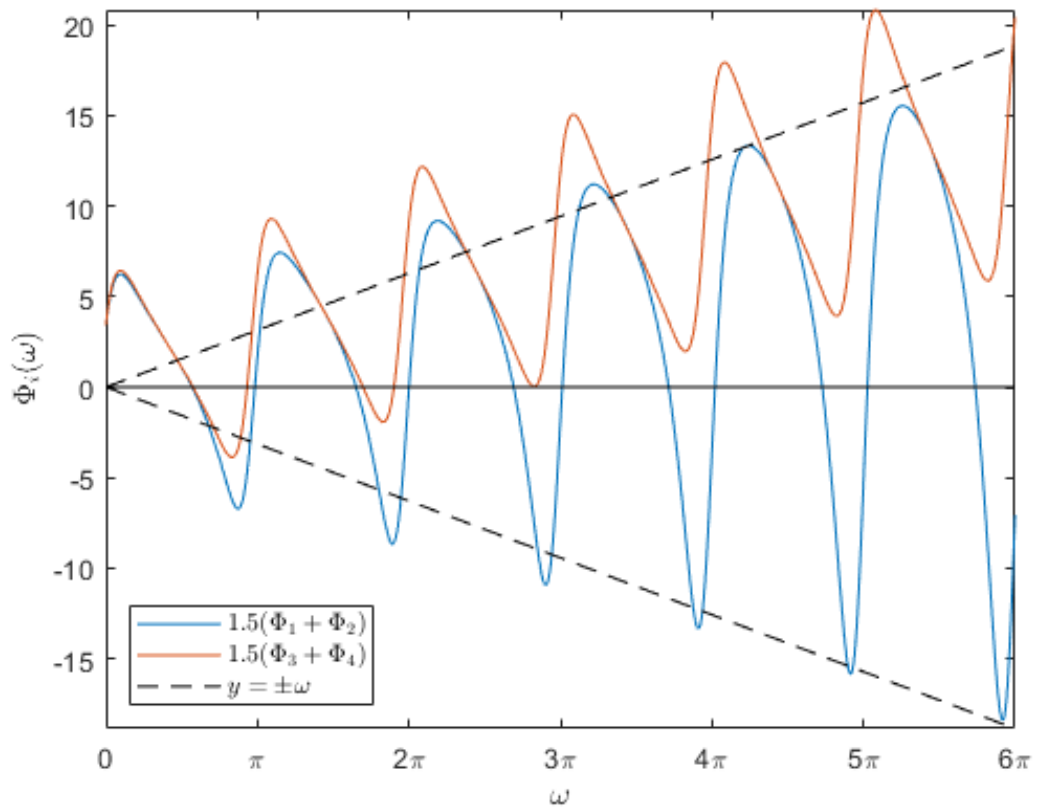


Figure 4.22. Profile of $y_1 = 1.5(\Phi_1 + \Phi_2)$ and $y_2 = 1.5(\Phi_3 + \Phi_4)$ as a function of ω . The magnitude of both functions are mainly less than $y = \pm\omega$ when $\omega > \pi$. $V_i(\omega)$ is assumed to be $3 \cos(\omega)$.

convergence. In one word, given $(\alpha l)/(\beta \pi)$ is not too large (for instance, > 2), the bigger the value the better the convergence could be improved.

4.4.3 Test case to evaluate boundary effect

In Section 4.1, a Poisson's equation is solved by RR-SP0 with $S_i I$ taken as constant and real value respectively. The result is shown in Table 4.1. The difference between $S_i I$ and $S_i^* I$ varies with A : the middle-valued A (such as $4/8$) has greater difference than those large/small-valued A , which is certainly due to the boundary effect. To see the difference, the profile of $S_i I$ applied to the Poisson's equation with $A = 1/8, 3/8$ and $7/8$ is given in Figs.4.23 and 4.24. Note the two profiles are symmetric in terms of A so that $S_1 I|_{A=1/8}$ is nearly equal to $S_2 I_{A=7/8}$. Therefore, only $A = 1/8$ case will be studied next.

According to Eq.(4.23), the modelled pike functions for $A = 1/8$ and $3/8$ cases are summarised in Table 4.3. It shows that for $A = 1/8$ case, one model is enough to fit the pike function of $S_1 I$, while 2 models are needed to fit the pike function of $S_2 I$. It means the pike function $P_2(x)$ is represented by:

$$P_2(x) = [40e^{-36(x-\frac{1}{64})} + 40e^{-36(x-\frac{63}{64})}] + [23e^{-8(x-\frac{5}{64})} + 23e^{-8(x-\frac{59}{64})}].$$

For $A = 3/8$ case, two-modelled fitting is always needed for both $S_1 I$ and $S_2 I$ pike functions. Next, having known α, β, σ_l and σ_r , the Fourier transform of $P_i(x)$ could be obtained and their profile is given in Figs.4.25 and 4.26. Note that the case for $\alpha = 40, \beta = 36$ is not plotted because it highly resembles the case of $\alpha = 34, \beta = 49$. The two cases could be considered almost equal. It could be estimated that $l \approx 1, 2$ respectively for $(\alpha, \beta) = (34, 49), (23, 8)$ cases. The final step is to calculate $\frac{\alpha l}{\beta \pi}$, which yields to the result that the value for $(34, 49)$ case is far less than 1.5 (about 0.2) while the value for $(23, 8)$ case is about 1.8. Clearly $(23, 8)$ case is much more effective in improving the convergence. Since $A = 3/8$ has $(23, 8)$ -type pike function in both $S_1 I$ and $S_2 I$, its convergence is better improved than $A = 1/8$ case that only has $(23, 8)$ -type pike function in $S_2 I$. This is consistent with the numerical test result.

The case of $(23, 8)$ represents the pike functions that are high in ratio of α/β . According to Eq.(4.23), α determines the peak height of the function and β determines the rise/fall speed of the function. Graphically, high ratio means the 'pike' is high and wide, while low ratio means the 'pike' is low and thin. It could be observed from Fig.4.5 as well: $P_1(x)$ is approximately 40 in height and 0.25 in width; $P_2(x)$ is approximately 30 in height and 0.125 in width. Correspondingly, the former is fitted by a high-ratio model and a low-ratio model, and the latter is fitted by a low-ratio model only. Therefore, the most direct way to evaluate the boundary effect is to plot $S_i I$ profile and observe the shape of the pike function. 'High and wide' means the boundary effect is strong while 'low and thin' means the opposite. The criterion of 'high' or 'low', 'wide' or 'thin' is relative. Plotting all profiles of various A in one figure creates the criterion automatically.

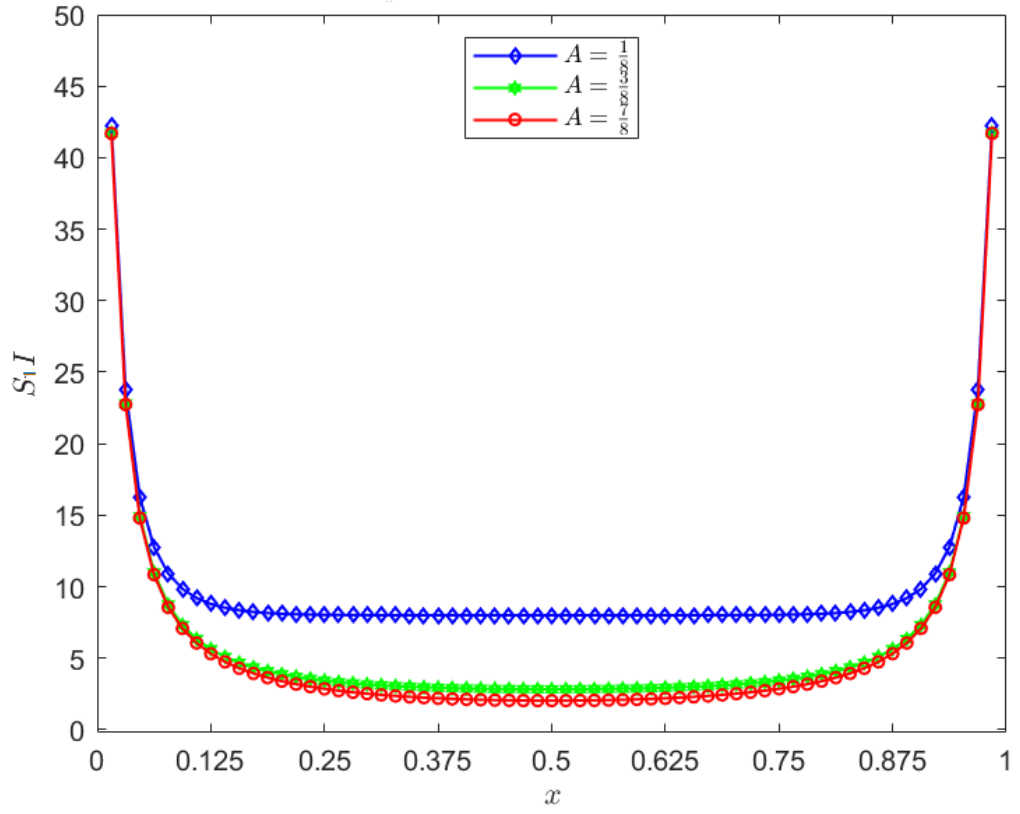


Figure 4.23. Profile of S_1I applied to a Poisson's equation, $A = \frac{1}{8}, \frac{3}{8}$ and $\frac{7}{8}$.

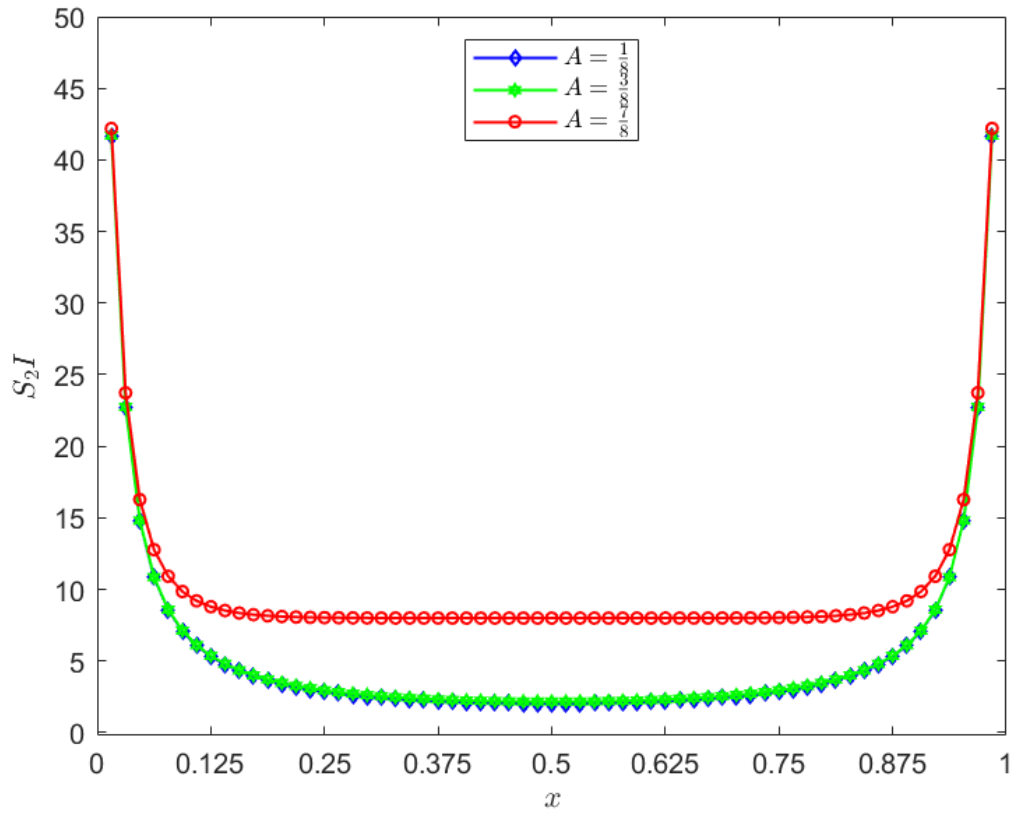


Figure 4.24. Profile of S_2I applied to a Poisson's equation, $A = \frac{1}{8}, \frac{3}{8}$ and $\frac{7}{8}$.

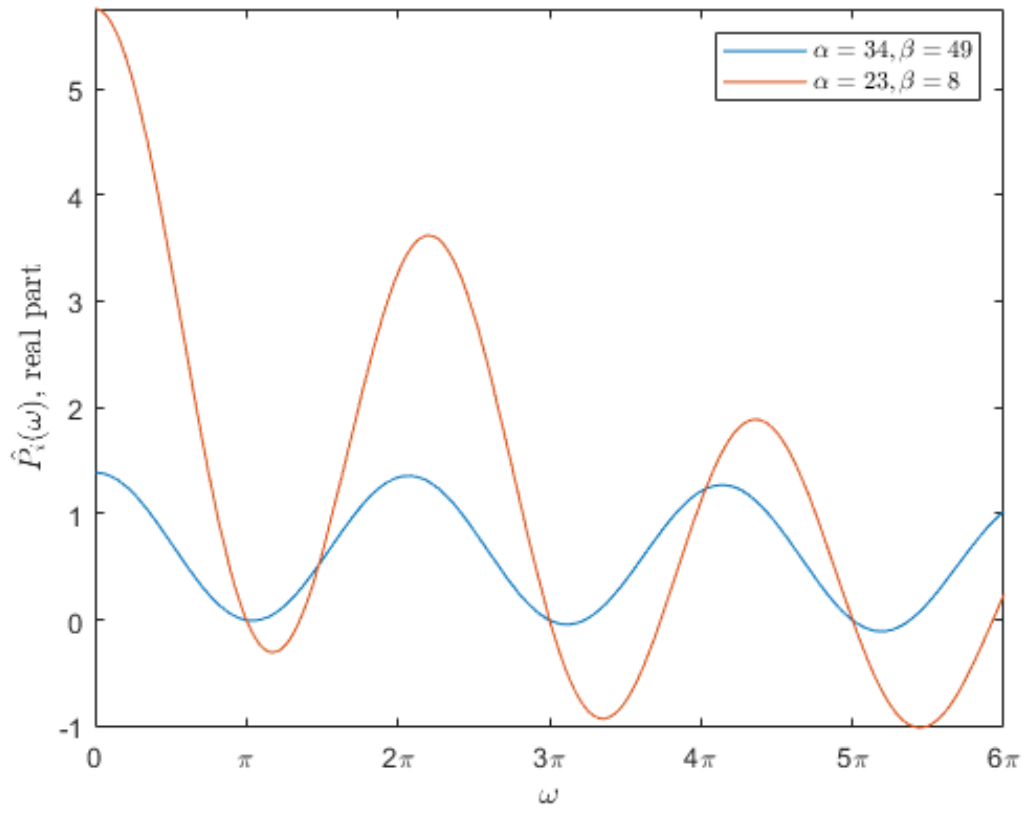


Figure 4.25. Profile of $\hat{P}_i(\omega)$, real part.

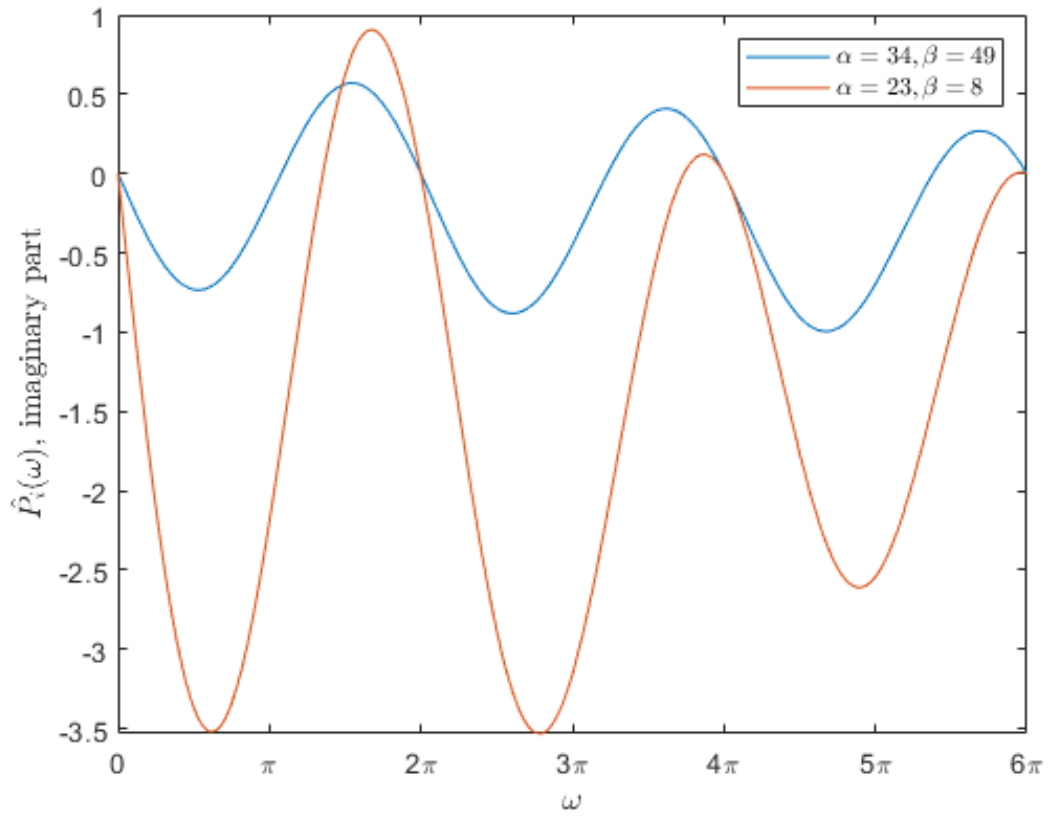


Figure 4.26. Profile of $\hat{P}_i(\omega)$, imaginary part.

Table 4.3. Parameters of the modelled pike functions.

	$S_1 I$				$S_2 I$			
	α	β	σ_l	σ_r	α	β	σ_l	σ_r
$A = 1/8$	34	49	$\frac{1}{64}$	$\frac{63}{64}$	40	36	$\frac{1}{64}$	$\frac{63}{64}$
					23	8	$\frac{5}{64}$	$\frac{59}{64}$
$A = 3/8$	40	36	$\frac{1}{64}$	$\frac{63}{64}$	40	36	$\frac{1}{64}$	$\frac{63}{64}$
					23	8	$\frac{5}{64}$	$\frac{59}{64}$

4.5 Convergence rate calculation of RR-SP1 and RR-SP2 with boundary effect

This section performs the convergence rate calculation of RR-SP1 and RR-SP2 algorithms with considering boundary effect. The so-called boundary effect is defined in the profile of $S_i x$ and $S_i x^2$ apart from that in $S_i I$, and is represented by (two other) pike functions similar to the pike function in $S_i I$. In one word, the calculation method and procedures totally follow those of RR-SP0, and the calculation result is in similar form to that of RR-SP0 as well.

4.5.1 Calculation of RR-SP1 with boundary effect

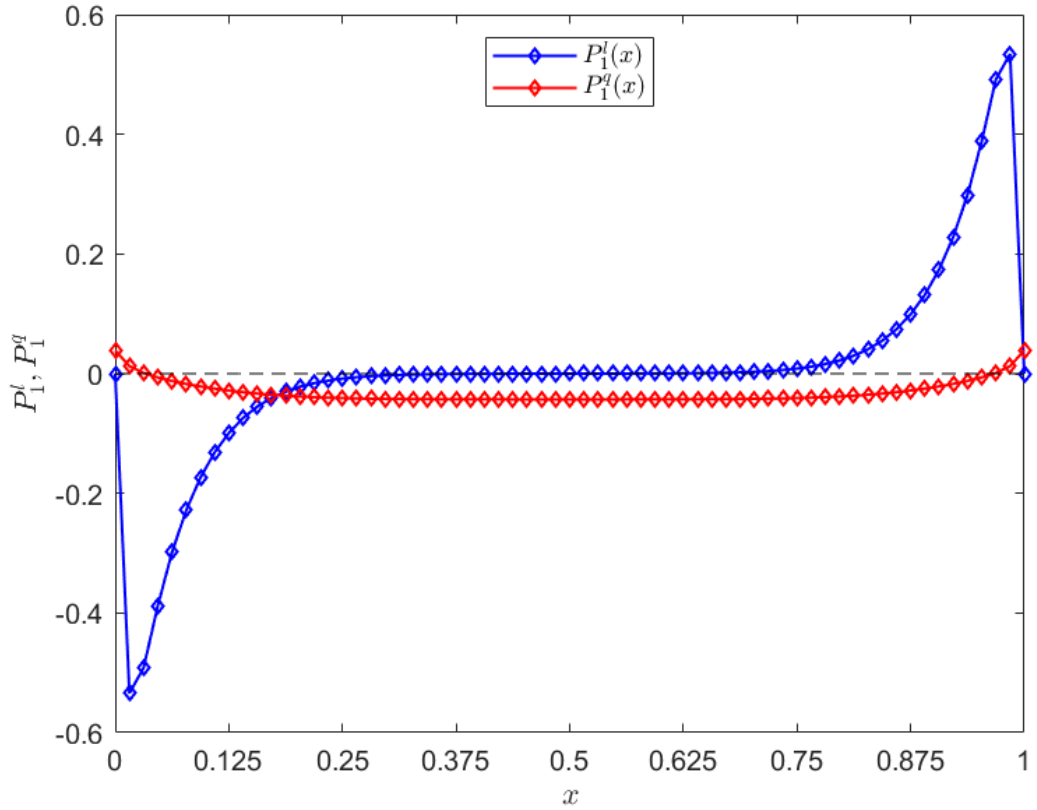


Figure 4.27. Profile of P_1^l and P_1^q in a Poisson test, $A = 1/8$. $P_1^l := S_2 x - S_2 I \cdot x - 0$, $P_1^q = (S_2 x^2 - 2x S_2 x + x^2 S_2 I) - (S_2^* x^2 - x^2 S_2^* I)$. $S_2^* x^2$ is obtained numerically with boundaries in x direction set at far place ($x = \pm 5$).

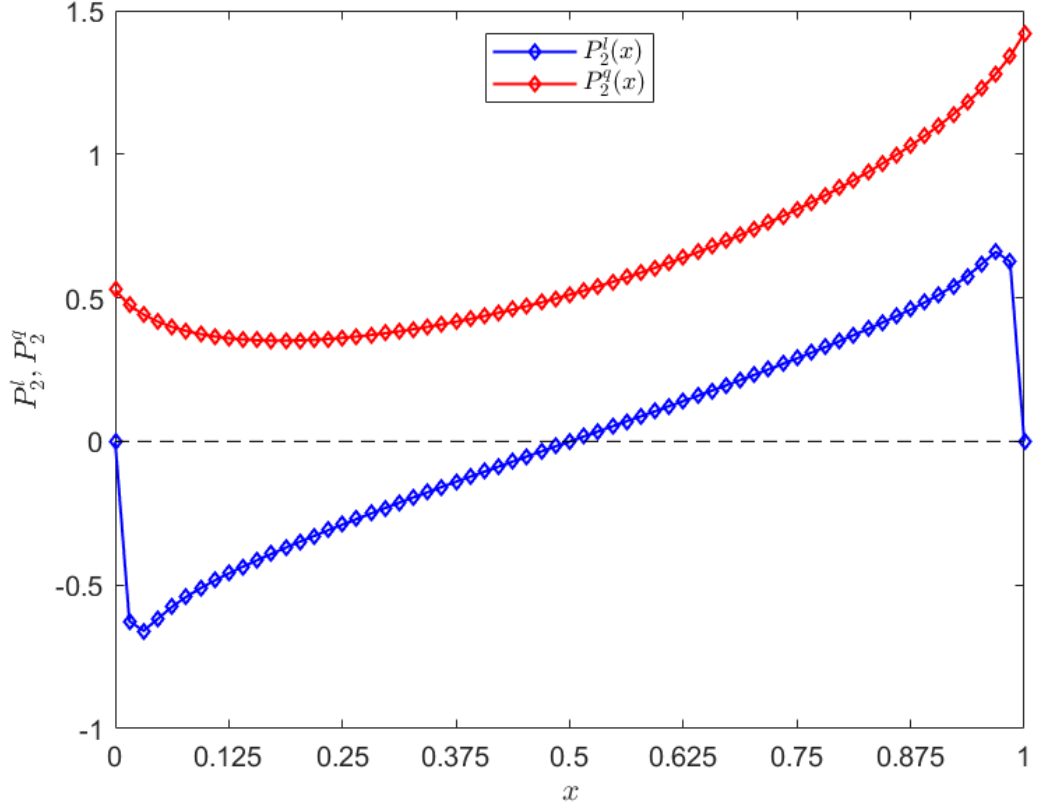


Figure 4.28. Profile of P_2^l and P_2^q in a Poisson test, $A = 1/8$. $P_2^l := S_1x - S_1I \cdot x - 0$, $P_2^q = (S_1x^2 - 2xS_1x + x^2S_1I) - (S_1^*x^2 - x^2S_1^*I)$. $S_1^*x^2$ is obtained numerically with boundaries in x direction set at far place ($x = \pm 5$).

Recall the error BVPs of (parallel) RR-SP1 reads

$$\begin{aligned} e_{1y}^{n+1} + S_2I \cdot e_1^{n+1} + e_{1x}^{n+1} \cdot T_{1l} &= \gamma_1^{n+1} \quad \text{on } \Gamma, \\ -e_{2y}^{n+1} + S_1I \cdot e_2^{n+1} + e_{2x}^{n+1} \cdot T_{2l} &= \gamma_2^{n+1} \quad \text{on } \Gamma, \end{aligned}$$

where

$$\begin{aligned} \gamma_1^{n+1} &:= e_{2y}^n(x, A) + S_2I \cdot e_2^n(x, A) + e_{2x}^n \cdot T_{1l}, \\ \gamma_2^{n+1} &:= -e_{1y}^n(x, A) + S_1I \cdot e_1^n(x, A) + e_{1x}^n \cdot T_{2l}, \\ T_{1l} &:= S_2x - S_2I \cdot x, \\ T_{2l} &:= S_1x - S_1I \cdot x. \end{aligned}$$

In the standard approach, T_{il} is taken as 0 but it is actually not. Therefore, T_{il} could be considered as the sum of 0 and a pike function:

$$T_{il} = 0 + P_j^l(x) \rightarrow P_j^l(x) := T_{il} - 0, \quad (4.49)$$

where $P_j^l(x)$ represents the pike function in T_{il} profile, as shown in Figs.4.27 and 4.28.

Next, consider the convergence rate calculation with $P_i^l(x)$. It is actually the term that contains

T_{il} differs from the standard approach, so the calculation hereby focus on the difference and starts from some intermediate stage:

$$\begin{aligned}
& \hat{e}_{1y}^{n+1}(A) + \mathcal{F}[(S_2^* I + P_2(x))e_1^{n+1}(A)] + \frac{j}{2\pi} \mathcal{F}[P_2^l(x)] * (\omega \hat{e}_1^{n+1}(A)) = \hat{\gamma}_1^{n+1}(A), \\
& \rightarrow \hat{e}_{1y}^{n+1}(A) + \hat{S}_2(0) \hat{e}_1^{n+1}(A) + \frac{1}{2\pi} \hat{P}_2(\omega) * \hat{e}_1^{n+1}(A) + \frac{j}{2\pi} \hat{P}_2^l(\omega) * (\omega \hat{e}_1^{n+1}(A)) = \hat{\gamma}_1^{n+1}(A), \\
& \rightarrow C_1(\omega)(\omega \cosh(\omega A) + \hat{S}_2(0) \sinh(\omega A)) + \frac{1}{2\pi} (\hat{P}_2(\omega) * \hat{e}_1^{n+1}(A) + j \hat{P}_2^l(\omega) * (\omega \hat{e}_1^{n+1}(A))) \\
& = \hat{\gamma}_1^{n+1}(A), \\
& \rightarrow C_1(\omega) = \frac{\hat{\gamma}_1^{n+1}(A) - \frac{1}{2\pi} (\hat{P}_2(\omega) * \hat{e}_1^{n+1}(A)) + j \hat{P}_2^l(\omega) * (\omega \hat{e}_1^{n+1}(A))}{\omega \cosh(\omega A) + \hat{S}_2(0) \sinh(\omega A)}. \\
\\
& - \hat{e}_{2y}^{n+1}(A') + \mathcal{F}[(S_1^* I + P_1(x)) \cdot e_2^{n+1}(A')] + \frac{j}{2\pi} \mathcal{F}[P_1^l(x)] * (\omega \hat{e}_2^{n+1}(A')) = \hat{\gamma}_2^{n+1}(A'), \\
& \rightarrow - \hat{e}_{2y}^{n+1}(A') + \hat{S}_1(0) \hat{e}_2^{n+1}(A') + \frac{1}{2\pi} \hat{P}_1(\omega) * \hat{e}_2^{n+1}(A') + \frac{j}{2\pi} \hat{P}_1^l(\omega) * (\omega \hat{e}_2^{n+1}(A')) = \hat{\gamma}_2^{n+1}(A'), \\
& \rightarrow C_2(\omega)(-\omega \cosh(\omega A') + \hat{S}_1(0) \sinh(\omega A')) + \frac{1}{2\pi} (\hat{P}_1(\omega) * \hat{e}_2^{n+1}(A') + j \hat{P}_1^l(\omega) * (\omega \hat{e}_2^{n+1}(A'))) \\
& = \hat{\gamma}_2^{n+1}(A'), \\
& \rightarrow C_2(\omega) = \frac{\hat{\gamma}_2^{n+1}(A') - \frac{1}{2\pi} (\hat{P}_1(\omega) * \hat{e}_2^{n+1}(A')) + j \hat{P}_1^l(\omega) * (\omega \hat{e}_2^{n+1}(A'))}{-\omega \cosh(\omega A') + \hat{S}_1(0) \sinh(\omega A')}.
\end{aligned}$$

Note $P_i(x)$ still represents the pike function in $S_i I$ profile, as is denoted in Section 4.4, Eq.(4.23).

The solutions to the BVPs are:

$$\begin{aligned}
\hat{e}_1^{n+1}(\omega, y) &= \frac{[\hat{\gamma}_1^{n+1}(A) - \frac{1}{2\pi} (\hat{P}_2(\omega) * \hat{e}_1^{n+1}(A) + j \hat{P}_2^l(\omega) * (\omega \hat{e}_1^{n+1}(A)))] \sinh(\omega A)}{\omega \cosh(\omega A) + \hat{S}_2(0) \sinh(\omega y)}, \\
\hat{e}_2^{n+1}(\omega, y) &= \frac{[\hat{\gamma}_2^{n+1}(A) - \frac{1}{2\pi} (\hat{P}_1(\omega) * \hat{e}_2^{n+1}(A) + j \hat{P}_1^l(\omega) * (\omega \hat{e}_2^{n+1}(A)))] \sinh(\omega(y - B))}{-\omega \cosh(\omega(A - B)) + \hat{S}_1(0) \sinh(\omega(A - B))}.
\end{aligned}$$

Particularly, on $y = A$ the (implicit) solution reads:

$$\hat{e}_1^{n+1}(\omega, A) = \frac{\hat{\gamma}_1^{n+1}(A) - \frac{1}{2\pi} (\hat{P}_2(\omega) * \hat{e}_1^{n+1}(A) + j \hat{P}_2^l(\omega) * (\omega \hat{e}_1^{n+1}(A)))}{\omega \coth(\omega A) + \hat{S}_2(0)}, \quad (4.50)$$

$$\hat{e}_2^{n+1}(\omega, A) = \frac{\hat{\gamma}_2^{n+1}(A) - \frac{1}{2\pi} (\hat{P}_1(\omega) * \hat{e}_2^{n+1}(A) + j \hat{P}_1^l(\omega) * (\omega \hat{e}_2^{n+1}(A)))}{-\omega \coth(\omega(A - B)) + \hat{S}_1(0)}. \quad (4.51)$$

Next, expand $\hat{\gamma}_i^{n+1}(A)$:

$$\begin{aligned}
\hat{\gamma}_1^{n+1}(A) &:= \hat{e}_{2y}^n(A) + \hat{S}_2(0) \hat{e}_2^n(A) + \frac{1}{2\pi} \hat{P}_2(\omega) * \hat{e}_2^n(A) + \frac{j}{2\pi} \hat{P}_2^l(\omega) * (\omega \hat{e}_2^n(A)), \\
&= (\omega \coth(\omega(A - B)) + \hat{S}_2(0)) \hat{e}_2^n(A) + \frac{1}{2\pi} (\hat{P}_2(\omega) * \hat{e}_2^n(A) + j \hat{P}_2^l(\omega) * (\omega \hat{e}_2^n(A))),
\end{aligned}$$

$$\begin{aligned}\hat{\gamma}_2^{n+1}(A) &:= -\hat{e}_{1y}^n(A) + \hat{S}_1(0)\hat{e}_1^n(A) + \frac{1}{2\pi}\hat{P}_1(\omega) * \hat{e}_1^n(A) + \frac{j}{2\pi}\hat{P}_1^l(\omega) * (\omega\hat{e}_1^n(A)), \\ &= (-\omega \coth(\omega A) + \hat{S}_1(0))\hat{e}_1^n(A) + \frac{1}{2\pi}(\hat{P}_1(\omega) * \hat{e}_1^n(A) + j\hat{P}_1^l(\omega) * (\omega\hat{e}_1^n(A))).\end{aligned}$$

Substitute the expanded $\hat{\gamma}_i^{n+1}(A)$ into the solution at $y = A$, the solution could be re-arranged as:

$$\begin{aligned}\hat{e}_1^{n+1}(A) + \frac{\frac{1}{2\pi}(\hat{P}_2(\omega) * \hat{e}_1^{n+1}(A) + j\hat{P}_2^l(\omega) * (\omega\hat{e}_1^{n+1}(A)))}{\omega \coth(\omega A) + \hat{S}_2(0)} = \\ \frac{\omega \coth(\omega(A-B)) + \hat{S}_2(0)}{\omega \coth(\omega A) + \hat{S}_2(0)}\hat{e}_2^n(A) + \frac{\frac{1}{2\pi}(\hat{P}_2(\omega) * \hat{e}_2^n(A) + j\hat{P}_2^l(\omega) * (\omega\hat{e}_2^n(A)))}{\omega \coth(\omega A) + \hat{S}_2(0)}, \\ \hat{e}_2^{n+1}(A) + \frac{\frac{1}{2\pi}(\hat{P}_1(\omega) * \hat{e}_2^{n+1}(A) + j\hat{P}_1^l(\omega) * (\omega\hat{e}_2^{n+1}(A)))}{-\omega \coth(\omega(A-B)) + \hat{S}_1(0)} = \\ \frac{-\omega \coth(\omega A) + \hat{S}_1(0)}{-\omega \coth(\omega(A-B)) + \hat{S}_1(0)}\hat{e}_1^n(A) + \frac{\frac{1}{2\pi}(\hat{P}_1(\omega) * \hat{e}_1^n(A) + j\hat{P}_1^l(\omega) * (\omega\hat{e}_1^n(A)))}{-\omega \coth(\omega(A-B)) + \hat{S}_1(0)}.\end{aligned}$$

Next, move the 2nd term of the left side to the right side and re-arrange:

$$\begin{aligned}\hat{e}_1^{n+1}(A) &= \frac{\omega \coth(\omega(A-B)) + \hat{S}_2(0)}{\omega \coth(\omega A) + \hat{S}_2(0)}\hat{e}_2^n(A) + \\ &\quad \frac{\frac{1-X_1(\omega)}{2\pi}(\hat{P}_2(\omega) * \hat{e}_2^n(A)) + \frac{1-X_1^l(\omega)}{2\pi}(j\hat{P}_2^l(\omega) * (\omega\hat{e}_2^n(A)))}{\omega \coth(\omega A) + \hat{S}_2(0)}, \\ \hat{e}_2^{n+1}(A) &= \frac{-\omega \coth(\omega A) + \hat{S}_1(0)}{-\omega \coth(\omega(A-B)) + \hat{S}_1(0)}\hat{e}_1^n(A) + \\ &\quad \frac{\frac{1-X_2(\omega)}{2\pi}(\hat{P}_1(\omega) * \hat{e}_1^n(A)) + \frac{1-X_2^l(\omega)}{2\pi}(j\hat{P}_1^l(\omega) * (\omega\hat{e}_1^n(A)))}{-\omega \coth(\omega(A-B)) + \hat{S}_1(0)}.\end{aligned}\tag{4.52}$$

Similarly, $X_i^l(\omega)$ is defined as $X_i^l := (\hat{P}_j^l * \hat{e}_i^{n+1}(A))/(\hat{P}_j^l * \hat{e}_j^n(A))$, and it could be estimated that $|X_i^l(\omega)| < 1$ although it remains unknown. The reason is similar to $X_i(\omega)$ in RR-SP0, see the explanation at end of Subsection 4.4.1.

It is the second term on the right-hand-side of above equations that determines the boundary effect in RR-SP1. More precisely, divide both side of above equations by $\hat{e}_j^n(A)$, and it could be seen that the term below actually represents the boundary effect:

$$\frac{\hat{P}_i(\omega) * \hat{e}_i^n(A) + j\hat{P}_i^l(\omega) * (\omega\hat{e}_i^n(A))}{\hat{e}_i^n(A)}.$$

4.5.2 Calculation of RR-SP2 with boundary effect

Recall the error BVPs of (parallel) RR-SP2 reads

$$\begin{aligned} e_{1y}^{n+1} + S_2 I \cdot e_1^{n+1} + e_{1x}^{n+1} \cdot T_{1l} + \frac{1}{2} e_{1xx}^{n+1} \cdot T_{1q} &= \gamma_1^{n+1} \quad \text{on } \Gamma, \\ -e_{2y}^{n+1} + S_1 I \cdot e_2^{n+1} + e_{2x}^{n+1} \cdot T_{2l} + \frac{1}{2} e_{2xx}^{n+1} \cdot T_{2q} &= \gamma_2^{n+1} \quad \text{on } \Gamma, \end{aligned}$$

where

$$\begin{aligned} \gamma_1^{n+1} &:= e_{2y}^n(x, A) + S_2 I \cdot e_2^n(x, A) + e_{2x}^n \cdot T_{1l} + \frac{1}{2} e_{2xx}^n \cdot T_{1q}, \\ \gamma_2^{n+1} &:= -e_{1y}^n(x, A) + S_1 I \cdot e_1^n(x, A) + e_{1x}^n \cdot T_{2l} + \frac{1}{2} e_{1xx}^n \cdot T_{2q}, \\ T_{1l} &:= S_2 x - S_2 I \cdot x, \\ T_{2l} &:= S_1 x - S_1 I \cdot x, \\ T_{1q} &:= S_2 x^2 - 2x S_2 x + x^2 S_2 I, \\ T_{2q} &:= S_1 x^2 - 2x S_1 x + x^2 S_1 I. \end{aligned}$$

Compared with RR-SP1, The additional term $(\frac{1}{2} e_{ixx}^n \cdot T_{iq})$ is unique in RR-SP2, so this term will be paid special attention in this subsection. Recall in the standard approach, T_{iq} is simplified as (the superscript '*' means simplified version):

$$T_{iq}^* := S_j^* x^2 - x^2 S_j^* I,$$

where $S_j^* x^2$ represents S_j applied to x^2 that is defined in $(-\infty, \infty)$.

Define a pike function $P_i^q(x)$ in T_{iq} profile, so that:

$$T_{iq}(x) = T_{iq}^*(x) + P_i^q(x) \rightarrow P_i^q(x) := T_{iq}(x) - T_{iq}^*(x). \quad (4.53)$$

The profile of $P_i^q(x)$ is illustrated in Figs.4.27 and 4.28. Note $S_i^* x^2$ could be obtained numerically: set boundaries in x direction at far place so that the boundary effect is negligible. In this case, the boundaries are set at $x = \pm 5$, while the real boundaries are at $x = 0, 1$.

Next, consider the convergence rate calculation of RR-SP2 with $P_i^q(x)$. Much of the calculation resembles that in RR-SP1 as performed in last subsection (4.5.1). Therefore, without presenting middle calculation, the following is given directly:

$$\begin{aligned} \hat{e}_1^{n+1}(A) + \frac{\frac{1}{2\pi}(\hat{P}_2(\omega) * \hat{e}_1^{n+1}(A) + j\hat{P}_2^l(\omega) * (\omega\hat{e}_1^{n+1}(A)) - \frac{1}{2}\hat{P}_2^q(\omega) * (\omega^2\hat{e}_1^{n+1}(A)))}{\omega \coth(\omega A) + \hat{S}_2(0) + \omega\hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)} = \\ \frac{\omega \coth(\omega(A-B)) + \hat{S}_2(0) + \omega\hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)}{\omega \coth(\omega A) + \hat{S}_2(0) + \omega\hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)} \hat{e}_2^n(A) + \\ \frac{\frac{1}{2\pi}(\hat{P}_2(\omega) * \hat{e}_2^n(A) + j\hat{P}_2^l(\omega) * (\omega\hat{e}_2^n(A)) - \frac{1}{2}\hat{P}_2^q(\omega) * (\omega^2\hat{e}_2^n(A)))}{\omega \coth(\omega A) + \hat{S}_2(0) + \omega\hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)}, \end{aligned}$$

$$\begin{aligned} \hat{e}_2^{n+1}(A) + \frac{\frac{1}{2\pi}(\hat{P}_1(\omega) * \hat{e}_2^{n+1}(A) + j\hat{P}_1^l(\omega) * (\omega\hat{e}_2^{n+1}(A)) - \frac{1}{2}\hat{P}_1^q(\omega) * (\omega^2\hat{e}_2^{n+1}(A)))}{-\omega \coth(\omega(A-B)) + \hat{S}_1(0) + \omega\hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)} = \\ \frac{-\omega \coth(\omega A) + \hat{S}_1(0) + \omega\hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)}{-\omega \coth(\omega(A-B)) + \hat{S}_1(0) + \omega\hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)} \hat{e}_1^n(A) + \\ \frac{\frac{1}{2\pi}(\hat{P}_1(\omega) * \hat{e}_1^n(A) + j\hat{P}_1^l(\omega) * (\omega\hat{e}_1^n(A)) - \frac{1}{2}\hat{P}_1^q(\omega) * (\omega^2\hat{e}_1^n(A)))}{-\omega \coth(\omega(A-B)) + \hat{S}_1(0) + \omega\hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)}. \end{aligned}$$

Next, move the 2nd term of the left side to the right side and re-arrange:

$$\begin{aligned} \hat{e}_1^{n+1}(A) = \frac{-\omega \coth(\omega(A-B)) + \hat{S}_2(0) + \omega\hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)}{\omega \coth(\omega A) + \hat{S}_2(0) + \omega\hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)} \hat{e}_1^n(A) + \\ \frac{\frac{1-X_1}{2\pi}(\hat{P}_2(\omega) * \hat{e}_2^n(A)) + \frac{1-X_1^l}{2\pi}(j\hat{P}_2^l(\omega) * (\omega\hat{e}_2^n(A)) - \frac{1-X_1^q}{4\pi}(\hat{P}_2^q(\omega) * (\omega^2\hat{e}_2^n(A))))}{\omega \coth(\omega A) + \hat{S}_2(0) + \omega\hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0)}, \end{aligned} \quad (4.54)$$

$$\begin{aligned} \hat{e}_2^{n+1}(A) = \frac{-\omega \coth(\omega A) + \hat{S}_1(0) + \omega\hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)}{-\omega \coth(\omega(A-B)) + \hat{S}_1(0) + \omega\hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)} \hat{e}_1^n(A) + \\ \frac{\frac{1-X_2}{2\pi}(\hat{P}_1(\omega) * \hat{e}_1^n(A)) + \frac{1-X_2^l}{2\pi}(j\hat{P}_1^l(\omega) * (\omega\hat{e}_1^n(A))) - \frac{1-X_2^q}{4\pi}(\hat{P}_1^q(\omega) * (\omega^2\hat{e}_1^n(A)))}{-\omega \coth(\omega(A-B)) + \hat{S}_1(0) + \omega\hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0)}. \end{aligned} \quad (4.55)$$

Note $X_i^q(\omega) := (\hat{P}_j^q * (\omega^2\hat{e}_i^{n+1}(A)))/(\hat{P}_j^q * (\omega^2\hat{e}_j^n(A)))$, and it could be estimated that $|X_i^q(\omega)| < 1$. The reason is similar to $X_i(\omega)$ in RR-SP0, see Eqs.(4.32),(4.33).

Divide above equations by $\hat{e}_j^n(A)$ on both sides, it could be seen that the following term plays the key role in determining the boundary effect:

$$\frac{\hat{P}_i(\omega) * \hat{e}_i^n(A) + j\hat{P}_i^l(\omega) * (\omega\hat{e}_i^n(A)) - \frac{1}{2}\hat{P}_i^q(\omega) * (\omega^2\hat{e}_i^n(A))}{\hat{e}_i^n(A)}.$$

4.6 Estimated analysis of boundary effect in RR-SP1 and RR-SP2

This section makes analysis on the convergence of RR-SP1 and RR-SP2 based on the convergence rate calculation result in Section 4.5. Since the boundary effect is represented by an individual term that could be separated from the standard convergence rate term, the section starts with listing the key terms in each algorithm. Next, similar analysis to RR-SP0 in Section 4.4 is made upon RR-SP1 and RR-SP2 respectively but the model study as shown in RR-SP0 is not conducted on RR-SP1 and RR-SP2. However, the model study result of RR-SP0 is used for understanding the boundary effect in RR-SP1 and RR-SP2, and helps evaluate the boundary effect in the two algorithms in qualitative manner. Analysis result suggests that the boundary effect in both RR-SP1 and RR-SP2 algorithms has positive influence on boosting the convergence, and RR-SP1 is supposed to converge faster than RR-SP0 and roughly at equal rate to RR-SP2.

4.6.1 Key terms of boundary effect

Recall in RR-SP0, the key term to represent the the boundary effect is $\frac{\hat{P}_i(\omega) * \hat{e}_i^n(A)}{\hat{e}_i^n(A)}$, see Eq(4.34) and below, and it is supposed to reduce the numerator of the standard convergence rate:

$$\hat{S}_1(0) - \omega \coth(\omega A) \text{ or } \hat{S}_2(0) - \omega \coth(\omega(B - A)).$$

Similar result could be obtained for RR-SP1 and RR-SP2:

Boundary effect of RR-SP0:

$$\frac{\hat{P}_i(\omega) * \hat{e}_i^n(A)}{\hat{e}_i^n(A)}.$$

Numerator of the standard convergence rate of RR-SP0:

$$\begin{aligned} \hat{S}_1(0) - \omega \coth(\omega A), \\ \hat{S}_2(0) - \omega \coth(\omega(B - A)). \end{aligned}$$

Boundary effect of RR-SP1:

$$\frac{\hat{P}_i(\omega) * \hat{e}_i^n(A) + j\hat{P}_i^l(\omega) * (\omega\hat{e}_i^n(A))}{\hat{e}_i^n(A)}. \quad (4.56)$$

Numerator of the standard convergence rate of RR-SP1:

$$\hat{S}_1(0) - \omega \coth(\omega A) + \omega\hat{S}_{1\omega}(0), \quad (4.57)$$

$$\hat{S}_2(0) - \omega \coth(\omega(B - A)) + \omega\hat{S}_{2\omega}(0). \quad (4.58)$$

Boundary effect of RR-SP2:

$$\frac{\hat{P}_i(\omega) * \hat{e}_i^n(A) + j\hat{P}_i^l(\omega) * (\omega\hat{e}_i^n(A)) - \frac{1}{2}\hat{P}_i^q(\omega) * (\omega^2\hat{e}_i^n(A))}{\hat{e}_i^n(A)}. \quad (4.59)$$

Numerator of the standard convergence rate of RR-SP2:

$$\hat{S}_1(0) - \omega \coth(\omega A) + \omega\hat{S}_{1\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{1\omega\omega}(0), \quad (4.60)$$

$$\hat{S}_2(0) - \omega \coth(\omega(B - A)) + \omega\hat{S}_{2\omega}(0) + \frac{1}{2}\omega^2\hat{S}_{2\omega\omega}(0). \quad (4.61)$$

4.6.2 Boundary effect evaluation of RR-SP1

This subsection considers the boundary effect in RR-SP1. From Eq.(4.56), it shows that one additional term distinguishes the boundary effect of RR-SP1 from RR-SP0: $j\frac{\hat{P}_i^l(\omega) * (\omega\hat{e}_i^n(A))}{\hat{e}_i^n(A)}$. To analyse the term, similar model study to RR-SP0 is referred to. Recall in Section 4.4, interface boundary value is assumed to be polynomials and the Fourier transform is given in

Eqs.(4.35) and (4.36). Assume N_r and N_i is close to one, the model to fit $\omega \hat{e}_i^n(A)$ could be roughly formulated as:

$$\mathcal{F}[\omega e_i^n(x, A)]_r \approx L_r \cos(\omega + \omega_0^r);$$

$$\begin{cases} \mathcal{F}[\omega e_i^n(x, A)]_i & \approx L_i (\cos(\xi\omega + \omega_0^i) - 1), & \text{if order} = 0 \\ \mathcal{F}[\omega e_i^n(x, A)]_i & \approx L_i \cos(\xi\omega + \omega_0^i). & \text{if order} \neq 0 \end{cases}$$

Next, assume $P_i^l(x)$ could be fitted by the same models as $P_i(x)$, see Eqs.(4.37) and (4.38), thus the term in Eq.(4.56): $\frac{\hat{P}_i^l(\omega) * (\omega \hat{e}_i^n(A))}{\hat{e}_i^n(A)}$ could be estimated the same way as Eq(4.34). The estimation process is mostly the same as that in Section 4.4, which leads to the estimation result as shown in Eq.(4.48). The only difference is that the result is perpendicular to $\frac{\hat{P}_i(\omega) * (\hat{e}_i^n(A))}{\hat{e}_i^n(A)}$ because of the imaginary unit j . Technically, the real and imaginary part of the result should be swapped. Roughly speaking, the term $j \frac{\hat{P}_i^l(\omega) * (\omega \hat{e}_i^n(A))}{\hat{e}_i^n(A)}$ should augment $\frac{\hat{P}_i(\omega) * (\hat{e}_i^n(A))}{\hat{e}_i^n(A)}$ in magnitude and it could be estimated that

$$\left| \frac{\hat{P}_i(\omega) * \hat{e}_i^n(A) + j \hat{P}_i^l(\omega) * (\omega \hat{e}_i^n(A))}{\hat{e}_i^n(A)} \right| > \left| \frac{\hat{P}_i(\omega) * \hat{e}_i^n(A)}{\hat{e}_i^n(A)} \right|.$$

On the other hand, the standard convergence rate of RR-SP0 and RR-SP1 is equivalent (and so are the numerators), so the real convergence of RR-SP1 should outperform (be faster) RR-SP0. The effect of the additional term $\frac{j \hat{P}_i^l(\omega) * (\omega \hat{e}_i^n(A))}{\hat{e}_i^n(A)}$ is equal to increasing the magnitude of $(\Phi_1 + \Phi_2)$ and $(\Phi_3 + \Phi_4)$ in RR-SP0, see Eq.(4.48). The standard convergence rate remaining unchanged means the base line in Figs.4.21 and 4.22 ($y = \pm\omega$) remains unchanged. Meanwhile, since both the interface boundary value and the pike functions of RR-SP1 are modelled the same way as RR-SP0, the analysis result for the latter applies to the former as well: the strength of the boundary effect depends on α/β or the shape of the pike function, which is elaborated in Subsection 4.4.3.

Lastly, as analysed before, both P_i and P_i^l play positive role in accelerating the convergence in RR-SP1, the boundary effect of RR-SP1 must make the algorithm converge faster than the standard version.

4.6.3 Boundary effect evaluation of RR-SP2

This subsection considers the boundary effect in RR-SP2. Similar to the analysis in last subsection (4.6.2), the boundary effect term of RR-SP2 is compared with RR-SP1 and the additional term is paid special attention. Eq.(4.59) shows that the additional term of RR-SP2 is $\frac{-\frac{1}{2} \hat{P}_1^q(\omega) * (\omega^2 \hat{e}_1^n(A))}{\hat{e}_1^n(A)}$. Since the profile of P_1^q does not resemble P_2^q (see Figs.4.27, 4.28) and they have different effect on the convergence, P_1^q and P_2^q need to be considered separately.

Fig.4.27 shows that P_1^q is small in value and could be nearly considered as constant throughout

the entire domain: $P_1^q \approx P^*$, ($P^* < 0$). It could be estimated that $\frac{-\frac{1}{2}\hat{P}_1^q(\omega) * (\omega^2 \hat{e}_1^n(A))}{\hat{e}_1^n(A)} \approx -\frac{1}{2}P^*\omega^2 > 0$. The effect of $-\frac{1}{2}P^*\omega^2$ is equal to shifting $(\Phi_3 + \Phi_4)$ upwards (in y direction) by P^* , see Figs.4.21 and 4.22.

Fig.4.28 shows that P_2^q is a positive function and it could be fitted by $P_2^q = C + w_2^q$, where C is a positive constant and w_2^q represents a function that could be modelled the same way as $P_i(x)$ by exponential functions such as Eqs.(4.37) and (4.38). Since C is a positive constant, it must impose inverse effect on the convergence compared to P^* , which means C shifts $(\Phi_3 + \Phi_4)$ downwards. Therefore, P^* and C are counterproductive in terms of influencing the convergence rate.

In addition, the term $\frac{w_2^q * (\omega^2 \hat{e}_2^n(A))}{\hat{e}_2^n(A)}$ remains unknown on how to change the standard convergence rate. Assume the approximation of the term is similar to $\frac{\hat{P}_i(\omega) * \hat{e}_i^n(A)}{\hat{e}_i^n(A)}$ in RR-SP0, consequently the left and right parts of P_2^q are counterproductive in influencing the convergence because they have different signs. The analysis of P_2 in Section 4.4 has shown that the positive pikes boost the convergence, and the right pike of P_2^q (which is negative, considering the coefficient of the term: $-\frac{1}{2}$) actually impairs the right pike of P_2 thus weakens the boost effect on the convergence. By contrast, the left pike of P_2^q (which is positive) consolidates the left pike of P_2 and strengthens the boost effect.

The numerator of the standard convergence rate of RR-SP2 differs itself from that of RR-SP1(0) by an additional term: $\frac{1}{2}\omega^2 \hat{S}_{i\omega\omega}(0)$, which makes the magnitude of the numerator smaller or unchanged, see Fig.4.29. Now assume Eq.(4.62) holds true, as a result, the real convergence rate of RR-SP2 is smaller than that of RR-SP0 and RR-SP1.

$$\left| \frac{\hat{P}_i(\omega) * \hat{e}_i^n(A) + j\hat{P}_i^l(\omega) * (\omega \hat{e}_i^n(A)) - \frac{1}{2}\hat{P}_i^q(\omega) * (\omega^2 \hat{e}_i^n(A))}{\hat{e}_i^n(A)} \right| \geq \left| \frac{\hat{P}_i(\omega) * \hat{e}_i^n(A) + j\hat{P}_i^l(\omega) * (\omega \hat{e}_i^n(A))}{\hat{e}_i^n(A)} \right| > \left| \frac{\hat{P}_i(\omega) * \hat{e}_i^n(A)}{\hat{e}_i^n(A)} \right|. \quad (4.62)$$

Considering the overall effect of P^* , C and w_2^q , it is not straightforward to evaluate the effect of P_i^q because it largely depends on how counterproductive P^* , C and $\frac{w_2^q * (\omega^2 \hat{e}_2^n(A))}{\hat{e}_2^n(A)}$ are working against each other. Despite much uncertainty, what could be assured is that due to the counterproductive effect, the overall effect of P_i^q should not be very strong (either positive or negative), although the real effect needs to be determined by more quantitative study on P^* , C and w_2^q .

Lastly, since P_i and P_i^l are both positive in boosting the convergence and P_i^q does not impose strong influence, the overall boundary effect of RR-SP2 should make the algorithm converge faster than the standard version.

To summarise, the standard convergence rate of RR-SP0 is equivalent to that of RR-SP1, and higher than that of RR-SP2. Taking the boundary effect into consideration, the real convergence rate is lower than the standard one for each single case. Meanwhile, the boundary effect strength is supposed to become increasingly stronger as N (RR-SPN) grows from 0 to

1, so the real convergence rate of RR-SP0 should also be higher than that of RR-SP1. RR-SP2 remains unknown about whether having faster or slower convergence than RR-SP1 but the difference should be minor. Therefore, it could be roughly considered as equal to RR-SP1.

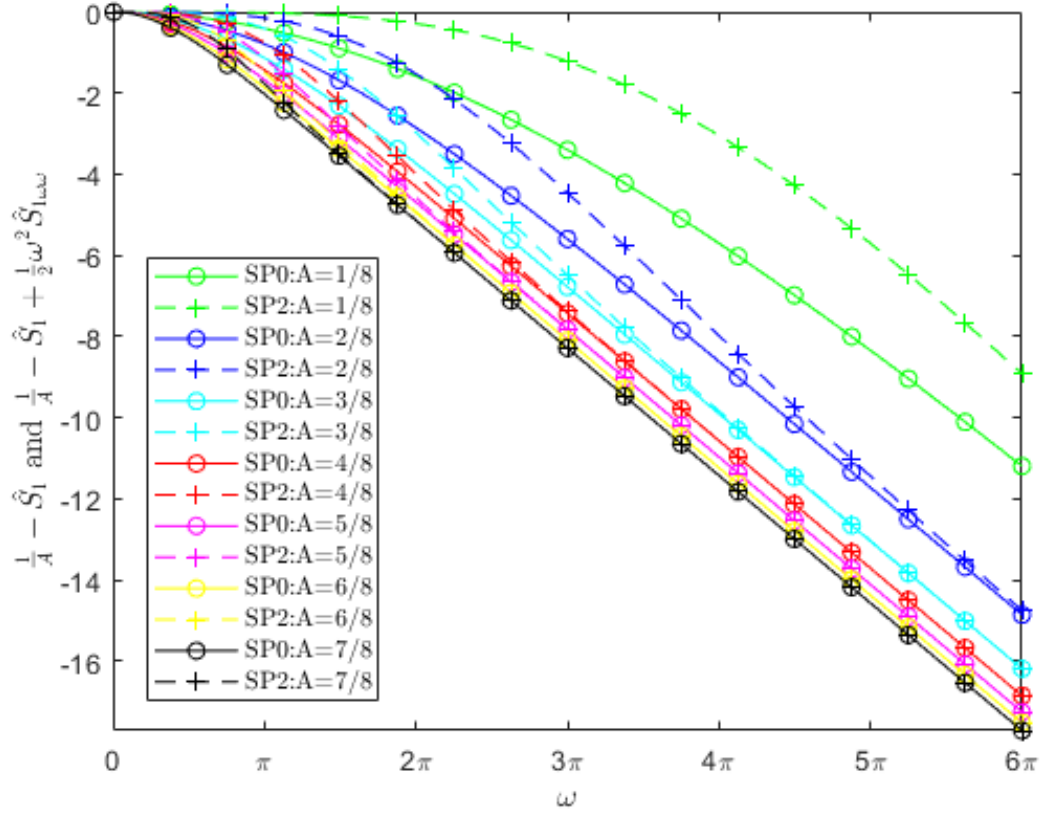


Figure 4.29. Numerator of the standard convergence rate CR_2 of RR-SP0(1) and RR-SP2. The magnitude of the former is always no less than the latter. The numerator of $CR_2(SP0) := \frac{1}{A} - \hat{S}_1(\omega)$, and the numerator of $CR_2(SP2) := \frac{1}{A} - \hat{S}_1(\omega) + \frac{1}{2}\omega^2 \hat{S}_{1\omega\omega}(0)$.

4.7 Summary

This chapter analyses the convergence rate result of the three RR algorithms obtained in Chapter 3, and considers the influence of the boundary effect.

The standard convergence rate of RR-SP0 is determined by the increment of SP operator symbol: $\Delta s_i(\omega) := \hat{S}_i(\omega) - \hat{S}_i(0)$. Smaller value of $\Delta s_1 \Delta s_2$ results in lower convergence rate. Similar result is obtained for RR-SP2 that the convergence rate is determined by $\Delta s_{1\omega\omega} \Delta s_{2\omega\omega}$ where $\Delta s_{i\omega\omega}(\omega) := \hat{S}_{i\omega\omega}(\omega) - \hat{S}_{i\omega\omega}(0)$. Clearly the nature of SP operator symbols is directly associated with the convergence of the algorithms. In practice, the analysis result is useful for evaluating the convergence of the algorithms with different interface boundary location y^* . A numerical test on Poisson's equation verifies the analysis result.

The boundary effect of RR-SP0 is estimated through a model study and it is supposed to help understand how the real convergence differs from the standard convergence and by how much. Through the study, one empirical law is summarised to estimate the strength of the boundary

effect, which varies from case to case in practice. The law provides a simple and quick way to evaluate the real convergence based on the standard convergence result. The evaluation of the boundary effect of RR-SP1 and RR-SP2 is performed the same way as RR-SP0 but in a qualitative manner. The result shows that the boundary effect could improve the standard convergence. Considering both the standard convergence nature and the boundary effect, the real convergence rate in reducing order reads: $RR-SP0 > RR-SP1 \approx RR-SP2$. A numerical test on Poisson's equation verifies the prediction.

Additionally, the convergence of DR algorithm is proved slower than any of the Robin-Robin type algorithms, which marks the significance of the project to study the latter.

Chapter 5

Numerical tests

This chapter conducts numerical tests on a Poisson's equation and two model equations of high-Reynolds-number flow with a boundary layer. The aim is to verify the convergence analysis result as shown in Chapters 3 and 4, and to test the feasibility of applying RR algorithms to near-wall turbulence modelling respectively.

In Section 5.1, the discrete scheme adopted in the thesis to implement the numerical tests is introduced. In particular, the discrete form of the SP operator approximations that make up the key components of the IBC formulations in the three domain decomposition algorithms is elaborated. The coding language is clarified.

In Section 5.2, a numerical test is conducted on Poisson's equation with various A . The test result fully supports the analysis result. Meanwhile, RR-SPN algorithms demonstrate their advantage over DR and the conventional RR algorithm [102] in the test.

In Section 5.3, a series of numerical tests are implemented on RR and DR algorithms to solve a RANS model equation with small parameter on second order derivative. The solution has different solution dependence on x . A free parameter α is used to manipulate the dependence: larger α accounts for greater dependence. The model equation represents the general equation form in RANS model. In low- α cases, the boundary effect is not powerful enough to influence the standard convergence so the real convergence is actually dominated by the standard convergence. In the test, RR-SP2 shows clear advantage over others as a result of lower standard convergence. When α grows larger, the boundary effect becomes more powerful so the real convergence differs more remarkably by the standard one. However, the boundary effect is not always positive. The numerical tests shows that RR-SP1 performs very slow convergence in high- α case. The speculation is that the solution u is composed of 2 parts: V and W , the former of which has variable-separation form and the latter does not. The slow convergence of RR-SP1 should be resulted by W . On the other hand, RR-SP2 demonstrates the ability of avoiding being negatively influenced by W and maintaining promising convergence in high- α case. The whole series of the tests show that RR-SP2 has better ability to perform faster convergence and maintain better stability in dealing with complex problem. In other words, the algorithm is better at reserving the non-local nature of SP operators applied to the given two-dimensional problem.

In the second part of Section 5.3, additional numerical tests are implemented to check the result as drawn previously. Firstly, interface boundary is set closer to the wall ($y = 0$) so that

the non-local effect of the inner region problem is weaker. Since RR-SP0, RR-SP1 and DR are less capable of retaining the non-local effect, the three algorithms especially benefit from the change and perform clearer improvement in the convergence. Next, Reynolds number of the model equations is set higher and the tests are repeated. Higher Reynolds number means the solution becomes relatively less dependent on x and the problem is easier to solve thus less iterations could be expected. The test result is comparatively studied with that from previous test and it confirms again that RR-SP2 shows better performance in terms of efficiency and stability.

In Section 5.4, a new method of floating interface boundary is proposed to achieve 'two-iteration' convergence. The idea is to run any convergent ENDD algorithm in outer region with an imaginary interface that is several nodes below the real interface. Then resume the interface to where it really is and resolve inner region. After the first iteration, the convergence is expected to be achieved and the second iteration is used to check the convergence. The method is proved efficient in solving the model equations regardless of α , and the two-iteration convergence is realised in all tested cases.

5.1 Discrete scheme and coding language

This section introduces how the numerical tests as presented in the thesis are implemented in terms of discrete scheme. The discrete form of SP operator approximations is particularly elaborated. The coding language is clarified.

The method of central finite difference is used as the discrete scheme. In particular on the interface, if the mesh are uniform, the first derivative (for instance, at node (i_0, j_0)) is obtained by

$$u_y(i_0, j_0) = \frac{1}{2h}(3u(i_0, j_0) - 4u(i_0, j_0 - 1) + u(i_0, j_0 - 2)),$$

where h is the meshing space between two adjacent nodes. If the mesh is not uniform, the derivative on interface is simply obtained by forward/backward differentiation.

Recall that the constant approximation of SP operators reads:

$$S_i u(x) \approx S_i I \cdot u(x_0).$$

Regarding the discretisation of $S_i u(x)$, there are potentially two schemes to consider:

$$\text{scheme 1: } S_i u(i_0) \approx S_i I(i_0) \cdot u(i_0); \quad (5.1)$$

$$\text{scheme 2: } S_i u(i_0) \approx S_i I(i_0) \cdot u(i_0 + 1). \quad (5.2)$$

In scheme one, x and x_0 are overlapping. In scheme two, x_0 is always one node ahead of the concerned node x . The value of x corresponds to the node index i_0 and x_0 has the index $(i_0 + 1)$. Although x and x_0 are intrinsically two distinctive nodes, numerical test on Poisson's equation (as will be elaborated in Section 5.2) shows that scheme one leads to convergence of

the test but scheme two causes divergence. The reason could be that the local error incurred by constant approximation is massively accumulated as the 'global' approximation is made. Note that x depends on its neighbour node x_0 , and the local approximation error is proportional to the space (h) between them. Then the error is transferred to the next node as the value of x_0 is locally approximated depending on the nodes ahead of it. The procedure repeats and the error accumulates. If there are $N + 1$ nodes, the local approximation repeats N times and the local error for the last node could be N times larger than the local error that is proportional to the meshing space h . In other words, the maximum error is about the size of the domain size (denote by L , $L = Nh$) and it could not be reduced by increasing the nodes number. By contrast, if x and x_0 are overlapping, the local error is always zero. However, it does not mean that the approximation is exact because x and x_0 are two distinctive nodes, and $u(x_0)$ is considered as constant while $u(x)$ as variable in the approximation. There is inevitable error incurred by neglecting the difference between the constant and the variable, for example $S_i u(x) \neq S_i I \cdot u(x)$. The error is related to the non-local nature of the function as analysed in Subsection 2.4.1. Only in special case (for instance, $u(x) \equiv \text{const.}$), the error could be eliminated.

Based on the result of constant approximation, the discrete form of linear and quadratic approximation of SP operators could be obtained:

$$\text{linear approximation: } S_i u(x) \approx (u(x_0) - x_0 u_x(x_0)) S_i I + u_x(x_0) S_i x(x);$$

$$\begin{aligned} \text{discrete form: } S_i u(i_0) \approx & (u(i_0) - x(i_0) \frac{u(i_0 + 1) - u(i_0 - 1)}{2h}) S_i I(i_0) + \\ & \frac{u(i_0 + 1) - u(i_0 - 1)}{2h} S_i x(i_0), \end{aligned}$$

$$\begin{aligned} \text{quadratic approximation: } S_i u(x) \approx & (u(x_0) - x_0 u_x(x_0) + \frac{x_0^2 u_{xx}(x_0)}{2}) S_i I + \\ & (u_x(x_0) - x_0 u_{xx}(x_0)) S_i x(x) + \frac{u_{xx}(x_0)}{2} S_i x^2; \end{aligned}$$

$$\begin{aligned} \text{discrete form: } S_i u(i_0) \approx & (u(i_0) - x(i_0) \frac{u(i_0 + 1) - u(i_0 - 1)}{2h} + \\ & x^2(i_0) \frac{u(i_0 + 1) + u(i_0 - 1) - 2u(i_0)}{2h^2}) S_i I(i_0) + \\ & (\frac{u(i_0 + 1) - u(i_0 - 1)}{2h} - \\ & x(i_0) \frac{u(i_0 + 1) + u(i_0 - 1) - 2u(i_0)}{h^2}) S_i x(i_0) + \\ & \frac{u(i_0 + 1) + u(i_0 - 1) - 2u(i_0)}{2h^2} S_i x^2(i_0). \end{aligned}$$

If the mesh is not uniform, the discrete form of derivative is slightly different:

$$\begin{aligned} u_x(i_0) &= \frac{u(i_0 + 1) - u(i_0 - 1)}{h_1 + h_2}, \\ u_{xx}(i_0) &= (\frac{u(i_0 + 1) - u(i_0)}{h_2} + \frac{u(i_0) - u(i_0 - 1)}{h_1}) / (h_1 + h_2), \end{aligned}$$

where h_1 is the meshing space between $x(i_0)$ and $x(i_0 - 1)$, and h_2 is the meshing space between $x(i_0)$ and $x(i_0 + 1)$.

In this thesis, all the numerical computation is implemented by Fortran 90 with GNU Fortran compiler.

5.2 Poisson's equation

In Subsection 4.1.2 of Chapter 4, a Poisson's equation test is performed to verify the theoretical analysis about RR-SP0 (both standard and real). In this subsection, the test is extended to RR-SP1 (real) and RR-SP2 (real). The test parameters and conditions are the same as Subsection 4.1.2 thus will not be introduced again. The test result is displayed in Table.5.1, which shows that RR-SP2, RR-SP1 and RR-SP0 perform the convergence in fast-to-slow order. Note RR-SP1 converges clearly faster than RR-SP0 but RR-SP2 shows no remarkable advantage over RR-SP1, which is possibly associated with the counterproductive effect of elements in \hat{P}_i^q , as discussed in Chapter 4, Section 4.6. The numerical result in the table is consistent with the theoretical analysis.

Table 5.1. Convergence iterations of (parallel) RR-SP0, RR-SP1 and RR-SP2 in application to a Poisson's equation.

A	1/8	2/8	3/8	4/8	5/8	6/8	7/8
RR-SP0	8	9	9	9	8	8	7
RR-SP1	6	7	7	7	7	7	6
RR-SP2	6	7	7	7	6	6	6

The Poisson's equation as shown above follows Lui's (conventional) case in [102] but the convergence criterion is different. Instead, Lui used the convergence criterion as displayed in Eq.(4.21), which monitors the convergence by checking absolute error on the interface. However, the exact interface solution varies with A and it makes the absolute error check unevenly strict for all cases: more strict for those A that have larger interface exact solution and less strict for those A that have smaller interface exact solution. To avoid it, the convergence criterion is replaced by Eq.(4.8) (relative criterion) in above test.

Next, consider repeating Lui's case first and compare Lui's algorithm with RR-SPN and DR afterwards. In the test, Eq.(4.21) is used as the convergence criterion. Recall in Subsection 4.3 of Chapter 4, it is proved that Lui's algorithm is essentially equivalent to RR-SP0 because the two algorithms only differ in parameters selection: Lui makes $\sigma_1 = \sigma_2 = \lambda = 6.794$, RR-SP0 makes $\sigma_1 = S_2 I$ and $\sigma_2 = S_1 I$. Meanwhile, at the end of Subsection 4.3, it is stated that RR-SP0 could prove its advantage over Lui's algorithm. To verify it, a numerical test is implemented and the result is shown in Table 5.2. The table also summarises the test result of RR-SP1, RR-SP2 and DR.

The test result shows that Lui's algorithm converges in the same iterations as RR-SP0 (parallel). However, RR-SP0 could prove its slight advantage from the convergence history

Table 5.2. Convergence iterations of RR-SPN,DR and Lui's algorithm in application to solving a Poisson's equation.

Algorithm	Parameter	$a = \frac{1}{8}$	$a = \frac{3}{8}$	$a = \frac{7}{8}$
Lui's	$\lambda = 6.794$	5	8	6
	RR-SP0	5	8	6
RR(parallel)	RR-SP1	4	5	4
	RR-SP2	4	5	4
RR(sequential)	RR-SP0	3	5	3
	RR-SP1	2	3	3
	RR-SP2	2	3	3
DR		4	7	7

plot, see Fig.5.1. Next, it is not surprising to see RR-SP1 and RR-SP2 show faster convergence than RR-SP0, which is also displayed in Table 5.1. The convergence history of (sequential) RR-SPN and DR algorithms is plotted in Figs.5.2 to 5.4. In addition, Table 5.2 compares the convergence iterations between the parallel and the sequential algorithms and it shows that the sequential algorithms take around half of iterations of the parallel algorithms to converge, which is consistent with the theoretical analysis. Lastly, DR algorithm shows clearly slower convergence than any RR-SPN algorithm. Note DR algorithm is supposed to compare with sequential algorithms of RR-SPN because DR algorithm itself is constructed as sequential algorithm: DR algorithm starts from solving u_2 , and u_1 uses the information of u_2 from the same iteration. This is the typical iterative procedures of sequential computing.

Comparing the numerical result of RR-SP0 shown in Tables 5.1 and 5.2, it could be found that $A = 3/8$ case converges three iterations slower (around 40% slower) than $A = 1/8$ case in Table 5.2 while only 1 iteration slower (around 10% slower) in Table 5.1. This is possibly because $A = 3/8$ case has larger exact value on interface and the absolute error check is more strict for it, which makes it take additionally more iterations to meet the convergence requirement.

5.3 Model equations

5.3.1 Model equation 1

This subsection applies the RR algorithms to solving two model equations that do not have analytical solution. The model equations are two-dimensional and have small parameter in second derivative term. The test result is supposed to be compared with that of Poisson's equation because the theoretical convergence analysis is made upon the latter.

Consider a model equation defined in a $[0, 1] \times [0, 1]$ square. The equation was solved earlier [81] by the so-called non-local wall function, which gives approximated solution to

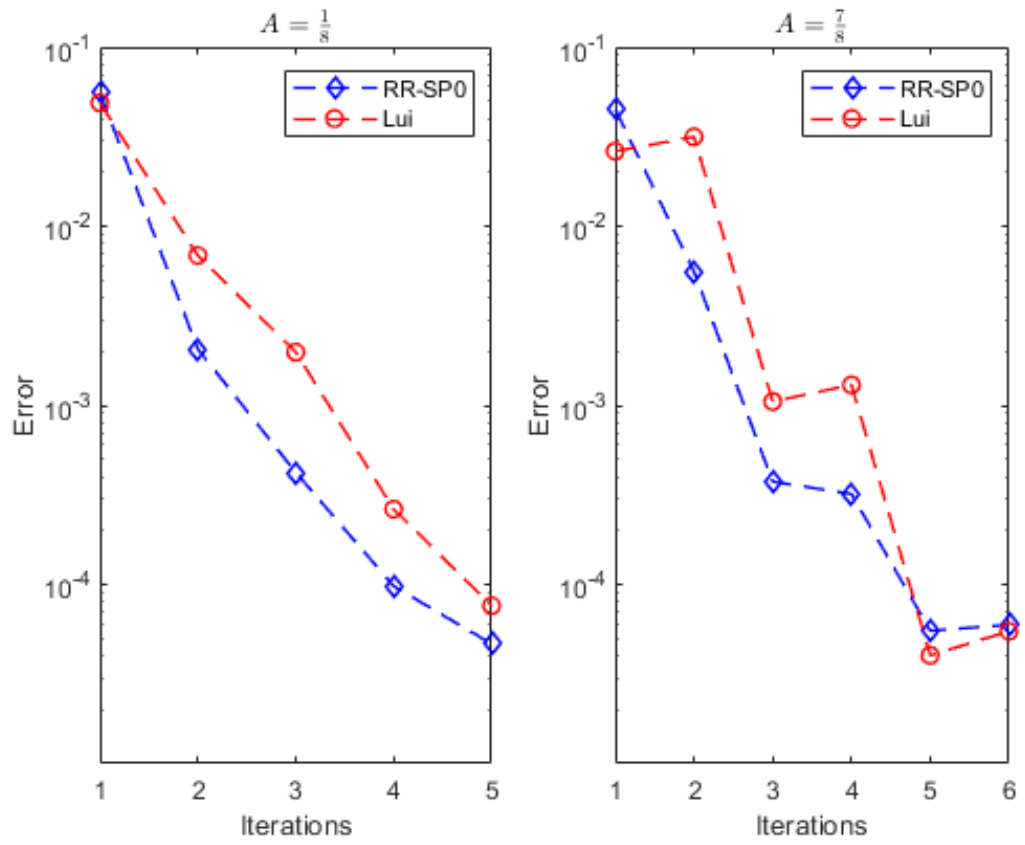


Figure 5.1. Convergence history of Lui's algorithm ($\lambda = 6.794$) and (parallel) RR-SP0 in comparison to solve a Poisson's equation. $A = \frac{1}{8}, \frac{7}{8}$

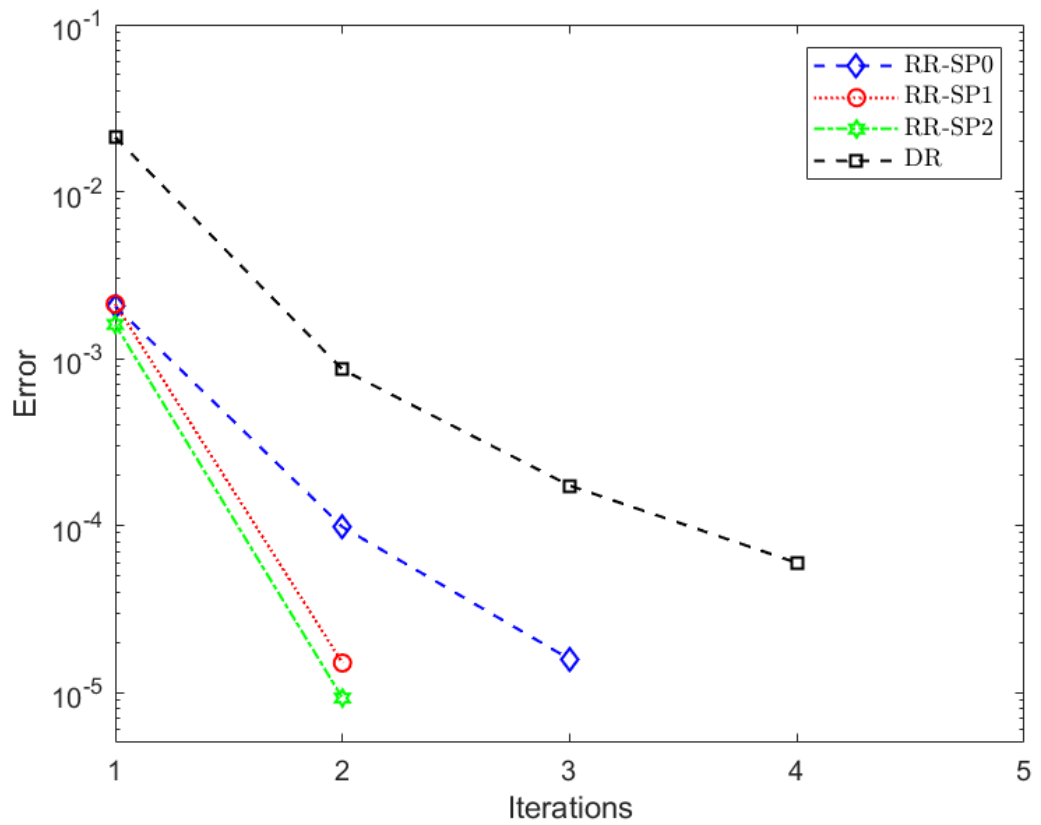


Figure 5.2. Convergence history of (sequential) RR-SPN ($N=0,1,2$) and DR algorithms in application to solving a Poisson's equation. $A = \frac{1}{8}$.

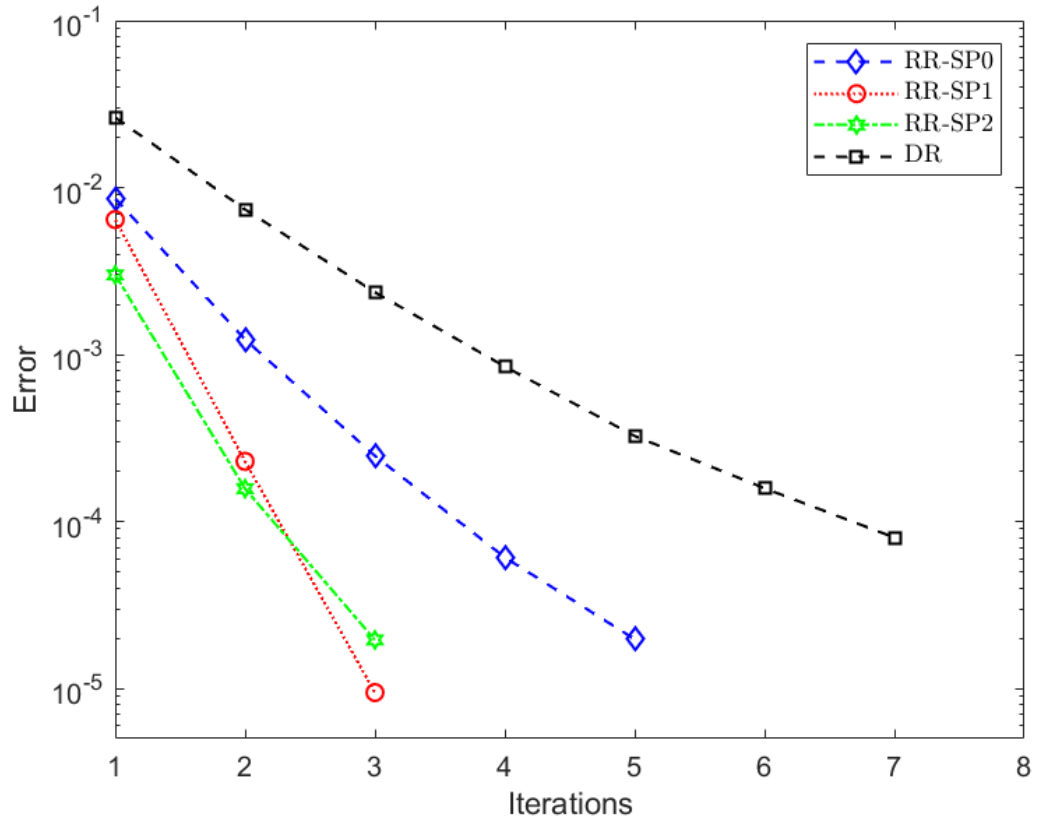


Figure 5.3. Convergence history of (sequential) RR-SPN ($N=0,1,2$) and DR algorithms in application to solving a Poisson's equation. $A = \frac{3}{8}$.

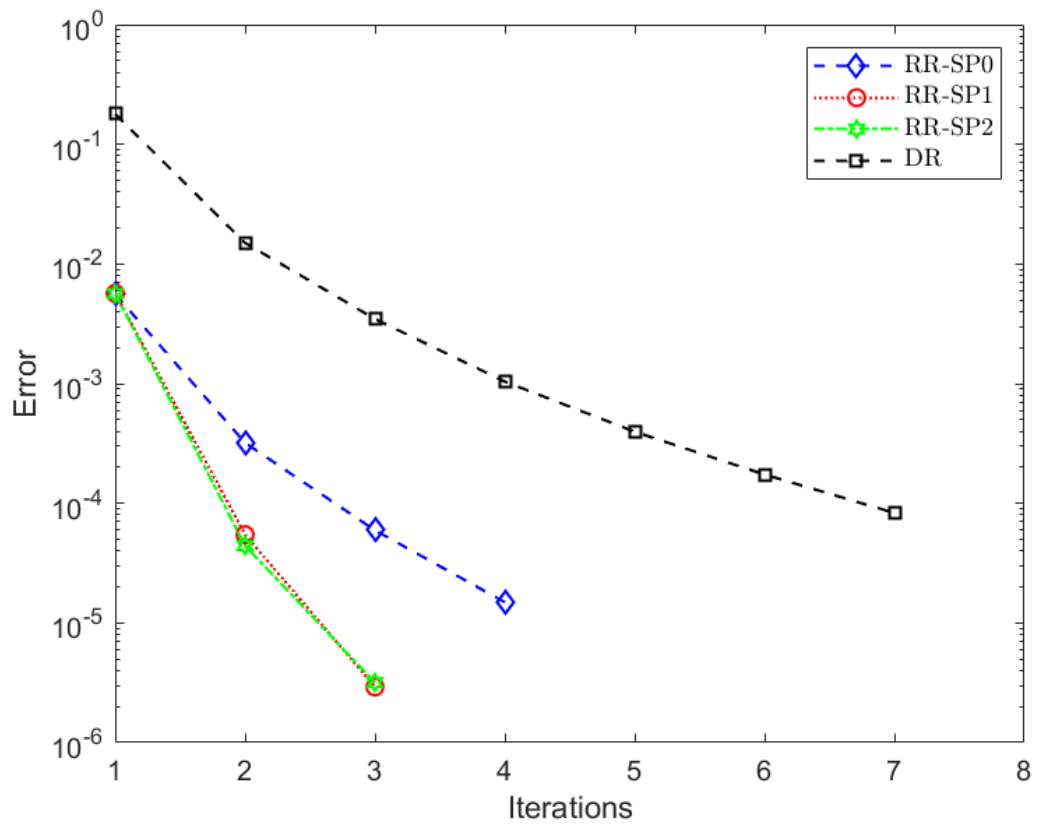


Figure 5.4. Convergence history of (sequential) RR-SPN ($N=0,1,2$) and DR algorithms in application to solving a Poisson's equation. $A = \frac{7}{8}$.

the equation. In this thesis, the equation is supposed to be solved exactly.

$$\begin{aligned}
(\mu U_y)_y + \sigma U_{xx} + \beta U_y + \gamma U &= f(x, y), \\
U(x, 0) &= 0, \\
U(x, 1) &= U_s(x), \\
U(0, y) &= U_0(y), \\
U(1, y) &= U_1(y).
\end{aligned} \tag{5.3}$$

Note $\sigma = \alpha\mu$, $\mu = \frac{1-e^{y/\epsilon}+\delta}{Re}$, $\epsilon = 3 \times 10^{-2}$, $\delta = 10^{-2}$, $\beta = y$, $\gamma = -y^2$, $Re = 10^4$, $f = -100x^2$, $U_s = 1 + 100x^2$.

The governing equation represents the general form of equations in RANS model. Apparently, the bigger σ , the stronger non-local effect the function has. In the vicinity of the wall, the function has boundary-layer profile and the mean flow feature prevails in off-wall region. The first two terms on the left hand side represent dissipative terms in RANS equations and the other terms are the representation of convective terms. The right hand side models source terms such as pressure gradient in momentum equation. The coefficient μ represents effective viscosity coefficient.

Note U_0 and U_1 are two one-dimensional functions that satisfy the following conditions:

$$(\mu U_y)_y + \beta U_y + \gamma U = 0, \tag{5.4}$$

$$U(0) = 0, \tag{5.5}$$

$$U(1) = U_s. \tag{5.6}$$

Such boundary conditions means flow becomes locally one-dimensional in the vicinity of the boundaries at $x = 0$ and $x = 1$.

In this test, $\alpha = 1, 10^2$ and 10^4 respectively. Here α is used to control the solution variation with x . If $\alpha = 0$, the problem becomes one-dimensional. The larger α , the stronger the solution is influenced by the term σU_{xx} and has greater dependence on x .

The mesh in x direction are uniformly generated with 101 nodes so $\Delta h_x = 0.01$. In y direction, adaptive mesh are generated separately in inner and outer region [104]:

$$y = -\frac{e^k}{b} \ln(1 - c\eta) \quad \text{if } 0 \leq \eta \leq \eta_1, \tag{5.7}$$

$$y = y(\eta_1) + \frac{c(\eta - \eta_1)}{b} \quad \text{if } \eta_1 \leq \eta \leq 1. \tag{5.8}$$

The parameters are set as follows: $k = -8.12$, $c = 1.67$, $b = 0.67$, $\eta_1 = 0.6$. The inner region mesh are generated as $\eta \leq \eta_1$ and those in outer region are generate as $\eta_1 \leq \eta \leq 1$. There are about 270 nodes in inner region with $y^* \approx 10^{-3}$ and about 230 nodes in outer region. The total node number is about 500.

The convergence criterion is:

$$\max\left(\frac{\|u_1^{n+1} - u_1^n\|_\infty}{\|u_1^{n+1}\|_\infty}, \frac{\|u_2^{n+1} - u_2^n\|_\infty}{\|u_2^{n+1}\|_\infty}\right) < 10^{-6}. \quad (5.9)$$

Table 5.3 shows that when the solution has weak dependence on x ($\alpha = 1$ case), the three algorithms converge at equal rate. This case matches the Poisson's equation best because they both have solutions that could be separated in terms of variables. In other words, the problem is basically one-dimensional for $\alpha = 1$ case. When the solution dependence on x becomes stronger ($\alpha = 100, 10^4$), RR-SP2 outperform the other 2 algorithms and RR-SP1 no longer shows any advantage over RR-SP0. Particularly at $\alpha = 10^4$ case, it converges especially slowly. This is different from the result in Poisson's equation test. The convergence history of the four algorithms is displayed in Figs.5.5 to 5.7.

Table 5.3. Convergence iteration of (sequential) RR-SP0, RR-SP1, RR-SP2 and DR in application to a model equation, $A = 10^{-3}$.

α	1	10^2	10^4
RR-SP0	3	7	10
RR-SP1	3	8	44
RR-SP2	3	4	10
DR	4	10	21

Table 5.4. Convergence iteration of (sequential) RR-SPN ($N=0,1,2$) and RR-SPN* in comparison to solve a model equation, $A = 10^{-3}$.

α	1	10^2	10^4
RR-SP0	3	7	10
RR-SP0*	4	8	16
RR-SP1	3	8	44
RR-SP1*	4	8	16
RR-SP2	3	4	10
RR-SP2*	4	5	15

To examine the boundary effect in the three cases, it is necessary to illustrate the profile of $S_i I$ of all α -cases, which is displayed in Figs.5.8 to 5.10.

Fig.5.8 shows that in $\alpha = 1$ case, there hardly exists any boundary effect so the convergence rate of the three algorithms is subjected to the standard convergence rate calculation. If $\hat{S}_{iw}(0) = 0$ still holds true for the model equation, RR-SP1 should have equal convergence rate as RR-SP0. Fig.5.5 confirms this: it shows that the error line of RR-SP1 almost coincides with RR-SP0. Meanwhile, RR-SP2 is shown to converge faster than the other two algorithms, which means the standard convergence rate of RR-SP2 is lower than that of RR-SP0(1).

Regarding the case of $\alpha = 100$, RR-SP2 shows clear superiority over the other 2 algorithms and RR-SP1 is still equal (actually slightly worse than) to RR-SP0. The reason of RR-SP2 performing better could be either it has lower standard convergence rate or stronger positive boundary effect. To study it, RR-SP2* is implemented to solving the problem and the convergence iteration is five, which is only one iteration slower than RR-SP2. Meanwhile,

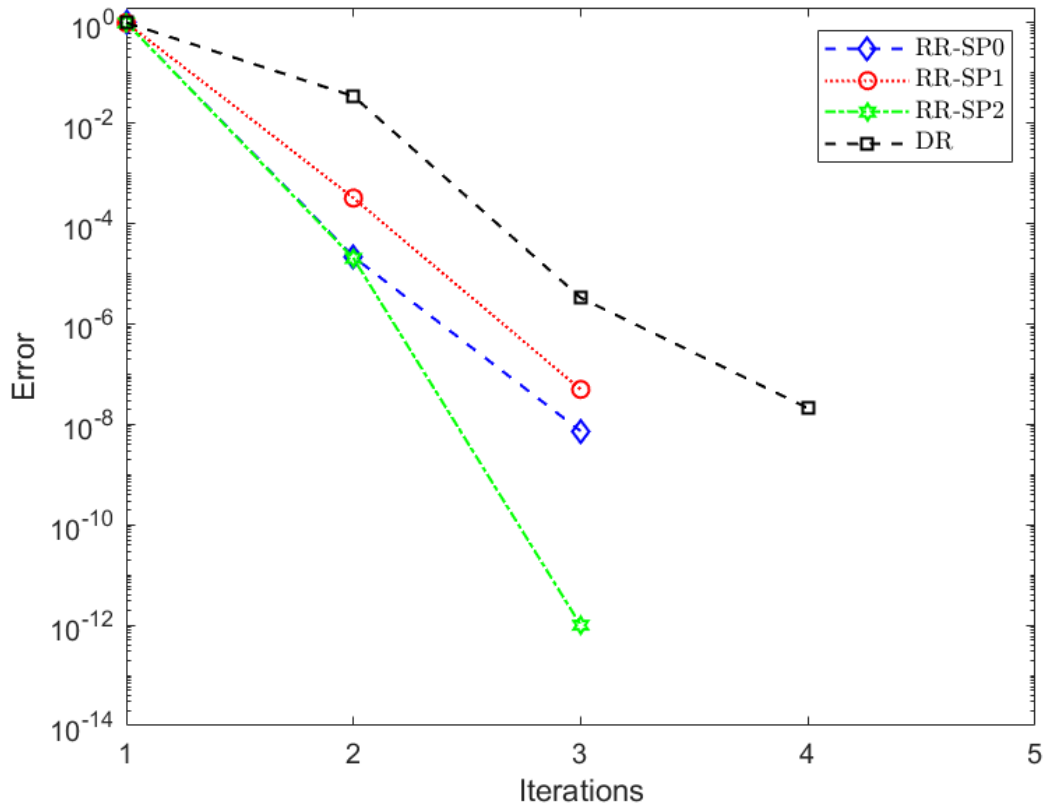


Figure 5.5. Convergence history of (sequential) RR-SPN ($N=0,1,2$) and DR algorithms in application to solving a model equation. $\alpha = 1$.

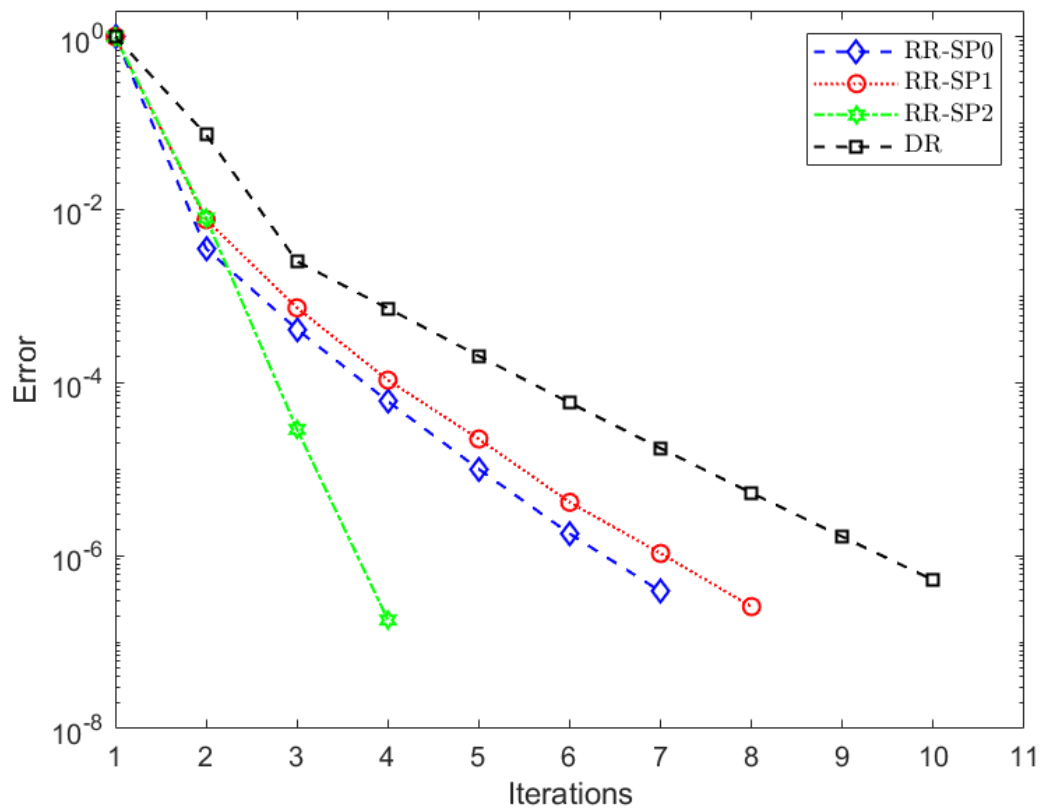


Figure 5.6. Convergence history of (sequential) RR-SPN ($N=0,1,2$) and DR algorithms in application to solving a model equation. $\alpha = 10^2$.

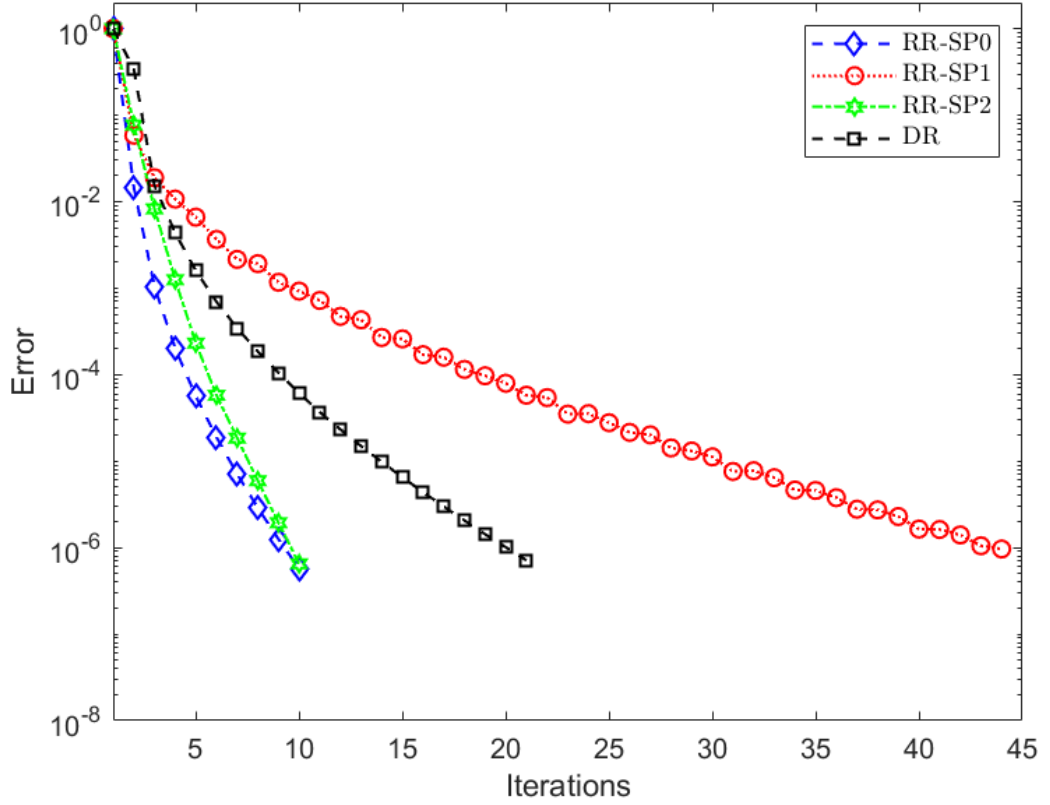


Figure 5.7. Convergence history of (sequential) RR-SPN (N=0,1,2) and DR algorithms in application to solving a model equation. $\alpha = 10^4$.

RR-SP0* is implemented to the same problem and it converges in eight iterations (three iterations slower than RR-SP2*), see Table 5.4. It means that the advantage of RR-SP2 over RR-SP0(1) is mainly due to RR-SP2* having faster (standard) convergence than its counterparts rather than the boundary effect. In addition, RR-SP1 shows no advantage (slight disadvantage instead) over RR-SP0 and it suggests that the boundary effect of RR-SP1 is not strong enough to make remarkable difference, either positive or negative. In other words, the only difference between RR-SP1 and RR-SP0 is P_i^l , which should be responsible for any performance difference between the two algorithms.

Figs.5.11 and 5.12 illustrate the profile of P_i^l and P_i^q in $\alpha = 10^2$ case. As can be seen, the size of P_i^q is nearly equal to 0 thus the boundary effect of RR-SP2 could hardly help the algorithm gain much advantage over the standard version (RR-SP2*), which has been verified by the numerical result as shown above. In addition, the shape of P_i^l is 'low and thin', suggesting a low $\frac{\alpha}{\beta}$ ratio and weak boundary effect. Note the height of P_i^l is about 0.01 times of that of P_i . Recall in Poisson's equation, $A = \frac{1}{8}$ case, the height of P_i^l (about 0.5, see Figs.4.27, 4.28) is also around 0.01 times of that of P_i (about 40, see Fig.4.5 in Chapter 4). The ratio is approximately equal but the width of P_i^l in Poisson's equation case is far larger.

Note RR-SP0* algorithm amounts to modifying RR-SP0 by replacing $S_i I$ with $S_i^* I$ so that $S_i^* I$ is a constant. RR-SP1* algorithm amount to letting $T_{il} = 0$ in RR-SP1 apart from the simplification about $S_i I$, which makes it identical to RR-SP0* thus the algorithm is

not specifically studied. The algorithm of RR-SP2* replaces T_{iq} in RR-SP2 with $T_{iq}^* := S_j x^2 - x^2 \cdot S_j I$, ($i \neq j$), apart from retaining the simplification about $S_i I$ and T_{il} . The above three algorithms correspond to the standard approach in Chapter 3.

The case of $\alpha = 10^4$ shows much different result from the above two. Firstly, RR-SP2 is no more faster (but equal to) than RR-SP0. Secondly, RR-SP1 shows far slower convergence than the other two algorithms. Thirdly, the standard version of RR algorithms makes remarkable difference from the real algorithms, see Table 5.4, which suggests that the standard approach plays secondary role in this case and the boundary effect is the main factor to influence the numerical result.

As mentioned earlier, RR-SP1 differs from RR-SP0 only by P_i^l , and RR-SP1 in $\alpha = 10^4$ case converges much slower than RR-SP0, so it could be deduced that P_i^l results in the slow convergence of RR-SP1. The profile of P_i^l is displayed in Figs.5.13 and 5.14 together with the profile of P_i^q . It could be seen that the two figures highly resemble Figs.4.27 and 4.28 in terms of shape, and the size of P_i^l and P_i^q in the former two figures is also comparable to that of the latter figures. It means that the model equation and the Poisson's equation should have been equally influenced by P_i^l and P_i^q and performed similar convergence iterations, but the numerical result fails to support it. The most likely reason might be that the solution to the model equation with large α has strong dependence on x hence no more has variable-separation form, so the analysis result drawn from Poisson's equation could not be fully applied to the model equation. Considering RR-SP1 performs poorly (as mentioned earlier, because of P_i^l) and RR-SP2 overcomes it with P_i^q , It could be deduced that P_i^q offsets the negative effect that P_i^l imposes on the convergence and the overall effect of P_i^l plus P_i^q is 0. That explains why RR-SP0 and RR-SP2 achieve the convergence within equal iterations.

Strictly speaking, the analysis result drawn from Poisson's equation could be partially applied to large- α model equation. Assume the solution $u := V + W$ where V has variable-separation form and W does not. Clearly the convergence of V is predictable via the convergence rate calculation so the slow convergence has nothing to do with it. Instead, the unusual convergence in $\alpha = 10^4$ case should be resulted by W . In addition, Fig.5.15 shows that the solution to $\alpha = 10^4$ case mainly changes with x , which means V is much larger than W so the former dominates the entire solution. Note in this case V is a one-dimensional function of x and it could be considered to have variable-separation form. If V does not prevail, the influence of W could be expected to be larger than this case, and the convergence in RR-SP1 might be even slower, as will be shown in next subsection (5.2.2).

To conclude, the real convergence is influenced by both the standard convergence and the boundary effect of the algorithms. The standard convergence nature plays the key role when α is small. As α grows, the strength of the boundary effect keeps increasing. In addition, comparing the three algorithms, RR-SP2 algorithm performs the most stable and fastest convergence among all cases regardless of α . When α is low and the boundary effect is not strong, the standard convergence rate mainly determines the real convergence, and RR-SP2 is proved (both theoretically and numerically) to have lower standard convergence rate than

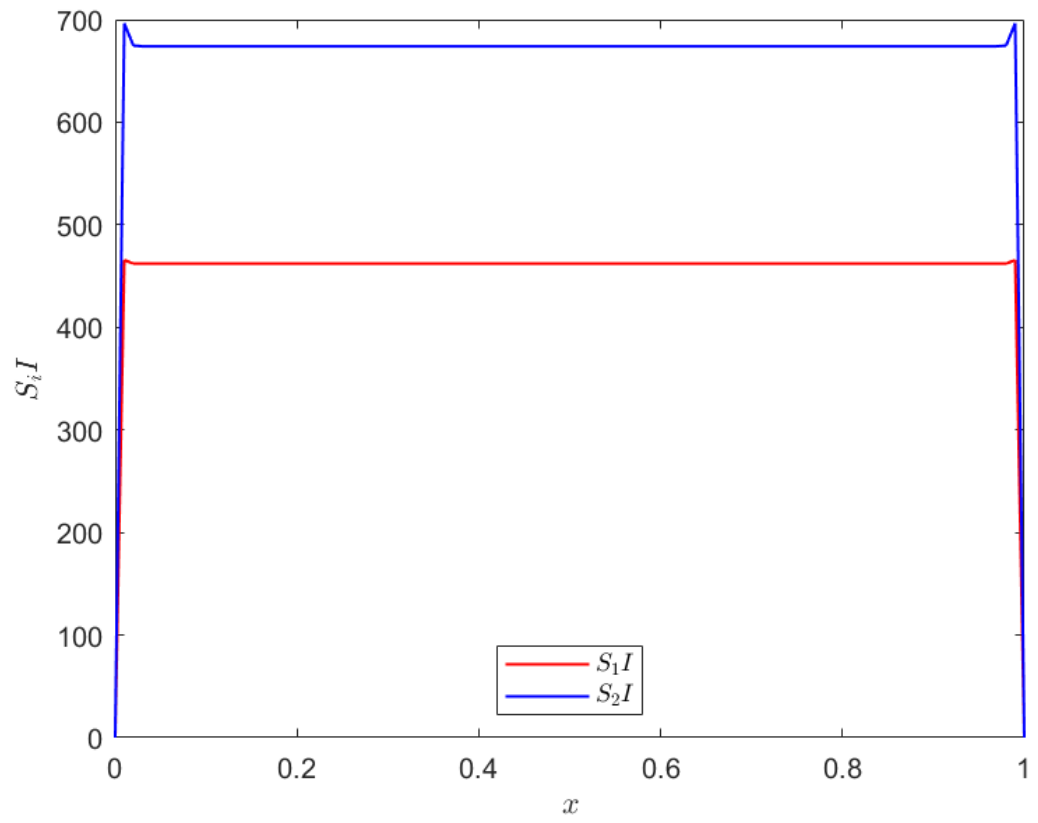


Figure 5.8. The profile of $S_i I$ in application to a model equation, $\alpha = 1$.

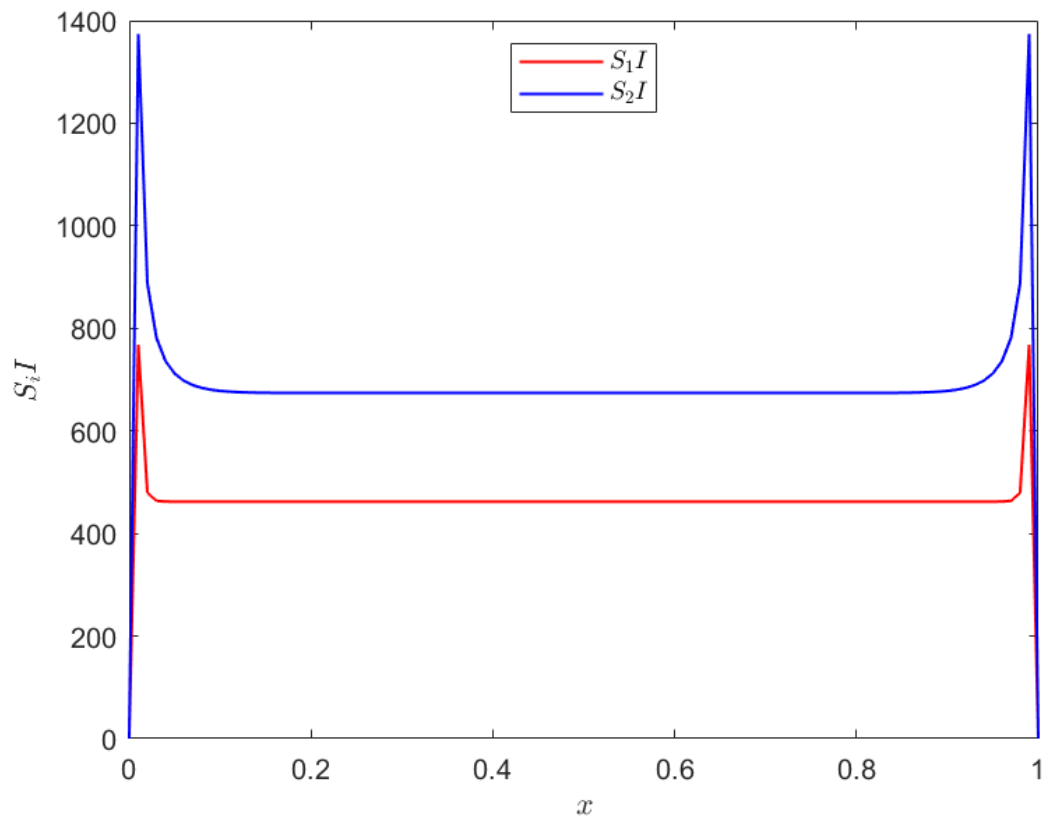


Figure 5.9. The profile of $S_i I$ in application to a model equation, $\alpha = 10^2$.

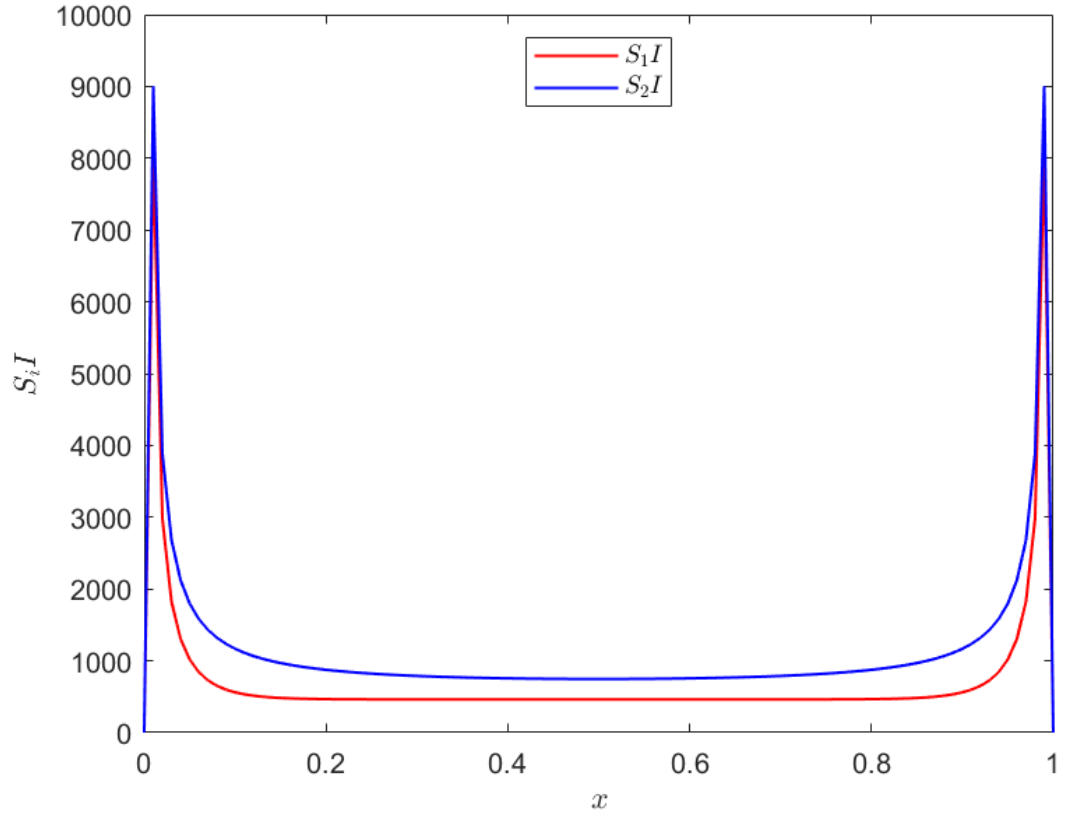


Figure 5.10. The profile of $S_i I$ in application to a model equation, $\alpha = 10^4$.

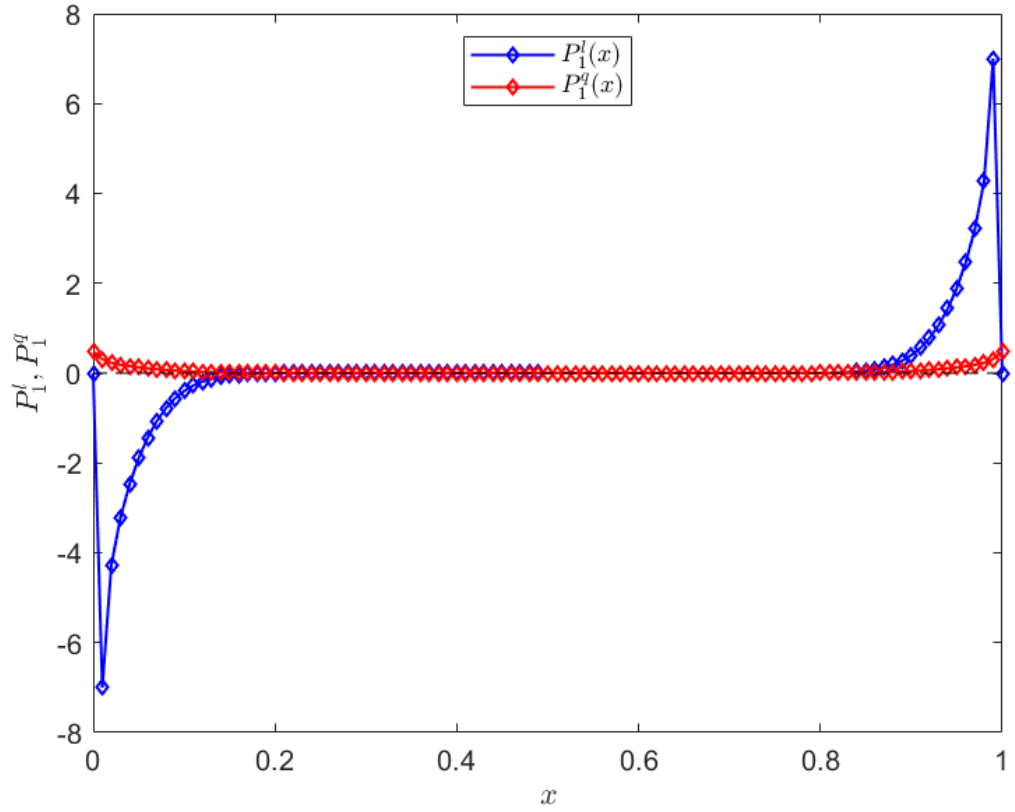


Figure 5.11. The profile of P_1^l and P_1^q in application to the model equation, $\alpha = 10^2$.
 $P_1^l := S_2 x - S_2 I \cdot x - 0$, $P_1^q := (S_2 x^2 - 2x S_2 x + x^2 S_2 I) - (S_2^* x^2 - x^2 S_2^* I)$. $S_2^* x^2$ is obtained numerically with boundaries in x direction set at far place ($x = \pm 2$).

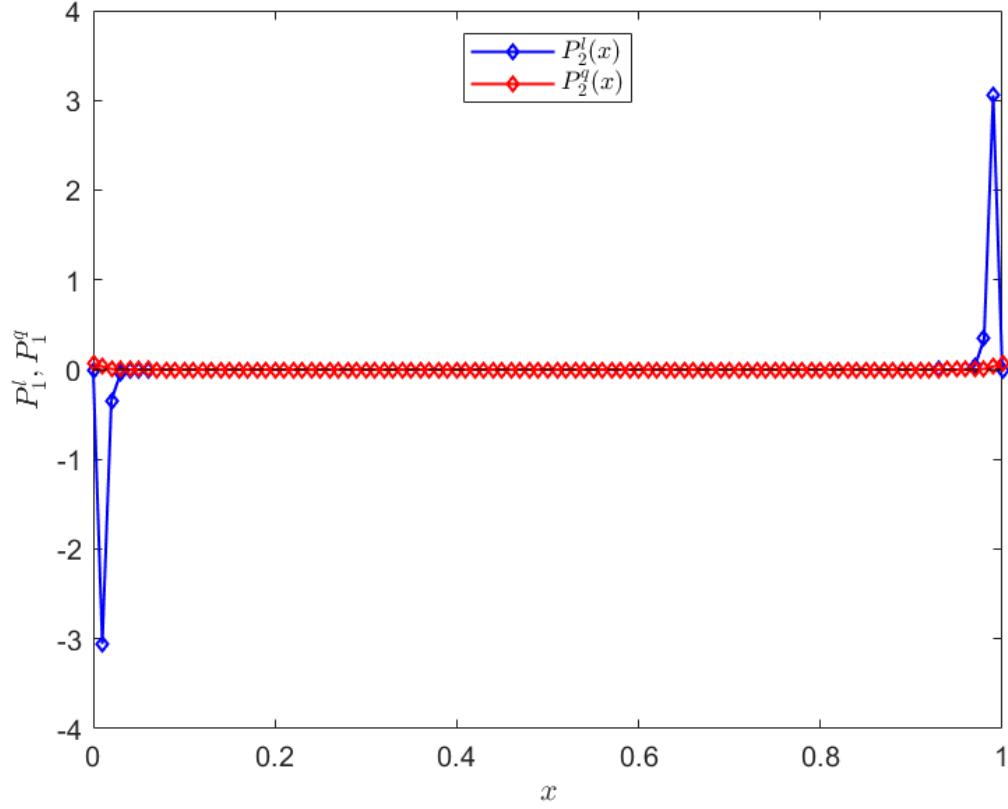


Figure 5.12. The profile of P_2^l and P_2^q in application to the model equation, $\alpha = 10^2$.
 $P_2^l := S_1 x - S_1 I \cdot x - 0$, $P_2^q := (S_1 x^2 - 2x S_1 x + x^2 S_1 I) - (S_1^* x^2 - x^2 S_1^* I)$. $S_1^* x^2$ is obtained numerically with boundaries in x direction set at far place ($x = \pm 2$).

RR-SP0(1) thus it shows faster convergence. When α is high and the boundary effect is strong enough to largely influence the real convergence, the convergence of RR-SP1 may be 'backfired' by the boundary effect of itself but RR-SP2 could offset the negative effect and maintains stable performance.

Both the standard convergence nature and the boundary effect could be considered as reflections of the non-local effect reservation of SP operators in application to given problems. Obviously, in the standard convergence rate calculation, $S_i I$, $S_i x$ and $S_i x^2$ are considered as two-dimensional functions thus the calculation reserves part of the non-local effect of the problem. Additionally, the rest part of the non-local effect is represented by the boundary effect (pike functions). The standard convergence calculation shows that RR-SP2 has lower (standard) convergence rate and the numerical test shows that RR-SP2 has stronger ability to avoid itself being negatively influenced by the boundary effect. The former makes RR-SP2 faster in achieving the convergence and the latter makes it more stable in dealing with complex problem.

5.3.2 Model equation 2

This subsection considers applying the algorithms to the same model equation as Subsection 5.2.1 but with two parameters adjusted: y^* and Re . Both adjustments change the non-local

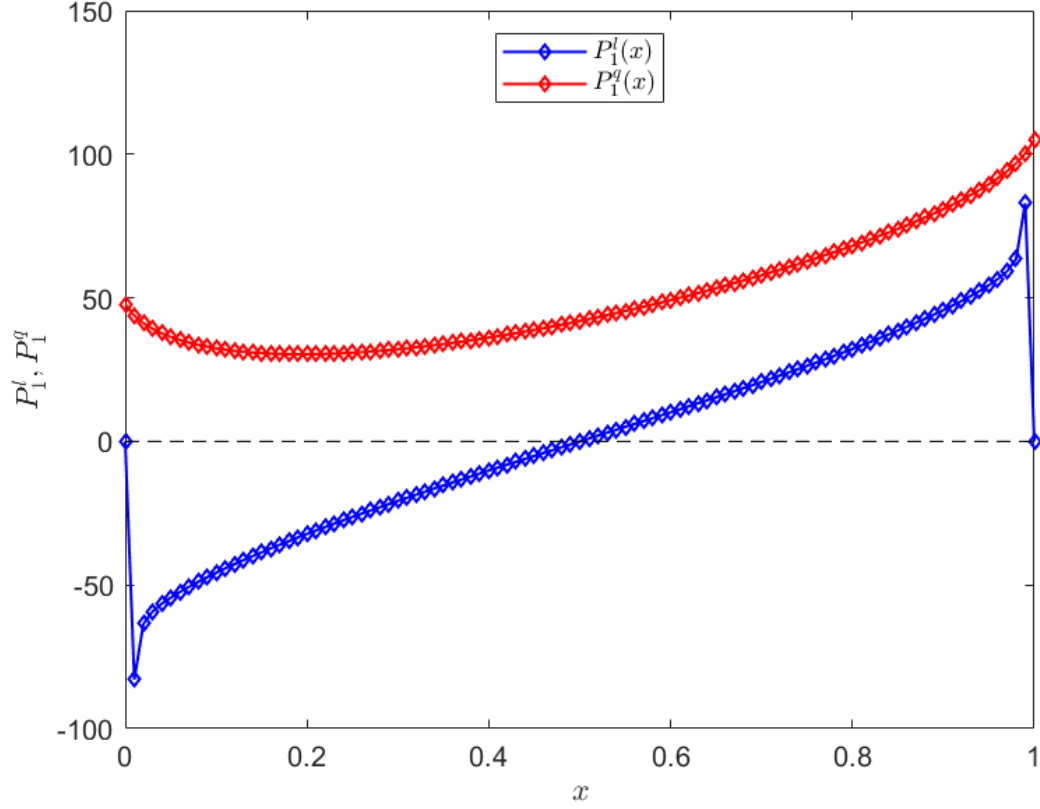


Figure 5.13. The profile of P_1^l and P_1^q in application to the model equation, $\alpha = 10^4$. $P_1^l := S_2x - S_2I \cdot x - 0$, $P_1^q := (S_2x^2 - 2xS_2x + x^2S_2I) - (S_2^*x^2 - x^2S_2^*I)$. $S_2^*x^2$ is obtained numerically with boundaries in x direction set at far place ($x = \pm 2$).

Table 5.5. Convergence iteration of RR-SP0, RR-SP1, RR-SP2 and DR algorithms in application to a model equation, $A = 5 \times 10^{-4}$.

α	1	10^2	10^4
RR-SP0	3	5	10
RR-SP1	3	5	29
RR-SP2	3	4	8
DR	4	7	19

effect strength of the problem on the interface. The additional test aims to check the analysis result drawn from Subsection 5.2.1 so the numerical result is comparatively studied with that from the subsection too.

Consider the case of implementing RR-SPN ($N=0,1,2$) and DR to the same model equations (parameters and mesh scheme all remain) but with interface boundary at lower position: $y^* = 5 \times 10^{-4}$. Moving y^* to nearer-wall position means the non-local effect of the near-wall problem becomes weaker so that the reservation of the non-local effect of S_1 becomes less important. In other words, the problem is closer to one-dimensional to some extent and the difficulty to solve it becomes lower. Table 5.5 confirms it that all algorithms take less iterations to converge, especially DR, RR-SP0 and RR-SP1. This is because the three algorithms are less capable of retaining the non-local nature of the problem hence are more sensitive to the reduction of it.

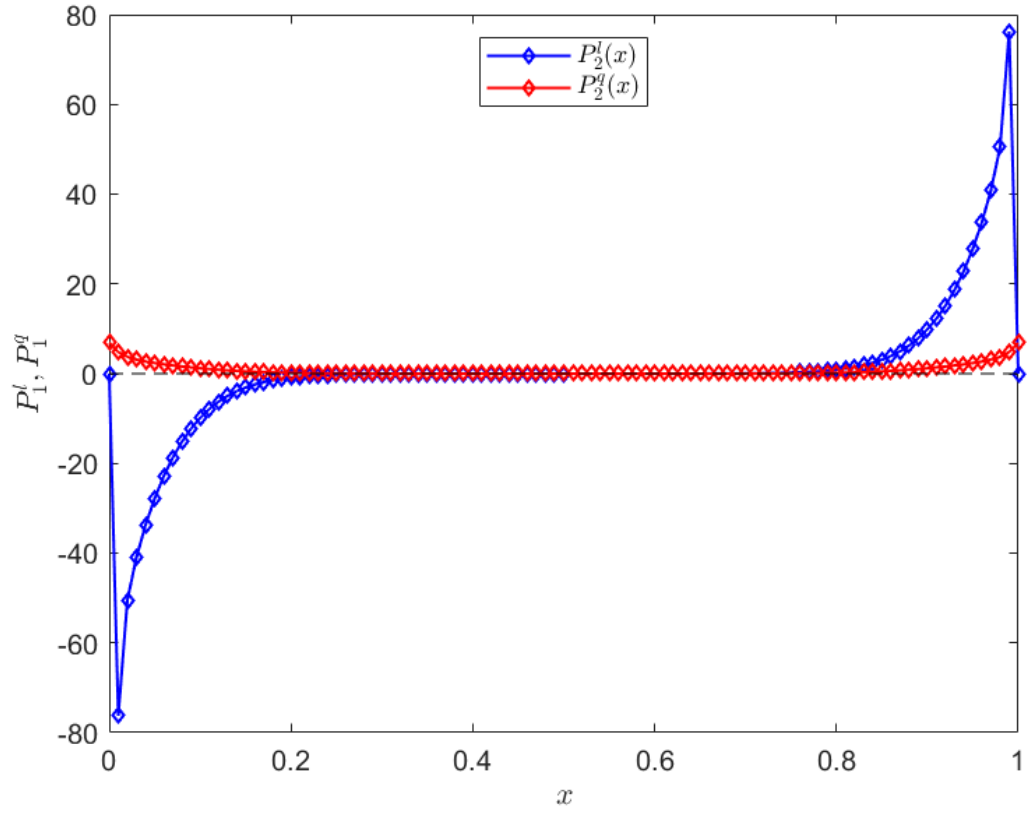


Figure 5.14. The profile of P_2^l and P_2^q in application to the model equation, $\alpha = 10^4$. $P_2^l := S_1 x - S_1 I \cdot x - 0$, $P_2^q := (S_1 x^2 - 2x S_1 x + x^2 S_1 I) - (S_1^* x^2 - x^2 S_1^* I)$. $S_1^* x^2$ is obtained numerically with boundaries in x direction set at far place ($x = \pm 2$).

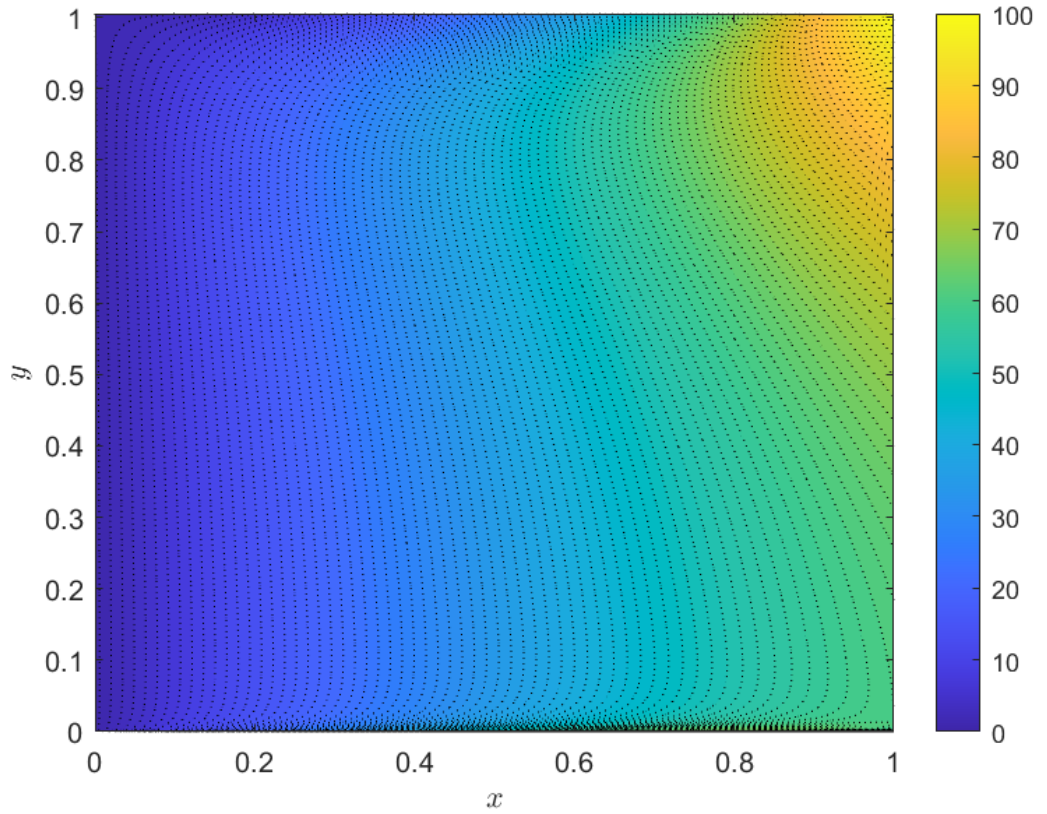


Figure 5.15. The contour plot of the solution to $\alpha = 10^4$ model equation. The solution mainly changes with x .

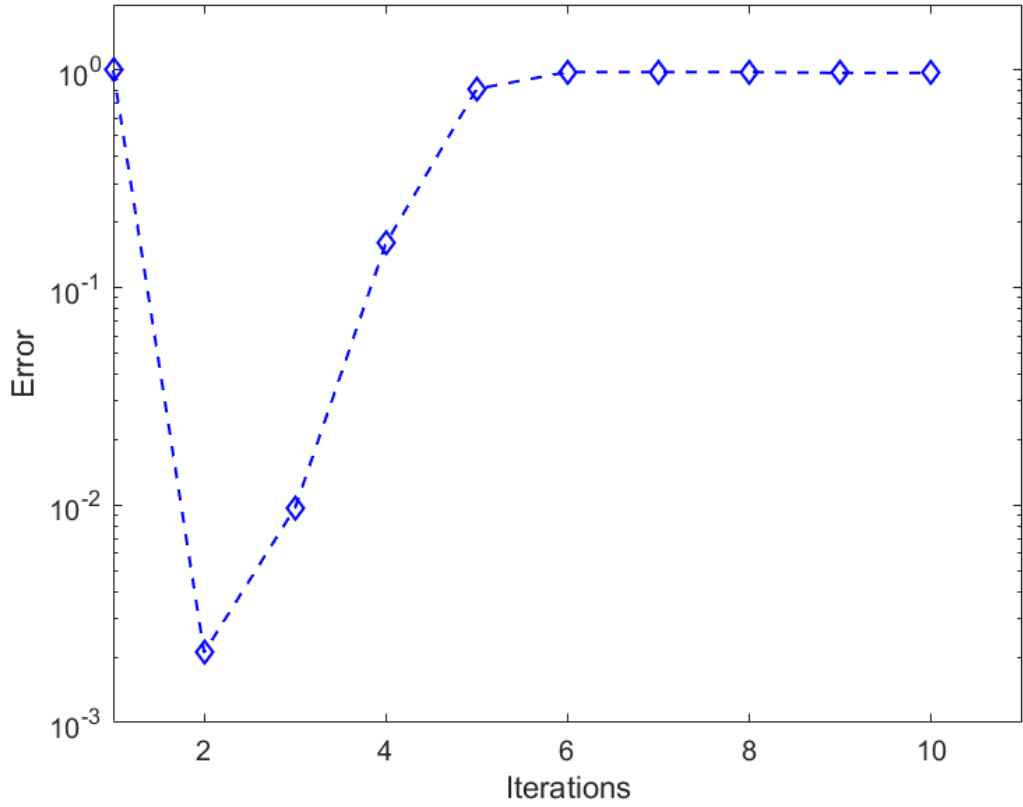


Figure 5.16. The divergence history of RR-SP0(1)* in application to the model equation, $\alpha = 10^6$. The convergence of V is speculated to be achieved within the first 5 iterations.

Next, consider the model equation with $Re = 10^6$ (all other parameters remain). The adjustment mainly affects σ that controls the solution variation in x direction. If Re becomes larger, σ relatively gets smaller and the solution less depends on x with the same α . Therefore, it could be expected that the problem is easier to solve.

Note different mesh scheme in y direction is adopted in the case:

$$y(i) = \delta \left(\frac{\tanh(\frac{R}{2}(i - \frac{n}{2}))}{\tanh(\frac{R}{2})} + 1 \right), \quad i = 0, 1, 2, \dots, n. \quad (5.10)$$

The space step $\Delta h_y(i) := y(i) - y(i - 1)$. The following parameters are given in the test: $n = 401, \delta = 1, R = 20$. The scheme is originally used for generating mesh in cylindrical pipe simulation so the mesh are symmetric along $i = n/2$. In this case, only half generated mesh are used so the final mesh have 202 nodes in total and 125 of them are located in inner region ($y \leq 10^{-3}$) and 78 are in the outer region ($10^{-3} < y \leq 1$). Compared with the mesh scheme (Eqs.(5.7), (5.8)) in Subsection 5.2.1, both have well-refined mesh in the inner region (accommodating over half number of nodes) with appropriate growing ratio, so they should both be able to capture the rapid velocity change in the boundary layer simulation. The numerical result based on the two schemes should be comparable.

The numerical test result is displayed in Table 5.6. Firstly, all cases converge in a few iterations including RR-SP1 that had slow convergence in high- α case of $Re = 10^4$. It proves the

Table 5.6. Convergence iteration of a model equation case. Note 'NA' denotes not applicable.

Parameter	parallel			sequential		
	$\alpha = 1$	$\alpha = 100$	$\alpha = 10^4$	$\alpha = 1$	$\alpha = 100$	$\alpha = 10^4$
RR-SP0	3	4	9	3	3	5
RR-SP1	4	4	9	3	3	5
RR-SP2	3	3	5	3	3	3
DR	NA	NA	NA	3	3	5

problem indeed easier to converge. Next, consider each α respectively. In low- α cases ($\alpha = 1, 100$), (parallel) RR-SP2 shows slight advantage in the convergence rate than the other two, which should be resulted in by the standard convergence nature difference. The $\alpha = 100$ case matches the case of $\alpha = 1$ in Subsection 5.2.1 because they both have $\sigma \sim 10^{-4}$. When α is high (10^4), (both parallel and sequential) RR-SP2 shows remarkable advantage over the other two. The case matches the case of $\alpha = 100$ in Table 5.3 (because they both have $\sigma \sim 10^{-2}$), so the explanation should be the same as well: the advantage is mainly due to the standard convergence rate and is also influenced by the boundary effect. To verify it, (parallel) RR-SP2* and RR-SP0(1)* is implemented to the case of $\alpha = 10^4$, and the convergence iterations are 5 and 9 respectively, which is exactly the same as the result with the boundary effect. It shows that the boundary effect of the algorithms almost has no impact on the convergence. The same test is implemented to sequential RR-SP2* and RR-SP0(1)* as well, and the result is three, four respectively, which shows the boundary effect of RR-SP0(1)* imposes slight impact on the convergence but the standard convergence still plays the leading role. To sum up, the test result with $Re = 10^6$ case is consistent with the previous analysis.

Table 5.6 also compares the iteration numbers between the parallel and the sequential algorithms. It shows that in whichever case, the former takes about twice number of iterations as the latter, as is in Poisson's equation test, see Table 5.2. Therefore, the solution to the model equation must also have (or dominated by) variable-separation form regardless of α . As will be seen in Section 5.3, the solution is speculated to be quasi-one-dimensional with respect to y and x in inner and outer region respectively.

Recall in last model equation test, $\alpha = 10^4$ case, RR-SP1 shows very slow convergence and RR-SP2 overcomes the negative trend and shows satisfactory stability in maintaining the outstanding convergence performance. The poor performance of RR-SP1 is deduced to be associated with part of the solution W that has no variable-separation form, and RR-SP2 proves itself capable of securing the convergence of W within small number of iterations. To test RR-SP2 again, the model equation studied in this subsection is particularly applied to (sequential) RR-SPN and RR-SPN* with $\alpha = 10^6$ so that $\sigma \sim 1$. The test result is shown in Table 5.7. In this test, RR-SP2* proves its superior standard convergence rate and RR-SP2 proves its stability that the convergence is not impeded by the boundary effect. Actually the error of RR-SP2 is slightly less than RR-SP2* although not resulting in iteration number deduction, and it shows that the boundary effect not only does not impede but also boosts the convergence. Therefore, RR-SP2 proves its advantage in having lower convergence rate and better stability again in dealing with complex problem.

In the $Re = 10^6$ case, although RR-SP0(1)* fails to converge, the divergence history of RR-SP0(1)* shows temporary decrease of the error (e_2^n) in the first two iterations. The error history is displayed in Fig.5.16. It suggests that error elimination takes place at the beginning as if the convergence could have been achieved, but shortly afterwards the error line goes up and keeps fluctuating at around 1, which is the symbol of 'mild divergence' (means error could neither be eliminated nor be enlarged). Clearly the algorithm managed to work at the beginning but crashed soon. The reason could be that the algorithm works well with V but does not with W . The early error elimination happens on V and it has actually reached the convergence at around iteration 5. By contrast, the error of W could not be eliminated all the time and the divergence is not visible during the first few iterations. Iteration 2-5 is the transient zone where V is converging but W remains in divergence.

Table 5.7. Convergence iteration of RR-SPN and RR-SPN* in application to a model equation, $\alpha = 10^6$. Note 'DIV' denotes divergence.

	SP0*	SP1*	SP2*	SP0	SP1	SP2
Iterations	DIV	DIV	4	DIV	DIV	4

5.4 Floating interface boundary method

This section introduces a new domain decomposition method to solve the model equations.

Figs.5.17 to 5.19 illustrate the solution profile of the model equations ($Re = 10^6$) with three different α : 10^2 , 10^4 and 10^6 . The increasing α represents stronger non-local effect of the solution in x direction. It could be seen that Fig.5.17 resembles Fig.5.18 where both profiles are mainly in blue (low function value) and only the right bottom corner reserves small area of yellow colour (high function value). The small yellow strip represents extremely high gradient zone in y direction (from top to bottom) in inner region, and it is supposed to simulate the boundary layer in turbulence flow. The outer region witnesses much milder transition in colour and the transition mainly happens in x direction. This region is supposed to simulate turbulence main flow. Roughly speaking, inner region could be approximated as a one-dimensional problem in y direction with super-high velocity gradient and outer region could be approximated as a one-dimensional problem in x direction with very low gradient. Reducing the inner problem to one-dimensional with respect to y fits the solution nature of the near-wall flow (exactly what ANDD algorithms do), but the outer region could suffer more from accuracy loss because it is basically a problem of x . In other words, outer region has stronger non-local effect than inner region. Recall it is previously stated that the boundary effect of $S_i I$ mostly represents the non-local effect of the operator (see end of Subsection 4.1.2 in Chapter 4). In the model equation tests, Fig.5.9 shows that $S_2 I$ has higher pike (boundary effect) than $S_1 I$, so the outer region should have stronger non-local effect. Figs.4.23 and 4.24 also confirm this: $A = 1/8$ has larger pike in $S_2 I$ than in $S_1 I$ and $A = 7/8$ has larger pike in $S_1 I$ than in $S_2 I$.

Fig.5.19 is remarkably different from the above two figures because it has the yellow zone

much bigger in size and located at top right of the domain (out of inner region). It means that the solution in inner region might still be largely influenced by the gradient in y direction but the gradient is much lower (might not dominate) thus not capable of representing boundary layer. By contrast, the outer region shows higher gradient in x direction: from right to left, so it becomes closer to a one-dimensional problem with respect to x . To summarise, inner region could not be simplified as one-dimensional problem of y in the case and outer region becomes closer to a one-dimensional problem with respect to x . The equation no longer simulates turbulence flow.

It is not surprising to see Fig.5.19 highly resembles Fig.5.15 because they both have $\alpha \sim 1$. The difference between the two figures is that the latter has bigger bright colour zone (or simply yellow zone) at the right top corner, which means the velocity gradient in x direction of the case is higher and the solution is closer to one-dimensional. Recall previous study assumes the solution to the case is made of two parts: $u = V + W$ where V could be solved with separation of variables and W could not. It could be speculated that V makes up higher portion of the solution as shown in Fig.5.15 so the negative effect of W on the convergence is weaker. In that case, RR-SP1 could still converge and RR-SP0 shows even better convergence, see Table 5.3. When W composes larger portion in the solution, as is the case as shown in Fig.5.19, the convergence condition could be worse or even divergence, see Table 5.7.

As has been analysed, in whichever case, the solution in outer region mainly varies with x as a result of strong non-local nature. Consequently, the solution gradient is low in y direction and the solution changes very slowly in this direction. Now considering outer region Ω_2 alone, throughout the iterative computing it has three exact boundary conditions (at $x = 0, 1, y = 1$) and one boundary condition not exact (at $y = A$). Presume the error on the interface is $e_2^0(x)$ before the iteration launches, and after the first iteration, $e_2^0(x)$ must propagate to other nodes in Ω_2 . Since the BVP is a quasi-one-dimensional problem with respect to x , the error mainly spreads along x direction and hardly travels along y direction, which means the solution $u_2^1(x, A + \delta y)$ should not be largely influenced by $e_2^0(x)$. The larger δy , the smaller the influence should be. If the propagation of e_2^0 could fade away within certain scope of δy , the solution beyond δy should be exact.

The method of floating interface boundary is proposed. The idea is that in the first iteration, define an imaginary interface several nodes below the real interface, denoted by $y_i^* := y^* - \delta y$. At the moment the outer region is slightly larger than it should haven been, and the real interface y^* is in outer region. Let the algorithm (whatever as long as it is convergent) run one iteration in outer region, and the value on $y = y^*$ in outer region should be very close to exact value. Then resume the real interface and let the algorithm run in inner region. So far the 1st iteration is completed. If the interface value at $y = y^*$ is close enough to exact value, the whole solution should already be exact. Afterwards let the algorithm run 1 more iteration to check error. It is expected that the algorithm will converge after the completion of iteration 2. As the interface boundary slightly moves around during the iterations, it is called 'floating interface boundary method'. It could be combined with any convergent ENDD algorithm, such as simply RR-SP0.

To approximately elaborate the error propagation in Ω_2 , consider the following one-dimensional BVP:

$$\begin{aligned}\mathcal{L}_y e_2 &= 0, \\ e_2(A) &= e_2^0, \quad e_2(1) = 0.\end{aligned}\tag{5.11}$$

The above BVP represents the error problem in Ω_2 and it is simplified to one-dimensional. To compare, similar error BVP could be defined in Ω_1 :

$$\begin{aligned}\mathcal{L}_y e_1 &= 0, \\ e_1(A) &= e_2^0, \quad e_1(0) = 0.\end{aligned}\tag{5.12}$$

If let $e_2^0 = 1$, the above BVPs could be used to obtain $S_1^* I$ and $S_2^* I$ by computing $e_{1y}(A)$ and $-e_{2y}(A)$. Instead of computing them, what is concerned here is the solution to e_1 and e_2 . Now consider computing the above BVPs with $Re = 10^6$. Let $L_y := \frac{\mu}{\rho} \frac{d}{dy} + \beta \frac{d}{dy} + \gamma$, where μ, β, γ are all defined the same as previous model equations. δy is 10 nodes so that the imaginary y_i^* is at 115th node. The solution to e_1 and e_2 is displayed in Fig.5.20.

The figure shows that the error is at maximum at $y = y_i^*$ and propagates to both sides of the domain. In Ω_2 , the error drops so quickly that after 10 nodes it reduces to almost 0. In Ω_1 , the error falls too but at much slower rate. It takes about half of the nodes in the region to let the interface error reduce to almost 0. This is consistent with the previous analysis: in inner region error mainly spreads along y so e_2^0 travels deep into the centre part of Ω_1 ; in outer region error mainly spreads along x and hardly transports along y direction, so the error is almost 'isolated' in the narrow strip along $y = y_i^*$.

To verify the method, the model equation ($Re = 10^6$) with $\alpha = 1, 10^2$ and 10^4 is solved by RR-SP0 with floating interface boundary. The test conditions remain all the same as above test. The result is that all three cases converge in two iterations, while RR-SP0 converges in three, three, five iterations respectively, see Table 5.6. Even RR-SP2 converges in three iterations, which is one iteration slower. What is more, the computing cost of 'floating RR-SP0' is the same as RR-SP0, which requires only the pre-computation of $S_i I$. By contrast, the implementation of RR-SP2 requires additionally the computation of $S_i x$ and $S_i x^2$ apart from $S_i I$.

The high efficiency of the method is closely associated with the nature of near-wall turbulent flow: boundary layer has high gradient in y direction and main flow is mainly a function of x . Therefore, the method is specifically aimed to solve near-wall turbulent flow but should not be expected to work effectively with other problems such as Poisson's equation.

5.5 Summary

This chapter conducts numerical tests to verify the convergence analysis result of RR algorithms, and to test the feasibility of applying the RR algorithms to resolving near-wall

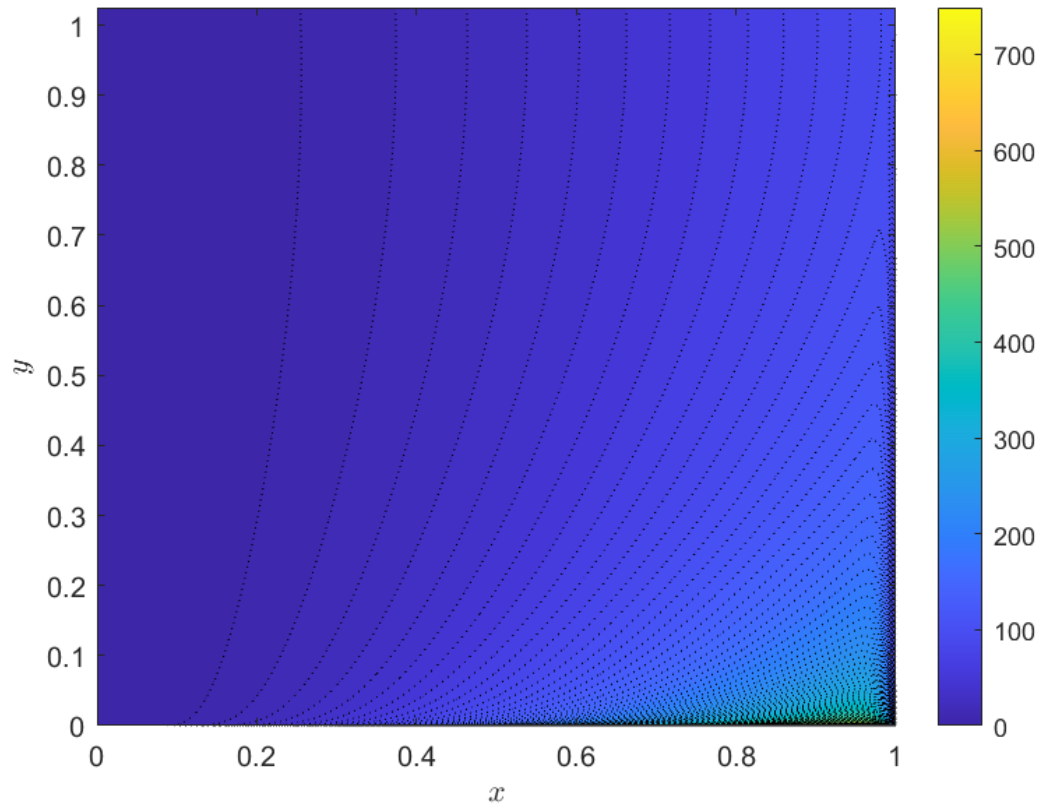


Figure 5.17. Solution profile of the model equation ($Re = 10^6$). $\alpha = 10^2$.

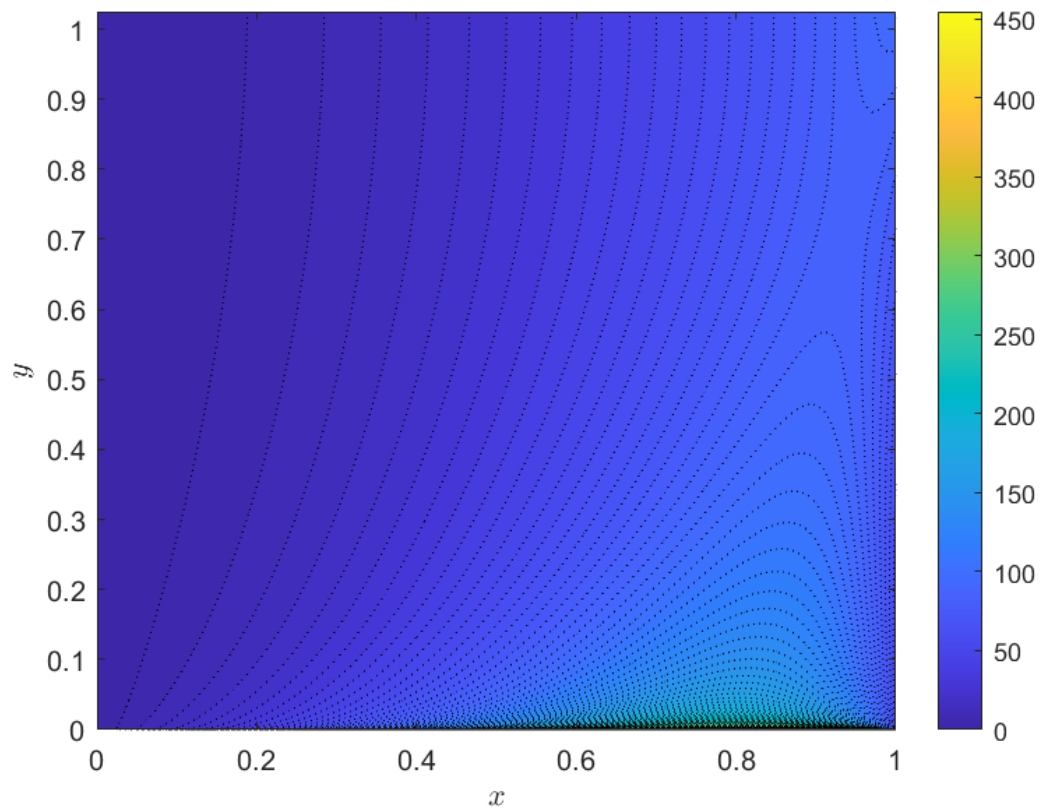


Figure 5.18. Solution profile of the model equation ($Re = 10^6$). $\alpha = 10^4$.

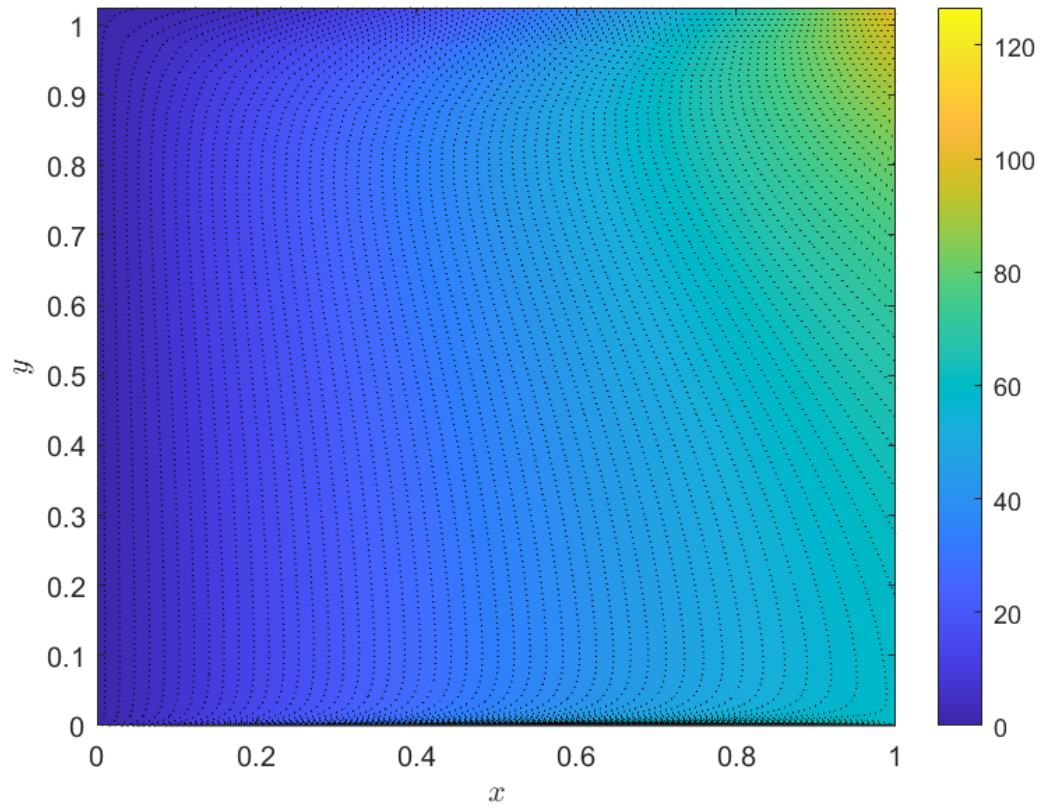


Figure 5.19. Solution profile of the model equation ($Re = 10^6$). $\alpha = 10^6$.

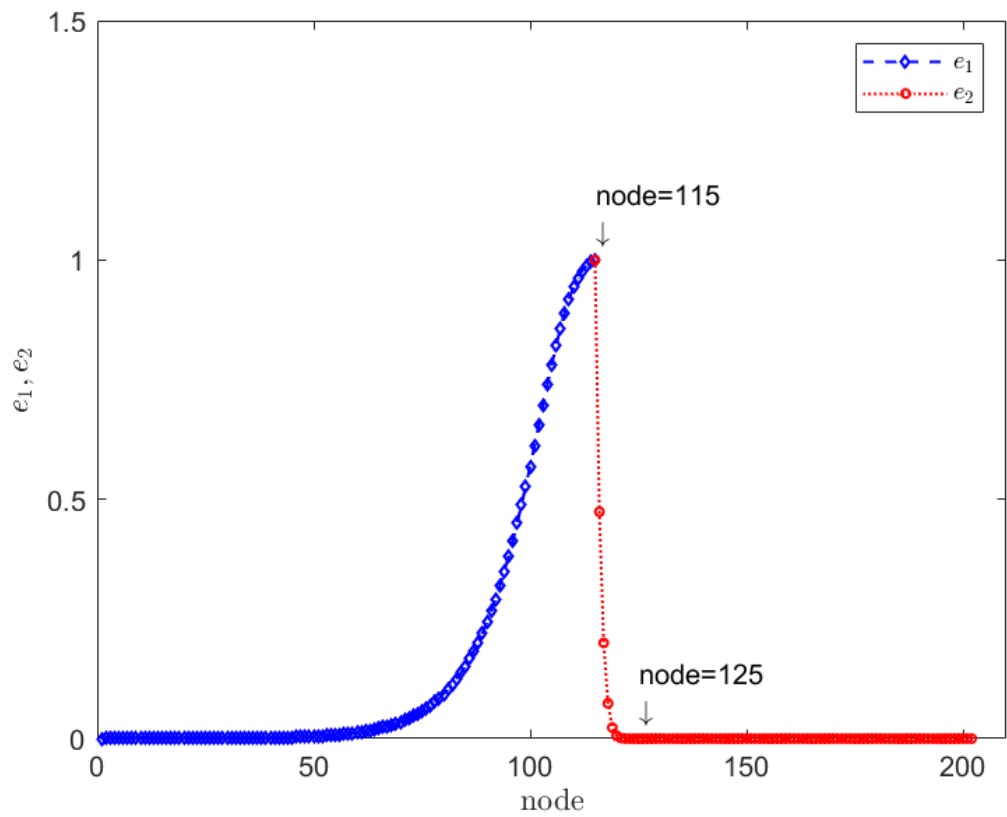


Figure 5.20. Profile of e_1 and e_2 , which are defined in Eqs.(5.11) and (5.12).

turbulence problem.

Numerical test result on Poisson's equation fully coincides with theoretical result. Next, numerical tests are extensively conducted to two model equations with different solution variation on x direction. Greater variation accounts for stronger non-local nature of the problem. Since the algorithms are designed to deal with non-local problems, they should not perform much differently when the non-local effect is weak. The numerical test confirms the analysis: all the three algorithms converge in a few iterations and RR-SP2 shows slight advantage, which is due to lower standard convergence rate. As the non-local effect gets stronger, the advantage of RR-SP2 becomes greater and this is because the boundary effect improves the convergence. Overall, RR-SP2 shows both superior stability and efficiency in solving the model equations.

Section 5.4 proposes a new algorithm based on movable interface boundary, which is proven to be as efficient as to achieve the convergence in two iterations. The high efficiency of the algorithm is found to be directly associated with the physical nature of near-wall turbulent flow: normal derivative of function dominates in near-wall region and tangential derivative dominates in off-wall region.

Chapter 6

Analysis and implementation of GMRES

This chapter discusses one of the most effective iterative algorithms to solve large sparse linear systems: the generalized minimum residual method [105], which is used in the project to solve the discrete BVPs as introduced in previous chapters. The method of GMRES usually goes with preconditioning procedures to achieve faster convergence. There are various preconditioning techniques applicable to GMRES and the so-called ILU(0) is elaborated in the chapter as it is implemented in the project.

The computing cost of solving a linear system is directly associated with the working mechanism of the solver that is used. For example, the size of the linear system could play an important role in determining the computing cost of applying direct solving method such as LU factorization, but it is usually not true for more complicated iterative solving method including GMRES. Therefore, the object of studying GMRES is to understand how it works to solve large sparse systems and find approach to evaluate the computing cost of applying GMRES to NDD method as opposed to one-block method.

6.1 Introduction of GMRES as a projection method

Consider a large sparse linear problem where $A \in \mathbb{R}^{n \times n}$ and $x, b \in \mathbb{R}^n$:

$$Ax = b. \tag{6.1}$$

This section introduces so-called projection method to solve the problem. The basic idea of projection method is to find an approximation of x (denoted by x_m) in the search subspace \mathcal{K}_m so that $b - Ax_m \perp \mathcal{L}_m$, in which both \mathcal{K}_m and \mathcal{L}_m are two m -dimensional subspace in \mathbb{R}^n . The approximated solution has the structure: $x_m = x_0 + \delta$, in which x_0 is an arbitrary initial guess and $\delta \in \mathcal{K}_m$. To extract such a x_m from \mathcal{K}_m , a typical technique is to impose m orthogonality conditions on the subspace \mathcal{K}_m so that for example, the residual vector $b - Ax_m$ is orthogonal to m linearly independent vectors, which form the so-called constraint subspace \mathcal{L}_m [106]. As stated above, this is the basic framework of so-called Petrov-Galerkin conditions. Particularly, when $\mathcal{L}_m = \mathcal{K}_m$, the Petrov-Galerkin conditions

are called Galerkin conditions [106].

$$\mathbf{x}_m = \mathbf{x}_0 + \boldsymbol{\delta}, \quad \boldsymbol{\delta} \in \mathcal{K}_m, \quad (6.2)$$

$$(\mathbf{b} - \mathbf{A}\mathbf{x}_m, \boldsymbol{\omega}) = 0, \quad \forall \boldsymbol{\omega} \in \mathcal{L}_m. \quad (6.3)$$

There are as many projection-method-based algorithms as the selections of \mathcal{K}_m . Particularly, Krylov subspace method is a method for which the search subspace \mathcal{K}_m is defined as Krylov subspace [107]:

$$\mathcal{K}_m(A, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \dots, \mathbf{A}^{m-1}\mathbf{r}_0\}. \quad (6.4)$$

where $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$. From now on, $\mathcal{K}_m \equiv \mathcal{K}_m(A, \mathbf{r}_0)$ would represent Krylov subspace of dimension m only unless otherwise indicated.

The implementation of GMRES usually starts from running Arnoldi algorithm [108]. The importance of Arnoldi algorithm largely lies in its property that it produces an orthogonal basis (denoted by \mathbf{V}_m) of \mathcal{K}_m , so that $\boldsymbol{\delta}$ could be expressed as $\boldsymbol{\delta} = \mathbf{V}_m\mathbf{y}_m$, where $\mathbf{y}_m \in \mathbb{R}^m$. Essentially, the algorithm uses (modified) Gram-Schmidt method to orthogonalize the non-orthonormal basis of \mathcal{K}_m and the relationship between \mathbf{A} and \mathbf{V}_m holds: $\mathbf{H}_m \equiv \mathbf{V}_m^T \mathbf{A} \mathbf{V}_m$, where \mathbf{H}_m is an $m \times m$ upper Hessenberg matrix. The Hessenberg matrix is not only a by-product of the orthogonalization operation on \mathcal{K}_m , as will be seen, it is also the matrix representation of \mathbf{A} projected in Krylov subspace \mathcal{K}_m with \mathbf{V}_m as basis and it contains important information of eigenvalues of matrix \mathbf{A} . In addition, there could be other ways to obtain the orthogonal basis of \mathcal{K}_m : Householder method and Givens rotation for example. The following discussion would demonstrate that modified Gram-Schmidt(MGS) has its unique advantage over all others in application to GMRES method.

There are two classes of projection methods in term of \mathcal{L}_m : $\mathcal{L}_m := \mathcal{K}_m$ and $\mathcal{L}_m := \mathbf{A}\mathcal{K}_m$, namely orthogonal and oblique projection respectively [106]. In the first case, Galerkin conditions becomes $(\mathbf{b} - \mathbf{A}\mathbf{x}_m, \boldsymbol{\omega}) = 0, \forall \boldsymbol{\omega} \in \mathcal{K}_m$ and it yields to the estimation of $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}$, where $\mathbf{y} = \mathbf{H}_m^{-1} \|\mathbf{r}_0\| \mathbf{e}_1$ and \mathbf{e}_1 is the first column of $m \times m$ identical matrix [109]. This directly leads to the full orthogonalization method (FOM), an algorithm highly similar (but less efficient than) to GMRES. In the second case when $\mathcal{L}_m = \mathbf{A}\mathcal{K}_m$, the estimation \mathbf{x}_m could be found in such a way that the residual norm $\|\mathbf{b} - \mathbf{A}\mathbf{x}_m\|_2$ is minimized over $\mathbf{x}_0 + \mathcal{K}_m$. Actually this is the optimal process of a least square problem because $\mathbf{x}_m \in \mathcal{K}_m$ and $m < n$. Together with Arnoldi algorithm, the process of solving the least square problem directly forms the standard GMRES.

The convergence of GMRES is guaranteed that it must converge in at most n iterations when \mathbf{A} is a $n \times n$ matrix. In this case, the solution is exact. However, it is rarely practical because n is usually very large and GMRES is expected to converge with acceptable error in m iterations (sometimes re-start after m iterations) where m is a pre-determined parameter and $m \ll n$. It could be proven that the rate of convergence is highly related to the eigenvalue distribution of \mathbf{A} : higher clustered distribution of eigenvalues makes GMRES converge faster, which means

the convergence could be achieved within less iterations.

6.2 Derivation of GMRES

6.2.1 Arnoldi algorithm

In both Petrov-Galerkin and Galerkin conditions, the estimated solution x_m is a member of \mathcal{K}_m , therefore an orthogonal basis of the subspace \mathcal{K}_m is always useful in revealing the structure of the subspace and simplifying the analysis due to the orthogonality of the basis. With Galerkin condition, $\mathcal{L}_m = \mathcal{K}_m$, intuitively this is simpler because the search subspace and constrain subspace share the common orthogonal basis that could be obtained by Arnoldi algorithm.

Arnoldi algorithm was firstly proposed in 1951 to decompose a dense matrix into Hessenberg matrix with unitary transformation. Then it was discovered that the Hessenberg matrix also offers good approximation of the eigenvalues of the decomposed matrix even if it is large and sparse. What is more, the unitary matrix V actually builds up the orthogonal basis of Krylov subspace $\mathcal{K}_m(A, v) \equiv \{v, Av, A^2v, \dots, A^{m-1}v\}$, which is exactly the search (also constraint) subspace in GMRES (FOM). Finally, from technical viewpoint, Arnoldi algorithms using Gram-Schmidt method produces the Hessenberg matrix and unitary matrix in an 'one-column-by-another' manner, which means it could stop at any iteration with an incomplete decomposition product. It makes it convenient for the implementation of re-started GMRES.

With classic Gram-Schmidt (CGS) method, Arnoldi algorithm reads as follows:

ALGORITHM: Arnoldi algorithm (CGS)

1. Choose a vector r_1 , let $v_1 = r_1 / \|r_1\|_2$
2. For $j = 1, 2, \dots, m$, Do

$$\begin{aligned}
 h_{i,j} &= (Av_j, v_i), i = 1, 2, \dots, j \\
 w_j &= Av_j - \sum_{i=1}^j h_{i,j} v_i \\
 h_{j+1,j} &= \|w_j\|_2 \\
 \text{if } h_{j+1,j} &= 0, \text{ then stop} \\
 \text{else } v_{j+1} &= w_j / h_{j+1,j}
 \end{aligned}$$

End Do

The matrix representation of the above orthogonalization process is as follows:

$$AV_m = V_{m+1}\bar{H}_m. \quad (6.5)$$

The unitary matrix composed of v_1, v_2, \dots, v_m defined by Arnoldi algorithm is denoted by V_m . The upper Hessenberg matrix whose non-zero elements are determined by Arnoldi algorithm is denoted by \bar{H}_m . The dimension of the upper Henssenberg matrix is $(m+1) \times m$.

Expand the matrix formation:

$$V_m = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 & \dots & \mathbf{q}_m \end{bmatrix} \quad (6.6)$$

$$V_{m+1} = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 & \dots & \mathbf{q}_m & \mathbf{q}_{m+1} \end{bmatrix} \quad (6.7)$$

$$\bar{H}_m = \begin{bmatrix} h_{11} & h_{12} & & & & h_{1m} \\ h_{21} & h_{22} & & & & h_{2m} \\ 0 & h_{32} & h_{33} & & & h_{3m} \\ & 0 & h_{43} & h_{44} & & h_{4m} \\ \dots & \dots & 0 & & & \dots \\ 0 & & \dots & & h_{m,m-1} & h_{m,m} \\ 0 & \dots & 0 & \dots & 0 & h_{m+1,m} \end{bmatrix}. \quad (6.8)$$

When $h_{j+1,j} = 0$, it marks the breakdown of the orthogonalization process or when $m = n$, it marks the full orthogonalization of n -order Krylov subspace. The latter happens when $h_{j+1,j} = 0$ at the last iteration of Arnoldi algorithm, and there would be no new column generated to complement V_m because $w_j = 0$. The matrix representation is

$$AV_n = V_n H_n, \quad (6.9)$$

where H_n is a $n \times n$ matrix whose non-zero element h_{ij} has indices $(i-1) \geq j$.

What is more concerned is in general case, where there is no breakdown, no $m = n$, the relationship between AV_m and $V_m H_m$. To see this, transform Eq.(6.5) into

$$AV_m = V_{m+1} \bar{H}_m = V_m H_m + h_{m+1,m} \mathbf{q}_{m+1} e_m^T, \quad (6.10)$$

where e_m^T is the m -th column of $m \times m$ identical matrix. In other words, the last column of $V_m H_m$ plus vector $h_{m+1,m} \mathbf{q}_{m+1}$ is equal to the last column of $V_{m+1} \bar{H}_m$. Note the added vector $h_{m+1,m} \mathbf{q}_{m+1}$ is equal to w_m in Arnoldi iteration. Meanwhile, the latest-generated orthogonal vector \mathbf{q}_{m+1} is actually normalized w_m as shown in Arnoldi process, so when Arnoldi algorithm breaks down or n -order Krylov subspace has been fully orthogonalized, no new orthogonal basis is produced and the term $h_{m+1,m} \mathbf{q}_{m+1} e_m^T$ vanishes, and Eq.(6.9) naturally holds true.

Next consider left-multiplying V_m^T on both sides of Eq.(6.10) and due to orthogonality of V_k , it yields to

$$V_m^T AV_m = H_m, \quad (6.11)$$

where H_m is equal to \bar{H}_m excluding the last row whose only non-zero element is $h_{m+1,m}$,

see below:

$$\mathbf{H}_m = \begin{bmatrix} h_{11} & h_{12} & \dots & & h_{1m} \\ h_{21} & h_{22} & \dots & & h_{2m} \\ 0 & h_{32} & h_{33} & \dots & h_{3m} \\ & 0 & h_{43} & h_{44} & \dots & h_{4m} \\ \dots & & \dots & & \dots \\ 0 & \dots & 0 & \dots & h_{m,m-1} & h_{m,m} \end{bmatrix}. \quad (6.12)$$

Rearranging Eq.(6.11) yields to the similarity transformation $\mathbf{V}_m^T \mathbf{A} \mathbf{V}_m = \mathbf{H}_m$, which indicates that \mathbf{H}_m has the same eigenvalues as \mathbf{A} . Due to the simpler structure of \mathbf{H}_m , it is easier to obtain the eigenvalues of \mathbf{A} (also \mathbf{H}_m) by numerical method like QR algorithm. As will be seen in Section 6.4, the distribution of eigenvalues of \mathbf{A} plays a crucial role in estimating the convergence rate of GMRES.

Arnoldi algorithm could also be achieved via modified Gram-Schmidt method, which is mathematically equivalent to the one by CGS method. The former is more reliable due to better elimination of round-off error in computing process. It takes the form as follows:

ALGORITHM: Arnoldi algorithm (MGS)

1. Choose a vector r_1 , let $v_1 = r_1 / \|r_1\|_2$
2. For $j = 1, 2, \dots, m$, Do

Compute $w_j := \mathbf{A}v_j$
 For $i = 1, \dots, j$, Do
 $h_{ij} = (w_j, v_i)$
 $w_j := w_j - h_{ij}v_i$
 End Do
 $h_{j+1,j} = \|w_j\|_2$
 if $h_{j+1,j} = 0$, then stop
 else $v_{j+1} = w_j / h_{j+1,j}$

End Do

6.2.2 Full orthogonalization method

Consider the simpler case first where $\mathcal{K}_m = \mathcal{L}_m = \text{span}\{r_0, Ar_0, \dots, A^{m-1}r_0\}$. With the orthogonal basis $v_i (i = 1, 2, \dots, m)$ of Krylov subspace produced by Arnoldi method, the

Galerkin condition requires that $\mathbf{b} - \mathbf{A}\mathbf{x}_m \perp \mathcal{K}_m$:

$$\begin{aligned}
& \mathbf{b} - \mathbf{A}\mathbf{x}_m \perp \mathcal{K}_m, \\
& \rightarrow \mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m) \perp \mathcal{K}_m, \\
& \rightarrow \mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y}_m \perp \mathcal{K}_m, \\
& \rightarrow \mathbf{V}_m^T(\mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y}_m) = \mathbf{0}, \\
& \rightarrow \mathbf{V}_m^T\mathbf{r}_0 - \mathbf{H}_m\mathbf{y}_m = \mathbf{0}, \quad (\text{by Eq.(6.11),}) \\
& \rightarrow \beta\mathbf{e}_1 - \mathbf{H}_m\mathbf{y}_m = \mathbf{0}, \quad (\text{by orthogonality of } \mathbf{V}_m,)
\end{aligned} \tag{6.13}$$

where $\beta := \|\mathbf{r}_0\|_2$. Then, the estimated solution $\mathbf{y}_m = \beta\mathbf{H}_m^{-1}\mathbf{e}_1$.

ALGORITHM: Full Orthogonalization Method

1. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, let $v_1 = \mathbf{r}_0/\beta$, where $\beta := \|\mathbf{r}_0\|_2$
2. Arnoldi process: For $j = 1, 2, \dots, m$, Do

$$\begin{aligned}
& h_{i,j} = (\mathbf{A}v_j, v_i), i = 1, 2, \dots, j \\
& \mathbf{w}_j = \mathbf{A}v_j - \sum_{i=1}^j h_{i,j}v_i \\
& h_{j+1,j} = \|\mathbf{w}_j\|_2 \\
& \text{if } h_{j+1,j} = 0, \text{ then stop, let } m := j \\
& \text{else } v_{j+1} = \mathbf{w}_j/h_{j+1,j}
\end{aligned}$$

End Do

3. Find the solution: $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m$, where $\mathbf{y}_m = \beta\mathbf{H}_m^{-1}\mathbf{e}_1$.

Here is the error estimation: the residual generated by the approximated solution \mathbf{x}_m is

$$\begin{aligned}
\mathbf{b} - \mathbf{A}\mathbf{x}_m &= \mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m), \\
&= \mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y}_m, \\
&= \beta\mathbf{v}_1 - \mathbf{V}_m\mathbf{H}_m\mathbf{y}_m - h_{m+1,m}\mathbf{q}_{m+1}\mathbf{e}_m^T\mathbf{y}_m, \quad (\text{by Eq.(6.10),}) \\
&= -h_{m+1,m}\mathbf{q}_{m+1}\mathbf{e}_m^T\mathbf{y}_m, \quad (\text{by } \mathbf{H}_m\mathbf{y}_m = \beta\mathbf{e}_1.)
\end{aligned} \tag{6.14}$$

6.2.3 Standard GMRES and restarted GMRES

Consider Petrov-Galerkin condition applied to the case when \mathcal{K}_m is the m -dimensional Krylov subspace with $\mathbf{v}_0 := \mathbf{r}_0$. It could be proved that the estimated solution \mathbf{x}_m satisfying the condition minimizes the 2-norm of the residual vector $\mathbf{b} - \mathbf{A}\mathbf{x}$ over $\mathbf{x}_0 + \mathcal{K}_m$. Therefore, the key problem turns into finding the optimal solution to a least square problem:

$$\min_{\delta \in \mathcal{K}_m} \|\mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \delta)\|_2. \tag{6.15}$$

Obviously δ could be expressed as $\delta = \mathbf{V}_m\mathbf{y}_m$, as is performed previously.

Recalling the equality relationship Eq.(6.5), the least square problem could be simplified as follows:

$$\min_{\delta \in \mathcal{K}_m} \|b - A(x_0 + \delta)\|_2 \quad (6.16)$$

$$= \min_{y_m \in \mathbb{R}^m} \|r_0 - AV_m y_m\|_2, \quad (6.17)$$

$$= \min_{y_m \in \mathbb{R}^m} \|r_0 - V_{m+1} \bar{H}_m y_m\|_2, \quad (6.18)$$

$$= \min_{y_m \in \mathbb{R}^m} \|\beta v_1 - V_{m+1} \bar{H}_m y_m\|_2, \quad (6.19)$$

$$= \min_{y_m \in \mathbb{R}^m} \|V_{m+1}(\beta e_1 - \bar{H}_m y_m)\|_2, \quad (6.20)$$

$$= \min_{y_m \in \mathbb{R}^m} \|\beta e_1 - \bar{H}_m y_m\|_2. \quad (6.21)$$

The $m \times (m+1)$ least square problem could be addressed by QR factorization and the method especially becomes easier due to the simple structure of \bar{H}_m .

ALGORITHM: GMRES (standard)

1. Compute $r_0 = b - Ax_0$, let $v_1 = r_0/\beta$, where $\beta := \|r_0\|_2$
2. Arnoldi process: For $j = 1, 2, \dots, m$, Do

$$h_{i,j} = (Av_j, v_i), i = 1, 2, \dots, j \quad (6.22)$$

$$w_j = Av_j - \sum_{i=1}^j h_{i,j} v_i \quad (6.23)$$

$$h_{j+1,j} = \|w_j\|_2 \quad (6.24)$$

$$\text{if } h_{j+1,j} = 0, \text{ then stop, let } m := j \quad (6.25)$$

$$\text{else } v_{j+1} = w_j/h_{j+1,j} \quad (6.26)$$

End Do

3. Form the solution by solving the least square problem: $x_m = x_0 + V_m y_m$, where y_m minimizes $\|\beta e_1 - \bar{H}_m y_m\|_2$.

The only difference between GMRES and FOM lies in the last step: the former finds an optimal solution to a least square problem while the latter computing y_m directly. Although the difference seems trivial, the two algorithms fall into two distinct categories of projection method: GMRES is a projection method based on taking \mathcal{K}_m as search subspace and $\mathcal{L}_m = A\mathcal{K}_m$ as constraint subspace, while FOM has the same search/constraint subspace \mathcal{K}_m . The derivation of the algorithms also varies correspondingly as has been demonstrated.

The breakdown of GMRES could only possibly occur at Arnoldi iteration when $h_{j+1,j} = 0$. As analyzed before, it means that the Krylov subspace basis has been fully orthogonalized if A is non-singular. In this situation, m must be equal to n and the solution must be exact because the degree of freedom matches the degree of restraint.

In cases when m is very large, the standard GMRES becomes increasingly expensive as m

keeps growing. One remedy is to re-start the algorithm after every m iterations and m is a pre-set integer and could even be dynamic. The algorithm is as follows:

ALGORITHM: GMRES(re-started)

1. Compute $r_0 = b - Ax_0$, let $v_1 = r_0/\beta$, where $\beta := \|r_0\|_2$
2. Arnoldi process: Produce V_m and \bar{H}_m using Arnoldi algorithm
3. Compute y_m which minimizes $\|\beta e_1 - \bar{H}_m y_m\|_2$ and $x_m = x_0 + V_m y_m$
4. Examine $r_m = b - Ax_m$. If satisfied, then stop. Else let $x_0 := x_m$ and go to step 1.

6.3 Solution to the least square problem and practical discussion

6.3.1 Solving the least square problem

This section discusses how to solve the least square problem:

$$\min_{y_m \in \mathbb{R}^m} \|\beta e_1 - \bar{H}_m y_m\|_2. \quad (6.27)$$

The normal way to solve the problem is by QR factorization that transforms \bar{H}_m into an upper triangular matrix R_m , and y_m could be directly obtained by backward substitution. Particularly, the special structure of \bar{H}_m simplifies the transformation process. Givens rotation is a widely-used method to implement the transformation. Consider a transformation matrix F applied to \bar{H}_m and transform it into an upper triangular matrix R :

$$F \bar{H}_m = R. \quad (6.28)$$

The transformation matrix $F = \prod_j F_j$ is composed of a series of 'rotation' matrix F_j which has the form:

$$F_j = \begin{bmatrix} 1 & \dots & & & & \\ & \dots & & & & \\ & & \dots & c & \dots & s & \dots \\ & & \dots & & 1 & & \dots \\ & & \dots & -s & \dots & c & \dots \\ & & \dots & & & & \\ & & \dots & & & & 1 \end{bmatrix}. \quad (6.29)$$

Note the ' $\dots c \dots s \dots$ ' row is the i th row of the rotation matrix and ' $\dots -s \dots c \dots$ ' is the j th row ($j > i$). Meanwhile, it is easy to see that F_j is unitary. The matrix is able to turn the j th entry of a vector $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_i, \dots, \alpha_j, \dots, \alpha_m)$ into 0 and update the i th entry accordingly while remaining all other entries unchanged:

$$F_j \alpha = (\alpha_1, \alpha_2, \dots, \alpha_i^{(1)}, \dots, \alpha_j^{(1)}, \dots, \alpha_m), \quad (6.30)$$

where $\alpha_i^{(1)} = \sqrt{\alpha_i^2 + \alpha_j^2}$ and $\alpha_j^{(1)} = 0$. Actually the formation of F_j is related to the vector that is being changed:

$$c = \frac{\alpha_i}{\sqrt{\alpha_i^2 + \alpha_j^2}}, \quad (6.31)$$

$$s = \frac{\alpha_j}{\sqrt{\alpha_i^2 + \alpha_j^2}}. \quad (6.32)$$

In one word, F_j 'rotates' (swaps position) the i th and j th entries of a vector α and updates their values in such a way that the newly updated j th entry is 0. All other entries except those two entries are not affected by the rotation. Therefore, applying multiple F_j (j must decrease in order) on a vector is able to turn multiple entries into 0, and applying the same rotation operations to multiple column vectors (matrix) is able to eliminate particular non-zero entries as demanded.

Consider applying the following rotation matrices on the Hessenberg matrix \bar{H}_m and transforming it into an upper triangular matrix \bar{R}_m :

$$F_m F_{m-1} \dots F_1 \bar{H}_m = F \bar{H}_m = \bar{R}_m. \quad (6.33)$$

Obviously, since F_j is unitary, so is F . Each rotation matrix operates on two particular entries in a vector: F_k , ($k = 1, 2, \dots, m$) rotates the $(k+1)$ th and k th entries of the k th column of \bar{H}_m and turns the new $(k+1)$ th entry into 0. For example, F_1 shifts h_{21} to 0, and F_2 shifts h_{32} to 0 ..., until $h_{m+1,m}$ becomes 0 by F_m . Note that F_k is applied to not only the k th column of \bar{H}_m although it is only this column turning a non-zero entry into zero, all other columns are affected by the rotation matrix imposed. Denoting \bar{H}_m composed by multiple column vectors h_i , ($i = 1, 2, \dots, m$), for those column vectors that $i < k$, they remain unchanged because their k th and $(k+1)$ th entries are already 0; for those vectors that $i > k$, their entries in position k and $k+1$ would be updated by F_k . Therefore, the rotation matrix F_k is actually determined by the two entries at position k and $(k+1)$ of vector column $h_k^{(k-1)} := F_{k-1} \dots F_1 h_k$ rather than original h_k .

Technically, F_k is built by modifying the k -th and $(k+1)$ -th rows of $(m+1) \times (m+1)$ identity matrix as follows:

$$F_k = \begin{bmatrix} 1 & \dots & & & \\ & \dots & & & \\ & & c_k & s_k & \dots \\ & & -s_k & c_k & \dots \\ & \dots & & & \\ & \dots & & & 1 \end{bmatrix}. \quad (6.34)$$

where

$$c_k = \frac{h_{k,k}^{(k-1)}}{\sqrt{(h_{k,k}^{(k-1)})^2 + (h_{k+1,k})^2}}, \quad (6.35)$$

$$s_k = \frac{h_{k+1,k}}{\sqrt{(h_{k,k}^{(k-1)})^2 + (h_{k+1,k})^2}}. \quad (6.36)$$

The outcome upper triangular matrix $\bar{\mathbf{R}}_m$ has the form:

$$\bar{\mathbf{R}}_m = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ 0 & r_{22} & \dots & r_{2m} \\ & 0 & r_{33} & \dots & r_{3m} \\ & & 0 & r_{44} & \dots & r_{4m} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & 0 & r_{m,m} \\ \dots & \dots & \dots & \dots & 0 \end{bmatrix}. \quad (6.37)$$

While transforming $\bar{\mathbf{H}}_m \mathbf{y}_m$ into a simpler form, the other side of the problem $\beta \mathbf{e}_1$ is also imposed by the same operation at the same time. Instead of eliminating any non-zero entry, the operation turns all 0 entries into non-zero:

$$\mathbf{F} \beta \mathbf{e}_1 := \bar{\mathbf{g}} = [\gamma_1, \gamma_2, \dots, \gamma_{m+1}]^T. \quad (6.38)$$

The original least square problem becomes

$$\min_{\mathbf{y}_m \in \mathbb{R}^m} \|\beta \mathbf{e}_1 - \bar{\mathbf{H}}_m \mathbf{y}_m\|_2, \quad (6.39)$$

$$\min_{\mathbf{y}_m \in \mathbb{R}^m} \|\mathbf{F} \beta \mathbf{e}_1 - \mathbf{F} \bar{\mathbf{H}}_m \mathbf{y}_m\|_2, \quad (\mathbf{F} \text{ is unitary,}) \quad (6.40)$$

$$\min_{\mathbf{y}_m \in \mathbb{R}^m} \|\bar{\mathbf{g}} - \bar{\mathbf{R}}_m \mathbf{y}_m\|_2. \quad (6.41)$$

The last row of $\bar{\mathbf{R}}_m$ is a null-vector and deleting it from $\bar{\mathbf{R}}_m$ gives a $m \times m$ upper triangular matrix \mathbf{R}_m . The size of \mathbf{y}_m is m , and together with \mathbf{g} that is the m -vector deleting γ_{m+1} from $\bar{\mathbf{g}}$, the least square problem could be transformed to

$$\min_{\mathbf{y}_m \in \mathbb{R}^m} \|\bar{\mathbf{g}} - \bar{\mathbf{R}}_m \mathbf{y}_m\|_2^2 = \min_{\mathbf{y}_m \in \mathbb{R}^m} \|\mathbf{g} - \mathbf{R}_m \mathbf{y}_m\|_2^2 + |\gamma_{m+1}|^2. \quad (6.42)$$

Now it is time to compute \mathbf{y}_m because the first term in the right hand side of Eq.(6.42) is a simple triangular system, which gives the solution

$$\mathbf{y}_m = \mathbf{R}_m^{-1} \mathbf{g}. \quad (6.43)$$

Note it is not necessary to calculate the inverse of \mathbf{R} in real calculation because Gauss elimination is good enough to solve \mathbf{y}_m .

In GMRES, the convergence criteria is formulated via \mathbf{r}_m rather than \mathbf{y}_m (or even \mathbf{x}_m) because there is an immediate way to obtain \mathbf{r}_m . Recall Eq.(6.42), the term $\|\mathbf{g} - \mathbf{R}_m \mathbf{y}_m\|_2^2$ is equal to 0 with optimal \mathbf{y}_m and consequently,

$$\min_{\mathbf{y}_m \in \mathbb{R}^m} \|\bar{\mathbf{g}} - \bar{\mathbf{R}}_m \mathbf{y}_m\|_2 = |\gamma_{m+1}|. \quad (6.44)$$

That means that after m -step of Arnoldi iteration and Givens rotation, the 2-norm of the minimum residual r_m is given directly by the last entry of $\bar{\mathbf{g}}_m$. With no more extra computation cost, the convergence condition could be examined.

6.3.2 Compressed row storage

In the project, the matrices to deal with are usually large and sparse, which means they have large number of entries ($\sim 10^8$ at least) but most of them (over 99%) are 0, which would not play any essential role in matrix calculation. It would be a huge waste if so many entries are stored but not using. Therefore, an effective storage strategy called compressed row storage method (CRS) is implemented to address the problem.

The idea of CRS is to create a system of much smaller size to store all the non-zero entries of the matrix and identify them with pre-designed order. This way the zero-entries are excluded from the computer memory unit occupation without any influence on the computing. Here demonstrates the way CRS works via an example:

Consider a $m \times (m + 2)$ matrix \mathbf{A} that has three non-zero entries at each row in diagonal and beside positions: $a_{i,i}$, $a_{i,i+1}$, $a_{i,i+2}$ at row i , as shown in Eq.(6.45). It has $3m$ non-zero entries, so a vector $\mathbf{nz}(3m)$ is defined to store the entries in so-called row-wise fashion that fills the entries into \mathbf{nz} by up-to-bottom and left-to-right order. In this case, for example, $\mathbf{nz}(1) = a_{11}$, $\mathbf{nz}(2) = a_{12}$, $\mathbf{nz}(5) = a_{23}$. Next, two vectors \mathbf{ja} and \mathbf{ia} are defined to help identify their positions in the matrix. Firstly, vector \mathbf{ja} is of size $3m$ and is used to store the column indices of the non-zero entries in row-wise fashion as well. The column index represents the position of the referred entry in its row. For example, a_{11} is the 1st entry in row 1, so its column index is 1, and similarly, a_{13} has column index '3', a_{24} has '4', $a_{m,m,m+1}$ has $m + 1$. Secondly, vector \mathbf{ia} that is of size $(m + 1)$ is used to store the row indices of the non-zero entries still in row-wise fashion. The row index represents the number of non-zero entries of each row in \mathbf{A} . In this case, each row of the matrix has three non-zero entries so that all entries of \mathbf{ia} are three except $\mathbf{ia}(1) = 1$ by default.

CRS uses three vectors whose size is merely of order m to store and identify all the entries needed for the matrix computation. If all the matrix \mathbf{A} is saved as conventional way, the memory cost would be of order m^2 , which means approximately m times less efficient in

storing and addressing the data. In the project, m is usually of order 10^4 .

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & & & \\ & a_{22} & a_{23} & a_{24} & & \\ & & a_{33} & a_{34} & a_{35} & \\ & & \dots & & & \\ & & & \dots & & \\ & & & & \dots & \\ & & & & & a_{m,m} & a_{m,m+1} & a_{m,m+2} \end{bmatrix}. \quad (6.45)$$

6.4 Convergence analysis

This section discusses the convergence rate of GMRES. It is known that GMRES must converge in up to n iterations if the linear system to solve has dimension of $n \times n$. However, it is often too expensive to let GMRES run to full scale in real problem. Instead, GMRES is usually allowed to run m iterations ($m \ll n$) and the residual r_m norm is then examined for the convergence check. Sometimes m iterations are not enough and another m -iteration-cycle is needed. Therefore, the key problem is by how many iteration could GMRES converge to required convergence criteria. In other words, the problem is how precise an approximation could be made from m -dimensional Krylov subspace to construct a n -dimensional solution. The intuition is that it depends on some property of matrix A since Krylov subspace is made of it apart from r_0 . As will be shown next, it is actually the eigenvalue distribution of A that determines the convergence rate of GMRES.

To demonstrate this, first a brief overview of the convergence study of GMRES by Saad is provided in Subsection 6.4.1, which is made in quite general basis using Chebyshev polynomials. Then in Subsection 6.4.2, a more detailed eigenvalue analysis is made without the assistance of Chebyshev polynomials. The conclusions in the above 2 subsections are consistent.

6.4.1 General analysis using Chebyshev Polynomial

This subsection briefly introduces the main conclusion of the GMRES convergence analysis made by Saad [106]. The provided conclusion is supposed to be compared with the analysis as will be shown in Subsection 6.4.2 as benchmark.

Assume that all the eigenvalues of A are contained within the ellipse $E(c, d, a)$, which represents an ellipse with center c , focal distance d and major semi axis a . The ellipse does not encircle origin in it. Meanwhile, suppose A could be diagonalizable that $A = X\Lambda X^{-1}$ where $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_n\}$, composed of eigenvalues of A . Then, the residual norm r_m

after m steps of GMRES has the norm

$$\|\mathbf{r}_m\|_2 \leq \kappa(X) \frac{C_m(\frac{a}{d})}{|C_m(\frac{c}{d})|} \|\mathbf{r}_0\|_2, \quad (6.46)$$

where

$$\kappa(\mathbf{X}) = \|\mathbf{X}\| \|\mathbf{X}^{-1}\|, \quad (6.47)$$

$$\frac{C_m(\frac{a}{d})}{C_m(\frac{c}{d})} \approx \left(\frac{a + \sqrt{a^2 - d^2}}{c + \sqrt{c^2 - d^2}} \right)^m. \quad (6.48)$$

Note C_m is the m -order complex Chebyshev polynomials. Let alone $\kappa(\mathbf{X})$, Eq.(6.48) shows the residual \mathbf{r}_m is affected by the geometry of the ellipse defined by eigenvalues of \mathbf{A} . It could be seen that a and d determines the distribution characteristic (loose or compressed) of eigenvalues of \mathbf{A} and c roughly determines the average eigenvalue. To make Eq.(6.48) small (≈ 0), a and d are supposed to be small and c is supposed to be large, which means the ellipse is small in size and located far from origin. That means eigenvalues of \mathbf{A} should be distributed around the average eigenvalue that is supposed to be large. Alternatively, very roughly speaking, $\frac{C_m(\frac{a}{d})}{C_m(\frac{c}{d})} \approx (\frac{a}{c})^m$ and that indicates the eigenvalues should be distributed in such a concentrated way that the difference among them is minor compared to the average value of them.

6.4.2 Eigenvalue analysis of GMRES convergence

Assume $\mathbf{A} \in \mathbb{R}^{n \times n}$ is non-singular with n eigenvalues: $\lambda_1, \lambda_2, \dots, \lambda_n$ and $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$. Meanwhile $\alpha_1, \alpha_2, \dots, \alpha_n$ are the corresponding eigenvectors. Apparently \mathbf{r}_0 could be written as:

$$\mathbf{r}_0 = b_1 \alpha_1 + b_2 \alpha_2 + \dots + b_n \alpha_n, \quad (6.49)$$

where b_i ($i = 1, 2, \dots, n$) are coefficients.

The second column vector of Krylov subspace $\mathbf{A}\mathbf{r}_0$ is:

$$\mathbf{A}\mathbf{r}_0 = b_1 \lambda_1 \alpha_1 + b_2 \lambda_2 \alpha_2 + \dots + b_n \lambda_n \alpha_n. \quad (6.50)$$

Similarly, expand Krylov subspace up to column n :

$$\mathbf{A}^{n-1} \mathbf{r}_0 = b_1 \lambda_1^{n-1} \alpha_1 + b_2 \lambda_2^{n-1} \alpha_2 + \dots + b_n \lambda_n^{n-1} \alpha_n. \quad (6.51)$$

Recall that $\mathbf{A}(\mathbf{x} - \mathbf{x}_0) = \mathbf{r}_0$, and $\mathbf{x} - \mathbf{x}_0$ could be expressed as members of Krylov subspace:

$$(\mathbf{x} - \mathbf{x}_0) = z_1 \mathbf{r}_0 + z_2 \mathbf{A}\mathbf{r}_0 + \dots + z_n \mathbf{A}^{n-1} \mathbf{r}_0, \quad (6.52)$$

where z_i ($i = 1, 2, \dots, n$) are coefficients.

Clearly,

$$\mathbf{A}(\mathbf{x} - \mathbf{x}_0) = z_1 \mathbf{A} \mathbf{r}_0 + z_2 \mathbf{A}^2 \mathbf{r}_0 + \dots z_n \mathbf{A}^n \mathbf{r}_0. \quad (6.53)$$

Substitute Eqs.(6.50) and (6.51) into Eq.(6.53) and transform:

$$\begin{aligned} & \mathbf{A}(\mathbf{x} - \mathbf{x}_0) \\ &= z_1 \mathbf{A} \mathbf{r}_0 + z_2 \mathbf{A}^2 \mathbf{r}_0 + \dots + z_n \mathbf{A}^n \mathbf{r}_0, \end{aligned} \quad (6.54)$$

$$= z_1 (b_1 \lambda_1 \boldsymbol{\alpha}_1 + b_2 \lambda_2 \boldsymbol{\alpha}_2 + \dots + b_n \lambda_n \boldsymbol{\alpha}_n) + \quad (6.55)$$

$$\begin{aligned} & z_2 (b_1 \lambda_1^2 \boldsymbol{\alpha}_1 + b_2 \lambda_2^2 \boldsymbol{\alpha}_2 + \dots + b_n \lambda_n^2 \boldsymbol{\alpha}_n) + \\ & \dots \\ & z_n (b_1 \lambda_1^n \boldsymbol{\alpha}_1 + b_2 \lambda_2^n \boldsymbol{\alpha}_2 + \dots + b_n \lambda_n^n \boldsymbol{\alpha}_n), \\ &= \boldsymbol{\alpha}_1 b_1 (z_1 \lambda_1 + z_2 \lambda_1^2 + \dots + z_n \lambda_1^n) + \\ & \boldsymbol{\alpha}_2 b_2 (z_1 \lambda_2 + z_2 \lambda_2^2 + \dots + z_n \lambda_2^n) + \\ & \dots \\ & \boldsymbol{\alpha}_n b_n (z_1 \lambda_n + z_2 \lambda_n^2 + \dots + z_n \lambda_n^n). \end{aligned} \quad (6.56)$$

Clearly, $\mathbf{A}(\mathbf{x} - \mathbf{x}_0)$ must be equal to \mathbf{r}_0 as expanded in Eq.(6.49). In matrix formation the equality could be expressed as follows:

$$\begin{bmatrix} b_1 \lambda_1 & b_1 \lambda_1^2 & \dots & b_1 \lambda_1^n \\ b_2 \lambda_2 & b_2 \lambda_2^2 & \dots & b_2 \lambda_2^n \\ \dots & \dots & \dots & \dots \\ b_n \lambda_n & b_n \lambda_n^2 & \dots & b_n \lambda_n^n \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}, \quad (6.57)$$

which is equal to

$$\begin{bmatrix} \lambda_1 & \lambda_1^2 & \dots & \lambda_1^n \\ \lambda_2 & \lambda_2^2 & \dots & \lambda_2^n \\ \dots & \dots & \dots & \dots \\ \lambda_n & \lambda_n^2 & \dots & \lambda_n^n \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix}. \quad (6.58)$$

This is the full-scale problem formulation. Apparently the solution would be exact and completely depend on the eigenvalue matrix. Next, consider the case when Krylov subspace is not expanded to n -th column and the solution is hence approximated (denoted by \mathbf{x}_m).

Assume m iterations has been run and $\mathbf{x}_m - \mathbf{x}_0$ are members of m -dimensional Krylov subspace $\mathbf{x}_m - \mathbf{x}_0 = \text{span}\{\mathbf{b}, \mathbf{A}\mathbf{b}, \mathbf{A}^2\mathbf{b}, \dots, \mathbf{A}^{m-1}\mathbf{b}\}$. Therefore, $\mathbf{A}(\mathbf{x}_m - \mathbf{x}_0)$ could be

written as

$$\begin{aligned}
& \mathbf{A}(\mathbf{x}_m - \mathbf{x}_0) \\
&= \bar{z}_1 \mathbf{A} \mathbf{r}_0 + \bar{z}_2 \mathbf{A}^2 \mathbf{r}_0 + \dots + \bar{z}_m \mathbf{A}^m \mathbf{r}_0, \\
&= \bar{z}_1 (b_1 \lambda_1 \boldsymbol{\alpha}_1 + b_2 \lambda_2 \boldsymbol{\alpha}_2 + \dots + b_n \lambda_n \boldsymbol{\alpha}_n) + \\
&\quad \bar{z}_2 (b_1 \lambda_1^2 \boldsymbol{\alpha}_1 + b_2 \lambda_2^2 \boldsymbol{\alpha}_2 + \dots + b_n \lambda_n^2 \boldsymbol{\alpha}_n) + \\
&\quad \dots \\
&\quad \bar{z}_m (b_1 \lambda_1^m \boldsymbol{\alpha}_1 + b_2 \lambda_2^m \boldsymbol{\alpha}_2 + \dots + b_n \lambda_n^m \boldsymbol{\alpha}_n), \\
&= \boldsymbol{\alpha}_1 b_1 (\bar{z}_1 \lambda_1 + \bar{z}_2 \lambda_1^2 + \dots + \bar{z}_m \lambda_1^m) + \\
&\quad \boldsymbol{\alpha}_2 b_2 (\bar{z}_1 \lambda_2 + \bar{z}_2 \lambda_2^2 + \dots + \bar{z}_m \lambda_2^m) + \\
&\quad \dots \\
&\quad \boldsymbol{\alpha}_n b_n (\bar{z}_1 \lambda_n + \bar{z}_2 \lambda_n^2 + \dots + \bar{z}_m \lambda_n^m).
\end{aligned} \tag{6.59}$$

Then, the equation $\mathbf{A}(\mathbf{x}_m - \mathbf{x}_0) = \mathbf{r}_0$ has the matrix form as follows

$$\begin{bmatrix} \lambda_1 & \lambda_1^2 & \dots & \lambda_1^m \\ \dots & \dots & \dots & \dots \\ \lambda_m & \lambda_m^2 & \dots & \lambda_m^m \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \lambda_n & \lambda_n^2 & \dots & \lambda_n^m \end{bmatrix} \cdot \begin{bmatrix} \bar{z}_1 \\ \bar{z}_2 \\ \dots \\ \bar{z}_m \end{bmatrix} = \begin{bmatrix} 1 \\ \dots \\ \dots \\ 1 \\ \dots \\ \dots \\ 1 \end{bmatrix}. \tag{6.60}$$

Note that the equality relationship $\mathbf{A}(\mathbf{x}_m - \mathbf{x}_0) = \mathbf{r}_0$ is not strictly true because \mathbf{x}_m is approximated solution to a least square problem. It is preferable to write $\mathbf{A}(\mathbf{x}_m - \mathbf{x}_0) \approx \mathbf{r}_0$.

Denote the matrix equation as shown in Eq.(6.58) by $\mathbf{G}z = \mathbf{f}$. In order to match the dimension of matrix equation as shown in Eq.(6.60) with $\mathbf{G}z = \mathbf{f}$, expand Eq.(6.60) to the same dimension as $\mathbf{G}z = \mathbf{f}$ and denote the expanded matrix equation as $\bar{\mathbf{G}}\bar{z} \approx \mathbf{f}$:

$$\begin{bmatrix} \lambda_1 & \lambda_1^2 & \dots & \lambda_1^m & 0 & \dots & 0 \\ \lambda_2 & \lambda_2^2 & \dots & \lambda_2^m & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & 0 & \dots & 0 \\ \lambda_m & \lambda_m^2 & \dots & \lambda_m^m & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & 0 & \dots & 0 \\ \lambda_n & \lambda_n^2 & \dots & \lambda_n^m & 0 & \dots & 0 \end{bmatrix} \cdot \begin{bmatrix} \bar{z}_1 \\ \bar{z}_2 \\ \dots \\ \bar{z}_m \\ 0 \\ \dots \\ 0 \end{bmatrix} \approx \begin{bmatrix} 1 \\ \dots \\ \dots \\ 1 \\ \dots \\ \dots \\ 1 \end{bmatrix}. \tag{6.61}$$

Comparing $\mathbf{G}z = \mathbf{f}$ and $\bar{\mathbf{G}}\bar{z} \approx \mathbf{f}$, it could be seen that:

$$\mathbf{G}z = (\bar{\mathbf{G}} + \Delta\mathbf{G})(\bar{z} + \Delta z) = \mathbf{f}, \tag{6.62}$$

where it is defined that $\mathbf{G} = \bar{\mathbf{G}} + \Delta\mathbf{G}$ and $z = \bar{z} + \Delta z$.

Formulate Eq.(6.62) in matrix equation as follows:

$$\begin{bmatrix} \lambda_1 & \lambda_1^2 & \dots & \lambda_1^m & \delta_{1,m+1} & \dots & \delta_{1,n} \\ \lambda_2 & \lambda_2^2 & \dots & \lambda_2^m & \delta_{2,m+1} & \dots & \delta_{2,n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \lambda_m & \lambda_m^2 & \dots & \lambda_m^m & \delta_{m,m+1} & \dots & \delta_{m,n} \\ \dots & \dots & \dots & \dots & \delta_{m+1,m+1} & \dots & \delta_{m+1,n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \lambda_n & \lambda_n^2 & \dots & \lambda_n^m & \delta_{n,m+1} & \dots & \delta_{n,n} \end{bmatrix} \cdot \begin{bmatrix} \bar{z}_1 + \delta_{z1} \\ \bar{z}_2 + \delta_{z2} \\ \dots \\ \bar{z}_m + \delta_{zm} \\ \delta_{z(m+1)} \\ \dots \\ \delta_{zn} \end{bmatrix} = \begin{bmatrix} 1 \\ \dots \\ \dots \\ 1 \\ \dots \\ \dots \\ 1 \end{bmatrix}. \quad (6.63)$$

This system is equal to Eq.(6.58) and displays ΔG and Δz in every entry separately. It is clear that $\delta_{i,j} = \lambda_i^j$ ($j \geq (m+1)$) in the eigenvalue matrix. The concerned problem now is how large ΔG could be as a result of Δz in the eigenvalue matrix. Note Δz represents the difference between the exact solution to the original problem and the approximated solution to the least square problem. In GMRES programming, applying certain convergence criterion on residual r means the error Δz in solution is also restraint to a certain level. In other words, suppose that GMRES has found x_m close enough to x after m iterations (hence $\|\Delta z\|$ is given and small), how much close is the approximated eigenvalue matrix \bar{G} to the full eigenvalue matrix G ? If ΔG is small, it means the Krylov subspace must be expanded to relative high order so that \bar{G} has more columns equivalent to G . This would cost more GMRES steps and thus causes slower convergence. By contrast, if ΔG is large, it means \bar{G} is less resembling G , hence less GMRES steps to run and faster convergence. Therefore, it is expected that ΔG is as large as possible if certain Δz is given.

To address this problem, transform Eq.(6.62):

$$\Delta z = -(\bar{G} + \Delta G)^{-1} \Delta G \bar{z}, \quad (6.64)$$

$$= -(M + (1 - \theta) \Delta G)^{-1} \Delta G \bar{z}, \quad (6.65)$$

where $M = \bar{G} + \theta \Delta G$ and $\theta \in [0, 1]$. Note there is a simplification $f = \bar{G} \bar{z}$ when performing the transformation. The simplification could be justified because $f - \bar{G} \bar{z}$ is the monitored residual to check the convergence in GMRES programming and it is usually set very small (usually about 10^{-8}).

Next, take the norm on both sides:

$$\|\Delta z\| \leq \|(M + (1 - \theta) \Delta G)^{-1}\| \|\Delta G\| \|\bar{z}\|. \quad (6.66)$$

Now consider $(M + (1 - \theta) \Delta G)^{-1}$ for a while. Assume $M + (1 - \theta) \Delta G$ is non-regular (A has n distinct eigenvalues), and its inverse could be written as

$$(M + (1 - \theta) \Delta G)^{-1} = M^{-1} - (1 - \theta) M^{-1} \Delta G (\bar{G} + \Delta G)^{-1}, \quad (6.67)$$

and the norm follows

$$\begin{aligned} \|(\mathbf{M} + (1 - \theta)\Delta\mathbf{G})^{-1}\| &\leq \|\mathbf{M}^{-1}\| + (\theta - 1)\|\mathbf{M}^{-1}\|\|\Delta\mathbf{G}\|\|(\mathbf{G} + \Delta\mathbf{G})^{-1}\|, \\ \rightarrow 1 &\leq \frac{\|\mathbf{M}^{-1}\|}{\|(\mathbf{M} + (1 - \theta)\Delta\mathbf{G})^{-1}\|} + (\theta - 1)\|\mathbf{M}^{-1}\|\|\Delta\mathbf{G}\|, \end{aligned} \quad (6.68)$$

$$\rightarrow \|(\mathbf{M} + (1 - \theta)\Delta\mathbf{G})^{-1}\| \leq \frac{\|\mathbf{M}^{-1}\|}{1 - (\theta - 1)\|\mathbf{M}^{-1}\|\|\Delta\mathbf{G}\|}. \quad (6.69)$$

Substitute Eq.(6.69) into Eq.(6.66):

$$\begin{aligned} \|\Delta\mathbf{z}\| &\leq \frac{\|\mathbf{M}^{-1}\|\|\Delta\mathbf{G}\|\|\bar{\mathbf{z}}\|}{1 - (\theta - 1)\|\mathbf{M}^{-1}\|\|\Delta\mathbf{G}\|}, \\ \rightarrow \frac{\|\Delta\mathbf{z}\|}{\|\bar{\mathbf{z}}\|} &\leq \frac{\|\mathbf{M}^{-1}\|\|\mathbf{M}\|\frac{\|\Delta\mathbf{G}\|}{\|\mathbf{M}\|}}{1 - (\theta - 1)\|\mathbf{M}^{-1}\|\|\Delta\mathbf{G}\|}, \end{aligned} \quad (6.70)$$

$$\rightarrow \frac{\|\Delta\mathbf{z}\|}{\|\bar{\mathbf{z}}\|} \leq \|\mathbf{M}^{-1}\|\|\mathbf{M}\|\frac{\|\Delta\mathbf{G}\|}{\|\mathbf{M}\|}, \quad (\text{let } \theta \approx 1). \quad (6.71)$$

In the case when $\theta \approx 1$, the denominator of right hand side of Eq.(6.70) ' $1 - (\theta - 1)\|\mathbf{M}^{-1}\|\|\Delta\mathbf{G}\|$ ' is approximately equal to 1 and \mathbf{M} is roughly equal to \mathbf{G} . Therefore, the above inequality could also be written as

$$\frac{\|\Delta\mathbf{G}\|}{\|\mathbf{G}\|} \geq \frac{1}{\|\mathbf{G}^{-1}\|\|\mathbf{G}\|} \frac{\|\Delta\mathbf{z}\|}{\|\bar{\mathbf{z}}\|}. \quad (6.72)$$

As can be seen, the relative 'error' of eigenvalue matrix $\frac{\|\Delta\mathbf{G}\|}{\|\mathbf{G}\|}$ is no less than $\frac{1}{\|\mathbf{G}^{-1}\|\|\mathbf{G}\|}$ times of the relative 'error' in the solution vector $\frac{\|\Delta\mathbf{z}\|}{\|\bar{\mathbf{z}}\|}$. Considering $\frac{1}{\|\mathbf{G}^{-1}\|\|\mathbf{G}\|}$ is the inverse of condition number of matrix \mathbf{G} : $\frac{1}{\kappa(\mathbf{G})}$, it means higher $\kappa(\mathbf{G})$ leads to lower projection of $\frac{\|\Delta\mathbf{z}\|}{\|\bar{\mathbf{z}}\|}$ onto error band of \mathbf{G} , which means $\Delta\mathbf{G}$ is restricted to be smaller. By contrast, lower $\kappa(\mathbf{G})$ leads to higher projection of $\frac{\|\Delta\mathbf{z}\|}{\|\bar{\mathbf{z}}\|}$ onto error band of \mathbf{G} , which allows higher 'error' ($\Delta\mathbf{G}$) in formulating \mathbf{G} by m -dimensional Krylov subspace. Therefore, the condition number of \mathbf{G} affects the convergence rate of GMRES and it is preferably supposed to be small.

The condition number $\kappa(\mathbf{G})$ depends on the eigenvalues of matrix \mathbf{A} as could be seen from its composition. Therefore, studying $\kappa(\mathbf{G})$ is equal to studying the eigenvalue nature of \mathbf{A} . Denote $\mathbf{v}_0 := [\lambda_1, \lambda_2, \dots, \lambda_n]^T$ where λ_i is the eigenvalue of \mathbf{A} (as defined previously). The eigenvalue matrix \mathbf{G} could be written as

$$\mathbf{G} = \begin{bmatrix} \mathbf{v}_0 & \Lambda\mathbf{v}_0 & \Lambda^2\mathbf{v}_0 & \dots & \Lambda^{n-1}\mathbf{v}_0 \end{bmatrix}, \quad (6.73)$$

where

$$\Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \dots & \\ & & & \lambda_n \end{bmatrix}. \quad (6.74)$$

Vector \mathbf{v}_0 is specially composed of eigenvalues of \mathbf{A} , and on the other hand, like any arbitrary vector, could be rewritten as $\mathbf{v}_0 = (a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + \dots a_n\mathbf{x}_n)^T$ where a_i ($i = 1, 2, \dots, n$) are

coefficients and $\mathbf{x}_i (i = 1, 2, \dots, n)$ are eigenvectors of $\mathbf{\Lambda}$ (also \mathbf{A}) corresponding to eigenvalues $\lambda_i (i = 1, 2, \dots, n)$ respectively. Then, the second column of \mathbf{G} could be written as

$$\mathbf{\Lambda} \mathbf{v}_0 = a_1 \lambda_1 \mathbf{x}_1 + a_2 \lambda_2 \mathbf{x}_2 + \dots a_n \lambda_n \mathbf{x}_n. \quad (6.75)$$

Repeat the multiplication k times:

$$\mathbf{\Lambda}^k \mathbf{v}_0 = a_1 \lambda_1^k \mathbf{x}_1 + a_2 \lambda_2^k \mathbf{x}_2 + \dots a_n \lambda_n^k \mathbf{x}_n, \quad (6.76)$$

$$= \lambda_1^k [a_1 \mathbf{x}_1 + a_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k \mathbf{x}_2 + \dots a_n \left(\frac{\lambda_n}{\lambda_1}\right)^k \mathbf{x}_n], \quad (6.77)$$

the above vector represents the k -th column of matrix \mathbf{G} .

Since λ_1 is the dominant eigenvalue, other coefficients $\|(\frac{\lambda_i}{\lambda_1})^k\| (i \neq 1)$ could be no greater than 1 and tend to be 0 if k is large or $\|\frac{\lambda_i}{\lambda_1}\| \ll 1 (i \neq 1)$. In such case, $\mathbf{\Lambda}^k \mathbf{u} \approx \lambda_1 \mathbf{\Lambda}^{k-1} \mathbf{u}$, which means the two neighbor columns of \mathbf{G} are roughly linearly dependent and \mathbf{G} tends to be regular and very ill-conditioned. Therefore, the condition number $\kappa(\mathbf{G})$ actually depends on the ratio between the dominant eigenvalue of \mathbf{A} and its any other eigenvalue, especially the second largest. The ideal scenario to let $\kappa(\mathbf{G})$ keep low is that \mathbf{A} has all eigenvalues almost equal to each other (not exactly equal because the above analysis presumes \mathbf{G} is not singular). At least the dominant eigenvalue should not be too larger than the second largest eigenvalue. In eigenvalue distribution, it is preferably supposed to have the first few largest eigenvalues clustered together.

Next consider the case when \mathbf{A} has multiple identical eigenvalues: \mathbf{G} is singular and the system $\mathbf{G} \mathbf{z} = \mathbf{f}$ does not have unique solution ($\mathbf{A} \mathbf{x} = \mathbf{b}$ still does). Recall the system as shown in Eq.(6.58), and assume any two eigenvalues identical, $\lambda_{k-1} = \lambda_k$, and the system becomes

$$\begin{bmatrix} \lambda_1 & \lambda_1^2 & \dots & \lambda_1^n \\ \lambda_2 & \lambda_2^2 & \dots & \lambda_2^n \\ \dots & \dots & \dots & \dots \\ \lambda_{n-1} & \lambda_{n-1}^2 & \dots & \lambda_{n-1}^n \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_{n-1} \\ z_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix}. \quad (6.78)$$

The solution is not unique. There are as many solutions as the selection of 'free' entry z_n . The solution vector $[z_1, z_2, \dots, z_{n-1}]^T$ depends on z_n . Assume z_n is given, and the solution vector $[z_1, z_2, \dots, z_{n-1}]^T$ could be uniquely obtained by solving the following $(n-1) \times (n-1)$ system: ($\mathbf{G}' \mathbf{z}' = \mathbf{f}'$)

$$\begin{bmatrix} \lambda_1 & \lambda_1^2 & \dots & \lambda_1^{n-1} \\ \lambda_2 & \lambda_2^2 & \dots & \lambda_2^{n-1} \\ \dots & \dots & \dots & \dots \\ \lambda_{n-1} & \lambda_{n-1}^2 & \dots & \lambda_{n-1}^{n-1} \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} 1 - \lambda_1^n z_n \\ 1 - \lambda_2^n z_n \\ \dots \\ 1 - \lambda_{n-1}^n z_n \end{bmatrix}. \quad (6.79)$$

There might be infinite selections of the free entry z_n but the selection of z_n only affects \mathbf{f}' but does not change the structure of \mathbf{G}' and \mathbf{z}' . The matrix \mathbf{G}' is non-singular and the norm

analysis performed with $Gz = f$ could be applied to this system. After m steps, GMRES finds an approximated solution \bar{z}' from the m -dimensional Krylov subspace whose size is $(n - 1)$. The conclusion regarding the convergence remains: closer distribution of eigenvalue λ_i makes less columns of G' generated hence enables faster GMRES convergence.

Therefore, the convergence rate of GMRES is directly affected by $\kappa(G)$ and essentially the eigenvalue distribution of A . Sometimes it is also possible to examine $\kappa(G)$ by checking $\kappa(A)$ (condition number of A) if it could be estimated cheaply. Here is the explanation:

$$|\lambda_1| \leq \|A\|, \quad (6.80)$$

$$\frac{1}{|\lambda_n|} \leq \|A^{-1}\|. \quad (6.81)$$

clearly, $\kappa(A) = \|A\|\|A^{-1}\| \geq \frac{|\lambda_1|}{|\lambda_n|} \geq 1$. If $\kappa(A)$ is small, the ratio between $|\lambda_1|$ and $|\lambda_n|$ could be even less and tends to 1 ideally. That also means the eigenvalue distribution of A is highly clustered.

6.5 ILU(0) preconditioning of GMRES

Preconditioning plays a significant role in GMRES. Usually the implementation of GMRES comes with some kind of preconditioner to improve its convergence. There are a number of techniques to design preconditioner and this project only uses the so-called zero fill-in incomplete LU (ILU(0)) factorization. This section is going to briefly introduce the way preconditioner works in GMRES followed by the implementation of ILU(0).

Recall that the linear system to solve is as Eq.(6.1) shows. Preconditioning is to transform the system into another system that is computationally cheaper to solve while maintaining the same solution by constructing a preconditioning matrix M (left preconditioning):

$$M^{-1}Ax = M^{-1}b, \quad (6.82)$$

or applying M on the right(right preconditioning):

$$AM^{-1}y = b, \quad (6.83)$$

where $x := M^{-1}y$.

The preconditioning matrix M is usually constructed to be close to A and M^{-1} must not be singular. The two techniques mainly differ on program-implementation level as will be seen next.

ALGORITHM: Left preconditioned GMRES

1. Compute $r_0 = M^{-1}(b - Ax_0)$, let $v_1 = r_0/\beta$, where $\beta := \|r_0\|_2$

2. Arnoldi process: For $j = 1, 2, \dots, m$, Do

$$h_{i,j} = (\mathbf{M}^{-1} \mathbf{A} \mathbf{v}_j, \mathbf{v}_i), i = 1, 2, \dots, j$$

$$\mathbf{w}_j = \mathbf{M}^{-1} \mathbf{A} \mathbf{v}_j - \sum_{i=1}^j h_{i,j} \mathbf{v}_i$$

$$h_{j+1,j} = \|\mathbf{w}_j\|_2$$

if $h_{j+1,j} = 0$, then stop, let $m := j$

else $\mathbf{v}_{j+1} = \mathbf{w}_j / h_{j+1,j}$

End Do

3. Define $\mathbf{V}_m = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m\}$ and $\bar{\mathbf{H}}_m = \{h_{ij}\}$, $1 \leq i \leq (j+1)$, $1 \leq j \leq m$.

4. Form the solution: $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m \mathbf{y}_m$, where \mathbf{y}_m minimizes $\|\beta \mathbf{e}_1 - \bar{\mathbf{H}}_m \mathbf{y}_m\|_2$.

ALGORITHM: Right preconditioned GMRES

1. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \mathbf{x}_0$, let $\mathbf{v}_1 = \mathbf{r}_0 / \beta$, where $\beta := \|\mathbf{r}_0\|_2$

2. Arnoldi process: For $j = 1, 2, \dots, m$, Do

$$h_{i,j} = (\mathbf{A} \mathbf{M}^{-1} \mathbf{v}_j, \mathbf{v}_i), i = 1, 2, \dots, j$$

$$\mathbf{w}_j = \mathbf{A} \mathbf{M}^{-1} \mathbf{v}_j - \sum_{i=1}^j h_{i,j} \mathbf{v}_i$$

$$h_{j+1,j} = \|\mathbf{w}_j\|_2$$

if $h_{j+1,j} = 0$, then stop, let $m := j$

else $\mathbf{v}_{j+1} = \mathbf{w}_j / h_{j+1,j}$

End Do

3. Define $\mathbf{V}_m = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m\}$ and $\bar{\mathbf{H}}_m = \{h_{ij}\}$, $1 \leq i \leq (j+1)$, $1 \leq j \leq m$.

4. Form the solution: $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{M}^{-1} \mathbf{V}_m \mathbf{y}_m$, where \mathbf{y}_m minimizes $\|\beta \mathbf{e}_1 - \bar{\mathbf{H}}_m \mathbf{y}_m\|_2$.

The two preconditioning techniques are similar and require almost equal amount of computation. In left preconditioning algorithm, the system aims to solve a transformed system $(\mathbf{M}^{-1} \mathbf{A}) \mathbf{x} = \mathbf{M}^{-1} \mathbf{b}$ and the original residual is substituted by preconditioned residual $\mathbf{M}^{-1} \mathbf{b}$. As a result, the convergence check is imposed on the preconditioned residual too. In right preconditioning algorithm, the solution vector does not participate the computation explicitly but in the form of $\mathbf{M}^{-1} \mathbf{y}$. That is why the original solution \mathbf{x}_m needs to be retrieved in step 4. However, the residual remains throughout the iterative process, where it differs from left preconditioning algorithm.

One commonly used technique to build preconditioner \mathbf{M} is so-called incomplete LU (ILU) factorization. The idea is inspired by Gauss elimination that is widely applied to decompose a matrix into two matrices $\mathbf{A} = \mathbf{L} \mathbf{U}$, where \mathbf{L} is an unit lower triangular matrix and \mathbf{U} is an upper triangular matrix. LU factorization is usually effective in solving small dense linear system but the idea is adopted here to precondition large sparse linear systems. Due to the

large size of the system to solve, the factorization result is usually not fully accurate in return for low computing cost.

The standard LU factorization goes through the entire matrix A in one-by-one fashion then generates and stores the result in the corresponding position of A to save computer memory. Therefore, the resultant matrices L and U are supposed to be stored in A as well, and the zero-entries of A are expected to still remain zero after being updated. Now consider L and U have the same non-zero pattern as A ,

$$R = LU - A, \quad (6.84)$$

the problem is that the product matrix LU could not be precisely equal to A . There are always some extra near-diagonal entries that fails to match the zero entries in A . Those entries are called fill-in elements. In order to minimize R in a simple way, the idea is to ignore those fill-in elements and make LU merely equal to A in those non-zero locations of A . This is so-called ILU(0).

Due to the fact that the Gauss elimination is only imposed on the entries that belong to the sparsity pattern of A , the computing cost is very low with the expense of accuracy lost. There should have been much more entries filled in with non-zero entries by Gauss elimination in L and U . To get more accurate factors L and U , the sparsity pattern should be augmented obviously, but that rises computing cost. The trade-off is that the sparsity pattern is slightly augmented and drop those fill-in elements that are 'too small' during iterative process. There is a model to measure the value of fill-in elements with an index $k \in [0, \infty)$. The higher k , the smaller elements and vice versa. Zero entries have $k = \infty$. All elements with k greater-equal than a certain value would be round off to 0 (dropped) during the computing. The ILU(0) preconditioning method drops all entries out of the sparsity pattern because all filled-in elements have $k \geq 0$. Therefore, ILU(0) is the cheapest but also the most inaccurate factorization among its family.

Algorithm: ILU(0)

1. For $i = 2, \dots, n$ Do
2. For $k = 1, \dots, i-1$ and $(i, k) \in NZ(A)$ Do
3. $a_{ik} = a_{ik}/a_{kk}$
4. For $j = k+1, \dots, n$ and $(i, j) \in NZ(A)$ Do
5. $a_{ij} = a_{ij} - a_{ik}a_{kj}$
6. End Do
7. End Do
8. End Do

Lastly, consider a practical issue: how to calculate M^{-1} effectively. Clearly $M^{-1} = U^{-1}L^{-1}$ and the problem turns to calculating the inverse of L and U . Assume $LX = I$ where $X := L^{-1} = [x_1, x_2, \dots, x_n]$ and I is the identity matrix. Addressing the problem is equal to solving

the following linear system n times:

$$\mathbf{L}\mathbf{x}_i = \mathbf{e}_i \quad (i = 1, \dots, n). \quad (6.85)$$

It is efficient to solve it by Gauss elimination and due to the high sparsity of \mathbf{L} , the computing cost would not be much more than pure backward substitution. In each column there are only a few non-zero entries, therefore in line three of Gauss elimination algorithm, only a few a_{ik} whose $k \lesssim i$ needs to be computed rather than all $(i - 1)$ a_{ik} . Then, in line 5, only those a_{ij} where $j \lesssim i$. In one word, the computation only takes place around the left-sided diagonal of the matrix instead of everywhere in the lower triangular domain.

Similarly, the inverse of \mathbf{U} could be obtained efficiently by solving the linear systems n times:

$$\mathbf{U}\mathbf{y}_i = \mathbf{e}_i \quad (i = 1, \dots, n). \quad (6.86)$$

The elimination process merely takes place around the right-sided diagonal of the upper triangular matrix, and $\mathbf{U}^{-1} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n]$.

Algorithm: Gauss elimination

1. For $i = 2, \dots, n$ Do
2. For $k = 1, \dots, i - 1$ Do
3. $a_{ik} = a_{ik}/a_{kk}$
4. For $j = k + 1, \dots, n$ Do
5. $a_{ij} = a_{ij} - a_{ik}a_{kj}$
6. End Do
7. End Do
8. End Do

In either left or right preconditioned GMRES, it is always required to compute $\mathbf{M}^{-1}\mathbf{A}\mathbf{v}_j$ in each Arnoldi step (take left-preconditioned GMRES as example). Recall Eq.(6.84), then

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{v}_j = \mathbf{M}^{-1}(\mathbf{M} - \mathbf{R})\mathbf{v}_j = \mathbf{v}_j - \mathbf{M}^{-1}\mathbf{R}\mathbf{v}_j. \quad (6.87)$$

The matrix \mathbf{R} is more sparse than \mathbf{A} hence more efficient in computing implement. Notice that \mathbf{R} is just a matrix made up of the dropped fill-in elements in ILU(0). It is not difficult to compute and store those elements when running ILU(0) algorithm.

6.6 Summary

This chapter introduces an effective algorithm to solve large linear equation system: GMRES, which is widely used in the project to solve every given linear system. It is highly important to understand how the given linear system is solved by the computer because it lays the foundation to evaluate the computing time and enables the comparison of the computing

time between NDD method and normal method. The key finding is that the convergence of GMRES is directly associated with the eigenvalue distribution of the coefficient matrix A where a linear system $Ax = b$ is solved. The ideal scenario is to let A have almost equal eigenvalues to achieve the fastest convergence of GMRES. One practical way to evaluate eigenvalue distribution of A is to estimate the condition number of the matrix: small condition number accounts for desired eigenvalue distribution. The derivation, implementation and preconditioning technique are also elaborated in the chapter.

Chapter 7

Conclusions and recommendations

This chapter summarises the main work of the thesis. More detailed conclusions can be found at the end of each chapter. Recommendations for future work that could follow this thesis are also given.

7.1 Summary of the work

The thesis has developed a novel non-overlapping domain decomposition approach with non-local interface boundary conditions that is applied to second order differential equations with small parameters at second derivative. The method is aimed at solving two-dimensional boundary value problems where the non-local effect is essential. The approach transfers the boundary condition from the wall to an intermediate interface in such a way that the non-local effect of the given problem is fully retained. Therefore, the approach provides a completely exact solution to the problem.

The derivation of the non-local interface boundary conditions is implemented based on the approximation of Steklov-Poincaré operators by local Taylor series expansions. Steklov-Poincaré operators contain the non-local nature of the given problem, so the accuracy of the operator approximations determine the convergence of the NDD approach. The approximations are realised by local Taylor expansions. Although the expansion itself is local, an almost non-local expansion could be obtained by considering the expansion everywhere at a global scale, formulating all function values in implicit equations, and solving all the equations in an entire system. The resulting interface boundary condition is of Robin-to-Robin type. Three variants (algorithms) of the NDD approach are proposed depending on how the Taylor series are locally expanded: the constantly-approximated algorithm, the linearly-approximated algorithm and the quadratically-approximated algorithm.

The convergence of the three algorithms is studied analytically in application to Poisson's equation. The product of Steklov-Poincaré operators applied to a given problem is not only affected by the governing equation but also the boundary position with respect to x (so-called 'boundary effect' in the thesis) which could vary from case to case. To simplify the analysis, the analysis does not consider the boundary effect at first and the way the analysis is performed is denoted by the standard approach. Technically, excluding the boundary effect means that certain functions (I , x and x^2 in the thesis), to which Steklov-Poincaré

operators are applied, are defined in $(-\infty, \infty)$, so that the boundary effect is too far to impose any influence. The standard analysis shows that the algorithm with constant approximation of Steklov-Poincaré operators has an equivalent convergence rate to that with linear approximation but a higher convergence rate than that with quadratic approximation. In addition, the factors that determine the convergence are also studied: the convergence rate of the constantly (linearly)-approximate algorithm is determined by the multiplication of Steklov-Poincaré operator symbol increment: $\Delta s_1 \Delta s_2$ where $\Delta s_i := \hat{S}_i(\omega) - \hat{S}_i(0)$, $i = 1, 2$, and the convergence rate of the quadratically-approximated algorithm is determined by the multiplication of the increment of the second-order derivative of Steklov-Poincaré operator symbol: $\Delta s_{1\omega\omega} \Delta s_{2\omega\omega}$ where $\Delta s_{i\omega\omega} := \hat{S}_{i\omega\omega}(\omega) - \hat{S}_{i\omega\omega}(0)$, $i = 1, 2$. Note the symbol of operator S_i is denoted by \hat{S}_i . In other words, the convergence rate of the algorithms are determined by the symbols (or the derivative of the symbols) of Steklov-Poincaré operators. What is more, it is proved that the result applies to all equations that have variable-separation form rather than only Poisson's equation. Besides, every algorithm has two versions: parallel and sequential, and the latter achieves convergence twice as fast as the former.

The study of the boundary effect shows that it has positive effect on boosting the convergence for all the three algorithms, which means that the real convergence rate of the algorithms is lower than that obtained from the standard approach. Considering the convergence result from both the standard approach and the boundary effect enables the complete (real) convergence evaluation. Comparatively speaking, quadratically-approximated algorithm has the lowest (real) convergence rate, followed by linearly-approximated and constantly-approximated algorithms respectively. The difference between quadratically-approximated and linearly-approximated algorithms is not remarkable. For practical use, an empirical law is proposed to evaluate the strength of the boundary effect: the boundary effect could be depicted/represented as a function (called 'pike function') in the profile of Steklov-Poincaré operators applied to I , x or x^2 , and the taller and wider pike function accounts for stronger boundary effect.

The NDD algorithms are also compared with a conventional Robin-to-Robin algorithm and a Dirichlet-to-Robin algorithm to solve the same Poisson's equation. It is both analytically and numerically proven that the NDD algorithms have faster convergence than both the conventional Robin-to-Robin algorithm and the Dirichlet-to-Robin algorithm, especially quadratically-approximated algorithm. The comparative study demonstrates the superiority of the NDD method over other similar methods.

Beyond Poisson's equation, the three algorithms are further applied to solving model equations with changeable parameters (α) at second derivative. The equations are supposed to imitate the main mathematical features of high-Reynolds-number flow with a boundary layer. The changeable parameter α manipulates the solution dependence on x . Three values of α are tested, which represent low, medium and high non-local effect. Numerical result shows that when the non-local effect is low, the problem could be considered as semi-one-dimensional and all the three algorithms converge equally fast. When the non-local effect strength is medium, quadratically-approximated algorithm shows remarkably faster

convergence than the other two due to faster standard convergence rate. When the non-local effect strength becomes strong, quadratically-approximated algorithm maintains promising convergence because the boundary effect does not undermine the standard convergence, which is not the case in linearly-approximated algorithm. Therefore, it could be stated that quadratically-approximated algorithm demonstrates both high efficiency and outstanding robustness in all conditions. The high efficiency is associated with the low standard convergence rate, and the great robustness is resulted by the positive boundary effect. The combination of high efficiency and good robustness amounts to the promising ability of quadratically-approximated algorithm in reserving the non-local effect of Steklov-Poincaré operators. The numerical test result on model equations suggests that the proposed NDD approach has great potential to be applied to modelling near-wall turbulent flow.

GMRES is an advanced method to solve large sparse linear systems and is widely used in the project. The convergence analysis of GMRES elaborates how by splitting a big matrix into two smaller ones and solving them separately could it save computing time. The convergence of GMRES is found associated with the eigenvalue distribution of the coefficient matrix A when the linear system $Ax = b$ is supposed to be solved. When the eigenvalues of a matrix is higher clustered, GMRES takes less iterations to converge and computing cost becomes lower too. The analysis lays a theoretical foundation for evaluating the computing time of the NDD approach and comparing efficiency with the standard (one-block) method.

7.2 Future work

In the project, an efficient non-overlapping domain decomposition method is studied both theoretically and numerically. Therefore, the possible future work is also recommended from the two aspects.

7.2.1 Theoretical future work

1. The theoretical convergence analysis is composed of two parts: the standard approach and the boundary effect analysis. The latter is only quantitatively studied for constantly-approximated algorithm but the rest two algorithms. Especially considering the boundary effect is not always positive (for example, in linearly-approximated algorithm case) in boosting convergence, more accurate model might be needed to elaborate the effect (especially the negative effect) that has been observed in the numerical test.
2. It is demonstrated that the symbol (as well as the derivative) of Steklov-Poincaré operators determines the convergence of the NDD algorithms. Therefore, it is important to gain the knowledge of the symbol when the operators are applied to random two-dimensional problem. In the project, the knowledge is obtained from analytical calculation of the symbol because Poisson's equation enables the calculation. However, in real turbulent flow problems it is no more practical to rely on analytical tools and there should be an easy and effective way to

estimate/compute the symbol in application to given problems.

3. Regarding GMRES, the eigenvalue characteristics of the coefficient matrix A is not easy to know without substantial amount of computational costs. As a matter of fact, Arnoldi algorithm, as part of GMRES, already offers good approximation of the eigenvalues of the decomposed matrix but the eigenvalue information is no more used in the later process. Therefore, it is possible to estimate the computational costs of implementing the NDD method via Arnoldi algorithm and even derive an empirical law to link the computational costs directly with linear system properties (such as mesh density in the two regions, interface boundary location, etc.).

7.2.2 Numerical future work

1. The numerical test is implemented on a Poisson's equation and two model equations. The former is to verify the theoretical result and the latter is to test the feasibility of extensively applying the method to turbulence modelling. Both goals have been achieved with the conclusion that quadratically-approximated algorithm is especially effective and robust to resolve near-wall turbulent flows. Next, the algorithm could be applied to RANS models and consider computational times as an important criterion to evaluate the efficiency.

2. A NDD algorithm with movable interface boundary is proposed at the end of Chapter 5 and is proven to be efficient in the numerical test of a model equation. The efficiency of the method is associated with the nature of the problems it solves: solution to the problems has high gradient in y direction in near-wall region, and off-wall region solution is dominated by x -variation. The nature is widely shared by near-wall turbulent flow quantities thus the algorithm has large potential to be further explored. Any other quantity characterised by such distinct patterns in two regions (not necessarily the same two regions as velocity) could be possibly resolved by the method.

References

- [1] F. Karimpour and S. K. Venayagamoorthy, “Some insights for the prediction of near-wall turbulence,” *Journal of Fluid Mechanics*, vol. 723, pp. 126–139, 2013.
- [2] A. Patel, B. J. Boersma, and R. Pecnik, “The influence of near-wall density and viscosity gradients on turbulence in channel flows,” *Journal of Fluid Mechanics*, vol. 809, pp. 793–820, 2016.
- [3] T. Craft, S. Gant, A. Gerasimov, H. Iacovides, and B. Launder, “Development and application of wall-function treatments for turbulent forced and mixed convection flows,” *Fluid Dynamics Research*, vol. 38, no. 2-3, p. 127, 2006.
- [4] H. Schlichting and J. Kestin, *Boundary layer theory*. Springer, 1961, vol. 121.
- [5] S. B. Pope and S. B. Pope, *Turbulent flows*. Cambridge university press, 2000.
- [6] W. Rodi, “Low Reynolds number $k - \varepsilon$ modeling with the aid of direct simulation data,” in *Proceedings of the 1990 Summer Program*, Center for Turbulence Research, NASA Ames research center, 1990.
- [7] V. Michelassi, W. Rodi, and J. Zhu, “Testing a low-Reynolds number $k - \varepsilon$ turbulence model based on direct simulation data,” *AIAA journal*, vol. 31, no. 9, pp. 1720–1723, 1993.
- [8] P. A. Durbin, “Near-wall turbulence closure modeling without “damping functions”,,” *Theoretical and computational fluid dynamics*, vol. 3, no. 1, pp. 1–13, 1991.
- [9] B. D. Wood, X. He, and S. V. Apte, “Modeling turbulent flows in porous media,” *Annual Review of Fluid Mechanics*, vol. 52, pp. 171–203, 2020.
- [10] J. Wang, J. Gong, X. Kang, C. Zhao, and K. Hooman, “Assessment of rans turbulence models on predicting supercritical heat transfer in highly buoyant horizontal flows,” *Case Studies in Thermal Engineering*, vol. 34, p. 102 057, 2022.
- [11] J. Smagorinsky, “General circulation experiments with the primitive equations: I. the basic experiment,” *Monthly weather review*, vol. 91, no. 3, pp. 99–164, 1963.
- [12] Y. Zhiyin, “Large-eddy simulation: Past, present and the future,” *Chinese journal of Aeronautics*, vol. 28, no. 1, pp. 11–24, 2015.

- [13] E. Karchniwy, N. E. L. Haugen, A. Klimanek, Ø. Langørgen, and S. Śladek, “The effect of turbulence on mass transfer in solid fuel combustion: Rans model,” *Combustion and Flame*, vol. 227, pp. 65–78, 2021.
- [14] D. R. Chapman, “Computational aerodynamics development and outlook,” *AIAA journal*, vol. 17, no. 12, pp. 1293–1313, 1979.
- [15] U. Piomelli, “Wall-modeled large-eddy simulations: Present status and prospects,” in *Direct and Large-Eddy Simulation VII*, Springer, 2010, pp. 1–10.
- [16] W. Jones and B. Launder, “The prediction of laminarization with a two-equation model of turbulence,” *International journal of heat and mass transfer*, vol. 15, no. 2, pp. 301–314, 1972.
- [17] W. Jones and B. Launder, “The calculation of low-Reynolds-number phenomena with a two-equation model of turbulence,” *International Journal of Heat and Mass Transfer*, vol. 16, no. 6, pp. 1119–1130, 1973.
- [18] B. Launder and B. Sharma, “Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc,” *Letters in heat and mass transfer*, vol. 1, no. 2, pp. 131–137, 1974.
- [19] V. C. Patel, W. Rodi, and G. Scheuerer, “Turbulence models for near-wall and low Reynolds number flows-a review,” *AIAA journal*, vol. 23, no. 9, pp. 1308–1319, 1985.
- [20] K. Y. Chien, “Predictions of channel and boundary-layer flows with a low-Reynolds-number turbulence model,” *AIAA journal*, vol. 20, no. 1, pp. 33–38, 1982.
- [21] W. Rodi, “Experience with two-layer models combining the $k - \varepsilon$ model with a one-equation model near the wall,” in *29th Aerospace sciences meeting*, 1991, p. 216.
- [22] P. R. Spalart, “Direct simulation of a turbulent boundary layer up to $R_\theta = 1410$,” *Journal of fluid mechanics*, vol. 187, pp. 61–98, 1988.
- [23] Z. Yang and T. H. Shih, “New time scale based $k - \varepsilon$ model for near-wall turbulence,” *AIAA journal*, vol. 31, no. 7, pp. 1191–1198, 1993.
- [24] Y. Nagano and M. Tagawa, “An improved k-epsilon model for boundary layer flows,” *ASME Journal of Fluids Engineering*, vol. 112, pp. 33–39, 1990.
- [25] M. A. Pakhomov and U. K. Zhapbasbayev, “Rans modeling of turbulent flow and heat transfer of non-newtonian viscoplastic fluid in a pipe,” *Case Studies in Thermal Engineering*, vol. 28, p. 101455, 2021.
- [26] A. N. Kolmogorov, “Equations of turbulent motion in an incompressible fluid,” in *Dokl. Akad. Nauk SSSR*, vol. 30, 1941, pp. 299–303.

- [27] P. G. Saffman, "A model for inhomogeneous turbulent flow," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 317, no. 1530, pp. 417–433, 1970.
- [28] D. C. Wilcox and I. Alber, "A turbulence model for high speed flows," in *Proceedings of the 1972 Heat Transfer and Fluid Mechanics Institute*, Stanford University Press, vol. 231, 1972, p. 252.
- [29] D. C. Wilcox, "Reassessment of the scale-determining equation for advanced turbulence models," *AIAA journal*, vol. 26, no. 11, pp. 1299–1310, 1988.
- [30] D. C. Wilcox *et al.*, *Turbulence modeling for CFD*. DCW industries La Canada, CA, 1998, vol. 2.
- [31] S. A. A. Mirjalily, "Lambda shock behaviors of elliptic supersonic jets; a numerical analysis with modification of rans turbulence model," *Aerospace Science and Technology*, vol. 112, p. 106 613, 2021.
- [32] F. R. Menter, M. Kuntz, and R. Langtry, "Ten years of industrial experience with the SST turbulence model," *Turbulence, heat and mass transfer*, vol. 4, no. 1, pp. 625–632, 2003.
- [33] C. G. Speziale, R. Abid, and E. C. Anderson, "Critical evaluation of two-equation models for near-wall turbulence," *AIAA journal*, vol. 30, no. 2, pp. 324–331, 1992.
- [34] F. R. Menter, "Improved two-equation k-omega turbulence models for aerodynamic flows. nasa sti," *Recon Technical Report*, vol. 93, p. 22 809, 1992.
- [35] J. C. Kok, "Resolving the dependence on freestream values for the $k - \omega$ turbulence model," *AIAA journal*, vol. 38, no. 7, pp. 1292–1295, 2000.
- [36] A. Hellsten, "New advanced $k - \omega$ turbulence model for high-lift aerodynamics," *AIAA journal*, vol. 43, no. 9, pp. 1857–1869, 2005.
- [37] P. Huang, "Physics and computations of flows with adverse pressure gradients," in *Modeling Complex Turbulent Flows*, Springer, 1999, pp. 245–258.
- [38] T. Coakley, "Turbulence modeling methods for the compressible Navier-Stokes equations," in *16th Fluid and Plasmadynamics Conference*, 1983, p. 1693.
- [39] D. C. Wilcox, "Formulation of the $k - \omega$ turbulence model revisited," *AIAA journal*, vol. 46, no. 11, pp. 2823–2838, 2008.
- [40] H. Fadhila, H. Medina, S. Aleksandrova, and S. Benjamin, "A new non-linear rans model with enhanced near-wall treatment of turbulence anisotropy," *Applied Mathematical Modelling*, vol. 82, pp. 293–313, 2020.

- [41] E. Casartelli, L. Mangani, D. Roos Launchbury, and A. Del Rio, “Application of advanced rans turbulence models for the prediction of turbomachinery flows,” *Journal of Turbomachinery*, vol. 144, no. 1, 2022.
- [42] D. Spalding, “Monograph on turbulent boundary layers,” *Imperial College: Mechanical Engineering Department Report TWF/TN*, vol. 33, 1967.
- [43] S. V. Patankar, “Heat and mass transfer in turbulent boundary layers,” Ph.D. dissertation, University of London, 1967.
- [44] M. Wolfshtein, “The velocity and temperature distribution in one-dimensional flow with turbulence augmentation and pressure gradient,” *International Journal of Heat and Mass Transfer*, vol. 12, no. 3, pp. 301–318, 1969.
- [45] B. Launder and D. Spalding, “The numerical computation of turbulent flows,” *Computer Methods in Applied Mechanics and Engineering*, vol. 3, no. 2, pp. 269–289, 1974.
- [46] C. Chieng and B. Launder, “On the calculation of turbulent heat transport downstream from an abrupt pipe expansion,” *Numerical heat transfer*, vol. 3, no. 2, pp. 189–207, 1980.
- [47] R. Amano, “Development of a turbulence near-wall model and its application to separated and reattached flows,” *Numerical Heat Transfer*, vol. 7, no. 1, pp. 59–75, 1984.
- [48] T. H. Shih, L. A. Povinelli, and N.-S. Liu, “Application of generalized wall function for complex turbulent flows,” *Journal of Turbulence*, vol. 4, no. 1, p. 015, 2003.
- [49] M. Ciofalo and M. Collins, “ $k - \varepsilon$ Predictions of heat transfer in turbulent recirculating flows using an improved wall treatment,” *Numerical Heat Transfer, Part B: Fundamentals*, vol. 15, no. 1, pp. 21–47, 1989.
- [50] T. Craft, A. Gerasimov, H. Iacovides, and B. Launder, “Progress in the generalization of wall-function treatments,” *International Journal of Heat and Fluid Flow*, vol. 23, no. 2, pp. 148–160, 2002.
- [51] R. Johnson and B. Launder, “Discussion of “on the calculation of turbulent heat transport downstream from an abrupt pipe expansion”,,” *Numerical Heat Transfer, Part A Applications*, vol. 5, no. 4, pp. 493–496, 1982.
- [52] S. Kim, “A near-wall treatment using wall functions sensitized to pressure gradient,” in *Separated and Complex Flows, ASME/JSME Fluids Engineering and Laser Anemometry Conference and Exhibition, Hilton Head, SC, 1995*, 1995.

- [53] H. Grotjans and F. R. Menter, “Wall functions for general application CFD codes,” *Computational fluid dynamics’98*, pp. 1112–1117, 1998.
- [54] S. Gant, “Development and application of a new wall function for complex turbulent flows,” Ph.D. dissertation, University of Manchester, 2003.
- [55] T. Craft, “Development and application of a new wall function for a complex turbulent flows,” in *ECCOMAS Computational Fluid Dynamics Conference*, 2001.
- [56] T. Craft, S. Gant, H. Iacovides, and B. Launder, “A new wall function strategy for complex turbulent flows,” *Numerical Heat Transfer, Part B: Fundamentals*, vol. 45, no. 4, pp. 301–318, 2004.
- [57] A. Gerasimov, “Development and application of an analytical wall-function strategy for modelling forced, mixed and natural convection flows,” Ph.D. dissertation, University of Manchester Institute of Science and Technology, 2003.
- [58] T. Craft, A. Gerasimov, H. Iacovides, and B. Launder, “Progress in the generalization of wall-function treatments,” *International Journal of Heat and Fluid Flow*, vol. 23, no. 2, pp. 148–160, 2002.
- [59] R. Amano, H. Arakawa, and K. Suga, “Turbulent heat transfer in a two-pass cooling channel by several wall turbulence models,” *International Journal of Heat and Mass Transfer*, vol. 77, pp. 406–418, 2014.
- [60] K. Suga, Y. Ishibashi, and Y. Kuwata, “An analytical wall-function for recirculating and impinging turbulent heat transfer,” *International journal of heat and fluid flow*, vol. 41, pp. 45–54, 2013.
- [61] K. Suga, T. Craft, and H. Iacovides, “An analytical wall-function for turbulent flows and heat transfer over rough walls,” *International Journal of Heat and Fluid Flow*, vol. 27, no. 5, pp. 852–866, 2006.
- [62] K. Suga, “Computation of high prandtl number turbulent thermal fields by the analytical wall-function,” *International journal of heat and mass transfer*, vol. 50, no. 25-26, pp. 4967–4974, 2007.
- [63] K. Suga and S. Nishiguchi, “Computation of turbulent flows over porous/fluid interfaces,” *Fluid dynamics research*, vol. 41, no. 1, p. 012 401, 2009.
- [64] K. Suga and M. Kubo, “Modelling turbulent high schmidt number mass transfer across undeformable gas–liquid interfaces,” *International journal of heat and mass transfer*, vol. 53, no. 15-16, pp. 2989–2995, 2010.

- [65] K. Suga, S. Nishiguchi, H. Asano, and M. Kitada, "Application of the combination of an analytical wall-function and a realizable two component limit second moment closure to hvac related complex turbulent flows," in *The 14th International Conference of Fluid Flow Technologies, Budapest, Hungary*, 2009.
- [66] C. Katsamis, T. Craft, H. Iacovides, and J. C. Uribe, "Use of 2-d and 3-d unsteady rans in the computation of wall bounded buoyant flows," *International Journal of Heat and Fluid Flow*, vol. 93, p. 108 914, 2022.
- [67] G. Kalitzin, G. Medic, G. Iaccarino, and P. Durbin, "Near-wall behavior of RANS turbulence models and implications for wall functions," *Journal of Computational Physics*, vol. 204, no. 1, pp. 265–291, 2005.
- [68] F. Billard, D. Laurence, and K. Osman, "Adaptive wall functions for an elliptic blending eddy viscosity model applicable to any mesh topology," *Flow, Turbulence and Combustion*, vol. 94, no. 4, pp. 817–842, 2015.
- [69] M. Popovac, "Modelling and simulation of turbulence and heat transfer in wall-bounded flows," Ph.D. dissertation, Delft University of Technology, 2006.
- [70] M. Popovac and K. Hanjalic, "A combined WF and ItW treatment of wall boundary conditions for turbulent convective heat transfer," in *9th UK National Heat Transfer Conference, Manchester, UK*, 2005.
- [71] M. Popovac and K. Hanjalic, "Compound wall treatment for RANS computation of complex turbulent flows and heat transfer," *Flow, turbulence and combustion*, vol. 78, no. 2, pp. 177–202, 2007.
- [72] R. Tatschl, B. Basara, J. Schneider, *et al.*, "Advanced turbulent heat transfer modeling for IC-engine applications using AVL FIRE," in *Proceedings of International Multidimensional Engine Modeling User's Group Meeting, Detroit, USA*, 2006.
- [73] B. Basara, F. Aldudak, S. Jakirlić, *et al.*, "Experimental investigations and computations of unsteady flow past a real car using a robust elliptic relaxation closure with a universal wall treatment," *SAE Transactions*, pp. 22–32, 2007.
- [74] S. V. Utyuzhnikov, "Some new approaches to building and implementation of wall-functions for modeling of near-wall turbulent flows," *Computers & fluids*, vol. 34, no. 7, pp. 771–784, 2005.
- [75] S. V. Utyuzhnikov, "Generalized wall functions and their application for simulation of turbulent flows," *International journal for numerical methods in fluids*, vol. 47, no. 10-11, pp. 1323–1328, 2005.
- [76] S. V. Utyuzhnikov, "Robin-type wall functions and their numerical implementation," *Applied Numerical Mathematics*, vol. 58, no. 10, pp. 1521–1533, 2008.

- [77] S. V. Utyuzhnikov, “The method of boundary condition transfer in application to modeling near-wall turbulent flows,” *Computers & fluids*, vol. 35, no. 10, pp. 1193–1204, 2006.
- [78] V. S. Ryaben’Kii, *Method of difference potentials and its applications*. Springer Science & Business Media, 2001, vol. 30.
- [79] S. V. Utyuzhnikov, “Generalized calderón–ryaben’kii’s potentials,” *IMA journal of applied mathematics*, vol. 74, no. 1, pp. 128–148, 2009.
- [80] S. V. Utyuzhnikov, “Domain decomposition approach for near-wall turbulence modeling,” in *VII European Congress on Computational Methods in Applied Sciences and Engineering*, Springer Nature, 2016.
- [81] S. V. Utyuzhnikov, “Domain decomposition for near-wall turbulent flows,” *Computers & fluids*, vol. 38, no. 9, pp. 1710–1717, 2009.
- [82] M. Petrov, S. V. Utyuzhnikov, A. Chikitkin, and V. Titarev, “On extension of near-wall domain decomposition to turbulent compressible flows,” *Computers & Fluids*, vol. 210, p. 104 629, 2020.
- [83] S. V. Utyuzhnikov and C. Wang, “Exact non-overlapping domain decomposition for near-wall turbulence modeling,” *Computers & Fluids*, vol. 181, pp. 283–291, 2019.
- [84] S. V. Utyuzhnikov and H. Li, “Domain decomposition with nonlocal interface boundary conditions,” *Journal of Computational and Applied Mathematics*, vol. 421, p. 114 847, 2022.
- [85] S. V. Utyuzhnikov, “Towards development of unsteady near-wall interface boundary conditions for turbulence modeling,” *Computer Physics Communications*, vol. 185, no. 11, pp. 2879–2884, 2014.
- [86] S. V. Utyuzhnikov and N. Smirnova, “Unsteady interface boundary conditions for near-wall turbulence modeling,” *Computers & Mathematics with Applications*, vol. 79, no. 5, pp. 1483–1502, 2020.
- [87] A. Chikitkin, S. V. Utyuzhnikov, M. Petrov, and V. Titarev, “Non-overlapping domain decomposition for modeling essentially unsteady near-wall turbulent flows,” *Computers & Fluids*, vol. 202, p. 104 506, 2020.
- [88] M. Petrov, S. V. Utyuzhnikov, A. Chikitkin, and N. Smirnova, “Extension of near-wall domain decomposition to modeling flows with laminar-turbulent transition,” *Communications in Computational Physics*, 2021.
- [89] S. V. Utyuzhnikov, “Interface boundary conditions in near-wall turbulence modeling,” *Computers & fluids*, vol. 68, pp. 186–191, 2012.

- [90] W. Cabot and P. Moin, “Approximate wall boundary conditions in the large-eddy simulation of high Reynolds number flow,” *Flow, Turbulence and Combustion*, vol. 63, no. 1, pp. 269–291, 2000.
- [91] M. Wang and P. Moin, “Dynamic wall modeling for large-eddy simulation of complex turbulent flows,” *Physics of Fluids*, vol. 14, no. 7, pp. 2043–2051, 2002.
- [92] A. Jones and S. V. Utyuzhnikov, “Application of a near-wall domain decomposition method to turbulent flows with heat transfer,” *Computers & Fluids*, vol. 119, pp. 87–100, 2015.
- [93] A. Quarteroni, “Domain decomposition techniques using spectral methods,” *Calcolo*, vol. 24, no. 2, pp. 141–177, 1987.
- [94] A. Quarteroni and A. Valli, *Domain decomposition methods for partial differential equations*, BOOK. Oxford University Press, 1999.
- [95] V. I. Agoshkov, “Poincaré-Steklov operators and domain decomposition methods in finite dimensional spaces,” in *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1988, pp. 73–112.
- [96] F. Nataf, “Interface connections in domain decomposition methods,” in *Modern methods in scientific computing and applications*, Springer, 2002, pp. 323–364.
- [97] M. Aletti and D. Lombardi, “A reduced-order representation of the Poincaré–Steklov operator: An application to coupled multi-physics problems,” *International Journal for Numerical Methods in Engineering*, vol. 111, no. 6, pp. 581–600, 2017.
- [98] H. Schwarz, “Ueber einige abbildungsaufgaben.,” *Journal für die reine und angewandte Mathematik*, vol. 71, pp. 105–120, 1869.
- [99] P. L. Lions *et al.*, “On the Schwarz alternating method,” in *First international symposium on domain decomposition methods for partial differential equations*, Paris, France, vol. 1, 1988, p. 42.
- [100] W. P. Tang, “Generalized schwarz splittings,” *SIAM Journal on Scientific and Statistical Computing*, vol. 13, no. 2, pp. 573–595, 1992.
- [101] Q. Deng, “Timely communicaton: An analysis for a nonoverlapping domain decomposition iterative procedure,” *SIAM Journal on Scientific Computing*, vol. 18, no. 5, pp. 1517–1525, 1997.
- [102] S. Lui, “On accelerated convergence of nonoverlapping schwarz methods,” *Journal of computational and applied mathematics*, vol. 130, no. 1-2, pp. 309–321, 2001.
- [103] M. Ruzhansky, “Introduction to pseudo-differential operators,” 2014.
- [104] V. D. Liseikin, *Grid generation methods*. Springer, 1999, vol. 1.

- [105] Y. Saad and M. H. Schultz, “Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems,” *SIAM Journal on scientific and statistical computing*, vol. 7, no. 3, pp. 856–869, 1986.
- [106] Y. Saad, *Iterative methods for sparse linear systems*. SIAM, 2003.
- [107] A. N. Krylov, “On the numerical solution of the equation by which in technical questions frequencies of small oscillations of material systems are determined (in russian),” *Izvestija AN SSSR (News of Academy of Sciences of the USSR), Otdel. mat. i estest. nauk*, vol. 7, no. 4, pp. 491–539, 1931.
- [108] W. E. Arnoldi, “The principle of minimized iterations in the solution of the matrix eigenvalue problem,” *Quarterly of applied mathematics*, vol. 9, no. 1, pp. 17–29, 1951.
- [109] Y. Saad, “Krylov subspace methods for solving large unsymmetric linear systems,” *Mathematics of computation*, vol. 37, no. 155, pp. 105–126, 1981.

Appendices

Appendix A

Convergence rate calculation of DN and NN algorithms

A.1 Convergence rate calculation of Neumann-to-Neumann algorithm

Consider a Neumann-to-Neumann algorithm in application to Poisson's equation. The algorithm starts from u_2 :

$$\begin{aligned} u_{2y}^{n+1} &= u_{1y}^n \quad \text{on } \Gamma, \\ u_{1y}^{n+1} &= u_{2y}^{n+1} \quad \text{on } \Gamma. \end{aligned}$$

The corresponding IBCs for the error problem are:

$$\begin{aligned} e_{2y}^{n+1} &= e_{1y}^n \quad \text{on } \Gamma, \\ e_{1y}^{n+1} &= e_{2y}^{n+1} \quad \text{on } \Gamma. \end{aligned}$$

Since the error BVPs have homogeneous boundary conditions except on interface, the IBCs in error problem can be also formulated as:

$$-S_2 e_2^{n+1} = S_1 e_1^n \quad \text{on } \Gamma, \tag{A.1}$$

$$S_1 e_1^{n+1} = -S_2 e_2^{n+1} \quad \text{on } \Gamma. \tag{A.2}$$

The eigenfunctions of SP operators (N2D map) in application to Poisson's equation ($0 < x < 1, 0 < y < B$) follows that in [84][97]:

$$\phi_k(x) = \sin(k\pi x), \quad k = 1, 2, \dots \tag{A.3}$$

The corresponding eigenvalues (λ_1, λ_2) could be obtained by solving the following BVPs in the two sub-domains (Ω_1 and Ω_2) respectively:

$$\begin{aligned} -\nabla^2 e_1 &= 0, \quad \text{on } \Omega_1, & -\nabla^2 e_2 &= 0, \quad \text{on } \Omega_2, \\ e_1 &= 0, \quad \text{on } \partial\Omega_1/\Gamma, & e_2 &= 0, \quad \text{on } \partial\Omega_2/\Gamma, \\ e_{1y} &= \lambda_1 e_1, \quad \text{on } \Gamma, & e_{2y} &= \lambda_2 e_2, \quad \text{on } \Gamma. \end{aligned} \tag{A.4}$$

The eigenvalues are:

$$\lambda_{1k} = k\pi \coth(k\pi A), \lambda_{2k} = k\pi \coth(k\pi(B - A)). \quad (\text{A.5})$$

Next, expand e_i into the eigenfunctions $\phi_k(x)$ and consider the coefficient (\hat{e}_i) equations:

$$-\lambda_{2k}\hat{e}_2^{n+1} = \lambda_{1k}\hat{e}_1^n, \quad (\text{A.6})$$

$$\lambda_{1k}\hat{e}_1^{n+1} = -\lambda_{2k}\hat{e}_2^{n+1}. \quad (\text{A.7})$$

By transforming above equations, the convergence rate could be obtained:

$$\|\hat{e}_i^{n+1}\| = \left| \frac{\lambda_{1k}\lambda_{2k}}{\lambda_{1k}\lambda_{2k}} \right| \|\hat{e}_j^n\| = \|\hat{e}_j^n\|. \quad (i \neq j.)$$

A.2 Convergence rate calculation of Dirichlet-to-Neumann algorithm

Consider a (relaxed) Dirichlet-to-Neumann algorithm in application to Poisson's equation. The algorithm starts from u_2 :

$$\begin{aligned} u_{2y}^{n+1} &= u_{1y}^n \quad \text{on } \Gamma, \\ u_1^{n+1} &= \theta u_2^{n+1} + (1 - \theta)u_1^n \quad \text{on } \Gamma. \end{aligned}$$

Note $0 \leq \theta \leq 1$. The corresponding IBCs for the error problem are:

$$\begin{aligned} e_{2y}^{n+1} &= e_{1y}^n \quad \text{on } \Gamma, \\ e_1^{n+1} &= \theta e_2^{n+1} + (1 - \theta)e_1^n \quad \text{on } \Gamma. \end{aligned}$$

Since the error BVPs have homogeneous boundary conditions except on interface, the IBCs in error problem can be also formulated as:

$$-S_2 e_2^{n+1} = S_1 e_1^n \quad \text{on } \Gamma, \quad (\text{A.8})$$

$$e_1^{n+1} = \theta e_2^{n+1} + (1 - \theta)e_1^n \quad \text{on } \Gamma. \quad (\text{A.9})$$

Expand e_i into the eigenfunctions as shown in Eq.(A.3) and consider the coefficient (\hat{e}_i) equations:

$$-\lambda_{2k}\hat{e}_2^{n+1} = \lambda_{1k}\hat{e}_1^n, \quad (\text{A.10})$$

$$\hat{e}_1^{n+1} = \theta \hat{e}_2^{n+1} + (1 - \theta)\hat{e}_1^n. \quad (\text{A.11})$$

The eigenvalues $\lambda_{1k}, \lambda_{2k}$ are given in Eq.(A.5). By transforming the above two equations, the

convergence rate of the algorithm could be obtained:

$$\begin{aligned}
\|\hat{e}_i^{n+1}\| &= \left| 1 - \theta \left(1 + \frac{\lambda_{1k}}{\lambda_{2k}} \right) \right| \|\hat{e}_j^n\| \\
&= \left| 1 - \theta \frac{k\pi \coth(k\pi(B-A)) + k\pi \coth(k\pi A)}{k\pi \coth(k\pi(B-A))} \right| \|\hat{e}_j^n\|. \quad (i \neq j, k = 1, 2, \dots)
\end{aligned}$$