Doktors der Naturwissenschaften (Dr. rer. nat.)
PhD Thesis: *Applied mathematics*

Subspace correction methods for solving nonlinear parabolic equations on GPUs

---

Mehdi Babamehdi

Fakultät für Mathematik und Naturwissenschaften
Bergische Universität Wuppertal
Gaußstraße 20
Wuppertal - 42119, Germany



BERGISCHE
UNIVERSITÄT
WUPPERTAL

Dissertation

Bergische Universität Wuppertal

Fakultät für Mathematik und Naturwissenschaften

Subspace correction methods for solving nonlinear parabolic equations on GPUs

Mehdi Babamehdi

Supervisor:
Prof. Dr. Matthias Bolten

Bergische Universität Wuppertal
Gaußstraße 20
Wuppertal - 42119, Germany

Examination Committee

Doctoral Thesis Examination Committee

Chair and Supervisor:	Prof. Dr. Matthias Bolten Bergische Universität Wuppertal
External Examiner:	Prof. Peter Jimack University of Leeds
Internal Examiner:	Prof. Dr. Andreas Frommer Bergische Universität Wuppertal
Internal Examiner:	Prof. Dr. Michael Günther Bergische Universität Wuppertal

Author's declaration

I, *Mehdi Babamehdi* declare that this work, submitted in fulfillment of the requirements for Doktors der Naturwissenschaften (Dr. rer. nat.) in *Applied mathematics* at the *Fakultät für Mathematik und Naturwissenschaften*, *Bergische Universität Wuppertal*, Germany, is in its entirety my own work presented as a compiled thesis. All information sources and related literature are indicated and cited. This document has not been submitted for assessment at any other academic institutions.

Signature: _____

Mehdi Babamehdi

Date: October 28, 2026

Place: Wuppertal, Germany

d

Dedication

To Mahsa, Sarina, Nasrin and Narges
who by their lives and perseverance changed the women's conditions in my country.

To the young talented people
whose beautiful lives and hopes were given brutally and inhumanly in the flight PS752.

And to my parents.

Contents

1. Introduction **1**

2. Theoretical aspects of the numerical method **3**
2.1. Nonlinear partial differential equations . . . 3
2.2. Subspace correction . . . 4
2.2.0.1. Successive subspace correction methods . . . 7
2.2.0.2. Parallel subspace correction methods . . . 8
2.2.0.3. Multilevel methods . . . 9
2.2.0.4. Convergence theory . . . 10
2.2.1. Multigrid method . . . 11
2.2.1.1. Introduction to linear multigrid method . . . 11
2.2.1.2. Convergence theory . . . 13
2.2.2. Full Approximation Scheme (FAS) . . . 14
2.2.3. Domain decomposition method . . . 17
2.2.3.1. Schwarz method . . . 17
2.2.3.2. Convergence theory of the additive Schwarz method . . . 18
2.2.4. Nonlinear additive Schwarz method . . . 20

3. Numerical methods **23**
3.1. GPU architecture . . . 23
3.2. Full Approximation Scheme . . . 27
3.3. Nonlinear restricted additive Schwarz method . . . 30
3.3.1. Globally convergent inexact Newton method . . . 34
3.3.1.1. Jacobian-Free Newton Krylov method . . . 36
3.3.2. GMRES . . . 38
3.3.3. GPU architecture and the FAS-NRAS algorithm . . . 38

4. Results **41**
4.1. Nonlinear equations . . . 41
4.1.1. Zero-order nonlinear Poisson's equation . . . 41
4.1.2. The Allen-Cahn equation . . . 42
4.2. Finite element method . . . 44
4.2.1. Triangulation . . . 45
4.2.2. Finite Element space . . . 45
4.2.3. Discretization . . . 45
4.2.4. Treatment of boundary conditions . . . 46

4.3. FAS as a nonlinear solver 47
4.3.1. Nonlinear Poisson's equation 48
4.3.2. The Allen-Cahn equation 50

5. NRASM as a preconditioner-minimizer for FAS as a multilevel optimizer 57
5.1. FAS as a multilevel optimization method 57
5.2. Result 65

6. Conclusion 76

Bibliography 79

A. Appendix - Brief on functional analysis A 1

List of Figures

3.1. GPU architecture 25
3.2. GPU memory heirarchy 25
3.3. multiGPU computing unit and memory hierarchy and the features provided by CUDA API 26
3.4. V-Cycle, W-Cycle and full multigrid schemes 30
3.5. Decomposing domain Ω, extending subdomains by overlap of size δ and adding artificial boundary Γ between subdomains 31
3.6. Restriction matrices $\mathbf{R}_i$ and $\tilde{\mathbf{R}}_i$ in one dimension 32
3.7. GPU architecture and the Algorithm (3.1) 39

4.1. Function u in phase field problem from the Allen-Cahn equation. . . . 43
4.2. Two strategies of combining FAS and NRAS methods. Here, to avoid confusion by overcrowded figures, circles represent cells in each subdomains not DoFs and lines the interfaces between subdomains, and in the current work only h-multigrid is used and there is not p-multigrid. 48
4.3. Time to solution for different sizes of problem. 51
4.4. Development of the radius of sphere R_t computed analytically and numerically for the Allen-Cahn equation 53
4.5. Solution of the Allen-Cahn equation in a 2D slice orthogonal to $(1,0,0)$ in left column and on the line $(0,y,0)$ in right column at different time (a) $t = 0$ sec (b) $t = 0.2$ sec (c) $t = 0.5$ sec 54
4.6. $\|e\|_{L2}$ for the Allen-Cahn equation in Backward Euler and Crank-Nicolson time schemes 56

5.1. Random initial guess $rand(0,100)$ on a 2D slice $(1,0,0)$ in middle of the cube on the left and on the line $(0,y,0)$ on the right 66
5.2. Reduction of $\|F(u)\|$ by applying NRASM (red) as NE and V-Cycle FAS (blue) for $\tau = 10^{-3}$; (a) $rand(0,10)$ with $\nu_{NE} = 5$ $\nu_1 = \nu_2 = 2$ (b) $rand(0,100)$ with $\nu_{NE} = 8$ $\nu_1 = \nu_2 = 2$ (c) $rand(0,1000)$ $\nu_{NE} = 500$ $\nu_1 = \nu_2 = 10$ 68
5.3. Reduction of $\|F(u)\|$ by applying NRASM (red) as NE and V-Cycle FAS (blue) for $\tau = 10^{-2}$; (a) $rand(0,100)$ with $\nu_{NE} = 8$ $\nu_1 = \nu_2 = 2$ (b) $rand(0,1000)$ $\nu_{NE} = 500$ $\nu_1 = \nu_2 = 10$ 68
5.4. Reduction of $\|F(u)\|$ by applying NRASM (red) as NE and V-Cycle FAS (blue) in $\tau = 10^{-1}$; (a) $\nu_{NE} = 10$, $\nu_1 = \nu_2 = 5$ (b) $\nu_{NE} = 20$, $\nu_1 = \nu_2 = 5$ with initial guess $rand(0,100)$ 69

5.5. Reduction of $\|F(u)\|$ by applying NRASM (red) as NE and V-Cycle FAS (blue) in $\tau = 4 \times 10^{-1}$; (a) with initial guess $rand(0,1)$ $\nu_{NE} = 20$, $\nu_1 = \nu_2 = 3$ (b) with initial guess $rand(0,10)$ $\nu_{NE} = 25$, $\nu_1 = \nu_2 = 5$ with initial guess $rand(0,100)$ $\nu_{NE} = 250$, $\nu_1 = \nu_2 = 20$ 70
5.6. (a) Initial guess $rand(0,100)$ with $\nu_{NE} = 20$, $\nu_1 = \nu_2 = 5$ (b) After applying 20 iteration of NRASM (c) Initial value and preconditioned values, plot over a line in $(0,y,0)$ (d) After first V-Cycle FAS (e) The solution after convergence by 3 iterations of V-Cycle FAS 73
5.7. (a) Initial $F(u_0)$ $rand(0,100)$ with (b) After applying 10 iteration of NRASM as NE method(c) After applying 20 iteration of NRASM as NE (d) initial $F(u_0)$, and after applying $\nu_{NE} = 10$ and $\nu_{NE} = 20$ on line $(0,y,0)$ (e) $F(u)$ after 3 number of V-Cycle FAS iteration pre- and post-smoothing $\nu_1 = \nu_2 = 5$ 74
5.8. Reduction of $\|F(u)\|$ for 6 steps of mesh refinement by applying NRASM (red) as NE and V-Cycle FAS (blue) with initial guess $rand(0,1)$; (a) $\tau = 10^{-1}$ and $\nu_{NE} = 8$, $\nu_1 = \nu_2 = 3$, $\tau = 4 \times 10^{-1}$ and $\nu_{NE} = 20$, $\nu_1 = \nu_2 = 3$ 75
5.9. development of the radius of sphere over time ; (blue) convex energy functional, (red) non-convex energy functional $E_{IM}(.,.)$, in $\tau = 0.4$, random number is $rand(0,1)$ for the rest $rand(0,100)$; left with 5 steps mesh refinement, right 6 steps mesh refinement 75

List of Tables

3.1. Major hardware features of NVIDIA A100. 24

4.1. Number of FAS V-Cycle necessary to get the error of the order of discretization in four different size of problem and the ratio of error reduction in each mesh refinement step 50
4.2. Number of V-Cycle FAS iterations necessary in each step of mesh refinement to get the convergence for solving the Allen-Cahn equation 53

5.1. relative *tol* in the iterative methods in NE-FAS-NRAS algorithm . . . 65

Nomenclature

$(.,.)$	inner product on function space $L^2(\Omega)$
$\mathbf{R}_i$	Restriction matrix $\in \mathbb{R}^{n \times n}$
$\mathfrak{D}n(u)(\nu)$	Fréchet derivative of $n(u)$ on ν
$\mathcal{V}$	Normed vector function space
$\mathcal{V}'$	Dual space of normed vector function space $\mathcal{V}$
Γ_{ij}	Interface between subdomain Ω_i and Ω_j
$\langle .,. \rangle$	Dual pairing
$\|.\|_a$	Energy norm $\|.\|_a \coloneqq \sqrt{a(.,.)}$
$\|.\|_{\mathcal{V}}$	norm on function space $\mathcal{V}$
$\|.\|_{L^2}$	norm on function space $L^2(\Omega)$
$\mathbb{I}$	Index set $i = 0, \ldots, J$
$\mathcal{N}(A)$	Null space of operator A
$\mathcal{R}(u^*; f)$	Smoothing operator
Ω	A bounded, open, and nonempty domain in $\mathbb{R}^3$
∂_G	Gâteaux derivative
$\tilde{\Omega}$	Interior subdomain
ξ	Perturbation parameter
$a(.,.)$	Bilinear form
$B(u)$	Approximation of A^{-1}
$B(u^*; r)$	Ball around u^* with radius r
E	Error propagation operator
$H^k(\Omega)$	Sobolev space with weak derivatives up to order k

$H_0^k(\Omega)$ Sobolev space with zero boundary value

I_i Interpolation operator

$L^p(\Omega)$ Lebesque space

P_i Projection operator $P_i : \mathcal{V} \to \mathcal{V}_i$

Q_i Projection operator $Q_i : \mathcal{V}' \to \mathcal{V}_i'$

Acknowledgements

First of all, I owe special thanks to my advisor Matthias Bolten for the great opportunity to write my thesis in his group, for all the time he spent in discussions with me, for his patience, and the encouragement he gave me during the almost last five years. I would also like to thank Robert Speck who also gave me the opportunity to be in his group and his support. I am grateful to Sriramkrishnan Muralikrishnan, my colleague, and my friend, who spent time discussing different topics related to my thesis, which was a significant help. I also want to express my sincere gratitude to Jacob Schroder who generously gave me time for the discussion, and the last chapter of the thesis is the fruit of our discussion. I should also thank my former colleagues, office-mate and friend Julian, with whom we started a regular scientific discussion, during the Corona pandemic, which the informal discussion led to a bigger group discussion. I would also like to thank all of my former and current colleagues in advance time integrator group (former PinT), namely Ruth Partzsch, Sriramkrishnan Muralikrishnan, Thomas Saube and Abdelouahed Ouardghi. Finishing this thesis would have been impossible without their support, their hints, and all the long discussions in front of whiteboards and blackboards, or in the video-conference room at JSC.

The author also sincerely thanks Matthias Bolten, Jacob Schroder, and Sriramkrishnan Muralikrishnan who kindly spent time proofreading the thesis and for their comments and feedback.

I am especially grateful to my former office-mate, Julian, Thomas and Sina for being good friends in all the years, who patiently supported me going through difficult phases of my study as a PhD student.

In particular, I would like to thank my parents, my sisters Fatemeh and Zahra. Finally, I am grateful for the encouragement and help of all my friends, especially Siamak, who for years has been my best friend and who helped me going through difficult phases of my life, Freshte, although far away, she helped me going through the difficult period of the Corona Pandemic and in the last few months, Sarah.

Abstract

Nonlinear problems are often more challenging to solve than linear ones because they may involve non-convexity, heterogeneity, and strong nonlinearities. These difficulties are compounded in modern high-performance computing systems, where algorithms must efficiently exploit heterogeneous architectures combining CPUs and multiple GPUs. Developing nonlinear solvers that address both the mathematical and computational challenges is therefore important.

This thesis develops and analyzes multilevel nonlinear methods for the numerical solution of the Allen–Cahn equation, with an emphasis on matrix-free implementation on graphics processing units (GPUs). The full approximation scheme (FAS) serves as the nonlinear multigrid solver, while the nonlinear restricted additive Schwarz method (NRASM) is used as a smoother in the FAS-NRASM algorithm. I prove that NRASM satisfies the smoothing condition required by FAS. Numerical results show that, for the strategy considered, the number of FAS V-cycles required to solve the discretized Allen–Cahn equation is independent of problem size. Because standard FAS relies on convexity, this requirement can constrain the time-step size.

To address more nonlinear and non-convex problems, I analyze FAS as a multilevel optimization algorithm and NRASM as a minimization method. This analysis motivates the NE-FAS-NRAM algorithm, which uses NRASM as a preconditioner-smoother. Theoretical analysis and numerical experiments demonstrate that the method can solve non-convex, highly nonlinear problems and allows time steps approximately 400 times larger than in the convex case considered. It converges for a moderate range of random initial guesses, although it may fail for very poor initial guesses. In GPU experiments on the Allen–Cahn equation, the new algorithm achieved a reported speed-up of 40,000× in convergence compared with solving the equation on a CPU using the FAS-NRASM algorithm.

1. Introduction

As human beings, we have tried to understand different phenomena in the universe. We are curious to understand how some processes in the human body develop, or how different things work in nature. This includes the behavior of tiny creatures like the development of the number of bacteria or some atomic motion to the movement of huge objects like stars or causes of different phenomena. In order to understand these different phenomena, what causes a thing to happen, and how one quantity changes in response to another quantity in time and space, we can rely on mathematics, finding the natural law governing these complicated phenomena, demonstrated in some form of *equations*, usually in some forms of differential equations. These equations determine the behavior of some phenomena or causes of some events. Partial differential equations are one of the forms of these equations that occur in different places, for instance, describing the motion of molecules with other molecules, how water flows through a river, how waves transmit through a medium, and so many others.

In order to understand these phenomena in a certain scenario, a specific spatial and temporal state, we need to solve the governing equation. In many problems, the conditions imposed by the phenomena are complicated, so that the governing equation can not be solved analytically, like some nonlinear relationship between different quantities. In such cases, we often rely on a numerical approximation of the quantity, which we usually call it *solution*. The numerical approach needs some methods which are the procedure for deriving the solution or an approximation of the solution from the differential equation.

Another component of the procedure is the *architecture* we rely on to use the numerical methods in solving the differential equations. If the problem is small, it can be solved with a pen and paper, which rarely happens, or we can rely on computers. Depending on the difficulty of the problem, which includes how large the problem is, the accuracy required, the computational power can vary. By advancing technology in the computer industry and introducing new devices like Graphical Processing Units (GPU), we can now look for the solution of more complicated problems and/or demand more accurate solutions. Then a challenge here is to see how the first component, a numerical method which is also called *algorithm*, works with the second component, the architecture, of this numerical procedure. This means that studying current numerical methods with respect to the new architecture of GPUs, trying to adapt the methods for the new devices or to find new numerical methods suited better to the architecture.

The primary nonlinear problem tackled in the current project is the Allen-Cahn equation. The Allen-Cahn equation is one of the governing equations in phase-field problems. The idea of the phase field method describes the material interfaces during phase transitions. The method can be traced back to Rayleigh, Gibbs, and Van der Waals [Ray64; Van79]. It represents material interfaces as thin layers of finite thickness over which material properties vary smoothly. Such a thin layer of the width $O(\epsilon)$ is often referred as a *diffuse interface*. Generally speaking, the phase-field method was introduced to model solid-liquid phase transition in which surface tension and non-equilibrium thermodynamics behavior become important at the interface. The phase-field method is not only used in phase transition, but it also has some other applications like the moving interface problem in multiphase flow, astrophysics, biology. The moving interface problem belongs to a larger category, which is *geometric moving interface problems* in which the motion of the interfaces is governed by some curvature-dependent geometric law. The most well-known geometric moving interface problem is *mean curvature flow*, and among the equations in this category, the Allen-Cahn equation (Equation (4.3)) is the best known. More details on the Allen-Cahn equation will be presented in Section (4.1.2). Research has been done in numerical analysis of the Allen-Cahn equation, including error analysis [GLV11; LM06; LMP], studying different time schemes [SS17; XT06], and spatial discretization schemes [FL15; Che+98; BBH09].

In the present thesis, the main focus is on developing a numerical algorithm to solve the Allen-Cahn equation. Since the main architecture on which the problem is solved numerically is GPU, the numerical algorithm should be suitable for GPU architecture. The main method studied in the following chapters is the *subspace correction method*. The main solver for the Allen-Cahn equation as a nonlinear parabolic PDE is nonlinear multigrid, called *full approximation scheme (FAS)*, which is a successive subspace correction method. As a multigrid method needs a proper smoother, the *nonlinear restricted additive Schwarz method*, as another form of subspace correction method, i.e., a parallel subspace correction method, is used for this purpose. The question is whether it is a proper smoother. This is the main topic of Chapter (2). In that chapter, the subspace correction methods will be presented, and then multigrid and restricted additive Schwarz method, first linear and then extend it to nonlinear, will be discussed and finally at the end of the chapter, the smoothing condition for nonlinear restricted additive Schwarz method is proved (Corrolary (2.11.1)). Chapter (3) is devoted to the numerical methods and is studied from the implementation point of view. In Chapter (4), the results of using nonlinear multigrid in solving two nonlinear equations, one being the Allen-Cahn equation, will be presented. And finally, in the last chapter, the idea of using nonlinear restricted additive Schwarz method as a *preconditioner-smoother* is introduced, studied theoretically and the numerical results of using the idea in practice will be presented.

2. Theoretical aspects of the numerical method

In this chapter, I focus mainly on the analytical aspects of the algorithm used to solve nonlinear parabolic partial differential equations, i.e. how to analyze the convergence of the method used to solve the nonlinear PDE. To build the foundation of the algorithm, I start by explaining subspace correction methods, where I mainly borrow material from [Xu92; Yse93]. The main solver, later used as a multilevel optimization method in Chapter (5), utilized to solve the nonlinear PDE is Full Approximation Scheme (FAS). This numerical method has different components, smoother and coarse-grid correction operators. In order to build the foundation for the theoretical analysis of FAS and its components, I start by studying the linear multigrid and the main condition necessary to demand from different components of the multigrid, i.e., the coarse grid correction and smoother, in order to have an efficient numerical method. Later, I expand these conditions to nonlinear multigrid, FAS, and introduced nonlinear restricted additive Schwarz method as a smoother (in that section I mainly followed [DH97]). In order to study the nonlinear restricted additive Schwarz (NRAS) method, here I also start from the linear restricted additive Schwarz method and go to the nonlinear version. Later I look at nonlinear restricted additive Schwarz as a smoother in FAS and see if the NRAS method can satisfy the necessary condition demanded from a proper nonlinear smoother. In Corollary (2.11.1), I prove the smoothing condition of the NRAS method.

2.1. Nonlinear partial differential equations

Let us assume $\mathcal{V}$ and $\mathcal{U}$ are two function spaces and $n : \mathcal{V} \to \mathcal{U}$ and $u \in \mathcal{V}$. In general, any PDE, linear or nonlinear, having the following form,

$$F(\mathfrak{D}^k n(u), \mathfrak{D}^{k-1} n(u), \dots, \mathfrak{D} n(u), n(u), u) = 0 \quad \forall u \in \mathcal{V} \tag{2.1}$$

where k^{th}-order PDE and $F : \mathcal{U}^{n^k} \times \mathcal{U}^{n^{k-1}} \times \cdots \times \mathcal{U} \times \mathcal{V} \to \mathcal{U}'$

PDEs in general can belong to different categories depends upon the nonlinearity as follows [Eva22; RR06]

Definition 2.1

i) PDE (2.1) is called ***linear*** *if it has the form*

$$\sum_{|\alpha|\leq k} a_\alpha(u)\mathfrak{D}^\alpha n(u) = f(u)$$

for given functions a_α, $|\alpha| \leq k$, *and* $f(u) \in \mathcal{U}$, *and if* $f = 0$ *it is homogeneous.*

ii) PDE (2.1) is called ***semilinear*** *if it has the form*

$$\sum_{|\alpha|=k} a_\alpha(u)\mathfrak{D}^\alpha n(u) + a_0\Big(\mathfrak{D}^{k-1}n(u), \ldots, \mathfrak{D}n(u), n(u), u\Big) = 0$$

iii) PDE (2.1) is called ***quasilinear*** *if it has the form*

$$\sum_{|\alpha|=k} a_\alpha\Big(\mathfrak{D}^{k-1}n(u), \ldots, \mathfrak{D}n(u), n(u), u\Big)\mathfrak{D}^\alpha n(u) + a_0\Big(\mathfrak{D}^{k-1}n(u), \ldots, \mathfrak{D}n(u), n(u), u\Big) = 0$$

iv) PDE (2.1) is called ***fully nonlinear*** *if it depends on nonlinearly to the highest order derivative,*

2.2. Subspace correction

In this section, the main subspace correction methods are briefly discussed. They are a group of algorithms where domain decomposition and multigrid are the main algorithms. We will return to each of these methods in the following section. The central idea behind these algorithms are simple and can be briefly explained in the following three steps. Defining a normed vector space $\mathcal{V}$ (e.g., Hilbert space H^1 or closed subspace H_0^1) which is equipped with an inner product $\langle ., . \rangle$ inducing the norm as

$$\|u\|^2 = \langle u, u \rangle$$

and defining a bilinear form as $a : \mathcal{V} \times \mathcal{V} \to \mathcal{R}$, we obtain an energy norm by using bilinear form $a(.,.)$ as

$$\|u\|_a^2 = a(u, u)$$

where $u \in \mathcal{V}$. Assuming $\mathcal{V} \subset H$ where H is a Hilbert space and using the Riesz representative [Kre91], we can find a bounded linear operator $A : \mathcal{V} \to \mathcal{V}'$ which has the following relation

$$(Au, v) = a(u, v), \ \ \forall v \in \mathcal{V}$$

If we assume that the linear operator A also has symmetry properties and is positive definite with respect to the inner product, then the primary goal is to solve a linear operator equation as

$$Au^* = f \tag{2.2}$$

where $f \in \mathcal{V}'$ in which $\mathcal{V}'$ is a dual space of $\mathcal{V}$. Defining $\mathcal{V}_h \in \mathcal{V}$ and using the Galerkin projection to project the problem into a finite dimension subspace, then having a basis for the function space $\mathcal{V}_h$, the continuous equation can be reformulated (for instance by using the finite element method [Bra97]) to discretized form as a linear system of

$$\mathbf{Au} = \mathbf{f} \tag{2.3}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$, is a finite dimension operator (matrix) of the original PDE, $\mathbf{f} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^n$. Then the primary goal is to solve the linear system in Equation (2.3) by using the subspace correction method as an iterative method. It should be noted that in the theoretical analysis of the methods introduced in this chapter I will look at the methods in the continuous form. Otherwise it is mentioned explicitly, later in the following chapters, in implementation the discrete form will be considered. The general subspace correction method can be formulated as follows:

Algorithm 2.1

Let $\mathcal{V}$ be a function space as defined above and $u \in \mathcal{V}$. Assume u^n is given, the new approximation of u as u^{n+1} computed in three steps as follows

1- Compute residual $r^n = f - Au^n$

2- Solve approximately $Ae^n = r^n$, i.e., by replacing $B \approx A^{-1}$

3- Correct current approximation of u by $u^{n+1} = u^n + e^n$

As is clear, finding the proper operator $B : \mathcal{V} \to \mathcal{V}'$ is the primary goal of subspace correction methods. The way in which subspace correction treats the problem is to decompose the space $\mathcal{V}$ into subspaces of $\mathcal{V}$ as $\mathcal{V}_i \subset \mathcal{V}$ where $\mathbb{I} = 1, 2, \ldots, J$ is an indices set, then the space $\mathcal{V}$ can be written as

$$\mathcal{V} = \sum_{i=1}^{J} \mathcal{V}_i$$

Therefore, for any vector in $\mathcal{V}$, $v \in \mathcal{V}$, it can be written as a unique linear combination of $v_i \in \mathcal{V}_i$ as

$$v = \sum_{i=1}^{J} v_i$$

In order to project onto subspaces $\mathcal{V}_i(\mathcal{V}_i')$, we need to have two orthogonal projections. The first projection operator $Q_i : \mathcal{V}' \to \mathcal{V}_i'$ is defined as

$$(Q_i u, v_i) = (u, v_i) \quad \forall v_i \in \mathcal{V}_i' \tag{2.4}$$

where $u \in \mathcal{V}'$ and the second orthogonal projection operator $P_i : \mathcal{V} \to \mathcal{V}_i$ is also defined as

$$a(P_i u, v_i) = a(u, v_i) \quad \forall v_i \in \mathcal{V}_i \tag{2.5}$$

where $u \in \mathcal{V}$. If $u \in \mathcal{V}$ is the solution of Equation (2.2), then $P_i u \in \mathcal{V}_i$ is the Ritz approximation of this equation in the subspace $\mathcal{V}_i$. By including the right-hand side of Equation (2.2) into the subspace correction and the approximation of $u_i \in \mathcal{V}_i$, we can introduce Ritz approximations $A_i : \mathcal{V}_i \to \mathcal{V}_i$ of the operator A with respect to subspace $\mathcal{V}_i$, then, we can define

$$(A_i u_i, v_i) = (A u_i, v_i) \ \forall u_i, v_i \in \mathcal{V}_i \tag{2.6}$$

We can then easily derive the relation between operators A_i, P_i and Q_i by

$$(A_i P_i u, v_i) = a(P_i u, v_i) = a(u_i, v_i) = (A u_i, v_i) = (Q_i A u, v_i) \tag{2.7}$$

which results in the following identity,

$$A_i P_i = Q_i A \tag{2.8}$$

The approximation $P_i u$ of the solution u, by using (2.8), satisfies also the following identity,

$$A_i P_i u = Q_i f \tag{2.9}$$

Defining $u_i = P_i u$ and $f_i = Q_i f$, then on each subspace we solve the following local problem as follows,

$$A_i u_i = f_i \tag{2.10}$$

Having the orthogonal projection operators and definitions, and if $u_i \in \mathcal{V}_i$ and $u = u_0 + u_1 + \cdots + u_J,$, the *subspace correction* method can be formulated as

$$\tilde{u} \leftarrow \tilde{u} + P_i(u - \tilde{u}) \tag{2.11}$$

with respect to subspace $\mathcal{V}_i$ and $\tilde{u}$ is the approximation. In iterative methods, we usually consider the residual of Equation (2.2), then by reformulating the iterative method, we can rewrite (2.11) as

$$\tilde{u} \leftarrow \tilde{u} + A_i^{-1} Q_i(f - A\tilde{u}) \tag{2.12}$$

In a large scale problem, usually computing A_i^{-1} can be too expensive to get a computationally reasonable method, therefore, we usually replace A_i^{-1} with another symmetric positive definite operator like $B_i : \mathcal{V}_i \to \mathcal{V}_i$ which represents an accurate enough approximation of the inverse of A_i but is computationally cheaper to compute. Therefore, the subspace correction method (2.12) changes to

$$\tilde{u} \leftarrow \tilde{u} + B_i Q_i(f - A\tilde{u}) \tag{2.13}$$

2.2.0.1. Successive subspace correction methods

In this algorithm, the approximation of one subspace is corrected using the most up-to-date approximation of u. If we start from $u^0 = u^{old}$ where u^{old} is the given approximation of the solution u of Equation (2.2), the correction of the residual in $\mathcal{V}_1$ gives

$$u^1 = u^0 + B_1 Q_1 (f - Au^0)$$

the new approximation of $u_2 \in \mathcal{V}_2$ by using $u_1 \in \mathcal{V}_1$ we can get

$$u^2 = u^1 + B_2 Q_2 (f - Au^1)$$

and proceeding further in a successive way for all $\mathcal{V}_i \; for \; i \in \mathbb{I}$. Then we can have a successive subspace correction algorithm as follows,

Algorithm 2.2

Let $u^0 \in \mathcal{V}$ be given and assume that $u^k \in \mathcal{V}$ has been obtained. Then for $i \in \mathbb{I}$, the index set $\mathbb{I} = 0, 1, \ldots, J$, $u^{k+1} \in \mathcal{V}$ is defined by

$$u^{k+i/J} = u^{k+(i-1)/J} + B_i Q_i (f - Au^{k+(i-1)/J})$$

This algorithm is like the *Gauss-Seidel* method. We can also look at the impact of the successive subspace correction method on the error. Let us define

$$T_i : \mathcal{V} \to \mathcal{V}_i$$
$$T_i = B_i Q_i A_i$$

then by using (2.8) we have $T_i = B_i A_i P_i$. Note that T_i is symmetric with respect to $a(.,.)$ and non-negative and $T_i = P_i$ if $B_i = A_i^{-1}$. If u^* is the exact solution of (2.2), we can reformulate Algorithm 2.2 as following,

$$u^* - u^{k+i/J} = (I - T_i)(u^* - u^{k+(i-1)/J}) \quad i \in \mathbb{I}$$

In studying the convergence rate of an iterative method, we usually study the proper norm of the error propagation operator. Then let us denote $E : \mathcal{V} \to \mathcal{V}$ as the *error propagation operator*, a successive subspace correction is reformulated as

$$u^* - u^{k+1} = E_J (u^* - u^k) \tag{2.14}$$

where

$$E_J = (I - T_J)(I - T_{J-1}) \ldots (I - T_1) \tag{2.15}$$

Assume $E_0 = I$ to be an identity operator in $\mathcal{V}$, and $E = E_J$, we can also see that

$$E_{i-1} - E_i = T_i E_{i-1}$$

then we can derive the following equality

$$I - E_i = \sum_{j=1}^{i} T_j E_{j-1}$$

The algorithm introduced above is non-symmetric. There is also a symmetric version of the algorithm. In the symmetrized version, Algorithm (2.2) is changed to the following form,

$$u^{k+i/(2J)} = u^{k+(i-1)/(2J)} + B_i Q_i (f - A u^{k+(i-1)/(2J)}) \ for \ i \in 0, 1, \ldots, J$$

and

$$u^{k+(J+i)/(2J)} = u^{k+(J+i-1)/(2J)} + B_i Q_i (f - A u^{k+(J+i-1)/(2J)}) \ for \ i \in 0, 1, \ldots, J$$

and the error propagation operator is changed to the form

$$E_J^{sym} = (I - T_1)(I - T_2) \ldots (I - T_J)(I - T_J)(I - T_{J-1}) \ldots (I - T_1)$$

We can also see that $E_J^{sym} = E_J^* E_J$ where E_J^* is the adjoint of E_j with respect to $a(.,.)$.

2.2.0.2. Parallel subspace correction methods

Let u^{old} be a given approximation of Equation (2.2). Calculating the residual based on the current approximation $r^{old} = f - Au^{old}$. Then we define the residual equation as

$$Ae = r^{old}.$$

Next, we correct the current approximation by the correction e_i, which is computed by the projected equation (2.2) to the subspace $\mathcal{V}_i$ as follows,

$$A_i e_i = Q_i r^{old}.$$

As mentioned previously, computing the inverse of A_i is computationally expensive, we use an approximated operator B_i and the equation to compute the new correction $\tilde{e} \in \mathcal{V}_i$ will be as follows,

$$\tilde{e_i} = B_i Q_i r^{old}.$$

Similar to the successive subspace correction method, we compute the correction $\tilde{e_i}$ for all subspaces $\mathcal{V}_i, \ i \in I$ in parallel based on the previous approximation of u^{old} and correct the current approximation as follows,

$$u^{new} = u^{old} + \sum_{i \in \mathbb{I}} \tilde{e}_i$$

Then substituting the equation for computing the residual into the equation of correction, we will have,

$$u^{new} = u^{old} + R(f - Au^{old})$$

where the operator $R : \mathcal{V} \to \mathcal{V}$ is the parallel subspace correction operator and is defined as

$$R = \sum_{i \in \mathbb{I}} B_i Q_i \tag{2.16}$$

Algorithm 2.3

Given $u^{old} \in \mathcal{V}$ applying the parallel subspace correction scheme to obtain new approximation of the solution we will have

$$u^{new} = u^{old} + R(f - Au^{old})$$

with the operator $R = \sum_{i \in \mathbb{I}} B_i Q_i$.

As is clear, in contrast to successive subspace correction, for parallel subspace correction, the equation on each subspace does not depend on the new correction of the other subspaces and therefore all the subspaces can work on the local problem independently in each iteration of the method. Analogously, parallel subspace correction is like the Jacobi method, whereas successive subspace correction methods can be seen as like the Gauss-Seidel method.
Finally, we define at the error propagation of parallel subspace correction as follows,

$$E_J = \sum_{i \in \mathbb{I}} I - T_i \tag{2.17}$$

We will need the error propagation operator later to analyze the convergence of FAS.

2.2.0.3. Multilevel methods

The multilevel algorithm is based on a nested sequence of subspaces as

$$\mathcal{M}_1 \subset \mathcal{M}_2 \subset \cdots \subset \mathcal{M}_J = \mathcal{V} \tag{2.18}$$

Similar to the subspace correction methods introduced above, we also define two orthogonal operators with respect to $(.,.)$ and $a(.,.)$ such that $\hat{Q}_k, \hat{P}_k : \mathcal{M}_J \to \mathcal{M}_k$, we also need to define the local operator as

$$\hat{\mathcal{A}}_k : \mathcal{M}_k \to \mathcal{M}_k$$

and then using the Ritz representation theorem [Kre91] we have

$$(\hat{\mathcal{A}}_k u_k, \nu_k) = a(u_k, \nu_k) \;\; \forall u_k, \;\; \nu_k \in \mathcal{M}_k$$

The multilevel method is then applied in two steps as we can see in the following algorithm,

Algorithm 2.4

Let $\hat{B}_k : \mathcal{M}_k \to \mathcal{M}_k$ be the approximation of the inverse of $\hat{A}_k$ and $g \in \mathcal{M}_k$, then $\hat{B}_k : \mathcal{M}_k \to \mathcal{M}_k$ is defined as follows,

i) $u^1 = \hat{B}_{k-1}\hat{Q}_{k-1}g$

ii) $\hat{B}_k g = u^1 + \hat{R}_k(g - \hat{\mathcal{A}}_k u^1)$

where the operator $\hat{R}_k : \mathcal{M}_k \to \mathcal{M}_k$ is called a *smoother*. We will return to this operator and the necessary properties we demand of it in Section (2.2.1).
The error propagation operator for the multilevel method can also be computed as

$$\hat{E}_k = I - \hat{B}_k\hat{A}_k = (I - \hat{R}_k\hat{A}_k)(I - \hat{B}_{k-1}\hat{A}_{k-1}\hat{P}_{k-1}) \tag{2.19}$$

Remark 2.1 *The SSC methods and multilevel methods are closely connected. To see the relationship, we can look at the recurrence relation for $I - \hat{B}_k\hat{A}_k\hat{P}_k$ (instead of $I - \hat{B}_{k-1}\hat{A}_{k-1}$). Multiplying both sides of (2.19) by $\hat{P}_k$ and using the equality $\hat{P}_{k-1}\hat{P}_k = \hat{P}_{k-1}$, we can have*

$$I - \hat{B}_k\hat{A}_k\hat{P}_k = (I - \hat{R}_k\hat{A}_k\hat{P}_k)(I - \hat{B}_{k-1}\hat{A}_{k-1}\hat{P}_{k-1})$$

and since $\hat{P}_J = I$, then the successive application of the above identity yields

$$I - \hat{B}_J\hat{A}_J = (I - \hat{T}_J)(I - \hat{T}_{J-1})\dots(I - \hat{T}_1) \tag{2.20}$$

where

$$\hat{T}_1 = \hat{P}_1, \quad \hat{T}_i = \hat{R}_i\hat{A}_i\hat{P}_i \quad i \in \mathbb{I}$$

2.2.0.4. Convergence theory

In this section, the general abstract theory for the subspace correction methods is presented. In the convergence analysis of an iterative method, we generally look at the properties of the error propagation operator E. Defining error as $e_i = u^* - u_i$ we can have the error propagation $E_i : \mathcal{V}_i \to \mathcal{V}_i$ defined above for each subspace correction method, we can compute e_i such that

$$e_i = E_i e_{i-1} \tag{2.21}$$

In order to study the convergence rate of any iterative method, we look at the bounds of a proper norm of the error propagation operator E. We will look at the error propagation for parallel and successive subspace correction methods in different ways.
For the successive subspace correction method, we will establish the contraction property such that defining $0 < \delta < 1$

$$\|E_J\|_A = \sup_{\nu \in \mathcal{V}} \frac{\|E_J\nu\|_A}{\|\nu\|_A} \leq \delta$$

where E_J is defined in (2.15). In the parallel subspace correction method, on the other hand, we look at the estimate of condition number of (2.17).
Then to analyze $\kappa(BA)$ and $\|E_J\|_A$ we look mainly at two constant parameters defined in the following assumptions:

Assumption 2.2

Let K_0 be the minimum constant such that $\forall u \in \mathcal{V}$ there exists a decomposition $u = \sum_{i\in\mathbb{I}} u_i \quad \forall u_i \in \mathcal{V}_i$ such that

$$\sum_{i\in\mathbb{I}} (B_i^{-1} u_i, u_i) \leq K_0 (Au, u) \tag{2.22}$$

*The condition defined above is also the **stability condition** of the decomposition of $\mathcal{V}$. We can also look at the condition introduced above in terms of the norm with respect to $a(u,u)$ and $a_i(u_i,u_i)$ which also refer to the energy of u and u_i.*

Assumption 2.3

For any $S \subset \{1,2,\ldots,J\} \times \{1,2,\ldots,J\}$ and $u_i, \nu_i \in \mathcal{V}$ for $i \in \mathbb{I}$, then exist a K_1 such that

$$\sum_{(i,j)\in S} a(T_i u_i, T_j \nu_i) \leq K_1 \left(\sum_{i\in\mathbb{I}} (T_i u_i, u_i) \right)^{\frac{1}{2}} \left(\sum_{j\in\mathbb{I}} (T_j \nu_j, \nu_j) \right)^{\frac{1}{2}} \tag{2.23}$$

*This inequality is called **strengthened Cauchy-Schwarz** inequality.*

These two assumptions, in particular the strengthened Cauchy-Schwarz inequality, will be used to analyze the upper bound of the additive Schwarz method operator in (2.2.3.2), which we later need to show if the nonlinear additive Schwarz method is a proper smoother for the full approximation scheme in (2.2.4).

2.2.1. Multigrid method

2.2.1.1. Introduction to linear multigrid method

In this section, we look at the multilevel V-Cycle multigrid. Consider a nested sequence of subspaces as we defined in (2.18). In addition, let $a(.,.)$ and $(.,.)_k$ be symmetric positive definite bilinear forms on the subspaces $\mathcal{V}_k$ for $k \in \mathbb{I}$. As we saw in the subspace correction methods, we also need two linear operators $Q_k^{mg} : \mathcal{V}'_{k+1} \to \mathcal{V}'_k$ and $P_k^{mg} : \mathcal{V}_{k+1} \to \mathcal{V}_k$ to project functions into coarse subspaces. For $u \in \mathcal{V}_k$, these two operators in the multigrid method are defined as follows,

$$(Q_k^{mg} u, \nu)_k = (u, \nu)_{k+1} \qquad \forall \nu \in \mathcal{V}'_k \tag{2.24}$$

$$a_k(P_k^{mg} u, \nu) = a_{k+1}(u, \nu) \qquad \forall \nu \in \mathcal{V}_k \tag{2.25}$$

Another component of the multigrid method is the *smoother operator* $\mathcal{R}_k : \mathcal{V}_k \to \mathcal{V}_k$ which is a symmetric operator with respect to the inner product $(.,.)_k$ and satisfies $u_k^* = \mathcal{R}_k(u^*; f)$, we also denote the adjoint of $\mathcal{R}_k$ with respect to $(.,.)$ by $\mathcal{R}_k^t$. Then by defining a linear operator $B_k^{mg} : \mathcal{V}_k \to \mathcal{V}_k$ as an approximation of the inverse of A_k (we will formulate it in the next section), the multigrid algorithm to solve Equation (2.2) is as follows,

Algorithm 2.5

1) *Smoothing step: By given initial value u^0, we can compute u^l for $l = 1, \ldots, m(k)$ by*

$$u^{l+1} = u^l + \mathcal{R}_k^t(f - A_k u^l). \tag{2.26}$$

2) *Coarse grid correction: Define $u^{m(k)+1} = u^{m(k)} + q^p$ where the correction, q^i for $i = 1, \ldots, p$, is defined by*

$$q^i = B_{k-1}^{mg}[Q_{k-1}^{mg}(f_k - A_k u^{m(k)})]$$

3) *Set $B_k^{mg} g = u^{2m(k)+1}$ where u^l is defined for $l = m(k)+2, \ldots, 2m(k)+1$ by equation (2.26)*

where $m(k)$ is the number of the smoothing step in level k. Before analyzing the convergence of the multigrid method, it is also worth looking at how these components should treat different components of error e in order for the multigrid method to work efficiently.

Let us then decompose the initial error as $e_k = s_k + t_k$ where t_k refers to oscillatory modes of error, i.e., the modes that have high frequency. On the other hand, s_k is the smooth modes of the error which have low frequency modes of the error in subspace $\mathcal{S}_k$. In the first step, the smoothing operator (step (1) in Algorithm (2.5)) is applied to the equation. The error after applying the smoothing operator, denoted as $\tilde{e}_k = \mathcal{R}(e)$. The oscillatory components of the error removed by the smoother operator and then $\tilde{e}_k$ has only the smooth component of the initial error e_k. The other component of multigrid is the coarse grid operator $\mathcal{C} : \mathcal{S}_k \to \mathcal{S}_k$.
Let us assume that we decompose the subspace $\mathcal{S}_k$ as $\mathcal{S}_k = \mathcal{S}_k^+ \oplus \mathcal{S}_k^-$. Defining the coarse grid problem as

$$a_k(\delta, \nu) = f(\nu) - a(u_k, \nu) \quad \forall \nu \in S_k^- \tag{2.27}$$

Substituting $\tilde{e}_k = u_k^* - u_k$ into (2.27), we will have

$$a(\tilde{e} - \delta, \nu) = 0 \quad \forall \nu \in S_k^- \tag{2.28}$$

which means that δ is the orthogonal projection of the error from $\mathcal{S}_k$ into $\mathcal{S}_k^-$ with respect to $a_k(.,.)$ inner product (*Galerkin orthogonality*). Then we can define the coarse grid correction operator by

$$\mathcal{C}(\tilde{e}) = \tilde{e} - \delta = u^* - (u + \delta) \tag{2.29}$$

Another way to look at how different components of the error should be treated in order to have an efficient multigrid method is to look at the null spaces of the coarse grid correction operator and the smoother operator. The condition necessary to have an efficient multigrid method is $s_k \in \mathcal{N}(\mathcal{C})$ and $t_k \in \mathcal{N}(\mathcal{R}_k)$. A detailed analysis of this approach can be found in [BHM00].

2.2.1.2. Convergence theory

V-Cycle multigrid is an iterative method to solve Equation (2.1) recursively. We can define the multigrid operator as follows,

$$\begin{aligned} &MG : \mathcal{V} \to \mathcal{V} \\ &u^{l+1} = MG_J(u^l, f) \end{aligned} \tag{2.30}$$

and then the generalization of the V-Cycle multigrid algorithm, Algorithm (2.5), can be defined as follows,

Algorithm 2.6
If $MG_1(\nu, f) = A_1^{-1} f$ and $k \in \mathbb{I}$ and $\nu \in \mathcal{V}_k$ and $f \in \mathcal{V}'_k$, and assuming $MG_{k-1}(.,.)$ is defined, then we define $MG_k(\nu, f)$ as

1) Smoothing step $\nu' = \nu + \mathcal{R}_k^t(f - A_k\nu)$

2) Coarse grid correction $\nu'' = \nu' + q$ where $q = MG_{k-1}(0, Q_{k-1}^{mg}(f - A_k\nu'))$

3) Smoothing step $\nu''' = \nu'' + \mathcal{R}_k^t(f - A_k\nu'')$

Here, we can also connect the operator B_k^{mg} in Algorithm (2.5) with $MG_k(.,.)$. Looking at Step 2 of both algorithms, we can write $B_k^{mg} f = MG(0, f)$.
Like the successive correction methods we introduced in Section (2.2.0.1), we can also define the error propagation operator E_J for the V-Cycle multigrid as

$$E_J(u - u^l) = u - u^{l+1} = MG_J(u, f) - MG_J(u^l, f) = MG_J(u - u^l, 0).$$

As mentioned in Section (2.2), in order to analyze an iterative method, we look at the act of the error propagation operator on error. Since multigrid has two main steps, smoothing and coarse grid correction, we can look at the influences of each step on e_k separately. The steps 1) and 3) of Algorithm (2.6) are the smoothing step. Defining $e' = u^* - u'$, we can have

$$\begin{aligned} e' &= e - \mathcal{R}_k^t A_k(e) \\ &= (I - \mathcal{R}_k^t A_k)e \\ &= \mathcal{K}_k e \end{aligned}$$

where $\mathcal{K}_k$ is the error propagation operator for the smoothing step.
Considering the equality (2.7) we can write the coarse grid correction step as

$$\begin{aligned} q &= MG_{k-1}(0, Q_{k-1}^{mg}(f - A_k u')) \\ &= MG_{k-1}(0, Q_{k-1}^{mg} A_k(u^* - u')) \\ &= MG_{k-1}(0, A_k P_{k-1}^{mg}(u^* - u')) \\ &= P_{k-1}^{mg}(u^* - u') - E_{k-1} P_{k-1}^{mg}(u^* - u') \end{aligned}$$

Then having $u'' = u' + q$ and computing $e'' = u^* - u''$ we can write

$$e'' = [(I - P_{k-1}^{mg}) - E_{k-1}P_{k-1}^{mg}]e'$$

and finally, combining two error propagation operators we can have the V-Cycle multigrid error propagation operator as

$$E_J = \mathcal{K}_k[(I - P_{k-1}^{mg}) - E_{k-1}P_{k-1}^{mg}]\mathcal{K}_k^t \tag{2.31}$$

where $\mathcal{K}_k^t$ is the adjoint of $\mathcal{K}_k$ with respect to $a(.,.)$.

Then in order to analyze the convergence of multigrid we look at different components in its error propagation operator and demand a condition from each component of multigrid, i.e., smoothing operator and coarse grid operator. The first assumption is *regularity and approximation property*, which describes how different subspaces (coarse and fine subspaces) interact with each other, and the second assumption is the *smoothing property* which essentially describes the necessary relation between the approximate subspace solver and the finite element equations.

Assumption 2.4

1) *The first assumption is* ***regularity and approximation*** *assumption of the form*

$$a((I - P_{k-1}^{mg})u, u)) \leq C_\alpha^2 \left(\frac{\|A_k u\|_k^2}{\lambda_k} \right)^\alpha a(u,u)^{1-\alpha} \quad \forall u \in M_k$$

for $0 < \alpha < 1$ *and* C_α *is independent of* k. λ_k *is the largest eigenvalue of* A.

2) *The second assumption is called the* ***smoothing assumption****. This condition we demand from the smoother operator* R_k *and it should satisfy the following inequality,*

$$\frac{\|u\|_k^2}{\lambda_k} \leq C_R(R_k u, u)_k \quad \forall u \in M_k$$

Then any proper smoother should satisfy the second assumption. Having these two assumptions, we can expect convergence from the multigrid method [BH83].

2.2.2. Full Approximation Scheme (FAS)

Let $\mathcal{V} \subset H$ be a real Hilbert space with dual space as $\mathcal{V}'$. Having a nonlinear operator $n : \mathcal{V} \to \mathcal{V}'$ and $m \in \mathcal{V}'$, we are interested in computationally approximating the unique solution of the nonlinear problem,

find $u^* \in \mathcal{V}$ such that

$$n(u^*)(v) = m(v) \quad \forall \nu \in \mathcal{V} \tag{2.32}$$

Suppose that we have a sequence of finite-dimensional subspaces $\mathcal{S}_k$ such that

$$\mathcal{S}_0 \subset \mathcal{S}_1 \subset \cdots \subset \mathcal{S}_k \subset \mathcal{V}$$

and $\overline{\bigcup_k \mathcal{S}_k} = \mathcal{V}$

The operator $n : \mathcal{V} \to \mathcal{V}'$ is the continuous operator, and we can define the local (level) operator $n_k : \mathcal{V}_k \to \mathcal{V}'_k$ in the subspace (level) $\mathcal{V}_k$. Defining a ball around u^* such as $B(u^*; r) \subset \mathcal{V}$ with $B(u^*; r) \supset \{u \in \mathcal{V} \mid \|u - u^*\| \leq \sigma_c\}$ we need to make the following assumptions about the nonlinear operator $n(u)$

Assumption 2.5

i) $n(u)$ *is Fréchet differentiable on* $B(u^*; r)$ *denoted by* $\mathfrak{D}n(u) : \mathcal{V} \to \mathcal{V}$ *and is a bounded linear operator*

ii) $\forall u \in B(u^*; r) : (\nu, w) \mapsto \langle \mathfrak{D}n(u)(v), w \rangle$ *defines a symmetric bilinear form on* $\mathcal{V} \times \mathcal{V}$

iii) $\mathfrak{D}n(u)$ *is coercive, i.e.* $\exists \gamma > 0 : \forall u \in B \ : \ \langle \mathfrak{D}n(u)(v), v \rangle \geq \gamma \|\nu\|^2 \ \forall \nu \in \mathcal{V}$

iv) $\mathfrak{D}n(u)$ *is Lipschitz continuous, i.e.* $\exists c_n > 0 : \|\mathfrak{D}n(u)(\nu) - \mathfrak{D}n(u)(w)\| \leq c_n \|\nu - w\| \ \forall \nu, w \in B(u^*; r)$

v) $\mathfrak{D}n(u)$ *is symmetric, i.e.* $\langle \mathfrak{D}n(u)(\nu), w \rangle = \langle \mathfrak{D}n(u)(w), v \rangle \ \forall \nu, w \in B(u^*; r)$

vi) $\mathfrak{D}n(u)$ *is monotone, i.e.* $\exists c_m > 0 : \|\langle \mathfrak{D}n(u)(\nu), \nu - w \rangle - \langle \mathfrak{D}n(u)(w), \nu - w \rangle\| \leq c_m \|\nu - w\|^2 \ \forall \nu, w \in B(u^*; r)$

And since $\mathfrak{D}n(u)$ is a symmetric positive definite operator, we can define a bilinear operator $b_u : \mathcal{V} \times \mathcal{V} \to \mathcal{V}'$ such that

$$b_u(\nu, w) = \langle \mathfrak{D}n(u)(v), w \rangle \ \forall \nu, w \in B(u^*; r) \tag{2.33}$$

Using the bilinear form $b(.,.)$, the energy norm $\|.\|_u$ on $\mathcal{V}$ can be defined as $\|\nu\|_u = (b_u(\nu, \nu))^{\frac{1}{2}}$.
Then the following lemma holds for the bilinear form $b(.,.)$ and the corresponding energy norm

Lemma 2.1

1) $b(.,.)$ *is bounded, i.e.* $\exists \Gamma > 0 : |b_\nu(u, z) - b_w(u, z)| \leq \Gamma \|\nu - w\|_{y_1} \|u\|_{y_2} \|z\|_{y_3} \ \forall \nu, w, y_i \in B(u^*; r), \forall u, z \in \mathcal{V}$

2) $\|u\|_\nu \leq (1 + \gamma^{\frac{1}{2}} \Gamma \|\nu - w\|) \|u\|_w^2 \quad \forall \nu, w \in B(u^*; r), \ \forall u \in \mathcal{V}.$

The proof of the lemma can be found in [Reu88]. Since we linearize the nonlinear operator $n(u)$ by using the Fréchet derivative, it is necessary to look at the bound of the error of the linearization. Usually we use the Taylor expansion of the operator $n(u)$ and compare it with the Fréchet derivative of the operator $n(u)$. We can define an operator $d_\nu : \mathcal{V} \to \mathcal{V}'$ such as $d_\nu(u) = n(u) - n(\nu) - b_\nu(u - v)\ \forall \nu, u \in B(u^*; r)$

Lemma 2.2
Regarding the definition above, the following holds:

$$\|d_\nu\|_\nu \leq \Gamma \|\nu - u\|_\nu^2$$

The proof of the lemma can be found in [Reu88]. This is a critical point of using FAS and we will return to this point later in Section (5.1).

As was mentioned, we need to study two main operators in the multigrid method; the first operator which is needed to be studied is the smoothing operator. This is the main focus of the following sections. The convergence of the Full Approximation Scheme can be studied from different approaches. The first approach introduced in [BD85] for linear multigrid, and was then used by [Reu88] in FAS. And the second approach is to use the subspace correction method to analyze the linear multigrid [BP87; Yse93; Xu92] and later to study FAS [Xie96].

In the first approach, the idea used is to study the characteristics of the relaxation operator by a number called "*generalized condition number*" [BD85; BH83] and denoted by κ. Originally, it was introduced in [BD85]. The method is initially used to study the convergence of the preconditioned conjugate gradient method. The condition number is generally computed for the entire subspace $\mathcal{V}_k$. However, in the analysis of the multigrid method, the generalized condition number is computed with respect to a certain subspace, i.e., the orthogonal complement of the coarse space $\mathcal{V}_{k-1}$. First we need to define a bilinear form $l_k : \mathcal{V}_k \times \mathcal{V}_k \to \mathcal{R}$ and assume that it is an approximation of inverse of bilinear $b(.,.)$ and also be continuous symmetric positive definite. Then the generalized condition number can be stated as

Lemma 2.3
For $\nu \in \mathcal{V}_k$, define $\mathcal{V}_{k-1}^\nu = \{u \in \mathcal{V}_k | b_\nu(y, u) = 0 \forall y \in \mathcal{V}_{k-1}\}$. With $l_{k,r}$ as defined above, the following holds

$$\forall r > 0, \exists \kappa_{B(u*;r)} : \forall k, \forall \nu \in B(u^*; r), \forall u \in \mathcal{V}_{k-1}^\nu : l_{k,r} \leq \kappa_{B(u*;r)} b_\nu(u, u) \tag{2.34}$$

In the second approach, introduced by Bramble [BP87], as described in Section (2.2.1.2) in detail, we look at the multigrid as a successive subspace correction method (2.2.0.1). In order to analyze the nonlinear multigrid method, we need two main assumptions, similar to the linear multigrid (2.4), but modified for the nonlinear multigrid. These assumptions in the nonlinear multigrid are as follows,

Assumption 2.6

i) the first assumption is ***regularity and approximation*** *assumption of the form*

$$b_u((I - P_{k-1})u, u)) \leq C_\alpha^2 \left(\frac{\|\mathfrak{D}n(u^*)u\|_k^2}{\lambda_k} \right)^\alpha b_u(u,u)^{1-\alpha} \quad \forall u \in M_k$$

for $0 < \alpha < 1$ *and* C_α *is independent of* k. λ_k *is the largest eigenvalue of* $\mathfrak{D}n(u)$.

ii) The second assumption is called ***smoothing assumption*** *which is on smoother operator* R_k *and it should satisfy the condition*

$$\frac{\|\mathfrak{D}n(u^*)u\|_k^2}{\lambda_k} \leq C_R b_u((I - \mathfrak{D}R_k(u*))u, u)_k \quad \forall u \in M_k \tag{2.35}$$

Then a proper smoother for FAS should be able to satisfy the smoothing assumption. One candidate as a smoother for FAS is the nonlinear restricted additive Schwarz method (NRASM), which is a nonlinear parallel subspace correction method. Then, as a proper smoother, it should satisfy the smoothing condition. The remainder of this chapter is devoted to this task. In the next subsection, the linear restricted additive Schwarz method (RASM) is introduced and then the convergence of the method will be studied. Following the linear RAS method, the nonlinear version of the RASM will be introduced, and then the convergence of the NRAS method will be analyzed, and at the end of this chapter, I will investigate whether the NRAS method satisfies the smoothing property.

2.2.3. Domain decomposition method

2.2.3.1. Schwarz method

Let us assume $\mathcal{V}$ is a function space (Hilbert space as we introduce earlier) with a proper norm and inner product. Then we want to solve the following problem in the space $\mathcal{V}$ is to find $u^* \in \mathcal{V}$ such that

$$a(u^*, \nu) = \langle f, \nu \rangle \quad \forall \nu \in \mathcal{V} \tag{2.36}$$

here $a(.,.)$ as we introduced before is a bilinear form and $\langle .,. \rangle$ is the inner product on $\mathcal{V}$. We assume that the problem (2.36) is well-posed [RR06]. As we mentioned earlier, by using the Ritz representation theory, we can find the linear operator $\mathcal{A} : \mathcal{V} \to \mathcal{V}'$ and then by defining a basis for the function space $\mathcal{V}$ we have a linear system of

$$Au = f$$

Assuming that there exists a set of interpolation operator $I_i : \mathcal{V}_i \to \mathcal{V}$. Defining the bilinear form $l_i : \mathcal{V}_i \times \mathcal{V}_i \to \mathbb{R}$ which is the approximation of the bilinear form $a(.,.)$ in the subspace $\mathcal{V}_i$, or $l(.,.)$ can also be $a_i(.,.)$, then

$$l(u_i, \nu_i) = a(I_i^{-1}u_i, I_i^{-1}\nu_i) \quad \text{for } u_i, \nu_i \in \mathcal{V}_i$$

We also assume that for $\nu \in \mathcal{V}_i$ $l_i(\nu,\nu) = 0$ implies that $\nu = 0$. Replacing P_i defined in Section (2.2) with a projection like operator $\tilde{T}_i : \mathcal{V} \to \mathcal{V}$ by

$$l_i(\tilde{T}_i u, \nu) = a(u, I_i \nu) \quad \forall \nu \in \mathcal{V}_i \tag{2.37}$$

since $\mathcal{V}_i$ is a finite-dimensional Hilbert space, the operator is well-posed. Now we need to define a new operator $T_i : \mathcal{V} \to \mathcal{V}$ such that

$$T_i = I_i \tilde{T}_i$$

and deriving the matrix form of the operator T_i as

$$T_i = R_i^T A_i^{-1} R_i A \tag{2.38}$$

where $R_i : \mathcal{V}_i \to \mathcal{V}_i$ is the discrete form of operator I_i. $A_i = R_i A R_i^T$ is the *exact local stiffness matrix* in $\mathcal{V}_i$ when we consider $l_i(u_i, \nu_i) = a(R_i^T u_i, R_i^T \nu_i)$ for $u_i, \nu_i \in \mathcal{V}_i$. We can easily show that the operator T_i is symmetric positive semi-definite with respect to the bilinear form $a(.,.)$.
The operator T_i can be used to construct different operators for Schwarz method by using different polynomials in T_i, i.e. $\mathcal{P}(T_i)$. Then, for instance the *additive Schwarz method* can be constructed by using the following polynomial

$$\mathcal{P}(T_1, T_2, \ldots, T_J) = T_1 + T_2 + \cdots + T_J$$

which, as we can see, is the parallel subspace correction method as introduced in Section §(2.2.0.2). The other variety of the Schwarz method which is widely used is the *multiplicative Schwarz method* in which the polynomial of T_i has the following form

$$\mathcal{P}(T_1, T_2, \ldots, T_J) = I - (I - T_1)(I - T_2) \ldots (I - T_J)$$

2.2.3.2. Convergence theory of the additive Schwarz method

In this section, the abstract convergence of the additive Schwarz method is studied. In order to study the convergence of the additive Schwarz method, we need three assumptions. The first two assumptions are similar to the assumptions (2.2) and (2.3). But there is a remark that needs to be emphasized here.

Remark 2.2

In assumption (2.2), the desirable value for K_0 is close to one, which means that the sum of the energies of u_i is close to the energy of u. In the following lemma, we can also see that K_0 is the lower bound of the spectrum of $T = \sum_{i \in \mathbb{I}} T_i$. This condition ensures weak continuity of the local bilinear form $l_i(.,.)$.

Lemma 2.4

Define $T = \sum_{i \in \mathbb{I}} T_i$ and $u = \sum_{i \in \mathbb{I}} I_i u_i$. Then

$$a(T^{-1}u, u) = \min_{u_i \in \mathcal{V}_i} \sum_{i \in \mathbb{I}} l_i(u_i, u_i) \tag{2.39}$$

The complete proof of this lemma can be found in [SBG98]. The assumption (2.3) for the Schwarz method is also changed as follows,

Assumption 2.7
Define $\mathcal{E}_{ij} \in [0,1]$ *to be the minimal values that satisfies*

$$|a(I_i u_i, I_j u_j)| \leq \mathcal{E}_{ij} a(I_i u_i, I_i u_i)^{\frac{1}{2}} a(I_j u_j, I_j u_j)^{\frac{1}{2}} \tag{2.40}$$

The spectral radius of $\rho(\mathcal{E})$ is a measure of the orthogonality of the subspaces. When $\mathcal{E}_{ij} = 0$, this means that $\mathcal{V}_i$ and $\mathcal{V}_j$ are orthogonal. In addition, $\rho(\mathcal{E})$ gives us an upper bound of the eigenvalues of T.
The third assumption is the *local stability* condition, which is the other side of the approximation along with the Assumption (2.2) of bilinear form $l_i(.,.)$. The assumption also ensures that the local bilinear forms are coercive.

Assumption 2.8
There exists $\omega \in [1,2)$ *such that*

$$a(I_i u, I_i u) \leq \omega l_i(u,u) \quad \forall u \in \mathcal{V}_i, i \in \mathbb{I}$$

In the case where $l_i(.,.) = a(.,.)$, we use the exact subdomain solver, then $\omega = 1$. Having all the necessary assumptions, we can find the upper bound of the eigenvalues of the additive Schwarz method, as we can see in the following lemma which is borrowed from [SBG98].

Lemma 2.5
The largest eigenvalue of the additive Schwarz method is $\omega\rho(\mathcal{E})$.

Proof:

$$\begin{aligned}
a(\sum_{i\in\mathbb{I}} T_i\nu, \sum_{i\in\mathbb{I}} T_i\nu) &= \sum_{i,j\in\mathbb{I}} a(T_i\nu, T_j\nu) \\
&\leq \sum_{i,j\in\mathbb{I}} \mathcal{E}_{ij} a(T_i\nu, T_i\nu)^{\frac{1}{2}} a(T_j\nu, T_j\nu)^{\frac{1}{2}} \\
&\leq \rho(\mathcal{E}) \sum_{i\in\mathbb{I}} a(T_i\nu, T_i\nu) \\
&\leq \omega\rho(\mathcal{E}) \sum_{i\in\mathbb{I}} a(\tilde{T}_i\nu, \tilde{T}_i\nu) \\
&= \omega\rho(\mathcal{E}) \sum_{i\in\mathbb{I}} a(\nu, T_i\nu) \\
&= \omega\rho(\mathcal{E}) a(\nu, \sum_{i\in\mathbb{I}} T_i\nu) \\
&\leq \omega\rho(\mathcal{E}) a(\nu,\nu)^{\frac{1}{2}} a(\sum_{i\in\mathbb{I}} T_i\nu, \sum_{j\in\mathbb{I}} T_j\nu)^{\frac{1}{2}}
\end{aligned}$$

therefore,

$$a([\sum_{i\in\mathbb{I}} T_i]^{\frac{1}{2}}\nu,\nu) = a(\sum_{i\in\mathbb{I}} T_i\nu, \sum_{j\in\mathbb{I}} T_j\nu) \leq \rho(\mathcal{E})^2\omega^2 a(\nu,\nu)$$

then we can conclude that

$$a(\sum_{i\in\mathbb{I}} T_i\nu,\nu) \leq \rho(\mathcal{E})\omega a(\nu,\nu) \tag{2.41}$$

which means that the largest eigenvalue of the additive Schwarz method is $\omega\rho(\mathcal{E})$. □

2.2.4. Nonlinear additive Schwarz method

As the nonlinear multigrid method needs a nonlinear smoother, then here we look at the nonlinear version of the restricted additive Schwarz method and investigate whether it is a proper smoother for the FAS, i.e., if it can satisfy the Assumption (2.34). In this section, we also need additional assumptions about the nonlinear operator $n(u)$. We need to assume that $n(:)$ is a quasilinear problem, as defined in (2.1). Under certain assumptions [LU68] we can prove that the quasilinear equation (2.32) has a unique solution $u^* \in B(u^*;r) \subset \mathcal{V}$. In this part, in particular with regard to assumptions and theorems, I follow mainly [DH97].

We also assume that $n(u)$ is Fréchet differentiable in $B(u^*;r)$, denoted by $\mathfrak{D}n(u^*)$ and assume that it is also non-singular.
In addition to aforementioned assumptions, we also need three more assumptions, later used to prove the uniqueness of the solution and also study the convergence of the nonlinear restricted additive Schwarz method. Defining the difference quotient operator $\mathcal{D}n \in \mathfrak{L}(\mathcal{V},\mathcal{V})$ as $\mathcal{D}n : \mathcal{V}\times\mathcal{V} \to \mathcal{V}'$, we need

Assumption 2.9

- $\forall u', u'' \in \mathcal{V}\ \exists C_d > 0 \quad such\ that \quad \|\mathcal{D}n(u',u'')\| \leq C_d$
- $\mathfrak{D}n(u') - \mathfrak{D}n(u'') = \mathcal{D}n(u',u'')(u'-u'')$
- $\|\mathcal{D}n(u',u'') - \mathfrak{D}n(u^*)\| \to 0 \ as\ u'', u' \to u^*$

These assumptions are very weak [DH97] and the possible choice for $\mathcal{D}n(u',u'')$ can be

$$\mathcal{D}n(u',u'') = \int_0^1 \mathfrak{D}n(u)(u' + \theta(u''-u'))d\theta$$

in which by assuming the continuity of $\mathfrak{D}n(u)$ we can easily see that $\mathcal{D}n(u',u'')$ is also continuous. Considering the operator $p_k : \mathcal{V} \to \mathcal{V}_k$ defined in section (2.2) and its

adjoint is denoted as $r_k : \mathcal{V}_k \to \mathcal{V}$. Then the nonlinear subspace iteration will be as find $\delta_\kappa \in \mathcal{V}_\kappa$ such that

$$r_\kappa n(u - p_\kappa \delta_\kappa) = m \quad \forall \kappa \in \mathbb{I} \tag{2.42}$$

where u is the current approximation such that $u \in B(u^*; r)$. In general, the solvability of (2.32) does not imply the solvability of (2.42) in subspace $\mathcal{V}_\kappa$, therefore, we need another assumption which is

$$r_\kappa \mathfrak{D} n(u)(u^*) p_\kappa : \mathcal{V}_\kappa \to \mathcal{V}_\kappa \tag{2.43}$$

should be invertible. By having the assumptions, we can guarantee the solvability of the nonlinear problem (2.42) by using the following theorem

Theorem 2.10

Having assumptions 2.9 and 2.43 and $B(u^*; r) \subset \mathcal{V}$ for small enough r. Then the subspace problem (2.42) has a unique solution $\forall u \in B(u^*; r)$ and all the indices $\kappa \in \mathbb{I}$ (Theorem 1.5 [HR89]).

The complete proof of the theorem can be found in [DH97]. The theorem proves that, having Newton-like iteration $\Phi : \mathcal{V}_\kappa \times B(u^*; r) \to \mathcal{V}_\kappa$, then $\delta(u)$ is the fixed point of $\Phi(\delta, u)$ by using Banach's fixed point [Cia13]. Having the solution of each subspace δ_κ by solving the local problem in (2.42), we can then project δ_κ to the space $\mathcal{V}$ by using p_κ as

$$\delta(u) := \sum_{\kappa \in \mathbb{I}} p_\kappa \delta_\kappa(u) \tag{2.44}$$

and the nonlinear parallel subspace correction iteration method is defined as

$$\Phi(u) := u - \alpha \delta(u) \tag{2.45}$$

where α can be seen as a damping factor (or step length). This factor in the context of a nonlinear iteration method can be determined by the *backtracking* algorithm (α might be chosen to satisfy one of the Wolfe, strong Wolfe or Goldstein conditions [NW06]). We will talk about it in section (3.3.1). If $\Phi(\delta, u)$ is a nonlinear subspace iteration method, we are also interested in studying its convergence rate. In the linear iteration method, we generally look at the norm of the iterative operator (or the error propagation operator E) and find its contraction number. On the other hand, in the nonlinear iteration method we look at the contraction number with respect to the norm of difference between two successive approximations, i.e. $\|\Phi(\delta^{n+1}, u) - \Phi(\delta^n, u)\|$. In the following theorem, we show that the nonlinear additive Schwarz method has the contraction number $\gamma < 1$

Theorem 2.11

By having the assumption (2.9), (2.43) and let γ be the contraction number of the mapping $\Phi(u) : \mathcal{V} \to \mathcal{V}$. Then, $\exists u \in B(u^*; r) \subset \mathcal{V}_\kappa$ such that

$$\|\Phi(u') - \Phi(u'')\| \leq \gamma < 1 \quad \forall u'', u' \in B(u^*; r) \tag{2.46}$$

(Theorem 1.2 [DH97])

The complete proof of the theorem can be found in [DH97]. Theorem (2.11) shows that defining $A_\kappa := r_\kappa \mathfrak{D}n(u)(u^*)p_\kappa$ and assuming $\sum_{\kappa\in\mathbb{I}} p_\kappa \mathcal{D}\delta_\kappa(u'', u') \to \sum_{\kappa\in\mathbb{I}} r_\kappa A^{-1} p_\kappa \mathfrak{D}n(u)(u^*)$, then the nonlinear subspace iteration Φ converges in $B(u^*; r)$ with the convergence rate γ and the convergence rate is the same as the linear subspace iteration applied to the linearized equations.

So far, the convergence of the nonlinear additive Schwarz method has been proven. Then we need to analyze the nonlinear restricted additive Schwarz with respect to the condition that is demanded from the NRAS method to be a proper smoother for nonlinear multigrid, the second assumption of 2.6. This is the primary result of the following corollary.

Corollary 2.11.1

Nonlinear restricted additive Schwarz method is a proper smoother for FAS, that is, it satisfies the smoothing condition (2.34).

Proof:

Let us consider the iterative method as defined in Section (2.3) in [DH97],

$$u_h^{k+1} = u_h^k + \omega\delta(u_h^k),$$

where

$$\delta(u_h^k) = \sum_{\kappa\in\mathbb{I}} p_\kappa \delta_\kappa$$

and δ_κ is the solution of

$$r_\kappa n(u_h^k - p_\kappa\delta_\kappa) = 0$$

where the subscript h refers to the discretized form of the nonlinear equation as explained in Chapter (4). Considering $A_\kappa := r_\kappa \mathfrak{D}n(u)(u^*)p_\kappa$, and having Assumptions (2.9), we need to analyze the additive Schwarz method for a linear system

$$Au_h = b$$

for $A := \mathfrak{D}n(u)(u^*)$. Looking at the linear problem, and we can use Theorem (2.5) to prove the satisfaction of the smoothing condition for the additive Schwarz method [Xu92]. As a consequence, by considering the Newton's like method defined in (2.45), we conclude that the nonlinear additive Schwarz method satisfies the smoothing condition in (2.35). □

3. Numerical methods

In the previous chapter, the algorithm developed to solve the nonlinear problem was investigated theoretically. As the algorithm should be implemented on GPUs to solve the problem numerically, the numerical aspects of the algorithm should also be studied. In this chapter, the main algorithm used to solve nonlinear partial differential equations will be explained in more detail and from the implementation point of view. As the primary goal of this project is to solve a nonlinear parabolic PDE on GPU, the properties of the algorithms with respect to the architecture of GPU will also be studied, in particular, how the properties of the algorithms take advantage of the GPU architecture, and also what is the drawback of these algorithms and with respect to the architecture of the GPU.

In the first section, the architecture of the GPU will be explained in more detail and how different components of the computing unit and memories work together. Then, in the three following main sections of this chapter, the Full Approximation Scheme, as a main nonlinear solver, will be introduced in detail and the main components and necessary parameters are numerically explored. This means that I will investigate the nonlinear restricted additive Schwarz method as a nonlinear smoother of FAS, with globally convergent inexact Newton's method as a local solver in the NRAS method and Krylov subspace method (GMRES) as the linear iterative solver to solve Newton's equation. And finally, at the end of this chapter, I look at the GPU architecture and the algorithm to see how the algorithm works with the specific components of the architecture of a GPU.

3.1. GPU architecture

In the last few years, the Graphic Processing Unit (GPU) became the main accelerator in the modern HPC system. They are usually massively parallel processors that contain hundreds of computing cores. Even if CPU cores are more powerful and flexible, GPUs can run thousands of threads in parallel and reach at least in theory Tera FLOPs.

The CUDA programming model utilizes this architecture and is based on a hierarchy of abstraction layers (see Figure (3.1)). The basic computing unit in a GPU is a thread. The next level of computing unit is a thread block, which is a batch of threads assigned to the same Streaming Multiprocessor (SM). Therefore, the blocks in a SM can utilize all the resources available in the SM such as registers and shared memory.

The threads in a block can communicate through shared memory. In a block, threads are divided into batches, usually 32 threads, which is also called *warp*. The warp is the scheduled unit, so the threads in each block are executed in a SM warp-by-warp in a SIMD (single instruction multiple data). All the threads in one warp should execute the same instruction at a time, which means if there is an if condition which forks the threads in a warp into different instruction, these instructions are executed sequentially and disabling the threads which are not on the same path. This is usually called *branching*. The threads in a warp will reconverge after all divergent paths are completed.

Table 3.1.: Major hardware features of NVIDIA A100.

GPU element	Feature	A100
SMs	SMs	108
	Total num. cores	6912
	Clock freq.	1410 MHz
Maximum number of threads	Per SM	2048
	Per block per SM	32
	Per warp	32
SRAM memory avaiable per SM	32-bit registers	65536
	Shared memory	164 KB
	L1 cache	192 KB/SM
	Memory Bandwidth	1555 GB/sec
	(shared+L1)	40 MB

In CUDA, in addition to the units mentioned above, another grouping concept is also introduced. Cooperative group is an API concept that defining grouping some threads in a block of threads together (See the Figure (3.3)). It provides partitioning of a thread block into smaller batches, usually called *tile*, and synchronization at this group level. In addition to that, it also provides the potency of clustering blocks of threads and synchronization of the cluster.

In GPUs, there are also different levels of memory hierarchy. Registers are the fastest memory components on a GPU, comprising the register file that supplies data directly into the CUDA cores. A kernel function uses registers to store the variables private to the thread and accessed frequently. Multiple levels of caches exist in modern processors. L1 cache is the fastest and closest to processing cores, handling immediate instructions. L2 cache provides a larger and slightly slower memory pool, often shared among SMs. Shared memory ensures efficient communication between processing units. Table (3.1) shows some hardware features of GPU NVIDIA A100.

One of the features that is quite often needed in numerical modeling is global synchronization. In general, in GPUs, the global synchronization can be achieved only

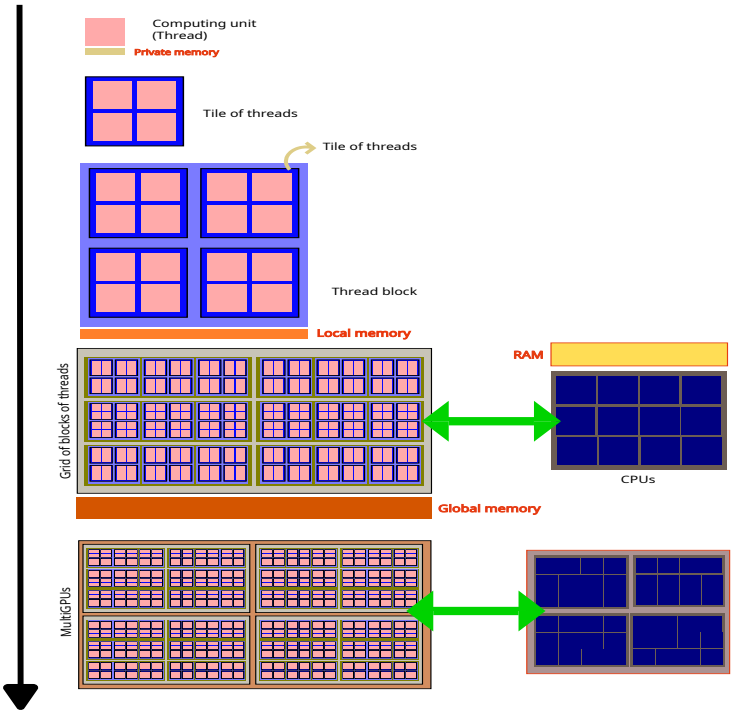


Figure 3.1.: GPU architecture

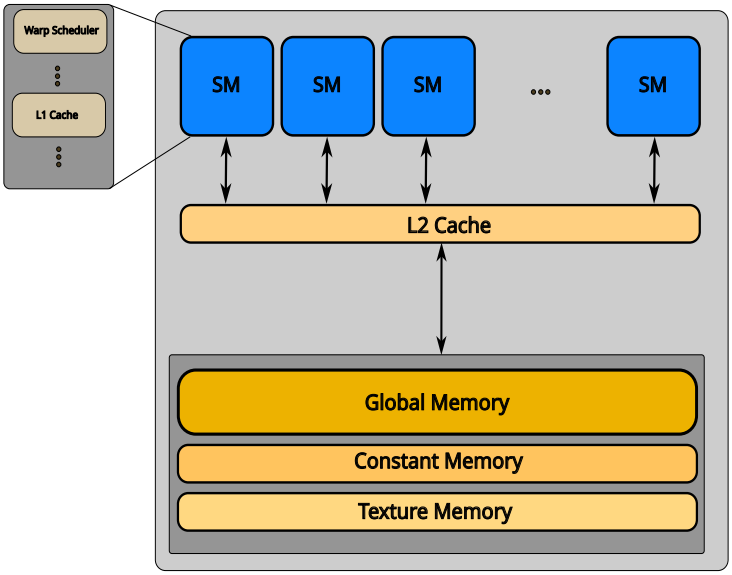


Figure 3.2.: GPU memory heirarchy

by leaving a kernel. Then, in numerical modeling, kernels should be queued to get global convergence. Another recent way to get global synchronization is using the

global synchronization possibility in the cooperative group, in which there is no need to leave the running kernel. However, in the cooperative launch, it should be ensured that all the blocks are running concurrently. Then, to avoid a deadlock, the GPU imposes some constraints on the number of block of threads can be assigned in a SM. By the availability of this potency in GPUs, we can think of new algorithms which

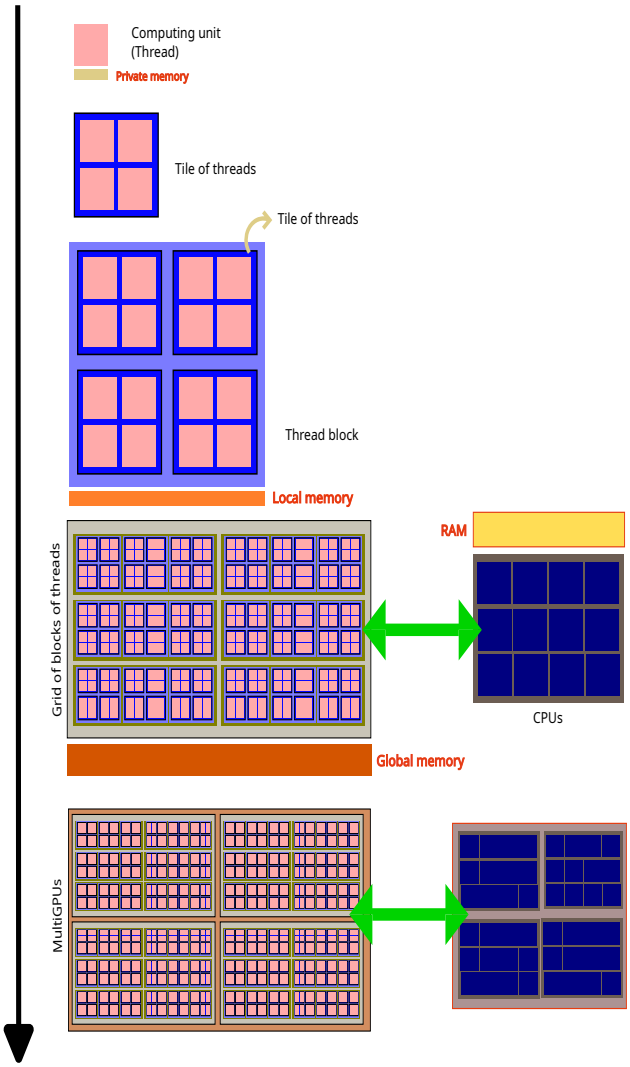


Figure 3.3.: multiGPU computing unit and memory hierarchy and the features provided by CUDA API

can benefit the architecture of the computing units and memory hierarchy, as well as the features provided by the API. Another important criterion of the computation in GPUs is to keep the data locally closer to the compute units as long as possible and avoid global to shared memory communications. In this regard, the use of shared memory is a critical point, although the use of too much shared memory can lead to

a lower occupancy. *Occupancy* is a factor that shows the ratio of active warp in a SM to the maximum possible warp in a SM. Therefore, there is a trade-off in using more shared memory and a lower occupancy. Another important factor to consider is to avoid *warp-branching*. In GPU, all the threads in a warp should execute the same command. This means that if there is a branching command (if or case condition), branches are executed sequentially, which reduces the efficiency of the algorithm in GPU.

Unfortunately, it is not always possible to consider optimality in all these aspects. Most of the time, the nature of the problem imposes some strong constraints. In the current work, the nonlinearity of the problem is an important factor which force us not be able to get a peak performance on the GPUs.

In the following, the algorithm used to solve nonlinear problems on GPUs is explained and at the end of this chapter, we can see how the algorithm and GPU architecture and memory hierarchy work together.

3.2. Full Approximation Scheme

In tackling the nonlinear problem in the Equation (2.32) we usually have two main approaches, applying *global linearization method* like using Newton's method for the nonlinear problem. In this approach, first, the nonlinear problem is linearized, and then the linear problem is solved either directly or by linear iterative methods. If the linearized problem is solved exactly (exact Newton's method) or iteratively by using some linear iterative method like the Krylov subspace method (inexact Newton's method) or applying the linear multigrid. This approach, however, has two main caveats, even though it can benefit from the rich resources on numerical linear algebra. The first issue is that the linearized problem can generate a dense Jacobian, which is expensive to solve. The second caveat is that there are strong assumptions required for the convergence of Newton's method to be considered, in particular the assumption to have a globally convergent Newton's method in particular in non-convex problems. This assumption will be explained in more detail in the Section (3.3.1). Another important issue is in the number of inner cycles in Newton's method when the inexact Newton's method is used to have an efficient approach. Finding a good balance on the inner (linear method) and outer (Newton's method) iteration numbers is challenging. The small number of iterations of the inner iterative method may result in a larger number of Newton iterations, which means that computing a more global Jacobian, on the other hand, oversolving linearized problem might not give us a benefit of faster convergence [BHM00; TOS01].

The second approach is to apply directly the multigrid method to the nonlinear problem [Bra77]. This method is, as we know from the previous chapter, Full Ap-

proximation Scheme. Consider the nonlinear initial boundary-value problem

$$\begin{aligned} n(u,t) &= m \text{ in } \Omega \\ u &= g \quad \text{on } \partial\Omega \end{aligned} \tag{3.1}$$

where $n(u,t) : \mathcal{V} \times [0,T] \to \mathcal{V}'$ is a nonlinear time-dependent operator (without loss of generality it can be only a function of u) and $\Omega \subset \mathbb{R}^d$ is a polyhedral domain. As we define the space $\mathcal{V} \subset H^1$ which subset of the standard Sobolev space. The weak formulation of the Equation (3.1) can be defined as

$$\langle n(u,t), \nu \rangle = (m, \nu) \quad \forall \nu \in H^1(\Omega) \tag{3.2}$$

Suppose $\mathcal{T}_i$ are non-overlapping simplies of size, $h \in (0,1]$ then $\Omega \in \mathbb{R}^3$ can be triangulated as $\Omega = \cup_i \tau_i$. Then the subspace $\mathcal{V}$ can be defined as

$$\mathcal{V} := \{\nu \in H^1 \mid \nu|_\tau \in \mathcal{P}_l(\tau_i) \ \forall \tau_i\}$$

where $\mathcal{P}_l$ is the space of a linear polynomial of order l. And considering nodal basis $\{\phi_i\}_{i=1}^{\hat{n}}$ where $\hat{n} = dim(\mathcal{V}_h)$ for subspace $\mathcal{V}_h \subset \mathcal{V}$ which should satisfy

$$\phi_i(x_l) = \delta_{il} \ \forall i, l = 1, \ldots, \hat{n}$$

where $\{x_i : i = 1, \ldots, \hat{n}\}$ is the set of all interior nodal points of $\mathcal{V}_h$. Using the nodal basis for the subspace $\mathcal{V}$, Equation (3.2) changes to solving an algebraic system of

$$\mathbf{N}[\mathbf{u}] = \mathbf{m} \tag{3.3}$$

where $\mathbf{N}[.] \in \mathbb{R}^n$ represents a nonlinear discrete operator on Ω, $\mathbf{u} \in \mathbb{R}^n$ and $\mathbf{m} \in \mathbb{R}^n$ is the discrete form of the right-hand side of the Equation (3.2). Then, for a given subspace $\mathcal{V}_k$, the discrete problem changed to

$$\mathbf{N}_k[\mathbf{u}_k] = \mathbf{m}_k \tag{3.4}$$

Thus as usual in the multigrid method, the smoothing operator S_k is first applied to the Equation (3.4) to remove the high-frequency modes of the error. In the case of the nonlinear multigrid, the proper smoother should be a nonlinear smoother like Gauss-Seidel-Newton or Jacobi-Newton algorithms [TOS01] in which *local linearization* is used (in Newton-Jacobi or Newton-Gauss-Seidel, the global linearization step is applied, and then the stationary iterative methods are used). In the current study, as it was explained theoretically in more detail in the Chapter (2) and later from the algorithmic perspective in the Section (3.3), the nonlinear restricted additive Schwarz method, is used as the smoother. After applying a few numbers of the smoothing iteration and having the approximation of the exact solution u_k^*, the next step is applying the *coarse grid correction*. In this step, the equation of the defect in subspace $\mathcal{V}_k$, denoted by $\mathbf{d}_k \in \mathbb{R}^n$, is formed as follows,

$$\mathbf{d}_k = \mathbf{N}[\mathbf{u}_k + \mathbf{e}_k] - \mathbf{N}[\mathbf{u}_k] \tag{3.5}$$

$$= \mathbf{m} - \mathbf{N}[\mathbf{u}_k]$$

where $\mathbf{e} \in \mathbb{R}^n$ is the correction. Then the solution in the coarse space can be written as $\mathbf{u}^*_{k-1} = I^{k-1}_k \mathbf{u}_k + \mathbf{e}_{k-1}$. The restriction operator I^{k-1}_k can be just an injection operator or be a more sophisticated operator [BHM00]. Then the formulation of the nonlinear Equation (3.3) in the coarse space $\mathcal{V}_k$ is obtained by solving the Equation (3.5) in the coarse space as

$$\mathbf{N}_{k-1}[\mathbf{u}_{k-1} + \mathbf{e}_{k-1}] - \mathbf{N}_{k-1}[\mathbf{u}_{k-1}] = \mathbf{d}_{k-1} \tag{3.6}$$

where $\mathbf{d}_{k-1} = I^{k-1}_k \mathbf{d}_k$ and $\mathbf{u}_{k-1} = I^{k-1}_k \mathbf{u}_k$. This means that in FAS in contrast to the linear multigrid two components should be transferred into the coarse space, the current approximation of the exact solution $\mathbf{u}^*$ and the defect $\mathbf{d}$. After solving or smoothing the problem in the coarse grid, the correction $\mathbf{e}_{k-1} = \mathbf{u}_{k-1} - I^{k-1}_k \mathbf{u}_k$ is computed to interpolate to the fine grid to correct the current approximation $\mathbf{u}_k$. We cannot directly interpolate $\mathbf{u}_{k-1}$ into the fine grid, and that is the coarse grid approximation of the smooth-out function $\mathbf{u}^*_k - \mathbf{u}_k$, since it introduces the error, probably oscillatory, of the full solution instead of the error of the correction. Thus, the correction step is

$$\mathbf{u}_k \leftarrow \mathbf{u}_k + I^k_{k-1}(\mathbf{u}_{k-1} - I^{k-1}_k \mathbf{u}_k) \tag{3.7}$$

After correction of the current approximation on the fine space, a few smoothing iterations are applied to smooth-out any high frequency error introduced to the interpolated correction through the interpolation process from the coarse to the fine space. The complete V-Cycle FAS is summarized in the following algorithm.

Algorithm 3.1 *FAS V-cycle scheme*

i) Pre-smoothing: apply ν_1 iterations of smoothing operation as

$$\mathbf{u}^l_k = S_k(\mathbf{u}^{l-1}_k, \mathbf{m}_k)$$

ii) Coarse-grid correction: Set

$$\mathbf{u}^{\nu_1}_k \leftarrow \mathbf{u}^{\nu_1}_k + I^k_{k-1}(\mathbf{u}_{k-1} - I^{k-1}_k \mathbf{u}^{\nu_1}_k)$$

where $\mathbf{u}^{\nu_1}_k$ is computed as

$$\mathbf{u}^{\nu_1}_k = \mathbf{B}_{k-1}(\mathbf{u}_{k-1})\left(I^{k-1}_k \mathbf{d}_k + \mathbf{N}_{k-1}[\mathbf{u}_{k-1}]\right)$$

iii) Post-smoothing: apply ν_2 iterations of smoothing operation as

$$\mathbf{u}^l_k = S_k(\mathbf{u}^{l-1}_k, \mathbf{m}_k)$$

where $\mathbf{B}_k(u)$, as defined in the Section (2.2), is the approximation of $\mathbf{N}^{-1}_k$. In addition to V-Cycle FAS, we can also define, like the linear multigrid, different cycles of FAS such as the W-Cycle FAS and the full multigrid (FMG) FAS [BHM00] (See the Figure (3.4)).

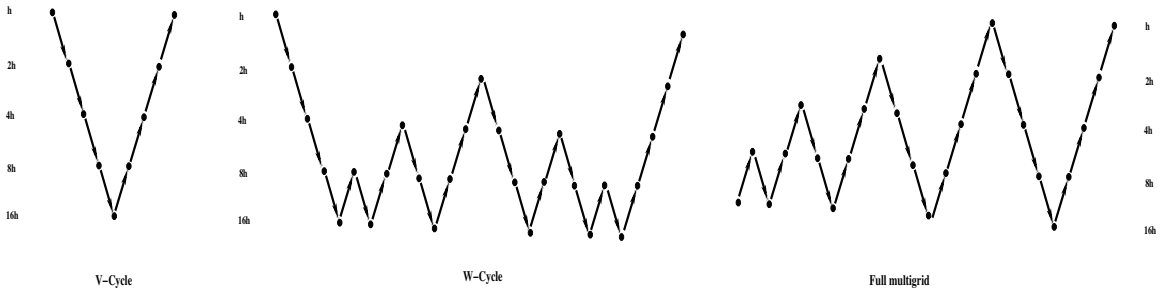


Figure 3.4.: V-Cycle, W-Cycle and full multigrid schemes

3.3. Nonlinear restricted additive Schwarz method

The nonlinear restricted additive Schwarz method from the theoretical aspect was explained in Section (2.2.4). Here, the algorithm and implementation of the method on multiple subdomains is introduced. Since from the algorithmic point of view the linear RASM and NRASM are similar, the linear additive Schwarz will be explained first, and then it will be expanded to the nonlinear problem.

The Schwarz method was first introduced by Schwarz [Sch70] in the 19th century for solving a continuous problem in a rather complex geometry, and by using the maximum principle he proved the convergence of the alternating Schwarz method. By the availability of parallel computing in 1970-80, Lions [Lio+88] introduced the parallel version of the alternating Schwarz method. In discrete form, the parallel Schwarz method began to be used as a solver for Poisson's equation [BGN70] and later for some more complex geometries [GM83]. As was explained in the Section (2.2), we can use subspace correction methods in the form of the multiplicative and the additive fashions. In the discrete form of the Schwarz method, there is the multiplicative Schwarz method [CM94; SBG98] in which the most updated information from the other subdomains is used to compute the new approximation of the solution in a given subdomain, whereas in additive form each subdomain, the local problem is solved in parallel using the old approximation of the solution [DW90].

Consider the continuous linear problem in Equation (2.2) or the discretized linear problem in Equation (2.3). We consider $\partial\Omega$ as the boundary of Ω. Then, we decompose the domain into subdomains in a non-overlapping way which is denoted by $\tilde{\Omega}_i$, then, construct the overlapping domain decomposition by adding overlap δ in such a way that it does not cut a cell, i.e., $\delta = ch$ where h is the size of a cell and $c \in \mathbb{N}$. The extended subdomains with overlap are called Ω_i, which contains $\tilde{\Omega}_i \subset \Omega_i$. This can be seen in Figure (3.5). The artificial boundary or interface $\Gamma_{i,j}$ is part of the boundary of the subdomain $\Omega_{ij} = \partial\Omega_i \cap \tilde{\Omega}_j$. In order to have an iterative method for a discrete problem, we need to partition the unknowns $\mathbf{u}$ into subsets $\mathbf{u}_i$ by using restriction matrices $\mathbf{R}_i$ which contain zeros except for the positions corresponding to $\mathbf{u}_i$ indicated by 1. With the given restriction matrices, the unknowns in each of the

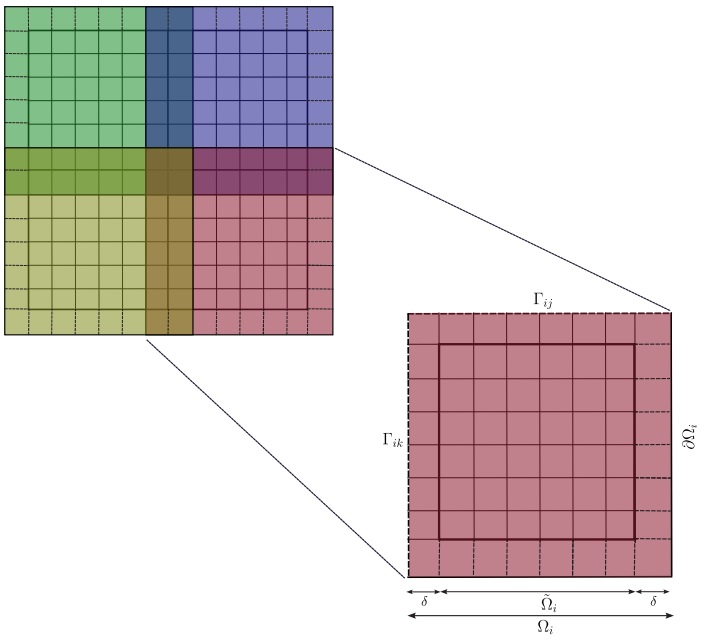


Figure 3.5.: Decomposing domain Ω, extending subdomains by overlap of size δ and adding artificial boundary Γ between subdomains

subdomains are obtained by projection of the global vector $\mathbf{u}$ into the subspace $\mathcal{V}_{hi}$ as $\mathbf{R}_i\mathbf{u}$. In each subspace, we also need the local matrices $\mathbf{A}_i$; These matrices $\mathbf{A}_i$ can be computed in two ways; one way is first discretizing the continuous PDE and then by using the restriction matrix $\mathbf{R}_i$, to project the problem into the subspace $\mathcal{V}_{hi}$ as follows:

$$\mathbf{A}_i \coloneqq \mathbf{R}_i\mathbf{A}\mathbf{R}_i^T,\ i \in \mathbb{I}$$

or first project the continuous form of the problem into the given subspace and then the local problem can be discretized in each Ω_i and compute $\mathbf{A}_i$. Then general formula of the additive Schwarz method can be as follows,

$$\mathbf{u}^{l+1} = \mathbf{u}^l + \sum_{i=\mathbb{I}} \mathbf{R}_i^T\mathbf{A}_i^{-1}\mathbf{R}_i(\mathbf{f} - \mathbf{A}\mathbf{u}^l) \tag{3.8}$$

Using the formula above unfortunately results in stagnation in the overlaps and as a result, a non-convergent algorithm [Gan+08]. To solve the problem, Cai and Sarkis [CS99] introduced the restricted additive Schwarz method, in which the matrices $\tilde{\mathbf{R}}_i$ introduced as the projection matrix from the subspace $\mathcal{V}_{hi}$ to the global subspace $\mathcal{V}_h$. These matrices are generally similar to $\mathbf{R}_i$ except the value 1s in overlap replaced by zeros. The Figure (3.6) illustrates both matrices, $\mathbf{R}_i$ and $\tilde{\mathbf{R}}_i$, in one dimension. Similar to the additive Schwarz method (Algorithm (3.9)), we can also introduce the method for the restricted additive Schwarz method by using the new restriction matrices

$$\mathbf{u}^{l+1} = \mathbf{u}^l + \sum_{i=\mathbb{I}} \tilde{\mathbf{R}}_i^T\mathbf{A}_i^{-1}\mathbf{R}_i(\mathbf{f} - \mathbf{A}\mathbf{u}^l) \tag{3.9}$$

There is a remark here before going to the nonlinear version of the restricted additive Schwarz method.

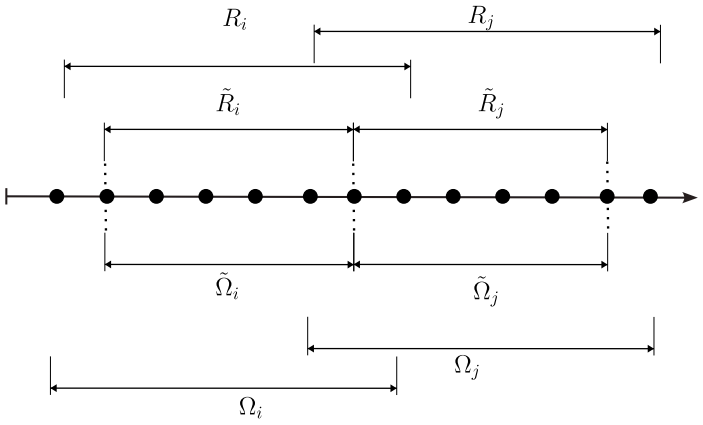


Figure 3.6.: Restriction matrices $\mathbf{R}_i$ and $\tilde{\mathbf{R}}_i$ in one dimension

Remark 3.1 *As it is desirable to have a matrix-free form in GPU implementation, we cannot construct the local matrix* $\mathbf{A}^{-1}$*, therefore, instead of directly solving the local problem, it is required to have an approximation of* $\mathbf{A}^{-1}$*, i.e.,* $\mathbf{B}$ *to be used in each subdomain. Then the local problem can be approximated by using iterative methods like Krylov subspace methods in the linear problem or Newton's method in nonlinear PDEs. Then, the question is if using* $\mathbf{B}$ *instead of* $\mathbf{A}^{-1}$ *has an influence on the efficiency of the algorithm. The answer is that if the local problem is solved accurately enough, then the convergence of the outer iteration is not affected [SBG98].*

In order to solve (or smooth) the nonlinear problem in the Equation (3.2) by using the nonlinear restricted additive Schwarz method, similar to the RAS method, we also need to project the problem into each subdomain by using matrices $\mathbf{R}_i$ (or $\tilde{\mathbf{R}}_i$) and then the local nonlinear problem should be solved or approximated. Then, the local nonlinear problem in discrete form will have the equation of

$$\mathbf{N}_i[\mathbf{u}_i] = \mathbf{m}_i \tag{3.10}$$

In order to have a general nonlinear form, let us assume that $\mathbf{F}_i[\mathbf{u}_i] = \mathbf{N}_i[\mathbf{u}_i] - \mathbf{m}_i = 0$. To solve the local nonlinear problem, we can rely on a nonlinear iterative method like Newton's iterative method. Then the new approximation of the solution in each NRAS iteration can be defined as

$$\mathbf{u}^{l+1} = \mathbf{u}^l + \alpha\tilde{\mathbf{R}}_i\mathbf{s}_i \tag{3.11}$$

where $\mathbf{s}_i \in \mathbb{R}^n$ is the direction computed by the local nonlinear iterative method like Newton's method. Since the local direction computed by local solver might not be the global descent direction, we need to check whether $\mathbf{s} = \sum_{i\in\mathbb{I}} \tilde{\mathbf{R}}_i\mathbf{s}_i$ gives us reasonable

progress.

Let us assume that instead of root finding of the continuous form of the nonlinear operator $n(u^*) = m$, as explained in the Section (5.1), we want to $\min_{u \in \mathcal{V}} \psi(u)$, as explained in the Section (5.1). To ensure that the algorithm NRAS gives a descent direction, it satisfies the Wolfe conditions [NW06]. The Wolfe conditions include two main conditions applied to the inexact line search algorithms. The first condition is a *sufficient decrease* in the objective functional $\psi(u)$ and measures the reduction of the nonlinear residual compared to the linear residual and is defined as follows:

$$\psi(u^l + \alpha^l s^l) \leq \psi(u^l) + c_1 \alpha^l \mathfrak{D}\psi(u^l) s^l \tag{3.12}$$

for $0 < c_1 < 1$. However, the condition can be satisfied by choosing a very small α^l. Then, to avoid an unreasonably short step length, we need another condition. This condition, *curvature condition*, is the second condition in the Wolfe conditions and is defined as follows,

$$\mathfrak{D}\psi(u^l + \alpha^l s^l) s^l \geq c_2 \alpha^l \mathfrak{D}\psi(u^l) s^l \tag{3.13}$$

for $0 < c_2 < 1$. Another common way to ensure that the line search algorithms progress reasonably is to choose α^l by using *backtracking* methods [EW94]. Then the condition (3.13) can be dispensed and we only need the condition (3.12). The simple strategy to fulfill the condition (3.12) can be as follows:

Algorithm 3.2 *Backtracking*

Choose $\alpha > 0$, $\rho \in (0,1)$ *and* $c \in (0,1)$

repeat *until* $\psi(u^l + \alpha s^l) \leq \psi(u^l) + c\alpha \mathfrak{D}\psi(u^l) s^l$

$\alpha \leftarrow \rho\alpha$

end

$\alpha^l = \alpha$

Then, in this scenario, we need to check the condition at the global level in each iteration of the NRAS method. This means that at each iteration of the NRASM, we need to check the condition for the direction $s = \sum_{i \in \mathbb{I}} \tilde{\mathbf{R}}_i s_i$ by using Algorithm (3.2). Finally, the complete globally convergent NRAS algorithm to compute the descent direction s^l can be written as follows,

Algorithm 3.3 *Nonlinear restricted additive Schwarz method*

i) Compute local direction s_i *by using the algorithm defined in Algorithm (3.4) until*

$$\|n(u^l + \tilde{R}_i s_i)\| \leq \eta \|n(u^l)\|$$

ii) compute α^l *by the Algorithm (3.2)*

iii) $s^l \leftarrow \alpha^l s^l$

The next step in the Algorithm FAS-NRASM is to solve the local nonlinear problem, this is the main goal of the rest of this chapter, to have the complete nonlinear algorithm.

3.3.1. Globally convergent inexact Newton method

To solve the nonlinear local problem $n_i(u_i) = m_i$ in a given subdomain Ω_i, we need a nonlinear iterative method. The root finding of the functional $n_i(u_i)$ can also be reformulated to the minimization of the problem

$$\min_{u \in \mathcal{V}_i} \psi_i(u)$$

where $\psi_i : \mathcal{V}_i \to \mathbb{R}$, and the first order necessary condition [NW06]

$$n_i(u) := \mathfrak{D}\psi_i(u) = m_i$$

Then in order to minimize $\psi_i(u)$ (or find the root of $n_i(u)$) we can use the line search method or the trust region [NW06]. In both scenarios, we usually look for the direction s_i and the step length α_i

$$u_i^{l+1} = u_i^l - \alpha_i s_i \tag{3.14}$$

In line search methods, we fix the direction and look for the step length whereas in trust region the maximum distance is chosen first and then we seek a direction. Here we look at the line search method and specifically Newton's method. In Newton's method, we demand that $\mathfrak{D}n(u) \in \mathfrak{L}(\mathcal{X}; \mathcal{Y})$ be bijective and Lipschitz continuous [Cia13]. Then, the direction $p_i = -\mathfrak{D}n(u_i^l)^{-1} n(u_i^l)$ is the descent direction in Newton's method, by which the second-order Taylor series of the functional $\psi(u)$ is minimized, and the convergence is geometric.

Remark 3.2 *There are also different variants of Newton's method. In the generalized Newton's method,* $\mathfrak{D}n(u^l)$ *is replaced by a linear bijective operator* $A \in \mathfrak{L}(\mathcal{X}; \mathcal{Y})$ *(as defined in the Fréchet derivative) [Cia13]. In the Chord method* $\mathfrak{D}n(u^0)$ *is used in all iterations of Newton's method [Kel95].*

In each step of Newton's method, the Newton's equation

$$\mathfrak{D}n(u^l)s^l = -n(u^l) \tag{3.15}$$

should be solved. If $u^0 \in B(u^*; r)$ for small r and functional $n(u)$ bijective, then u^l converges superlinearly. Having Lipschitz continuity, quadratic convergence can be achieved [OR00]. If the Newton's Equation (3.15) is solved exactly, the method is called *exact Newton's method*, otherwise, by using an iterative method, like the

Krylov subspace methods, *inexact Newton's method*. Newton's method has a local convergence, which means that a poor initial guess might lead to slow convergence or even divergence. In order to have a globally convergent algorithm, a sufficient condition (3.12) can be used with the inexact Newton's method. Generally speaking, in backtracking, the actual reduction and the predicted reduction are compared. If the actual reduction exceeds the predicted reduction with some constant factor, we accept the step. Otherwise, the step s^l is damped by the damping factor θ. The sufficient condition is usually reformulated in practice [EW94] as

$$\begin{aligned}\|n(u^l + \alpha s^l)\| &\approx \|n(u^l) + \alpha \mathfrak{D} n(u^l) s^l\| \\ &\leq \alpha \|n(u^l) + \mathfrak{D} n(u^l) s^l\| + (1-\alpha)\|n(u^l)\| \\ &\leq (1 - \theta(1-\eta^l))\|n(u^l)\| \qquad \text{By having inexact Newton condition}\end{aligned}$$

In order to have a practical inexact Newton's method, we need to look at the *inexact Newton condition* inequality, which is as follows,

$$\|n(u^l) + \mathfrak{D} n(u^l) s^l\| \leq \eta^l \|n(u^l)\| \tag{3.16}$$

The inequality looks at the local convergence of the inexact Newton's method and controls it by the forcing terms η^l. Then, the next question will be how to choose the value for η^l. In the inexact Newton's method if the solution of the Newton equation produces s^l too large, then there is a poor agreement between nonlinear functional $n(u)$ and its linear approximation. The forcing term can be chosen statically or dynamically, and the optimal choice can be problem specific [EW96]. The role of the forcing term is to force the linear residual to be small. If the initial guess u^0 is close to the exact solution u^*, the inexact Newton's method converges quadratically [DS83]. However, away from the exact solution, the local linear model may disagree with the nonlinear operator. Then, it might lead to choosing a small value for η^l and oversolving the Newton equation. Oversolving may result in little or no progress toward the solution. There are some numerical experiments on how to choose the forcing term [Cai+94; BS90; DS83]. Each of these forcing term leads to relatively different local convergence in nonlinear and linear iteration methods. Choosing a small value of the forcing term leads to more iterations of the Krylov iteration method per Newton iteration. On the other hand, large values for η^l results in less work of the Krylov method and more Newton iterations.

In the end, the globally convergent inexact Newton's algorithm can be as follows:

Algorithm 3.4 *Globally convergent inexact Newton's method [EW94]*

Let u^0, $\eta_{max} \in [0,1)$, $0 < \theta_{min} < \theta_{max} < 1$ *and* $t \in (0,1)$ *be given*

Solve iteratively the Newton's equation (3.15) (for example by Krylov subspace methods)

For $k = 0, \infty$ *do*

Find some $\eta_k \in [0, \eta_{max}]$ *and* s^l *such that*

$$\|n(u^l) + \mathfrak{D}n(u^l)s^l\| \leq \eta^l \|n(u^l)\| \quad \textit{inexcat Newton condition}$$

and While

$$\|n(u^l + s^l)\| \leq [1 - t(1 - \eta^l)]\|n(u^l)\| \quad \textit{sufficient condition} \tag{3.17}$$

Choose $\theta \in [\theta_{min}, \theta_{max}]$

Update $s^l \leftarrow \theta s^l$ *and* $\eta^l \leftarrow 1 - \theta(1 - \eta^l)$

Set $u^{l+1} = u^l + s^l$

3.3.1.1. Jacobian-Free Newton Krylov method

One of the critical bottlenecks of parallel computation, and in particular for GPU, is the communication. Compared to the cost in terms of time, communication is usually much more expensive in high-performance computing and is a bottleneck for algorithm optimization. This means that the ratio of computation per communication in an algorithm is a critical factor for the performance of an algorithm. Usually in numerical computations, matrices have a very large size and then reading matrices or a part of them are really communicationally expensive, and most of the time computation units must wait for the fetching data. One way to tackle the problem is to overlap communication and computation. If the matrices are sparse, there are some methods to store only nonzero values and reduce the size of data necessary to be communicated, but we need some metadata to map each array to the original matrix. Another way is to reduce the data that needs to be transferred. Interestingly, in most numerical problems, matrices computed with few values. For example, in discretization of a PDE by using the finite element method, in a structured grid, each element of a matrix is computed by the values or gradient of the basis function on the quadrature points, weights on the quadrature points, and the Jacobian in affine mapping to the reference element, as we can see in following,

$$a_{ij} = \sum_{q=1}^{m} \phi(j, q) \times det(\mathbf{J}) \times \mathbf{w}(q)$$

where ϕ is the basis function, q is the quadrature points in the reference element, $\mathbf{J}$ is the Jacobian in the finite element method and $\mathbf{w}$ are the weights on each quadrature points. These parameters will be discussed in more detail in Chapter (4). Another point in numerical algorithms is that we usually do not explicitly need the matrices, but we mostly need matrix-vector multiplication. Therefore, instead of generating the matrix and later multiplying it with a given vector, we can combine matrix generation and matrix-vector multiplication together.

$$v_i = \sum_{j=1}^{l} a_{ij} \times u_j$$

$$= \sum_{j=1}^{l} \sum_{q=1}^{m} \phi(j,q) \times det(\mathbf{J}) \times \mathbf{w}(q) \times \phi(i,q) \times u_j$$

This approach of treating matrix-vector multiplication increases the computational cost, since for each given vector $\mathbf{v} \in \mathbb{R}^n$, matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ should be computed, but, on the other hand, it reduces the communication cost. For GPU architectures, this approach is more desirable, since the ratio of computation to communication is reduced.
In the nonlinear problem, this approach can also be used, with a minor difference. Looking at the inexact Newton's method and using the Krylov subspace method like GMRES [SS86] as inner iterative solver, we can see the same procedure in which the algorithms need multiplication of the Jacobian of the nonlinear operator $n(u)$, $\mathfrak{D}n(u)$, with a given vector but not forming the Jacobian explicitly. And looking at this Jacobian-vector multiplication closely (this Jacobian refers to the Frèchet derivative of the nonlinear operator $n(u)$), we can easily see that it is just the Gâteaux derivative (See Appendix). The idea is to approximate Jacobian-vector multiplication by the directional (Gâteaux) derivative, and it was first introduced to solve ODEs [BH86; GS83] and later used to solve PDEs [BS90]. Then the matrix-vector multiplication can be approximated as

$$\mathfrak{D}n(u)v \approx \frac{n(u + \xi v) - n(u)}{\xi}$$

where ξ is a small perturbation. This formula is the first-order approximation of Jacobian-vector multiplication. It is also straightforward to construct a second-order approximation by central difference [KK04]. Then choosing the perturbation parameter ξ is the critical point of the approximation. Since u and v can change significantly in each iteration of linear and nonlinear iterations, the approximation is very sensitive to the parameter. If ξ is too large, the approximation is poor due to a very large truncation error. And if the perturbation parameter is very small, it might end up having singular Jacobian. In general, ξ can be chosen as a static or dynamic parameter. Some research is done on how to choose the parameter [CJ84; DS96; PW98; Kel03]. In the current project, the perturbation parameter is chosen using the following formula [Kel03]

$$\xi(u,v) := \frac{max(|u^T v|, \|v\|) sign(u^T v)}{\|v\|} \tag{3.18}$$

then, $\mathfrak{D}n(u)v$ can be computed by using the forward difference directional derivative as follows,

$$\mathfrak{D}n(u)v := \begin{cases} 0, & v = 0, \\ \|v\| \dfrac{n(u + \xi(u,v) h \frac{v}{\|v\|}) - n(u)}{\xi(u,v) h}, & v \neq 0. \end{cases} \tag{3.19}$$

3.3.2. GMRES

In the inexact Newton's method, as mentioned above, the linearized problem is solved iteratively. The Krylov subspace methods are the iterative method widely used to solve the linearized problem [BS90; Bro87]. Among the Krylov subspace method, GMRES is the algorithm used most in Newton-Krylov method, since the residual is minimized in each iteration. The only issue with the GMRES method is that the storage requirement increases with each iteration. The remedy for increasing storage in GMRES is to use the restarted version, although the price is deterioration of the convergence rate. An alternative to GMRES is the Lanczos based methods, like BiCGSTAB [Van92] or TFQMR [Fre93], but these methods are not robust when they are used in matrix-free form of the Newton-Krylov method. The number of iterations of the linear solver in the Newton-Krylov method is higher in the Jacobian-free implementation, compared to the standard implementation, in particular, by increasing the size of the problem [MK94]. Since Newton's method is also local solver and the size of the local problem in each subdomain is small, standard GMRES without restarting shows a good performance. GMRES relies on Arnoldi's method to generate an orthogonal basis for the Krylov subspace. Arnoldi's method [SS86] applied to the residual equation of (3.15) generates a sequence of the approximation of v_m to the solution of the residual satisfying the following condition,

$$y_m \in \mathcal{K}_m \quad \text{and } (r_0 - \mathfrak{D}n(u^l)y_m) \perp \mathcal{K}_m$$

where Krylov subspace $\mathcal{K}_m = \text{span}\{r_0, \mathfrak{D}n(u^l)r_0, \ldots, \mathfrak{D}n(u^l)^{m-1}r_0\}$. If $V_m := [v_1, v_2, \ldots, v_m] \in \mathbb{R}^{n\times m}$ is the orthogonal basis of the subspace $\mathcal{K}_m$, then Arnoldi's method iteratively generates the orthogonal basis of the Krylov subspace such that $V_m^T \mathfrak{D}n(u^l)V_m$ has an upper Hessenberg form. Having the Hessenberg matrix, we can change the residual equation as follows,

$$\|\mathfrak{D}n(u^l)u^l + n(u^l)\|_2 = h_{m+1,m}|e_m^T d_m|$$

where $e_m = (0, \ldots, 0, 1)^T$ and $d_m \in \mathbb{R}^m$ such that $y_m = V_m d_m$. The complete algorithm of the GMRES can be found in [SS86].

3.3.3. GPU architecture and the FAS-NRAS algorithm

After having all the components of the algorithm FAS-NRAS (Algorithm (3.1)), this includes Algorithm (3.3) as smoother, Algorithm (3.4) as a local nonlinear solver in each subdomain of domain decomposition, and GMRES as a linear solver for linearized problem (Newton's equation (3.15)), we can look at how the algorithm and hardware of GPU (Figure (3.1)) work together and what possible constraints, the GPU architecture can impose on the algorithm.

Let us assume that the finite element method is used to discretize a PDE, as described briefly in Section (4.2) in the next chapter, there is a hierarchy of the components of the algorithm. The smallest component of the algorithm is a Degree of Freedom (DoF) in a cell. The DoFs in a cell are the next level of the component of the numerical method. Depending on the order of the polynomial in a cell, there can be a different number of DoFs in the cells and there can be different orders of polynomial in each cell. In our experiment, orders of the polynomial in cells are identical. Each subdomain contains a specific number of cells, which is the third level of the hierarchy of the numerical method. And the last level of the numerical method is the global problem.

On the other side, the GPU has a specific memory and computing unit architecture,

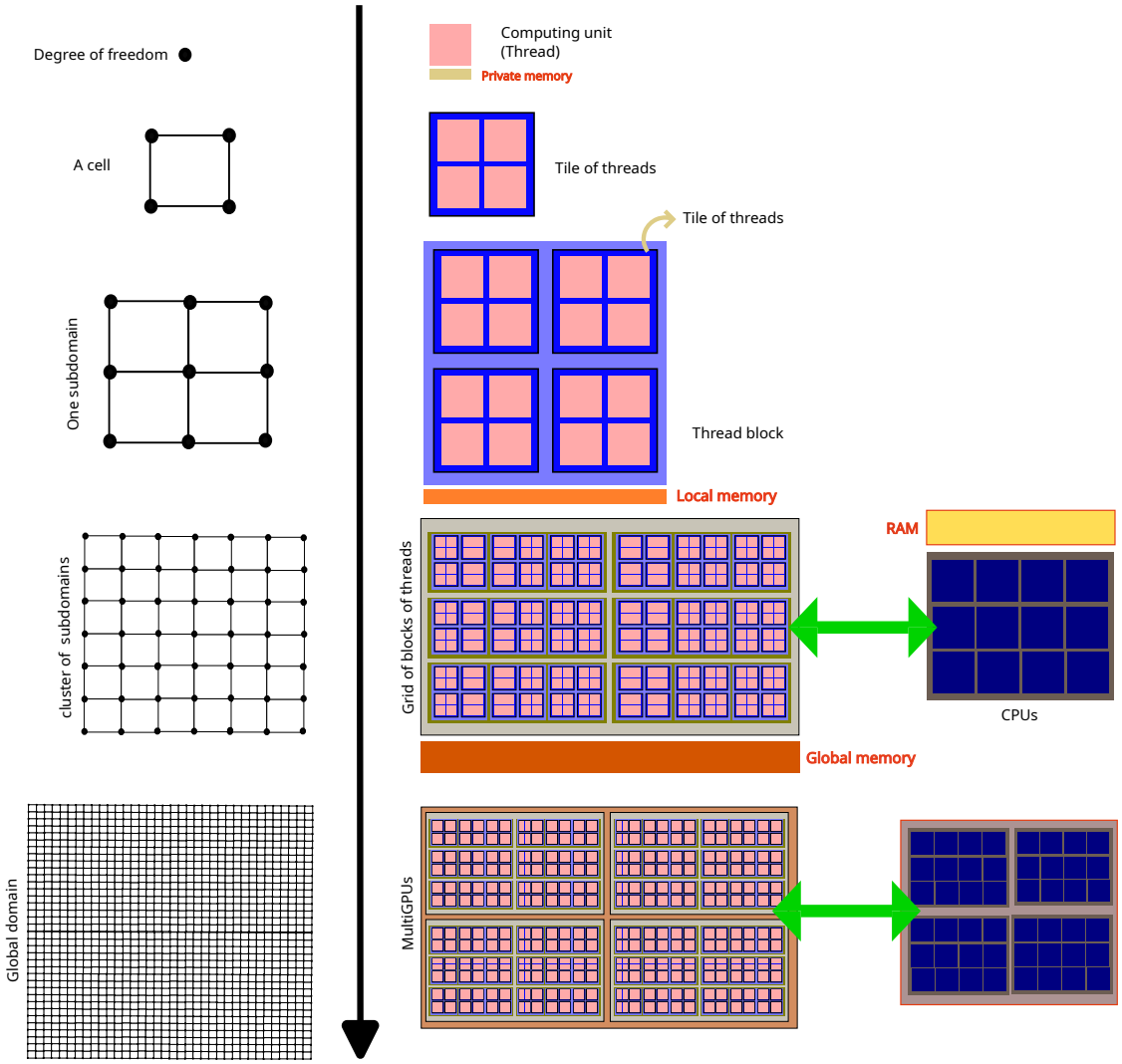


Figure 3.7.: GPU architecture and the Algorithm (3.1)

as was shown and explained in Section (3.1). Using CUDA and particularly utilizing the potency in the cooperative group, we can also have two more components in the computing unit hierarchy. Each block of threads can be partitioned into a smaller working unit, usually called a *tile*. This component has its own synchronization [HP17], indexing and other functionalities. In developing the algorithm for the GPU architecture, we can see that there is a one-to-one association between the GPU architecture and the algorithm, as we can observe in Figure (3.7). A degree of freedom can be associated to a thread, a cell to a tile, a block of threads to subdomain, and finally a GPU to the global domain. In the case of a large problem, a cluster of subdomains can be solved in a GPU and the global problem can be distributed on multi-GPUs.

As we can see in the algorithm, in each domain decomposition iteration, all the subdomains should solve the local problem in parallel and then update their interfaces from adjacent subdomain by the new approximation of the solution in the corresponding adjacent subdomain. Here is one of the points where global synchronization is necessary. In GPU, there are two ways of having global synchronization, one by leaving the running kernel and another one by using global synchronization introduced in cooperative group. In the current work, the second approach is chosen, since I want not to leave the kernel in order to avoid the latency of launching new kernel, possible data transferring between host and device and between global and local memories in GPU. However, using the global synchronization provided by the cooperative group has a caveat. Depending upon the size of shared memory and the number of threads in a block associated to a kernel, the number of threads block in SMs can be changed; for example, larger shared memory allocation for a kernel usually reduces the number of block of threads can be allocated in one SM. This means if there is a large problem we want to solve in GPU, due to constraints imposed by hardware on the number of block of threads in SMs, the number of subdomains are larger than the number of blocks of threads and, therefore, blocks of threads need to work on multiple subdomains. Then we need to transfer data in shared memory to the global memory and store them there to have enough memory space for following subdomains; this eventually leads to more communication between shared memory and global memory, although it is necessary to mention that in the first approach of the global synchronization, the information needed for the following steps of algorithm in the successor kernels should transfer to global memory, since shared memory life-cycle depends on the life-cycle of the kernel to which they are associated, and after leaving the kernel, the shared memory is also de-allocated automatically. The necessity of having large shared memory allocation comes from the necessity of storing data in the methods in the nonlinear solver, which results in growing the size of shared memory.

4. Results

In this chapter, the algorithm FAS-NRAS method analyzed theoretically in Chapter (2) and explained algorithmically in details in Chapter (3) is used in two main nonlinear PDEs, for a zero-order nonlinear Poisson's equation and Allen-Cahn equation. First, the PDEs are introduced and briefly their main properties are explained. Then, the method used to discretize the nonlinear PDEs, the Finite Element Method (FEM), is explained. In the last two sections, the numerical algorithm (Algorithm (3.1)) is used to solve the problem. First, by looking at the algorithm as a nonlinear solver and studying whether the NRAS method is a proper nonlinear smoother in numerical modeling as proven in this work in Corollary (2.11.1). And then, I will look at the FAS-NRAS method as a multilevel optimization technique and investigate if the NRAS method can also work as a preconditioner to reduce the nonlinearity of PDEs and help FAS to work or have a better performance as explained in Equation (5.7).

4.1. Nonlinear equations

4.1.1. Zero-order nonlinear Poisson's equation

The first nonlinear equation solved iteratively with the nonlinear solver FAS-NRAS method is zero-order nonlinear Poisson's equation. The equation is a nonlinear diffusion equation and defined as follows,

$$\begin{aligned} -\triangle u(x,y,z) + \gamma u(x,y,z) e^{u(x,y,z)} &= f(x,y,z) && \text{in } [0,1]^3 \\ u(x,y,z) &= g(x,y,z) && \text{on } \partial\Omega \end{aligned} \tag{4.1}$$

It has a diffusive term, a Laplacian operator, and a nonlinear term. In the nonlinear term, there is a constant parameter γ that can be changed and then the nonlinearity of the PDE will also be controlled. This is a good example for investigating the influence of changing the nonlinearity of the equation can influence the behavior of the nonlinear iterative method. The nonlinear operator in this example has the following form

$$n(u) = -\triangle u(x,y,z) + \gamma u(x,y,z) e^{u(x,y,z)} \tag{4.2}$$

The nonlinear operator has a positive Fréchet derivative, i.e., $\mathfrak{D}n(u)(u^*) > 0$ and also increasing γ makes the problem more local, since the influence of the Laplacian operator is reduced.

In order to compute the error, $e = u^* - u$, and study how the error changes over FAS-NRAS iteration, an analytical solution is plugged in the equation above and

then compute the right-hand side $f(u)$. The analytical solution used in this work is as follows,

$$u^*(x, y, z) = (x - x^2)(y - y^2)(z - z^2)$$

in which the boundary condition will be a homogeneous Dirichlet boundary condition, $g(x, y, z) = 0$. Then by plugging the analytical solution into the equation, the right-hand side will be as follows,

$$f(u) = 2((x - x^2) + (y - y^2) + (z - z^2)) + \gamma u e^u$$

4.1.2. The Allen-Cahn equation

The second example that will be studied using the FAS-NRAS method is the Allen-Cahn equation [AC79a]. It is an advective-diffusive nonlinear PDE and used to model different phenomena in the nature, mainly as a phase-field approach to interfacial evolution problems in material science and biological systems. The phase-field particularly describes application of diffuse-interface in solidification and dendrite growth [Bro+84; Fix82; KKL84; KKL85]. In general, phase-field models describe physical processes in which the system can be locally at different states, e.g., a solid and liquid state in mixing and melting processes. In mathematical form, it is also similar to the time-dependent Ginzburg-Landau equation [GL50] which was primarily developed for diffuse-interface of superconducting phase transitions and compositional clustering. In phase-field method, there is a labeling function $u \in \mathcal{V}$ which is assigned to value 0 (sometimes -1) for one phase or state, and another value, 1, for another state or phase. The function u changes rapidly but smoothly from 0 to 1 in the interfacial region with length ϵ. In Figure (4.1), the function $u(x, y, z, t)$ is depicted on a line in the initial time step and $t = 0.2$ sec.

The Allen-Cahn equation can generally be formulated as follows,

$$\begin{aligned} \frac{1}{K}\frac{\partial u}{\partial t} &= \triangle u - \frac{1}{\epsilon^2}\mathfrak{D}W(u) \qquad \text{on } [-0.5, 0.5]^3 \times [0, T]\ T > 0 \\ \frac{\partial u}{\partial n} &= 0 \quad \text{on } \partial\Omega \\ u(x, y, z, 0) &= u(t = 0) \end{aligned} \tag{4.3}$$

where K denotes the kinetic coefficient and $0 < \epsilon \ll 1$ determines the width of the interfaces that separate regions in different phases (see Figure (4.1)). The Allen-Cahn equation can be seen as the L^2 gradient flow dynamics for the Ginzburg-Landau free energy functional $E_\epsilon : W^{1,2}(\Omega) \to \mathbb{R}$ which is defined as follows,

$$E_\epsilon(u) := \int_\Omega \left(\frac{\epsilon}{2}|\nabla u|^2 + \frac{1}{2}W(u) \right) dx$$

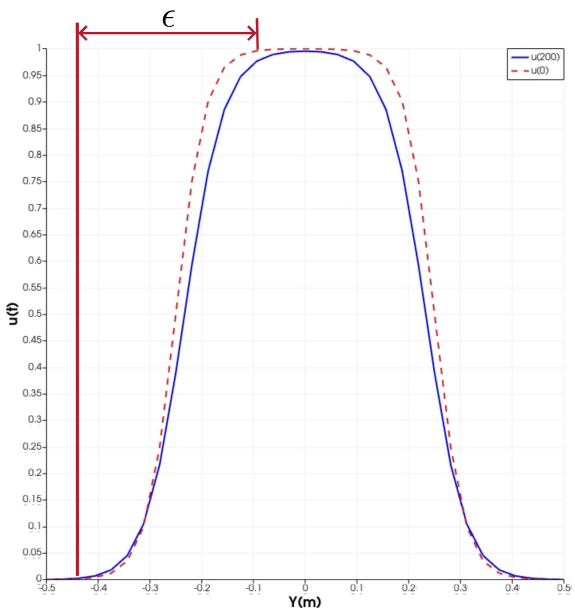


Figure 4.1.: Function u in phase field problem from the Allen-Cahn equation.

where $W(u) : \mathcal{V} \to \mathbb{R}$ is the double well potential which enforces the phase field function to be equal to 1 inside the interface and 0 (or -1) outside it.
To derive the Allen-Cahn equation from the Ginzburg-Landau energy function, as mentioned, we need to get the L_2 gradient of the functional as follows,

$$\frac{\partial u}{\partial t} = -\partial_G E_\epsilon \quad \text{in } L^2(\Omega)$$

where $\partial_G E_\epsilon$ denotes the Gâteaux derivative of E_ϵ at u. The Allen-Cahn equation is an evolution problem and has a time derivative; therefore, in addition to the space discretization method, which will be explained in the following section, we need a numerical scheme for the time derivative.
Let us assume that the Allen-Cahn equation is discretized using the finite element method described in Section (4.2). The simplest time discretization scheme is explicit Euler method, but it suffers from stability condition and in long time shows poor performance. Another variation of the Euler scheme is the implicit one, which can be written as follows,
find $u_h^{n+1} \in \mathcal{V}_h$ such that

$$\frac{1}{K}\left(\frac{u_h^{n+1} - u_h^n}{\tau}, \nu_h\right) + \left(\nabla u_h^{n+1}, \nabla \nu_h\right) + \frac{1}{\epsilon^2}\left(\mathfrak{D}W(u_h^{n+1}), \nu_h\right) = 0 \quad \forall \nu_h \in \mathcal{V}_h \tag{4.4}$$

where τ is the time step. Then, the Allen-Cahn equation can be seen as

$$u^{n+1} = u^n + \tau \partial_G E_\epsilon(u^{n+1}) \tag{4.5}$$

which can be reformulated as L^2 gradient flow for the energy functional E_ϵ as [AC79b]

$$\frac{\partial}{\partial t}E_\epsilon(u(t)) = \left(-\triangle u + \frac{1}{\epsilon^2}\mathfrak{D}W(u), u_t\right)_{L^2(\Omega)} = -\|u_t\|_{L^2(\Omega)} \leq 0$$

Defining discrete energy functional $E_{n,IM}(u_h; u_h^n)$ for implicit Euler as

$$E_{n,IM}(u_h; u_h^n) := E_\epsilon(u_h) + \frac{1}{2\tau}\int_\Omega (u_h - u_h^n)^2 dx$$

We can see the implicit Euler method of minimization of the energy functional $E(u_h; u_h^n)$ as

$$u_h^{n+1} = \underset{u_h \epsilon \mathcal{V}_h}{\operatorname{argmin}} E_{n,IM}(u_h; u_h^n). \tag{4.6}$$

Then, the question is if the discrete energy functional derived by using implicit Euler is convex. It can be proved, by using bifurcation analysis [Zha+24] or studying the energy functional introduced above [Xu+19] that under the condition of $\tau < \epsilon^2$ the functional $E_{n,IM}(u_h; u_h^n)$ is convex and the energy minimization in (4.6) is equivalent to (4.4). We can also use a higher-order time stepping scheme, like Crank-Nicolson. With the same analogy as above, it can be proved that the discrete energy functional $E_{n,CN}(u_h; u_h^n)$ defined as Formula 4.2 in [Xu+19] under the condition $\tau \leq 2\epsilon^2$ is strictly convex and the solution u_h^{n+1} is the minimum of the discrete energy functional $E_{n,CN}(u_h; u_h^n)$ as follows,

$$u_h^{n+1} = \underset{u_h \epsilon \mathcal{V}_h}{\operatorname{argmin}} E_{n,CN}(u_h; u_h^n)$$

4.2. Finite element method

The equations introduced above as test cases should be solved numerically. The first step for numerical simulation is to discretize the continuous PDEs with a discretization methods like the finite difference, finite element or finite volume methods. In the current project, the continuous Galerkin finite element method will be the discretization method [GRS07]. In this section, finite element discretization will be explained briefly. Implementing the Dirichlet boundary condition in finite element discretization is not straightforward. In particular, in Section (4.2.4), the treatment of this kind of boundary condition will be discussed. In the Galerkin's method, the discrete problem is defined over the finite dimension subspace $\mathcal{V}_h \subset \mathcal{V}$. In general, a finite element has three main components as $\{K, \mathcal{P}_K^p, \Sigma_K\}$ which are the geometry of the subdomain K, local ansatz function in K which is the space of polynomials of degree of p and, a unisolvent linear functional. These three components will be explained in the following sections.

4.2.1. Triangulation

The first step in finite element implementation is the triangulation. If $\mathcal{T}_h$ $(h > 0)$ be a partition of the closure $\bar{\Omega}$ of the domain Ω into a finite number of closed d-dimensional simplex T ($d = 3$ in current work) with mutually disjoint interiors such that

$$\bar{\Omega} = \bigcup_{T \in \mathcal{T}_h} T$$

The element T can be triangles or in the current work 3D parallelepiped (cube).

4.2.2. Finite Element space

Let

$$\mathcal{V}_h := \{\nu_h, \nu_h \in C(\bar{\Omega}) \cap H_0^1(\Omega), \nu_h|_T \in \mathcal{P}^1(T) \ \forall T \in \mathcal{T}_h\} \tag{4.7}$$

denote the space of continuous piecewise linear functions in $\mathcal{T}_h$ that vanishes on the boundaries, including the subdomain interfaces Γ_{ij} and $\partial\Omega$. Let the basis functions for $\mathcal{V}_h$ be denoted by $\{\phi_i\}_{i=1}^{\tilde{N}}$ and satisfy the following condition,

$$\phi_i \in \mathcal{V}_h, \ \phi_i(v_j) = \delta_{ij}, \ i, j = 1, \ldots, \tilde{N}$$

where δ_{ij} is the Kronecker symbol and v_i are mesh vertices. This function is also called a *hat function* and obviously has a very small support. Then, having the basis function for $\mathcal{V}_h$, any function $u \in \mathcal{V}_h$ can be represented as,

$$u(v) = \sum_{i=0}^{N} u_i \phi_i(v_i) \tag{4.8}$$

where each parameter $u_i \in \mathbb{R}$ is the unique value of the function u on each vertex v_i. In order to evaluate integral (volume integral) in the reference element *tri-Gauss* quadrature points are used.

4.2.3. Discretization

In general, as mentioned in Chapter (2), the weak formulation of any equation in a described as (2.32). Discretization of the equation using the conforming version of the Ritz-Galerkin method, which is based on a subspace $\mathcal{V}_h \subset \mathcal{V}$. In order to solve a nonlinear problem numerically, we can approach it in two different ways; First, discretizing the continuous form and then linearizing it, or in the second approach, first linearizing the nonlinear continuous form, and then discretizing the linearized continuous form. In the current project, the former approach will be the way to solve the problem numerically, since in Jacobian-free implementation, Jacobian is not generated explicitly, and only Jacobian-vector multipication, as explained in Section

(3.3.1.1), is needed. Then the discrete nonlinear problem is as follows,
find $u_h \in \mathcal{V}_h$ such that,

$$\langle n(u_h), \nu_h \rangle = (m, \nu_h) \quad \forall \nu_h \in \mathcal{V}_h$$

Then, using the basis function, we can describe $N_h = (n(u)\phi_i))_{i=1}^{\tilde{N}}$ and $m_h = (m(\phi_i))_{i=1}^{\tilde{N}}$. This discretized form can be used later, and by using Newton's method in Section (3.3.1), we can linearize it.

4.2.4. Treatment of boundary conditions

The target PDEs used in the current project to study the FAS-NRAS algorithm as a solver in the current chapter and in the next chapter as a multilevel optimizer method, requires two different boundary conditions to be implemented. In each subdomain, the boundary consists of two adjoint parts, the interfaces Γ_{ij} and boundaries $\partial\Omega_i$. In the first nonlinear equation, on interfaces Γ_{ij} and boundaries $\partial\Omega_i$ the Dirichlet boundary condition should be considered. In the Allen-Cahn equation, on interfaces, the Dirichlet boundary condition should be implemented, and on $\partial\Omega_i$, the periodic boundary condition.
Let us assume the Dirichlet boundary conditions on $\partial\Omega_i$ and Γ_{ij} are defined as follows,

$$\begin{aligned} u(t,x,y,z) &= u_b, \quad \text{on } \partial\Omega_i \\ u(t,x,y,z) &= u_\Gamma, \quad \text{on } \Gamma_{ij} \end{aligned} \tag{4.9}$$

where u_b and u_Ω are a trace of some $u^* \in H^1(\Omega)$.

In the implementation of the Dirichlet boundary condition, there are generally two ways to follow; strong implementation and weak implementation [Nit71]. In the strong implementation of the Dirichlet boundary condition, the finite element approximation of (2.42) can be written as
find $u_h \in \mathcal{V}_h$ such that

$$\begin{aligned} (u_h - u_{\partial\Omega}, \nu_h)_{L^2(\partial\Omega)} &= 0 && \forall \nu_h \in \mathcal{V}_h \\ a_h(u_h, \nu_h) &= m(\nu_h) && \forall \nu_h \in \mathcal{V}_h \end{aligned}$$

where $a_h(.,.)$ is the weak formulation of the nonlinear problem. On the other hand, in weak implementation of the Dirichlet boundary condition, we look at the Dirichlet boundary condition as a constraint imposed on the bilinear form. In this way, the subspace is defined as

$$\mathcal{V}_h := \{\nu_h, \nu_h \in H^1(\Omega), \nu_h|_T \in \mathcal{P}^1(T) \ \forall T \in \mathcal{T}_h\}$$

Then the bilinear form will also have face integrals, include normal derivatives of u and ν and a penalty term (Equation (3) in [FS95]). (In general, it is similar to the implementation of the Dirichlet boundary condition in the discontinuous Galerkin

finite element method (compare Equation (3) in [FS95] with $(M_e^{11})_{ij}$ on page 53 in [Riv08]). Then in the bilinear form we will have a penalty term which is added to have a coercive bilinear form. Choosing a small penalty term usually leads to loss coercivity, and a large value for the penalty term leads to deterioration of the convergence rate of the iterative method.
Since weak implementation of the Dirichlet boundary condition needs a careful choice of the penalty term in the bilinear form in order to achieve coercivity in the bilinear form close to the boundary, in the current work, the Dirichlet boundary condition on interfaces Γ_{ij} and on $\partial\Omega$ are imposed strongly.

Another boundary condition we need for the examples is periodic boundary condition. In order to implement periodic boundary condition, ghost cells added to at every side of the domain. Here, the values on the ghost cells are the value of boundary $\partial\Omega$ on the other side of the domain. The periodic boundary condition alters by Dirichlet boundary condition and is implemented in strong form as explained above.

4.3. FAS as a nonlinear solver

The first step in studying the FAS-NRAS method is to use the algorithm as a nonlinear solver for the two nonlinear problems, zero-order nonlinear Poisson's equation as a nonlinear elliptic PDE, and the Allen-Cahn equation as a nonlinear parabolic PDE. As explained in Section (3.3) in Chapter (3), the continuous PDEs are first projected to subspaces $\mathcal{V}$ using the projection operators P_i and Q_i as defined in Chapter (2) and then the projected equation in subspace $\mathcal{V}_i$ is discretized using FEM. The subspace $\mathcal{V}_i$ is a proper Sobolev space, as explained in the previous section, which depends upon the actual boundary $\partial\Omega$ and interfaces Γ_{ij}, and finally form the weak formulation of the PDE and impose the Dirichlet boundary conditions in the strong form.

Before going into the results of the numerical modeling, a remark should be mentioned here. Combining multigrid and domain decomposition, there are two ways of coarsening the grids. The first approach is to coarsen the grid in each subspace until the coarsest grid in the subspace, as shown on the left side of Figure (4.2). In this way, we cannot go to the coarsest space of the global problem, if at certain point composing the subdomains and decomposing with fewer number of subdomains or only having one subdomain and coarsening it until getting the coarsest grid of the problem. If we only coarsen in each subdomain until the coarsest subspace in each subdomain and avoid the steps of composing, decomposing to the new set of subdomains, we can not remove all the low frequency modes of error, therefore the problem needs more iteration of V-Cycle FAS to get the convergence and by increasing the size of the problem, we might need more multigrid iteration. This means that the number of V-Cycles necessary to get the convergence is not independent of the size of the problem. On the other hand, the second approach that we can rely on is to keep the number of degrees of freedom in the subdomains at each level constant, Strategy

2 on the right side of Figure (4.2). This means that at each level the local problem should project to the global problem, apply the restriction operator, and project to the fewer number of subdomains. In this way, we can go to the coarsest subspace of the global problem and therefore the number of iterations of V-Cycle FAS is expected to be independent of the size of the problem. However, the caveat is that in the second approach, the communication increases by the steps of composing and decomposing in each iteration of V-Cycle FAS. This is really important when the problem is solved on GPU, and is more critical in multiGPU due to the necessity of communication between nodes (MPI communication).

Remark 4.1 *The first strategy might be confused with the use of FAS as a local solver in each subdomain. In this scenario, the local problem is solved by a few number of FAS iterations (V-Cycle, W-Cycle or FMG), and then the values on the interfaces are updated. Whereas in Strategy 1 in Figure (4.2) at each level of multigrid, NRAS iterations are applied and in each iteration of the NRAS method the values on the interfaces are updated. When the smoothing process is done in the current level, the information is projected to the coarser level, and then the smoothing process (including solving the local problem and updating information on the interfaces) is carried out, but on a smaller local problem.*

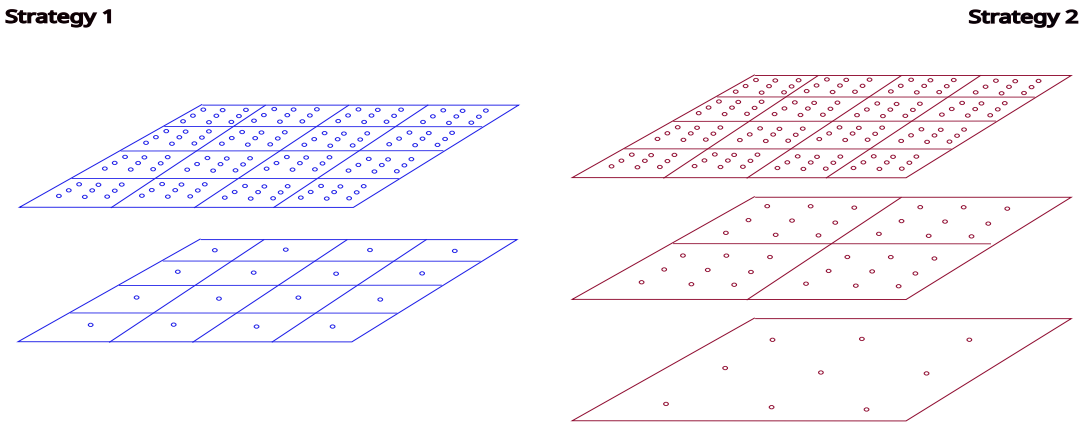


Figure 4.2.: Two strategies of combining FAS and NRAS methods. Here, to avoid confusion by overcrowded figures, circles represent cells in each subdomains not DoFs and lines the interfaces between subdomains, and in the current work only h-multigrid is used and there is not p-multigrid.

4.3.1. Nonlinear Poisson's equation

As it was explained in the previous section, FEM is used to discretize continuous PDEs and generate a discrete form. In the zero-order nonlinear Poisson equation, by

defining the function space as

$$\mathcal{V} := \{\nu \in H_0^1 \cap H^1 | u|_{\partial\Omega} = 0 \text{ and } u|_{\Gamma_{ij}} = u_{\Gamma_{ij}}\} \tag{4.10}$$

then, by having conforming finite element as $\mathcal{V}_h \subset \mathcal{V}$, we can define the weak formulation as follows:

$$\begin{aligned} &\langle n(u), \nu\rangle = \langle f, \nu\rangle \quad \forall \nu \in \mathcal{V}_h \\ &\langle n(u), \nu\rangle = (\nabla u, \nabla \nu) + \gamma(ue^u, \nu) \end{aligned} \tag{4.11}$$

where $n(u)$ is defined in (4.2). Here, as explained in Section (4.1.1), f is the right-hand side computed by plugging the analytical solution.
The algorithm was first developed and tested on CPU before analyzing its efficiency on GPU. In this example, the CPU version is serial, although it is expected that the algorithm with respect to the number of iterations will be the same in GPU as well. The detail in GPU implementation will be discussed in the next example, the Allen-Cahn equation.

The equation is solved for four different problem sizes, from 5 to 8 steps of mesh refinement. In solving the problem, the first strategy was used and the local problems are coarsened to the coarsest space in each subdomain, and then the prolongation process starts. Usually, the size of the overlap in the domain decomposition method is a critical parameter in the efficiency of the algorithm [QV99; SBG98; TW04]. The larger overlap helps the information transfer faster, treating the low frequency modes of error more efficiently, and therefore, the method converges faster [Nat07; GMN02; DGG09]. But in the algorithm FAS-NRAS having multigrid can also help us to have a coarse space and therefore the low frequency modes of error removed by the coarse-grid correction in FAS. The size of the overlap at all levels and in all problems is one cell in each direction, i.e., $\delta = h$. At each problem size, the size of local problem is kept the same; this means by increasing the size of problem, the number of subdomains increases. For example, in 5-step mesh refinement, there are 8 subdomains in each direction, which means that the number of subdomains in the problem is 512. In the next step of mesh refinement, there are 4096 subdomains. In this way, in the coarsest grid, there is a problem of the size of the number of subdomains. At each level, five number of iterations in the NRAS method is used, and the local problem is also approximated by using two numbers of iterations in Newton's method. The linearized problem is also solved by two number of iterations of GMRES iteration. The number of V-Cycle FAS necessary to obtain the error of the order of discretization is shown in Table (4.1). As expected, the number of multigrid necessary to get the convergence increases by increasing the size of the problem, here at each step of mesh refinement the number of V-Cycle FAS almost doubles compared with the previous step of mesh refinement. The reason of this behavior is that in multigird, all the low frequency modes of error are not eliminated, therefore, it needs more iteration to converge.

The third column in Table (4.1) shows the ratio of the error $\frac{\|e\|}{\|e_0\|}$ where e_0 is the initial error and e is the error after obtaining the convergence. The ratio of error

Table 4.1.: Number of FAS V-Cycle necessary to get the error of the order of discretization in four different size of problem and the ratio of error reduction in each mesh refinement step

Mesh refinement steps	Number of V-Cycles FAS	Ratio of $\|e\|_{L^2}$	time-to-solution (sec)
5	9	0.043	132
6	18	0.012	2025
7	36	0.003	31551
8	76	0.001	524327

reduces by a factor of 4 from one step of mesh refinement to the next. This is an optimal behavior we expect to see, which shows that the algorithm works as expected in reducing the error to the order of discretization error.

Another important factor in the evaluation of a numerical method is to study the behavior of time-to-solution by increasing the size of problem. The optimal behavior is a linear increase in the computation time with respect to the increase of the size of problem. In the last column of Table (4.1), the time-to-solution for each problem size and in Figure (4.3), the empirical time-to-solution and the optimal behavior are shown. A log-log plot of the time-to-solution against the number of degrees of freedom yields a complexity of approximately $O(N^{1.33})$. Thus, the solver exhibits superlinear scaling with problem size and does not achieve the ideal linear $O(N)$ complexity expected from an optimally scalable algorithm. This behavior is expectable since the number of the V-Cycle FAS increases by a factor of 2 by increasing the step of mesh refinement. The reason of this behavior, as it was explained before, is that by choosing the strategy 1 in (4.2), I could not go to the coarsest grid, and the number V-Cycle FAS to get convergence increases by factor of 2 (column 2 of the Table), then the time-to-solution increases by factor of ≈ 16 instead of 8.

4.3.2. The Allen-Cahn equation

The Allen-Cahn equation (4.3) is a time-dependent parabolic equation. To have a weak formulation, first the time discretization method is applied, and then the weak formulation for each time step is derived.
In this current experiment, the double-wall function $W(u)$ in Equation (4.3) is defined as follows,

$$W(u) := \frac{1}{2}u^2 - u^3 + \frac{1}{2}u^4$$

therefore, we will have

$$\mathfrak{D}W(u) := u(1-u)(1-2u)$$

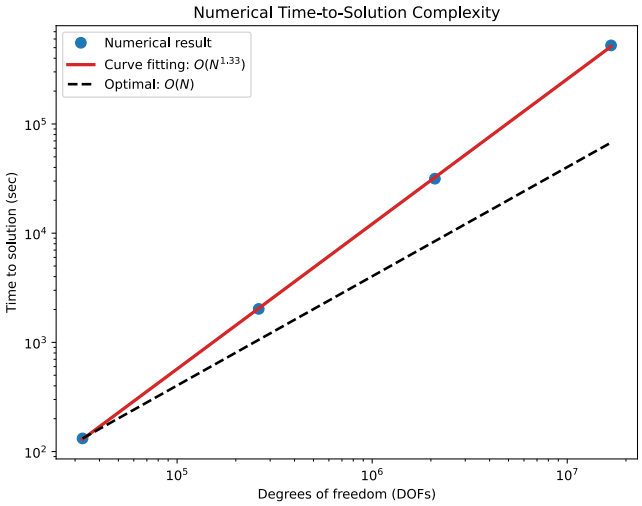


Figure 4.3.: Time to solution for different sizes of problem.

At first the backward Euler time discretization is used, and we will have the equation as

$$n(u^{k+1}) = u^{k+1} + K\tau\left(-\triangle u^{k+1} + \frac{1}{\epsilon^2}\mathfrak{D}W(u^{k+1})\right) \tag{4.12}$$
$$m(u^k) = u^k$$

and in using Crank-Nicolson for the discretization of time derivative we will have

$$n(u^{k+1}) = u^{k+1} + \frac{K\tau}{2}\left(-\triangle u^{k+1} + \frac{1}{\epsilon^2}\mathfrak{D}W(u^{k+1})\right) \tag{4.13}$$
$$m(u^k) = u^k + \frac{K\tau}{2}\left(-\triangle u^k + \frac{1}{\epsilon^2}\mathfrak{D}W(u^k)\right)$$

which has second-order accuracy in time [Ise09].
Multiplying $n(u^{k+1})$ and $m(u^k)$ with an ansatz function, we will have the weak formulation as
find $u^{k+1} \in \mathcal{V}_h$ such that

$$\langle n(u^{k+1}), \nu\rangle = \langle m(u^k), \nu\rangle \quad \forall \nu \in \mathcal{V}_h$$

for $m(u^k) \in \mathcal{V}_h'$. The Allen-Cahn equation is a time-dependent problem, and therefore, we need an initial value for the function $u(t=0)$. The initial value of the function u at $t=0$ can be computed using the following formula,

$$u(t=0) := \frac{1}{2}\left(1 + \frac{tanh(R_0 - \|r\|)}{\sqrt{2}\epsilon}\right)$$

where $R_0 > 0$ is the initial radius of the function u, where $u(x, y, z, 0) = 1$ and r is the distance of a given point (x, y, z) from the center. The profile is instable, and the radius of the sphere is reduced over time. In the sharp interface, $\epsilon \to 0$ the radius of the sphere can be expressed as follows,

$$R(t) = \sqrt{R_0^2 - 4K(t - t_0)} \tag{4.14}$$

Then in order to check whether the numerical method can capture the behavior of the problem properly, the radius computed numerically will be compared with the formula above.

In the experiment for numerically solving the Allen-Cahn equation, the parameters are chosen as follows,

$$\begin{aligned} R_0 &= 0.25 \\ \epsilon &= 0.04 \\ K &= 0.04 \end{aligned}$$

In the first example, in this section, for FAS-NRAS as a solver τ is chosen in the way that the discrete energy functional for backward Euler $E_{n,IM}(u_h; u_h^n)$ in Equation (4.6) and Crank-Nicolson $E_{n,CN}(u_h; u_h^n)$ in Equation (4.1) in [Xu+19] are convex. This means that regarding the value of $\epsilon = 0.04$, the time step in backward-Euler should be $\tau_{IM} < 1.6 \times 10^{-3}$ and in Crank-Nicolson $\tau_{CN} < 3.2 \times 10^{-3}$. In order to solve the problem using the iterative method, we need an initial guess, $u(t = 0)$. The initial guess for the iterative algorithm, the solution of the previous time step, $u_0^{k+1} = u^k$. This is a proper initial guess, since the time step is small.

In solving the Allen-Cahn equation on GPU, the strategy of the implementation was changed due to the constraints imposed by GPU architectures. In order to use the global synchronization introduced by cooperative group, the number of thread blocks in a SM should be calculated by using the function *cudaOccupancyMaxActiveBlocksPerMultiprocessor*. Then this function according to the size of a block of threads and allocated shared memory for the kernel can give us the number of block of threads in a SM. In the current algorithm, the size of shared memory is a critical factor which determines the number of block of threads. Since the methods used in the FAS-NRAS algorithms need to store some local vectors in each iteration of the V-Cycle FAS, the size of shared memory is relatively large ($\approx 65\ KB$, NVIDIA A100 has 164 KB), which dictates having only one block of threads allocated to a SM. And since the total number of SM in NVIDIA A100 is 108 (see Table (3.1)), each block of threads should work on multiple subdomains. This means that the local information for each subdomain should be transferred to global memory. This leads to the conclusion that if the problem is large, there is a need to communicate to global memory. Therefore, although the first strategy, is more efficient for the reduction of communication in theory, it cannot be practical for a large problem. As a result, the second strategy in Figure (4.2) was used to solve the Allen-Cahn equation in GPU.

Table 4.2.: Number of V-Cycle FAS iterations necessary in each step of mesh refinement to get the convergence for solving the Allen-Cahn equation

Number of mesh refinement steps	5	6	7	8
V-Cycles FAS	4	4	4	4

The first test for solving the Allen-Cahn equation is to see the number of V-Cylce FAS iterations to solve the nonlinear equation by changing the size of the problem. Similar to the previous nonlinear problem, the Allen-Cahn equation is solved for the problem of size of 5 to 8 steps of mesh refinement. The numbers of pre-smoothing and post-smoothing iterations are also the same, and $\nu_1 = \nu_2 = 2$. In each subdomain, 2 numbers of Newton's method iteration is used to solve the local nonlinear problem, and Newton's equation is also solved with 2 numbers of GMRES. Table (4.2) shows the number of V-Cycle FAS iterations to obtain the relative tolerance of 10^{-8}, i.e., $\|F(u)\|_{L2} < tol\|F(u_0)\|_{L2}$. As we can see, the number of multigrid iteration stays constant by increasing size of the problem, and this is the behavior expected from the multigrid method. And the reason of this behavior, compared to the results in solving Poisson's equation in Table (4.1), is that in the current case, we can go to the coarsest grid and remove the low frequency modes of error, whereas in the former we can only go to the coarsest grid within subdomain, therefore, some low frequency modes of error are not treated properly.

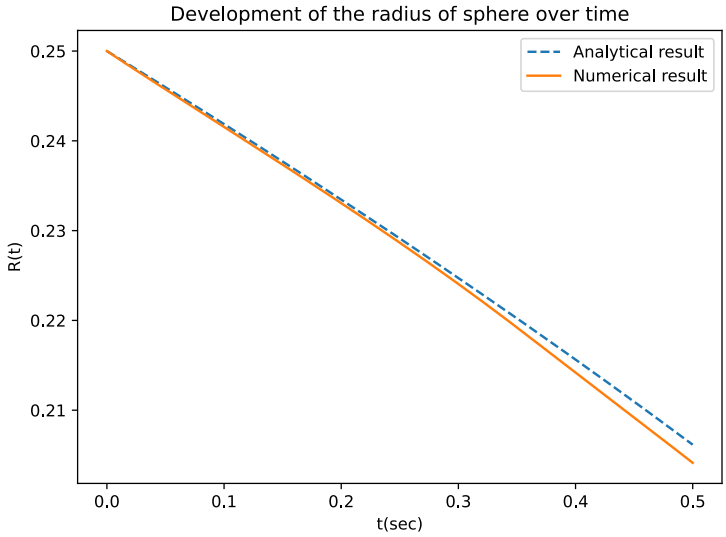


Figure 4.4.: Development of the radius of sphere R_t computed analytically and numerically for the Allen-Cahn equation

The Allen-Cahn equation is an evolution equation by which the development of the state can be obtained over time. The radius of the sphere in the Allen-Cahn equation shrinks over time until there is only one state (phase) left. The evolution of

the sphere can be seen in Figure (4.5). As we can see, the numerical result shows very good agreement with the analytical radius, even after 500 time steps, even considering the accumulated error over time steps, $\|r_a - r_n\|_{inf} \approx 2 \times 10^{-3}$. In order to validate the numerical result. As mentioned before, the radius of the sphere computed numerically is compared to the radius computed analytically. The numerical and analytical evolution of the radius over time can be seen in Figure (4.6). In solving the Allen-

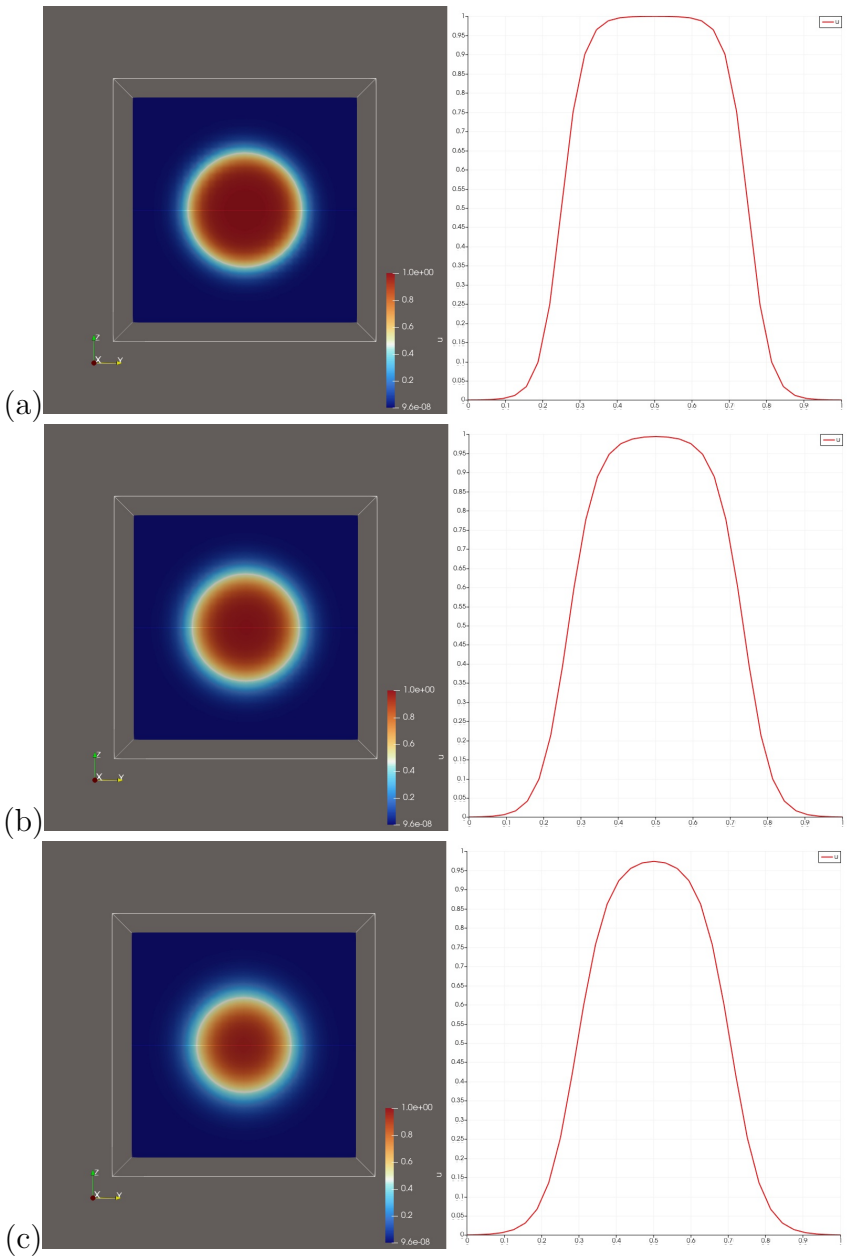


Figure 4.5.: Solution of the Allen-Cahn equation in a 2D slice orthogonal to $(1, 0, 0)$ in left column and on the line $(0, y, 0)$ in right column at different time (a) $t = 0$ sec (b) $t = 0.2$ sec (c) $t = 0.5$ sec

Cahn equation, both backward Euler and Crank-Nicolson methods are used. The backward Euler is a first-order time scheme. In order to have a higher accuracy in time, Crank-Nicolson can be used. In Crank-Nicolson, in each time step, there is only a necessity to evaluate the nonlinear operator on the solution of the previous time step ($n(u^{k-1})$), therefore, it does not increase the cost of computation significantly, and we can obtain second-order accuracy in time with a reasonable computational price. In addition, it also gives us a potency to have a convex energy functional for the larger time step, as described in (4.1.2). In order to study the development of error $e = u - u_h$, similar to the nonlinear Poisson's equation, an analytical function is plugged into the Allen-Cahn equation, and it introduces a source term $f(x, y, z)$ on the right-hand side of the equation. Then the equation will change as follows,

$$\frac{1}{K}\frac{\partial u}{\partial t} = \triangle u - \frac{1}{\epsilon^2}\mathfrak{D}W(u) + f(x, y, z) \qquad \text{on } [-0.5, 0.5]^3 \times [0, T]\ T > 0 \tag{4.15}$$

and the analytical solution used here is

$$u^* = (1-x)^2(1+x)^2(1-y)^2(1+y)^2(1-z)^2(1+z)^2e^{-t}$$

The function is then plugged into the equation above and we can easily compute $f(x, y, z)$. Using the analytical function changes the boundary condition from a periodic to a homogenous Dirichlet boundary condition. The source term $f(x, y, z)$ can also be computed. For this example, four different time steps, $\tau = 10^{-3},\ 5 \times 10^{-4},\ 10^{-4},\ 5 \times 10^{-5}$, and four problem sizes with mesh refinement from 5 to 8 are tested. Figure (4.6) shows how the error behaves in time discretization schemes, the backward Euler method, upper row, and the Crank-Nicolson method, bottom row, and with respect to the time step τ, left column, and the sizes of problem, right row. For backward Euler method in the case 5 steps mesh refinement reduction of time step does not reduce the error (top-left plot). In order to understand the behavior of the error, it is useful to study different sources of the error and analyze them separately. The main source of error in this problem comes from time discretization, spatial discretization, and error from a numerical method.

$$e_{total} := \underbrace{e_{time\ \ discr}}_{\leq\ C_t\tau_t^n} + \underbrace{e_{spatial\ \ discr}}_{\leq\ C_h h^2} + e_{numerical} \tag{4.16}$$

where $n_t = 1$ for the backward Euler method, and $n_t = 2$ for the Crank-Nicolson method. In order to reduce the total error, all the components should be reduced if they are in the same order or the dominated component(s) should be reduced. That now explains the behavior of the error in the top-left plot. The error from spatial discretization is a dominated component, and reduction in total error by reducing the time-step size is negligible in problem size with 5 steps of mesh refinement. Reducing spatial step size and time step size together gave us the behavior we expect from backward Euler method (green line). The same behavior can also be seen in the top-right for spatial step size reduction, i.e., as time step size is large, reduction in spatial

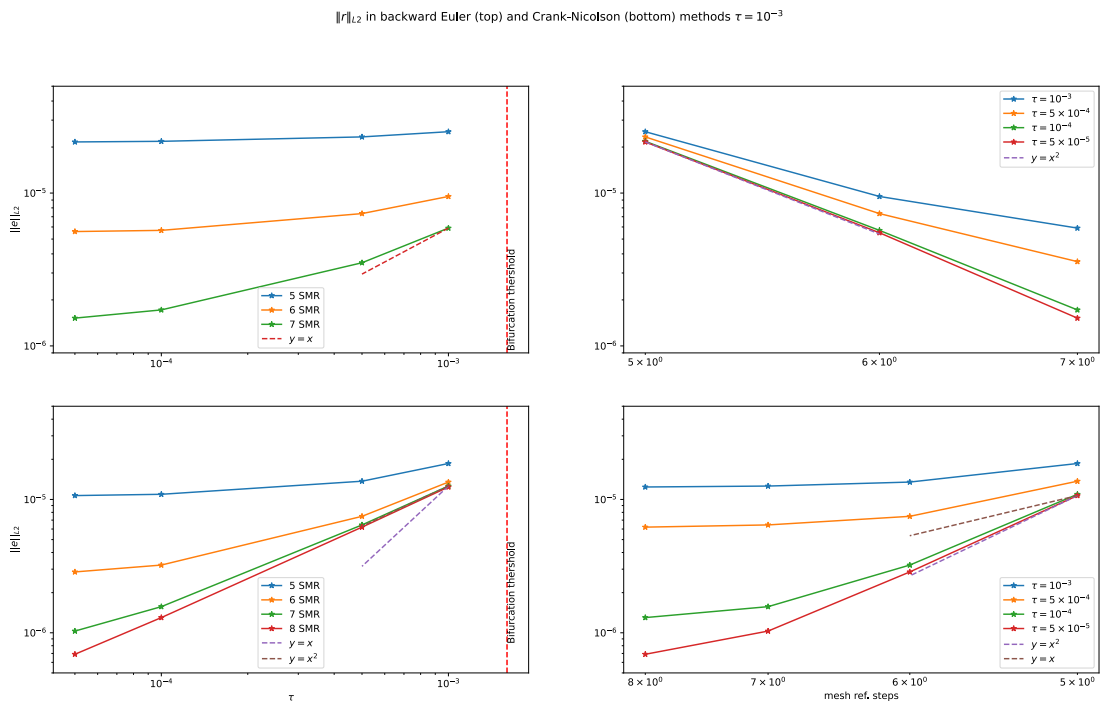


Figure 4.6.: $\|e\|_{L2}$ for the Allen-Cahn equation in Backward Euler and Crank-Nicolson time schemes

step size does not show a second-order reduction in the error.

In the Crank-Nicolson method, the behavior of the error with respect to the spatial step size and time step have also, in general, similar behavior to backward Euler. But there is a difference between the two cases. The Crank-Nicolson method, as we know it, is a second-order time discretization scheme. This means that by reducing in the time step size, the error should be reduced by an order of 2. But this is not what we can observe in the bottom-left plot of Figure (4.16). The main reason of this behavior is that in order to see the behavior of reduction of error by second order with respect to time, we need to make the spatial step size smaller, then the error stems from spatial discretization has no remarkable influence on the behavior of the error. However, currently the second term in (4.16) is the dominate term, hence we cannot see a second-order reduction. On the other hand, in the bottom-right plot, as the time step is small enough such that this component does not have significant influence on the total error, we can see the second-order reduction of the error in the case when $\tau = 5 \times 10^{-5}$. However, the behavior of the error changed by reducing the spatial step size, where the first term in (4.16) again becomes the dominate source (left part of the red line in $\tau = 5 \times 10^{-5}$). So, these results, similar to the previous results for backward Euler, show how the different components of the error change the behavior of the total error.

5. NRASM as a preconditioner-minimizer for FAS as a multilevel optimizer

In this chapter, I look at FAS as a multilevel optimization method. By changing the perspective, there is some valuable information that can be obtained from FAS and how it works with NRAS, in particular looking at the NRAS method as a right preconditioner-minimizer. In Equation (5.11), and the following discussion, I try to explain how using the NRAS method can help FAS to work more efficiently or in some scenarios even halp to work.

The first part of this chapter, I look at FAS as a multilevel optimization method and explain it briefly from this perspective. And then I use this perspective to analyze the multilevel optimization algorithm. I use the result of this analysis to develop a new algorithm based on FAS and NRASM as a nonlinear iterative method which they can also work efficiently on non-convex problems, i.e., the Allen-Cahn equation with bifurcation. I should here emphasize that the new algorithm is still a solver to solve $n(u) = m$ and not an optimization method to minimize $\psi(u)$.

5.1. FAS as a multilevel optimization method

There has been some research looking at the subspace correction methods, parallel or successive, for convex optimization problems [TX02]. In this section, we will analyze FAS as a multilevel optimization method. The first important theorem we need to have here is to show that we can find a functional that whose Fréchet derivative is the functional $n(u)$ by which the problem of root finding in Equation (3.1) will change to a minimization problem. The theorem is as follows,

Theorem 5.1
By using the assumption (2.5) and Theorem (5) in §4.1 [Jeg13] we can verify that there is a functional $\psi : \mathcal{V} \to \mathbb{R}$ such that $\mathfrak{D}\psi(u) = n(u)$, this means operator $n(u)$ is a gradient operator.

Then based on Theorem (5.1) we can find the functional $\psi : \mathcal{V} \to \mathbb{R}$ such that $\mathfrak{D}\psi(u) = n(u) - m$. This means that solving the problem (2.32) corresponds to the minimization of the functional ψ. And in the context of the subspace correction method, by solving the nonlinear problem $n_k(u_k) = m_k$ in the subspace $\mathcal{V}_k$ using the subspace correction method, we can minimize the functional $\psi_k : \mathcal{V}_k \to \mathbb{R}$ such that

$\mathfrak{D}\psi_k(u_k) = n_k(u_k) - m_k$ for $u_k \in \mathcal{V}_k$. Now the problem can be reformulated to solve the minimization problem as follows,

find $u_k \in \mathcal{V}_k$ such that

$$\min_{u_k \in \mathcal{V}_k} \psi_k(u_k) \tag{5.1}$$

where ψ_k is the *objective function*. Let us define the nonlinear multilevel optimization operator as

$$\begin{aligned} \mathcal{F}_k &: \mathcal{V}_k \times \mathcal{V}_k \to \mathcal{V}_k \\ u_k^{i+1} &= \mathcal{F}_k(m_k; u_k^i) \end{aligned} \tag{5.2}$$

where the operator on level k used to approximate the exact solution u_k^* of $\psi(u)$ (and indirectly solve $n_k(u_k) = m_k$). Then FAS by using the following assumption is a method to find a descent direction (quadratic formula with a damping factor $\alpha(u^*, u)$ [HR89] (Theorem 4.5) with respect to the following assumption (Lemma 4.1 in [HR89])

Assumption 5.2

For $k \in \mathbb{I}$ and $\nu, x, y \in \mathcal{V}_k$ and assuming $\mathfrak{D}\psi(x)(y) \geq 0$ and ζ be such that

$$0 \leq \zeta \leq 2\delta\mathfrak{D}\psi_k(x)(y)[\int_0^1 \mathfrak{D}^2\psi_k(x - t\zeta y)(y)(y)dt]^{-1} \quad \textit{for some } \delta \in [0, 1] \tag{5.3}$$

This assumption imposes an upper bound on the Hessian of ψ. This is a critical point that will be discussed later in this section. In addition to this assumption, there is a damping factor that acts as a *step length* in an optimization method similar to the line search methods. The damping factor introduced here to ensure that the following condition is satisfied by FAS as a multilevel optimization, which guarantees that FAS gives a descent direction,

$$\psi(\mathcal{F}_p(m; u)) \leq \psi(u) \tag{5.4}$$

which means that $\mathcal{F}_p(m; u)$ gives the descent direction. Then the damped nonlinear multilevel algorithm by using a suitable damping factor $\alpha(u^*, u)$ under the assumptions on the relaxation method in each level (Lemma 4.2 [HR89]) and on coarse grid correction (Lemma 4.3 and 4.4 [HR89]) can converge from any starting point (Theorem 4.5 in [HR89]).

Therefore, FAS under Assumption (2.5) is changed to a multilevel line search method for convex problems, in which global convergence can be obtained using a proper damping factor. In order to transform information into different spaces, we need two operators, the prolongation operator $I_H^h : \mathcal{V}_k \to \mathcal{V}_{k-1}$ and the restriction operator $I_h^H : \mathcal{V}_{k-1} \to \mathcal{V}_k$. We can define here the multilevel optimization operator as follows,

$$\begin{aligned} \mathcal{MGO} &: \mathcal{V} \to \mathcal{V} \\ u^{n+1} &= \mathcal{MGO}_J(\psi(u^k), u^n) \end{aligned} \tag{5.5}$$

Having these operators, we can define the MG/OPT algorithm as follows,

Algorithm 5.1 - *MGOpt*

* *If on coarsest space, compute* $u^{i+1} \leftarrow \mathcal{F}_p(m; u^i)$ *otherwise*
  1) *Partially minimize the problem in level H*
  2) *Compute residual as* $\tau \leftarrow \mathfrak{D}\psi_H(u_H) - I_h^H \mathfrak{D}\psi_h(u_h)$
  3) *Apply multilevel recursion* $u_H \leftarrow \mathcal{MGO}_J(\psi(u^k), u^n, \tau_H)$
  4) *Compute the search direction* $e_h \leftarrow I_H^h(u_H - I_h^H u_h)$
  5) *Use the line search to compute* $u_h \leftarrow u_h + \alpha e_h$
  6) *Partially minimize the problem using the local minimizer algorithm*

The algorithm above is the V-cycle multilevel MG / OPT, other forms of multigrid (W-cycle or full multigrid) can also be used [TOS01]. In the new multilevel optimization algorithm introduced above, in Steps 1) and 6) a local minimizer is needed. Therefore, in the FAS-NRASM algorithm, the nonlinear restricted additive Schwarz method works as a local minimization method to minimize ψ_k for a given subspace $\mathcal{V}_k$. In the algorithm the search direction is computed in the coarse level, and under the assumption (5.2) and Lemma 4.2-4.4 in [HR89] a search direction is the descent direction, which guarantees that in each iteration of the multilevel algorithm the approximation of the solution improves, i.e., assuming $0 < \alpha < 2$ we will have

$$\psi(u + \alpha e) - \psi(u) \leq \frac{1}{2}\alpha(\mathfrak{D}\psi(u), e)$$

The next topic we should investigate is studying the performance of FAS as a multilevel approximation method and the factors which have influence on the efficiency of the FAS-NRASM algorithm. The answer to this question is the main goal of the rest of this section. Changing the perspective and looking at FAS as a multilevel optimization method, we can benefit from the rich literature on the optimization of nonlinear problems and have a better picture of the performance of FAS.
First, let us look at the Fréchet derivative of the nonlinear operator $n(u)$ on the exact solution, we have if $\exists A \in \mathfrak{L}(\mathcal{X}; \mathcal{Y})$ a linear operator satisfies the following identity,

$$n(u^*) = n(u) + A(u^* - u) + \|u^* - u\|_{\mathcal{X}}\delta \; if \lim_{u \to u^*} \delta = 0$$

where $\mathcal{X}$ and $\mathcal{Y}$ are two normed vector spaces. Considering $u^* = u + \alpha e$ and using the first-order Taylor expansion of the $n(u)$ on u

$$n(u + \alpha e) = n(u) + \alpha \mathfrak{D}n(u)e + O(e^2, \mathfrak{D}^2 n(u)) \tag{5.6}$$

comparing these two equations, we can realize that $A := \mathfrak{D}n(u)$, i.e., replacing the linear operator A with $\mathfrak{D}n(u)$. Looking at FAS to solve Equation (2.32), we can see

that in each iteration of the FAS algorithm by computing the nonlinear residual (also called *defect* in the literature) as $r = n(u) - m$ we will have

$$\begin{aligned} r &= n(u) - m \\ &= n(u) - m - (n(u^*) - m) \\ &= n(u) - n(u^*) \\ &= \mathfrak{D}n(u)(u - u^*) + O\Big((u - u^*)^2, \mathfrak{D}^2 n(u)\Big) \text{ using (5.6)} \end{aligned} \tag{5.7}$$

This means that in each FAS iteration, we compute the nonlinear residual as $r = n(u) - m$, restrict it into the coarse space, and then we compute the correction in the coarse space which is later used to correct the current approximation of u in the finest space. By denoting $p = u - u^*$ and looking at the last line of (5.7) and comparing to the line search methods, we can also realize that in each iteration of FAS we compute the descent direction in the coarse space, which is computationally less expensive than fine spaces.

Due to the nonlinearity of the problem, it is not straightforward to use the approach used in linear multigrid to analyze the performance of the nonlinear multigrid. On the other hand, looking at the FAS as a multilevel optimization method gives us the potency of using the strategies usually utilized in the analysis of the optimization techniques.

In this respect, Nash and Lewis [NL11] introduced four diagnoses by which we can estimate the performance of FAS as a multilevel optimization method.

1) **Nonlinearity** Most optimization methods, including FAS as multilevel optimization, are based on the quadratic Taylor series approximation of the nonlinear functional. The Taylor expansion of the functional $\psi(u)$ around u is as follows,

$$\psi(u + p) = \psi(u) + \mathfrak{D}\psi(u)p + \frac{1}{2}\mathfrak{D}^2\psi(u)p^2 + O\Big(\|p\|_{\mathbb{X}}^3, \|\mathfrak{D}^3\psi(u)\|_{\mathbb{Y}}\Big) \tag{5.8}$$

where u is the current approximation of the exact solution u^* and p is the descent direction which the multilevel method should compute. Then the difference between the quadratic approximation and the nonlinear functional $\psi(u)$, i.e., the fourth term on the right-hand side of the above equality $\frac{1}{4}\mathfrak{D}^3\psi(u)p^3$, shows the nonlinearity of the problem. If the problem has high nonlinearity, the quadratic Taylor series can not be a good approximation of $\psi(u)$ around u.

2) **Consistency** In multilevel methods, as multigrid or multilevel optimization, in addition to get an improvement of the method on fine grid, solving the coarse problems is computationally cheaper. And there is the temptation to go to the coarsest possible problem. In linear problems we usually get desirable advantages from doing so, but in nonlinear problems the scenario might go in a different direction. Due to nonlinearity of the problem, we can get a benefit from

the coarser problem if the optimization model in the coarser spaces is a good approximation of the problem in the fine space. In the multilevel optimization, we obtain the search direction p_h by applying an optimization method to the coarse problem. Then the question is how good the coarse problem can or should approximate the search direction. The answer is if the coarse problem approximate the fine problem properly, the predicted reduction by the coarse problem should approximate the actual reduction in the fine problem after applying the line search. If we compute these two reductions as

$$\begin{aligned} R_p &= \psi(u_{H,1}) - \psi(u_{H,2}) && \text{predicted reduction} \\ R_a &= \psi(u_{h,1}) - \psi(u_{h,2}) && \text{actual reduction} \end{aligned}$$

where $u_{.,1}$ and $u_{.,2}$ are the solution before and after applying line search. By using Taylor expansion for both reductions we will have [NL11]

$$-R_p = p_H^T I_h^H \mathfrak{D}\psi_h(u_{h,1}) + \frac{1}{2} p_H^T I_h^H \mathfrak{D}^2\psi_H(u_{H,1}) p_H + O(\|p_H\|_{\mathcal{X}}^3)$$

and

$$-R_a = p_H^T I_h^H \mathfrak{D}\psi_h(u_{h,1}) + \frac{1}{2} p_H^T I_h^H \mathfrak{D}^2\psi_h(u_{h,1}) I_H^h e_H + O(\|p_H\|_{\mathcal{X}}^3)$$

then the difference between actual and predicted reduction is

$$R_a - R_p = \frac{1}{2} p_H^T \left[\mathfrak{D}^2\psi_H(u_{H,1}) - I_h^H \mathfrak{D}^2\psi_h(u_{h,1}) I_H^h \right] p_H + O(\|p_H\|_{\mathcal{X}}^3) \tag{5.9}$$

The formula has two components. The first two terms show the difference between the coarse Hessian and restricted fine Hessian. This shows how good the approximation of convexity properties of the $\psi(u)$ in the coarse space. And the last term in (5.9) shows the nonlinearity of the problem. If $R_a - R_p$ is very large, then in contrast to the linear multigrid, the direction computed by coarse space might not be descent direction, and therefore it is not desirable to go to the coarse space.

3) **Complementarity** MG/OPT and multigrid methods have two main component, the smoother (or the underlying minimization method on each level) and the multilevel recursion. As we know in linear multigrid, we demand that the high frequency modes of error be handled by the smoother and the low frequency modes of error by the coarse grid correction. This diagnosis is the generalization of how the smoother and the coarse grid correction in linear multigrid method should complement each other. In nonlinear multigrid as a multilevel optimization method, we look at how the underlying minimization method will complement the multilevel recursion. Then we demand that the component resolved by the minimization method at a level should be distinct from the component treated by the multilevel recursion, and this only possible if these two components are complementary to each other (more on [NL11]).

4) **Separability** To have an efficient multilevel optimization method, the same as multigrid, we demand the component resolved on the fine space must not be influenced by the component of the solution on the coarse space. In order to satisfy this condition, the Hessian must be separable across different levels. If we can write the Hessian as a block matrix like

$$G = \begin{pmatrix} G_{hh} & G_{Hh}^T \\ G_{Hh} & G_{HH} \end{pmatrix}$$

then this condition demands that the off-diagonal components of the block matrix should be small, $G_{Hh} \approx 0$. In the context of multigrid this means that the smoother should work on the high frequency modes of error whereas coarse grid correction on the low-frequency modes. This means none of the components of the multilevel algorithm should add components to the error which were already eliminated by the other component of the algorithm.

There are different challenges in applying these diagnoses in nonlinear multigrid or multilevel optimization methods, for instance lack of prior knowledge of Hessian or having constrained optimization which makes the Hessian complicated [NL11]. However, the main goal of introducing these diagnoses in the current work is to analyze the role and advantage of using the nonlinear restricted additive Schwarz method in the context of FAS as a quadratic approximation of the nonlinear functional $\psi(u)$.

As in Steps 1) and 6) of Algorithm (5.1) we need to minimize the problem at a given level; therefore, we need a proper local minimizer method. Then the NRAS method works here as a minimization algorithm. It is necessary to see how and when NRAS works in this context. Here are two important things to investigate. First, if the subspace problem (2.42) is equivalent to the minimization of $\psi(u - p_\kappa\delta_\kappa)$ at a given level. The second is if $\delta = \sum_{\kappa\in\mathbb{I}} p_\kappa\delta_\kappa$ is the global descent direction in a given level. The local minimization problem in each subspace $\mathcal{V}_\kappa$ will be

find $\delta_\kappa \in \mathcal{V}_\kappa$ such that

$$\psi(u - p_\kappa\delta_\kappa) = \min_{\delta'\in\mathcal{V}_\kappa} \psi(u - p_\kappa\delta') \tag{5.10}$$

In this respect, some assumptions are needed. The main assumption, we demand, is that the nonlinear operator $n(u)$ should be convex in $B(u^*; r)$ and must satisfy Assumptions (2.9). Hackbusch and Reusken [HR89] introduced a globally convergent algorithm in which each step is the minimization of the local problem. The algorithm is as follows,

Algorithm 5.2 *NRASM as a local minimization*

i) $u := u^i$

ii) if $u \in B(u^*; r)$*, solve (2.42) and* $\delta = \sum_{\kappa\in\mathbb{I}} p_\kappa\delta_\kappa$

iii) if $u \notin B(u^*; r)$*, solve (5.10) and compute* $\delta = \sum_{\kappa \in \mathbb{I}} p_\kappa \delta_\kappa$ *and*

iv) by using damping factor λ*, compute* $u^{i+1} = u^i + \lambda\delta$ *which gives* $\min_{u \in \mathcal{V}_k} \psi(u - \lambda\delta)$

In the algorithm, as we also pointed out, we need to choose a proper value for λ. In order to have a globally convergent algorithm, λ should satisfy Condition (5.4) (One possible way to choose λ is by using the Wolfe condition or backtracking).

So far, we saw that in order to have FAS as an efficient multilevel optimization, we need two critical assumptions, Assumption (5.2) and diagnosis 1 on the nonlinearity of $n(u)$ on u. In Chapter (2), it was shown that the NRAS method is a proper smoother for FAS, and in this chapter, a suitable local minimizer. In addition to these two properties, NRASM can also be used in another manner to relax the two assumptions mentioned above. As explained, the nonlinearity of the problem is a critical diagnosis, which consistency also depends on it (cf (5.9)). This means that if the problem has a high nonlinearity, FAS might not work well or not work at all. If we look carefully at the Taylor expansion of the nonlinear operator $n(u)$ around the current approximation u and considering the third term on the right-hand side we will have

$$n(u^*) = n(u) + \mathfrak{D}n(u)(u^* - u) + \frac{1}{2}\mathfrak{D}^2 n(u)(u^* - u)^2 + O\Big(\|\mathfrak{D}^3 n(u)\|_{\mathcal{Y}}, \|(u^* - u)^3\|_{\mathcal{X}} \Big) \tag{5.11}$$

Comparing this approximation with (5.6) and (5.7), we can see that $\mathfrak{D}^2 n(u)(u^* - u)^2$ tells us how nonlinear $n(u)$ in the current approximation is. This term has two main components, the second derivative of the operator $n(u)$ and the difference between the current approximation and the exact solution. If the third term on the right-hand side is large, we will have a poor approximation of n at u. Then the approximation in (5.6) can be improved by two ways; reducing the third term in (5.11) by reducing either $(u^* - u)$ or $\mathfrak{D}^2 n(u)$, resulting in a reduction of the nonlinearity of the problem (as we can see the second derivative of $n(u)$ is also evaluated on the current approximation, it is difficult to have prior knowledge of the behavior of the third term in (5.11)). The second way is to have a higher order approximation in FAS. However, computing the second derivative of $n(u)$ is usually expensive. Therefore, we usually use an approximation of the term $\mathfrak{D}^2 n(u)(u^* - u)^2$ [YD06].

In the current project, we aim for the first remedy and look for a way to reduce the second component of the third term, i.e., $(u^* - u)$ by providing a better initial guess $u_0 \in B(u^*; r_0)$ for small enough r_0 for FAS. One way to obtain a better initial guess is to use a right preconditioner. If we define a right preconditioner operator as

$$\mathcal{G} : \mathcal{V} \to \mathcal{V}$$
$$u := \mathcal{G}(u)$$

which can be used before starting the FAS iteration and provides a proper initial guess for the FAS algorithm. This is similar to *Nonlinear Elimination* (NE) method [LKK18; Liu+22; CL11]. The method NE was generally introduced to improve the performance of the inexact Newton's method by providing a better initial guess and removing the "bad components" which are associated with high nonlinearity. The idea is to decompose the domain into subdomains, restrict the nonlinear operator to the subspace $\mathcal{V}_i$ as

$$n_i(u) = R_i n(u)$$

and iterate on the local problem $n_i(u) = m_i$ until, by providing a better approximation of the solution, the nonlinearity of the problem reduces, and we have more homogeneous nonlinearity in problems with heterogeneous nonlinearity (such as turbulent flow or fracture). Then in order to evaluate the nonlinearity, we need to compute or approximate the nonlinear term. The nonlinear term $\mathfrak{D}^2 n(u)(u^* - u)^2$ is expensive to compute, therefore we can look at the following inequality in the subdomains for a constant ρ

$$\|n_i(u)\| \leq \rho \|n(u)\| \tag{5.12}$$

and in any subdomain, if the inequality is not satisfied, a few NRAS iterations are then applied to obtain the condition in the inequality. Another adaptive way to approximate nonlinearity is also introduced in [Liu+22] in which in the subdomains where we have a highly nonlinear problem do some nonlinear additive Schwarz method iterations until the nonlinearity is below a desired value. This provides a better initial guess u for the FAS algorithm. This can also be useful when we have a very heterogeneous or high nonlinearity in a domain where the nonlinearity is due to large $\mathfrak{D}^2 n(u)$ and then NE can help to isolate some part of the problem with high nonlinearity, the bad component, and apply the NRAS iteration to reduce the nonlinearity. Then the first and second diagnosis can be satisfied. Then we can redefine Algorithm (3.1) by adding NE to reduce the nonlinearity of the problem. Finally, the algorithm NE-NRASM-FAS can be defined as follows,

Algorithm 5.3 - *NE-FAS-NRAS*

i) NE iteration: check the condition (5.12) in subdomains and apply NRAS method on subdomains which does not satisfy the condition

ii) FAS Iteration: apply FAS Algorithm (3.1)

There is a remark that needs to be mentioned here.

Remark 5.1 *First, the pre-smoothing step in the first iteration of the FAS algorithm might not be necessary to apply, since in the NE iteration the current approximation is smoothed. However, the post-smoothing step is still necessary to remove the high-frequency mode of error introduced during the prolongation process.*

The numerical results of applying Algorithm (5.3) is presented in the following section.

5.2. Result

In order to test Algorithm (5.3) in practice, here I mainly focus on the Allen-Cahn equation. As explained in Section (4.1.2), the energy functional in the implicit Euler method $E_{IM}(.;.)$ is convex if the condition $\tau < \epsilon^2$ is satisfied, otherwise the discrete form of the energy functional is non-convex. The main idea here is to make the problem more difficult to be solved using the FAS-NRAS method until the point where even applying the NE algorithm as a preconditioner, the FAS cannot converge to the solution. For this purpose, two main parameters are changed to observe the behavior of the algorithm; first, the time step τ increases to reasonable large values in order to make the energy functional non-convex, and the second parameter is changed here so that instead of using the solution in the previous time step as an initial guess for the FAS algorithm, some random numbers $rand(0, \hat{n})$ are used. The $rand(0, \hat{n})$ means that a random number between 0 and $\hat{n}$, it can be seen for $rand(0, 100)$ in Figure (5.1). In both scenarios, the values τ and the range of random numbers increase to the point where the NE-FAS-NRAS method fails to converge. In the following, different combinations of the values for these two parameters will be studied, and some results are presented that can be useful to understand the behavior of the NE-FAS-NRAS algorithm. Just a quick remark that by using the solution in the previous time step, the norm of the initial nonlinear residual is $\|F(u_0)\| \approx 10^{-6}$ ($F(u) = n(u) - m$) and we consider it as a *reference nonlinear residual*, denoted by $\|\tilde{F}(u_0)\|$, to which the norm of nonlinear residuals in the following experiments are compared. This is important since by changing any of the parameters above, the order can increase remarkably, and then observing the development of each component of Algorithm (5.3) can help to understand the behavior of the nonlinear multilevel optimization technique.

In the experiments, ν_{NE} iterations of the nonlinear elimination method is used as right preconditioner is used before going to the second step of the NE-FAS-NRAS algorithm, the V-Cycle FAS with ν_1 number of pre-smoothing and ν_2 number of post-smoothing . In all tests, the relative *tol* for iterative methods in the NE-FAS-NRAS algorithm can be seen in Table (5.1). As a remark, since the NRAS method can be

Table 5.1.: relative *tol* in the iterative methods in NE-FAS-NRAS algorithm

GMRES	Newton	NRASM	FAS
10^{-12}	10^{-10}	10^{-9}	10^{-8}

used as a minimizer in multilevel optimization or smoother in the nonlinear multigrid,

we can use it interchangeably in this experiment.

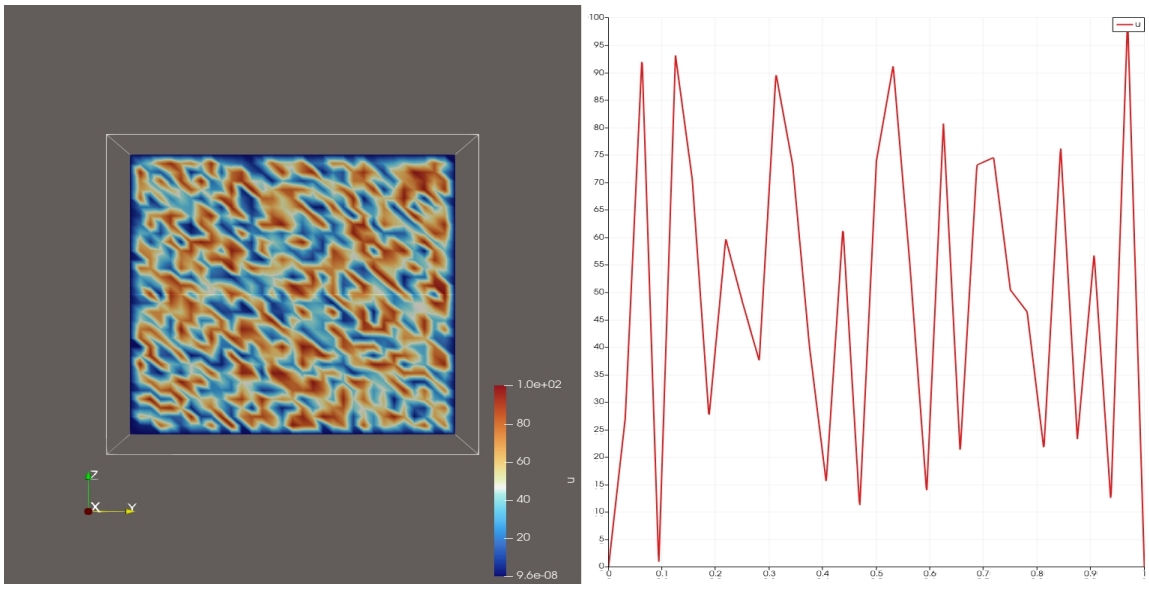


Figure 5.1.: Random initial guess $rand(0, 100)$ on a 2D slice $(1, 0, 0)$ in middle of the cube on the left and on the line $(0, y, 0)$ on the right

Before increasing the time step to the value that makes the energy functional E_{IM} become non-convex, the impact of increasing the range of random numbers is studied. The results can be seen in Figure (5.2). By increasing the range of initial guess values, the initial $\|F(u_0)\|$ increases and in order to get convergence with V-Cycle FAS, we need a few preconditioner steps and then apply V-Cycle FAS to converge to the solution. In the case with $rand(0, 10)$, the order of $\|F(u_0)\|$ increases by 3 orders of magnitude compared to the case where the solution of the previous time step is used as the initial guess, and then we need to use $\nu_{NE} = 5$ number of preconditioning, and finally with $\nu_1 = \nu_1 = 2$ pre- and post-smoothing, after 3 iterations of V-Cycle FAS the convergence is achieved. Increasing the range of random number to $rand(0, 100)$, the norm of the nonlinear residual again increases by 3 orders of magnitude compared to the previous test case, and the number of preconditioner iterations necessary to get the convergence in FAS increases to $\nu_{NE} = 8$, and then with the same number of pre- and post-smoothing steps as previous, the method converges in 3 iterations of V-Cycle FAS. In the case $rand(0, 1000)$, the scenario changes. Since the initial $\|F(u_0)\|$ is 10^{10} times larger than $\|\tilde{F}(u_0)\|$, we can see in the plot (c) in Figure (5.2) even with $\nu_{NE} = 500$ the norm of residual reduces to $\|F(u_0)\| = 702$ and the further reduction cannot be obtained by applying more iterations of the preconditioner, and starting FAS iteration, the nonlinear residual increases sharply and the FAS algorithm diverges.

In the next step, τ is increased to make the discrete energy functional E_{IM} and

the problem non-convex. In each step, τ increases by a factor of 10, which means $\tau = 0.01, 0.1$ and in the last step $\tau = 0.4$. The larger time steps are not reasonable since the truncated error in time discretization is large, in particular for a first-order method like implicit Euler. And in each time step, the same initial guesses as above are used.

Increasing the time step to $\tau = 0.01$, with an initial guess of $rand(0, 100)$, the norm of nonlinear residual increases by order of 10^9 (see Figure (5.3)) and to get the convergence, we need $\nu_{NE} = 8$ preconditioning step and $\nu_1 = \nu_2 = 2$ pre- and post-smoothing steps. Increasing the initial guess to $rand(0, 1000)$, $\|F(u_0)\|$ increases by a factor of 10^{12} compared to $\|\tilde{F}(u_0)\|$, and similar to the previous time step with the same range of random numbers, the nonlinear residual diverges even with $\nu_{NE} = 500$ iteration of the NRAS method as a preconditioner. These two cases ($\tau = 0.001, 0.01$ with initial guess $rand(0, 1000)$) show that if initial guess is far from the solution, which makes the second term on the right-hand side of Equation (5.7) very large, i.e., the nonlinearity becomes very large, and then even with NE method, FAS cannot converge to the solution, which is in agreement with the first and second diagnoses introduced in the previous section.

Then in order to study how large the time step can be to get the convergence for the Allen-Cahn equation as a non-convex problem, the time step increases by a factor of 10 ($\tau = 0.1$) and the three ranges of the random numbers are used as before. With an initial guess of random number $rand(0, 10)$ with $\nu_{NE} = 8$, $\nu_1 = \nu_2 = 3$, 4 iteration of V-Cycle FAS is enough to obtain the convergence (in this case $\|F(u_0)\| = 0.9$). Increasing the range of random numbers to $rand(0, 100)$ the norm of initial nonlinear residual increases to $\approx 10^3$ and then there is a necessity of having more preconditioner iterations. This necessity can be seen in Figure (5.4). With only $\nu_{NE} = 10$ iterations of the NRAS method as a preconditioner, the reduction of the nonlinear residual stalls (plot (a) on the left). Increasing the number of preconditioner iterations to 20, convergence can be achieved by 3 iterations of V-Cycle FAS with $\nu_1 = \nu_2 = 5$ smoothing steps.

And finally, in the last step, the time step increases to $\tau = 0.4$ and the experiment is repeated with the same random numbers. In cases with random numbers $rand(0, 100)$ and $rand(0, 1000)$, we cannot obtain convergence. The main reason for this behavior is that the initial guess is far away from the solution, which makes the $\|F(u_0)\|$ very large (for example in $rand(0, 1000)$, the norm on the initial nonlinear residual is 12 orders of magnitude larger than the reference value. For example, in the case with an initial guess of $rand(0, 100)$ even having $\nu_{NE} = 500$, $\|F(u)\|$ stalls around 0.7 and does not decrease further. But with a random number with a smaller range we can get the convergence (Plots (a) and (b)). Therefore, it shows the robustness of the algorithm NE-FAS-NRAS by which even by very large time step size (≈ 400 larger than time step size with convex energy functional) and initial guesses far from the solution, the method can converge to the correct solution.

In addition to studying the reduction of the norm of the nonlinear residual, look-

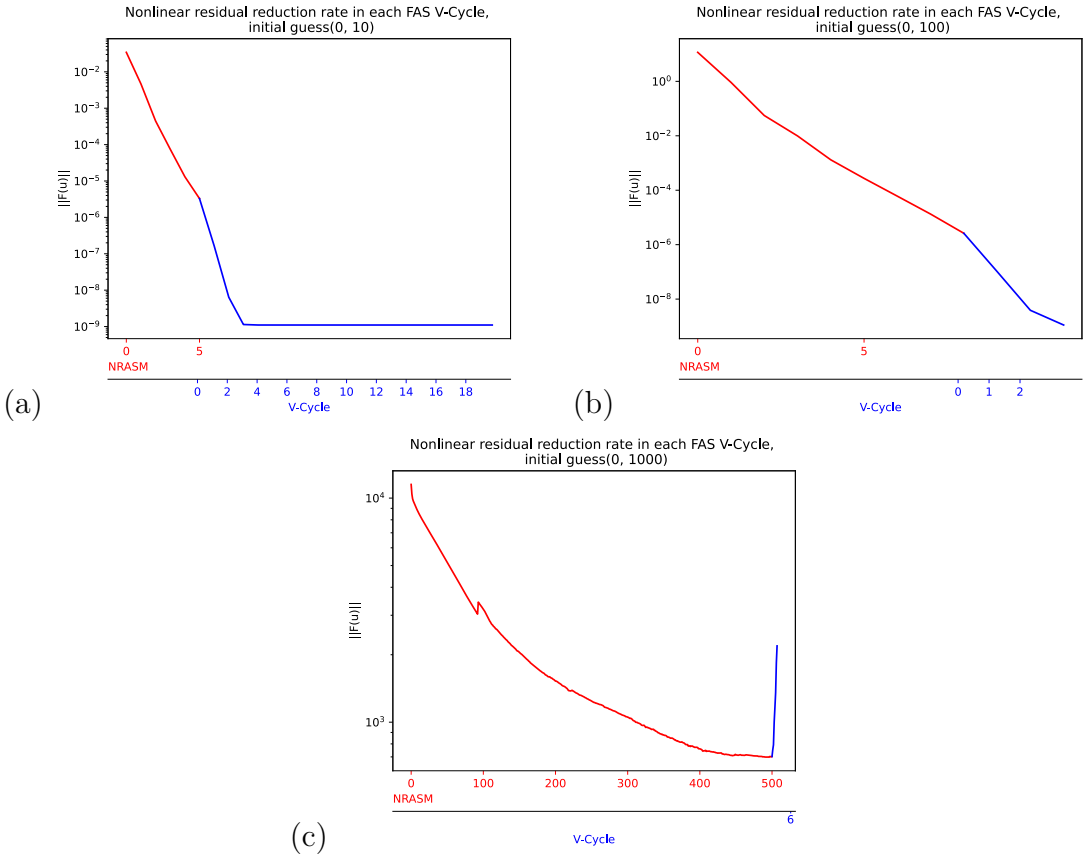


Figure 5.2.: Reduction of $\|F(u)\|$ by applying NRASM (red) as NE and V-Cycle FAS (blue) for $\tau = 10^{-3}$; (a) $rand(0, 10)$ with $\nu_{NE} = 5$ $\nu_1 = \nu_2 = 2$ (b) $rand(0, 100)$ with $\nu_{NE} = 8$ $\nu_1 = \nu_2 = 2$ (c) $rand(0, 1000)$ $\nu_{NE} = 500$ $\nu_1 = \nu_2 = 10$

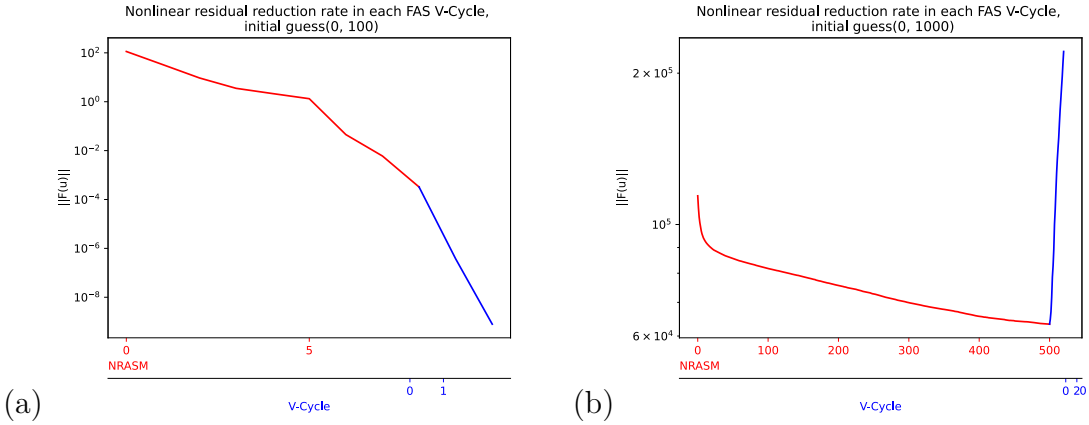


Figure 5.3.: Reduction of $\|F(u)\|$ by applying NRASM (red) as NE and V-Cycle FAS (blue) for $\tau = 10^{-2}$; (a) $rand(0, 100)$ with $\nu_{NE} = 8$ $\nu_1 = \nu_2 = 2$ (b) $rand(0, 1000)$ $\nu_{NE} = 500$ $\nu_1 = \nu_2 = 10$

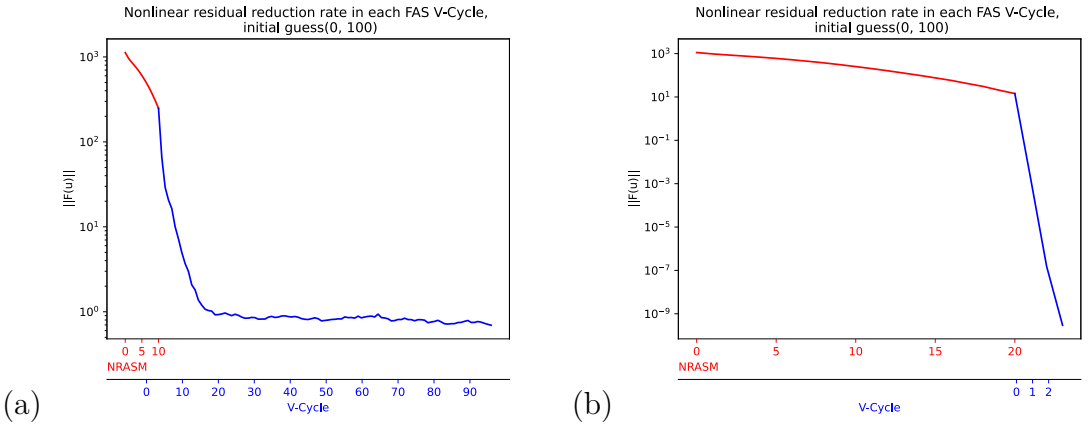


Figure 5.4.: Reduction of $\|F(u)\|$ by applying NRASM (red) as NE and V-Cycle FAS (blue) in $\tau = 10^{-1}$; (a) $\nu_{NE} = 10$, $\nu_1 = \nu_2 = 5$ (b) $\nu_{NE} = 20$, $\nu_1 = \nu_2 = 5$ with initial guess $rand(0, 100)$

ing at the behavior of the solution can also be insightful. Figure (5.6) shows the progress of the solution for the scenario with the time step size $\tau = 0.1$ and initial guess of a random number $rand(0, 100)$ and how u evolves in the different stages of the algorithm. The initial guess can be seen in plot (a). After applying 20 steps of the NRAS method as a preconditioner, the solution in a slice in the middle of the domain shows the values of the solution u, which is now much smoother. The change in the solution is more visible in the plot (c), which is on the line $(0, y, 0)$, although u has some fluctuation after applying preconditioner, compared to the solution in the initial guess, is smoother and the amplitude of fluctuations is also reduced. Then in the first V-Cycle FAS (plot (d)), there is still fluctuation in the left-bottom, but u has the general form of the solution. And finally, after 3 iterations of the V-Cycle FAS, u converges to the correct solution.

It is also useful to look at how the nonlinear residual evolves in the different steps of the NE-FAS-NRAS algorithm. This is what we can see in Figure (5.7). In particular, comparing the scenarios of the convergent and non-convergent experiments can give us a useful picture of the behavior of the NRAS as a preconditioner and FAS as the main solver/optimization technique. The time step and the range of a random number are the same as the previous case ($\tau = 0.1$ and $rand(0, 100)$). Similar to the solution, we first look at the initial nonlinear residual (plot (a) in Figure (5.7)). As expected, $F(u_0)$ is random and extremely heterogeneous. Then we look at the nonlinear residual after 10 iterations of the NRAS method as a preconditioner (plot (b)) where the numerical simulation does not get convergence and the simulation with a convergent result (plot (c)) in FAS with $\nu_{NE} = 20$. In the convergent scenario, $F(u_0)$ reduces from the order 10^0 to 10^{-2} whereas in the non-convergent simulation, of one orders of magnitude larger. Looking at the nonlinear residual on a line in the middle of the cube $((0, y, 0))$, it shows that the NRAS method as a preconditioner

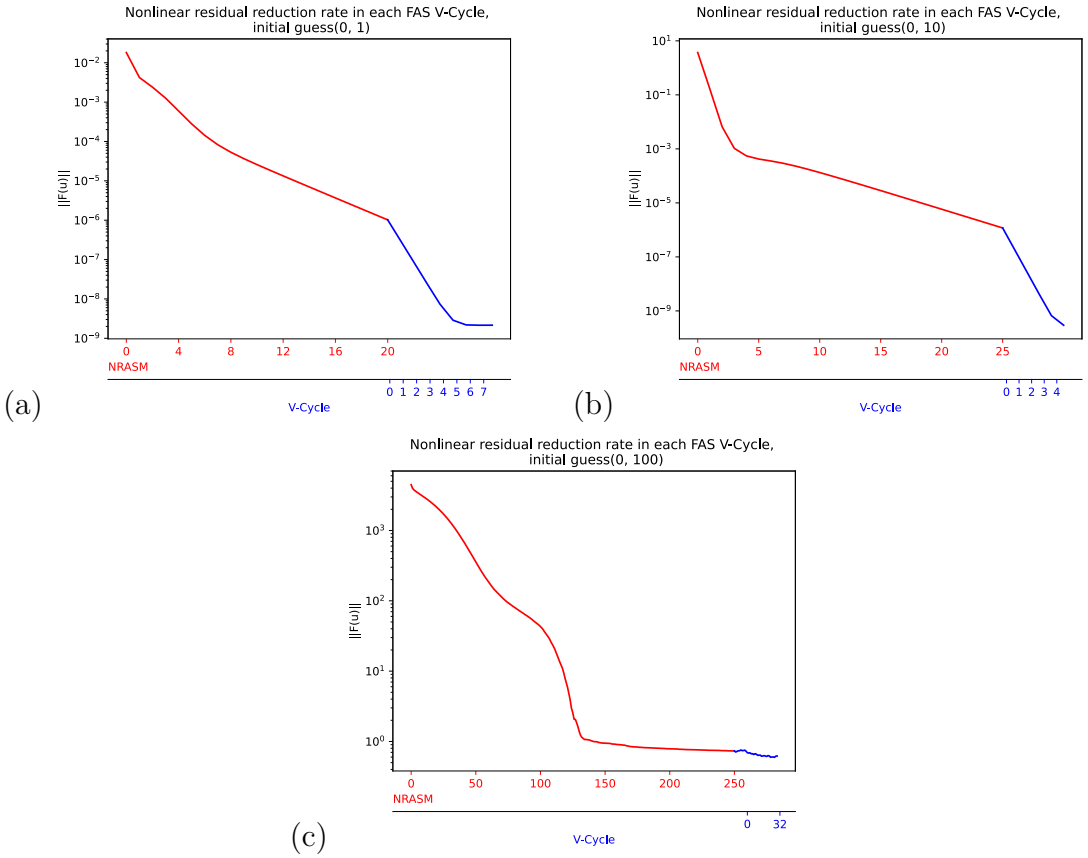


Figure 5.5.: Reduction of $\|F(u)\|$ by applying NRASM (red) as NE and V-Cycle FAS (blue) in $\tau = 4\times10^{-1}$; (a) with initial guess $rand(0,1)$ $\nu_{NE} = 20$, $\nu_1 = \nu_2 = 3$ (b) with initial guess $rand(0,10)$ $\nu_{NE} = 25$, $\nu_1 = \nu_2 = 5$ with initial guess $rand(0,100)$ $\nu_{NE} = 250$, $\nu_1 = \nu_2 = 20$

reduces $F(u)$ and makes the nonlinear residual more homogeneous. This is what is expected from the nonlinear elimination method as a right preconditioner [Liu+22]. Even though with $\nu_{NE} = 10$ the nonlinear residual is smaller and more homogeneous, it is still not a proper starting point to apply the FAS algorithm. And after making the nonlinear residual small enough and properly homogeneous (after $\nu_{NE} = 20$) FAS can work. This is what was explained in the first and second diagnosis in Section (5.1). By applying the NRAS method as a right preconditioner, the second term on the right-hand side of identity (5.7) shrinks and, therefore, the nonlinearity of the problem is reduced and, as a result, FAS can work as a multilevel optimization method. In the end, in plot (e) in Figure (5.7), the reduction of the nonlinear residual can be observed by applying three iterations of V-Cycle FAS.

All the results have been presented so far are with 5 steps mesh refinement. It is also useful to look at a larger problem size and the behavior of the NE-FAS-NRAS algorithm for a larger problem. To test the algorithm in this problem size, the two

largest time step sizes are considered ($\tau = 10^{-1}, 4 \times 10^{-1}$) and $rand(0, 1)$ as an initial guess. As we can see in Figure (5.8), for $\tau = 10^{-1}$ we need to have $\nu_{NE} = 8$ preconditioner steps and with $\nu_1 = \nu_2 = 3$ pre- and post-smoothing steps. Following by 3 iterations of V-Cycle FAS, the convergence is obtained. This is a similar behavior observed with the problem size of 5 steps of mesh refinement. In the larger time step, $\tau = 4 \times 10^{-1}$, the number of preconditioner increases to 20 and again with the same number of pre- and post-smoothing, the convergence is obtained after 5 iterations of V-Cycle FAS. Compared with the same scenario but with 5 steps of mesh refinement, in case $\tau = 10^{-1}$, we need the same number of iterations of V-Cycle FAS and for $\tau = 4 \times 10^{-1}$, we need one more iteration of V-Cycle, which is not unexpected, since the initial nonlinear residual for starting the V-Cycle iteration in problem with 6 steps of mesh refinement is one order of magnitude larger than the problem with 5 steps of mesh refinement. Therefore, it is fair to conclude that the number of nonlinear multigrid is independent of the size of the problems.

As explained in Section (4.3.2), in order to evaluate the accuracy of the numerical method in solving the Allen-Cahn equation, the evolution of the radius of the sphere in the solution u is compared with the analytical formula (4.14). The radius of the sphere for four time steps $\tau = 10^{-3}, 10^{-2}, 10^{-1}, 4 \times 10^{-1}$ of the problem size of 5 steps of mesh refinement on the left, and two time step sizes $\tau = 10^{-1}, 4 \times 10^{-1}$ for 6 steps of mesh refinement on the right, can be seen in Figure (5.9). In 5 steps of mesh refinement, in all time steps except $\tau = 4 \times 10^{-1}$ the random initial guess $rand(0, 100)$ is used. The red-color font in the legend of the figure denotes non-convexity of the energy functional E_{IM}. The lines corresponding to $\tau = 10^{-3}$ and 10^{-2} follow the same path and are very close. Increasing the time step to $\tau = 10^{-1}$ changes the behavior of the curve, at the beginning, the radius opposite to the previous cases shrinks faster than the analytical radius, but then around $t = 1.1$ the behavior changes and becomes slower. And in the case with time step $\tau = 4 \times 10^{-1}$ the radius deviates further, and it moves faster than the analytical radius. One reason for this deviation can be the error stems from the large truncation error in time discretization due to the larger time step to $\tau = 4 \times 10^{-1}$ (the truncation error has the order of $O(\tau^1)$, as explained in Equation (4.16) in the previous chapter). But in general in all the time steps, the behavior of the radius of the sphere are reasonable, and the deviation are acceptable. In the case with larger problem size on the right, the initial guess for both test cases are $rand(0, 1)$. The numerical results in these cases have also good agreements with the radius computed by analytical formula. And similar to previous problem size, in the larger time step size, we have more deviation from the analytical result. This shows that using the NRAS method as a preconditioner-smoother (or preconditioner-minimizer) can help FAS work in a non-convex problem with poor initial guess in the Allen-Cahn equation.

Here, it is also useful to mention the execution time to the final solution by using Algorithm (5.3). Using NRASM as a preconditioner-smoother with the largest time step size ($\tau = 0.4$), the execution time to compute the final solution of the Allen-

Cahn equation in the current experiment is reduced by factor ≈ 250 compared to the experiment in Chapter (4). Although the time step size is 400 times larger, it is necessary to apply NRASM as a preconditioner; as a result, only a speed-up of a factor of ≈ 250 can be achieved. Solving the problem on a GPU (A100 in Table (3.1)) compared to a CPU (Intel Core i7-10510U CPU @ 1.80GHz and RAM) , a speed-up can also be achieved by a factor of ≈ 160.

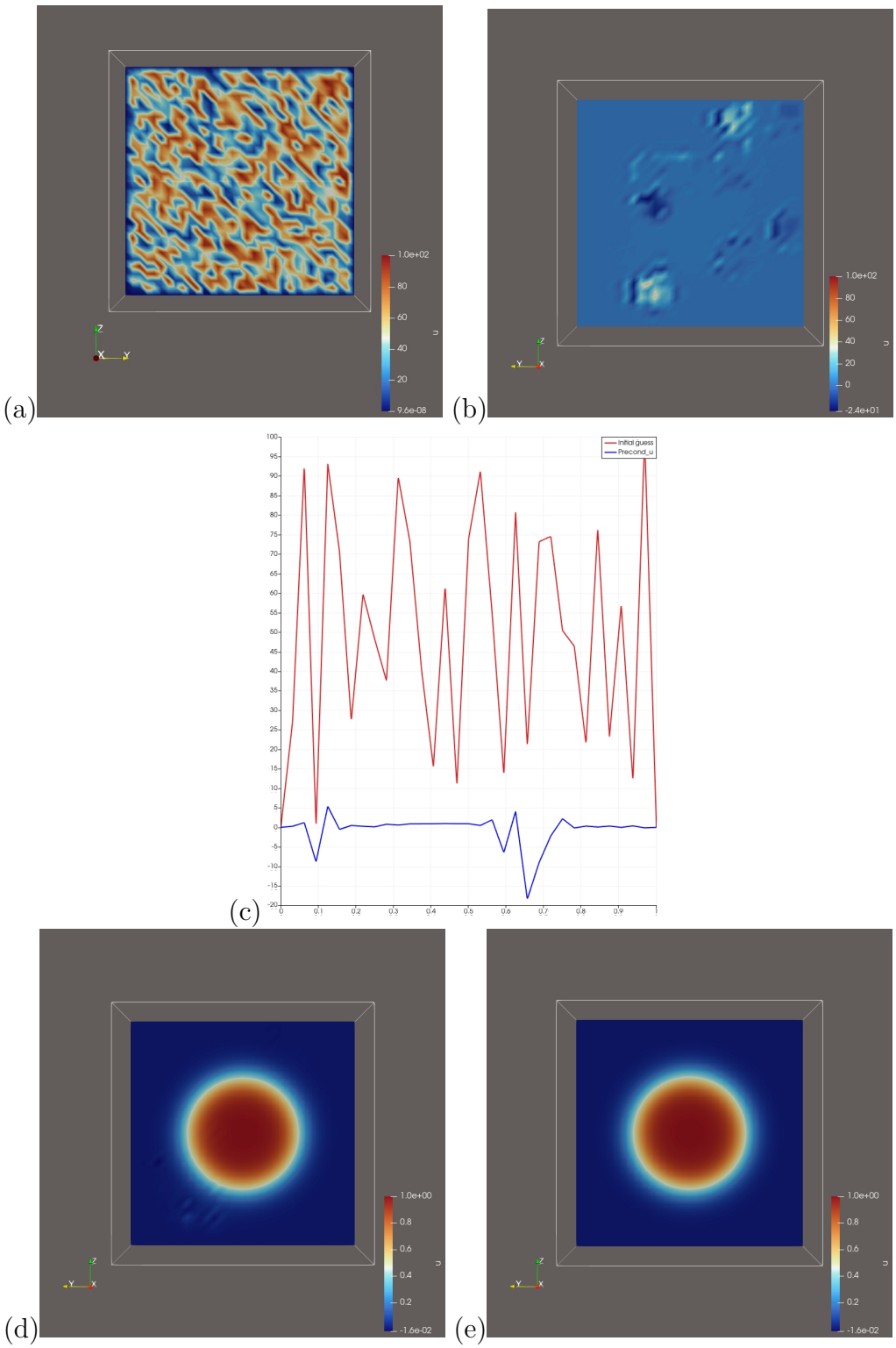


Figure 5.6.: (a) Initial guess $rand(0, 100)$ with $\nu_{NE} = 20$, $\nu_1 = \nu_2 = 5$ (b) After applying 20 iteration of NRASM (c) Initial value and preconditioned values, plot over a line in $(0, y, 0)$ (d) After first V-Cycle FAS (e) The solution after convergence by 3 iterations of V-Cycle FAS

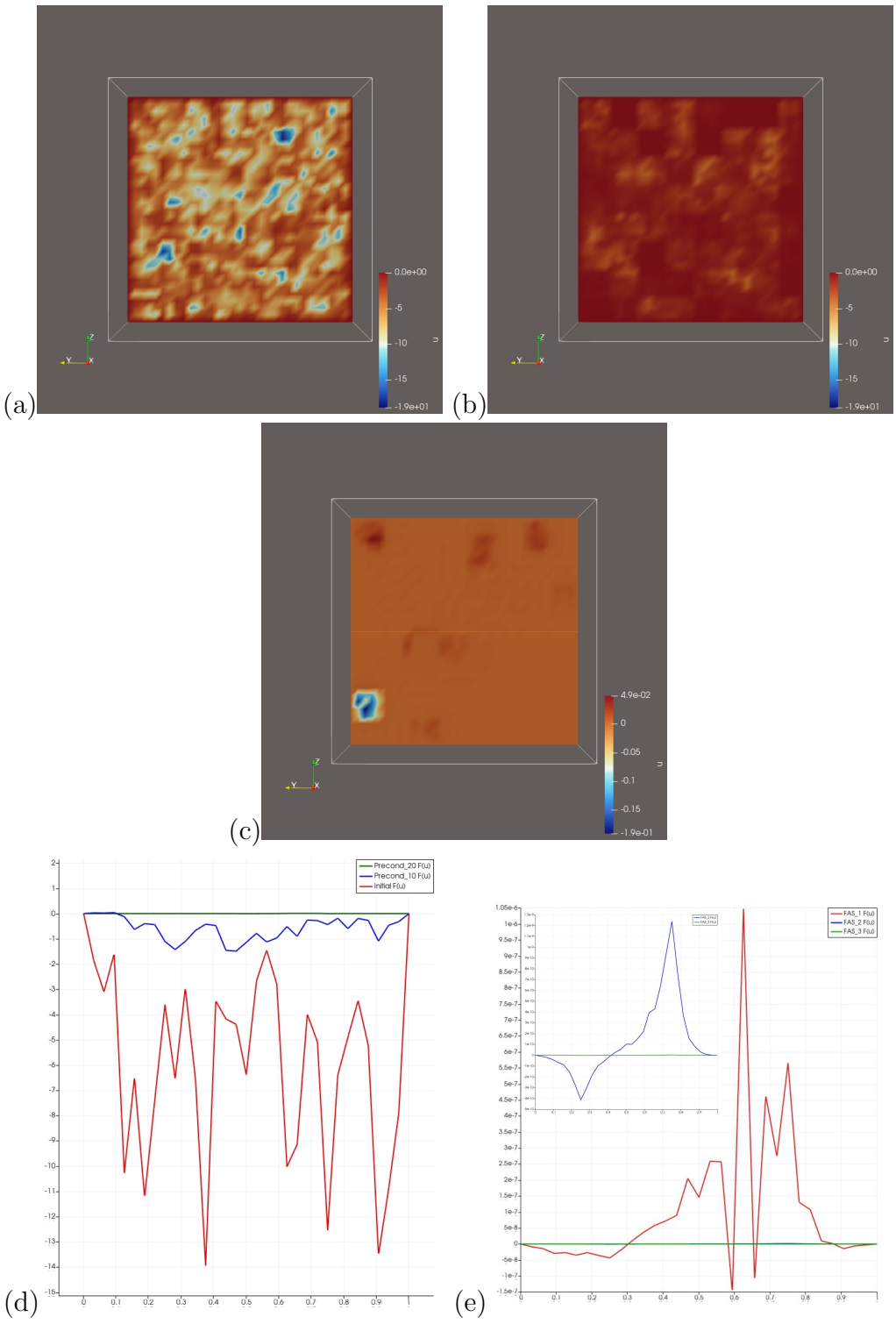
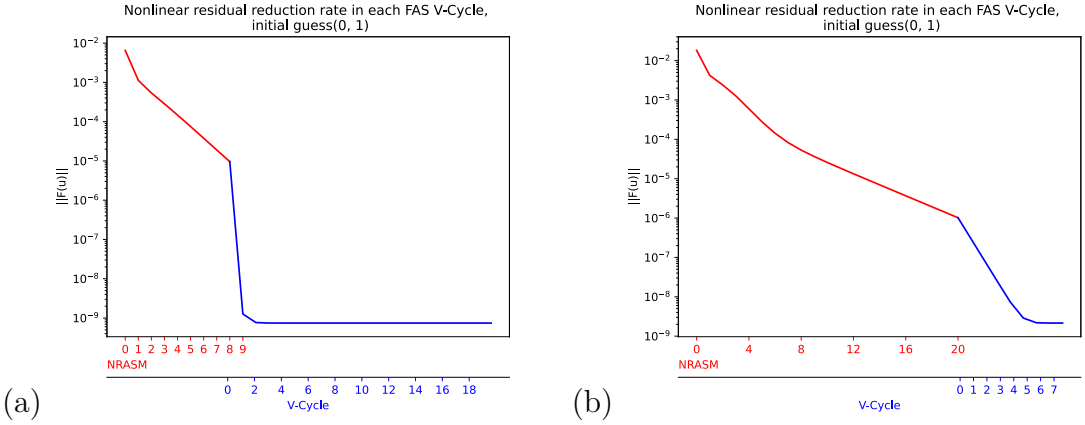


Figure 5.7.: (a) Initial $F(u_0)$ $rand(0, 100)$ with (b) After applying 10 iteration of NRASM as NE method(c) After applying 20 iteration of NRASM as NE (d) initial $F(u_0)$, and after applying ν_{NE} = 10 and ν_{NE} = 20 on line $(0, y, 0)$ (e) $F(u)$ after 3 number of V-Cycle FAS iteration pre- and post-smoothing $\nu_1 = \nu_2 = 5$



(a) (b)

Figure 5.8.: Reduction of $\|F(u)\|$ for 6 steps of mesh refinement by applying NRASM (red) as NE and V-Cycle FAS (blue) with initial guess $rand(0,1)$; (a) $\tau = 10^{-1}$ and $\nu_{NE} = 8$, $\nu_1 = \nu_2 = 3$, $\tau = 4 \times 10^{-1}$ and $\nu_{NE} = 20$, $\nu_1 = \nu_2 = 3$

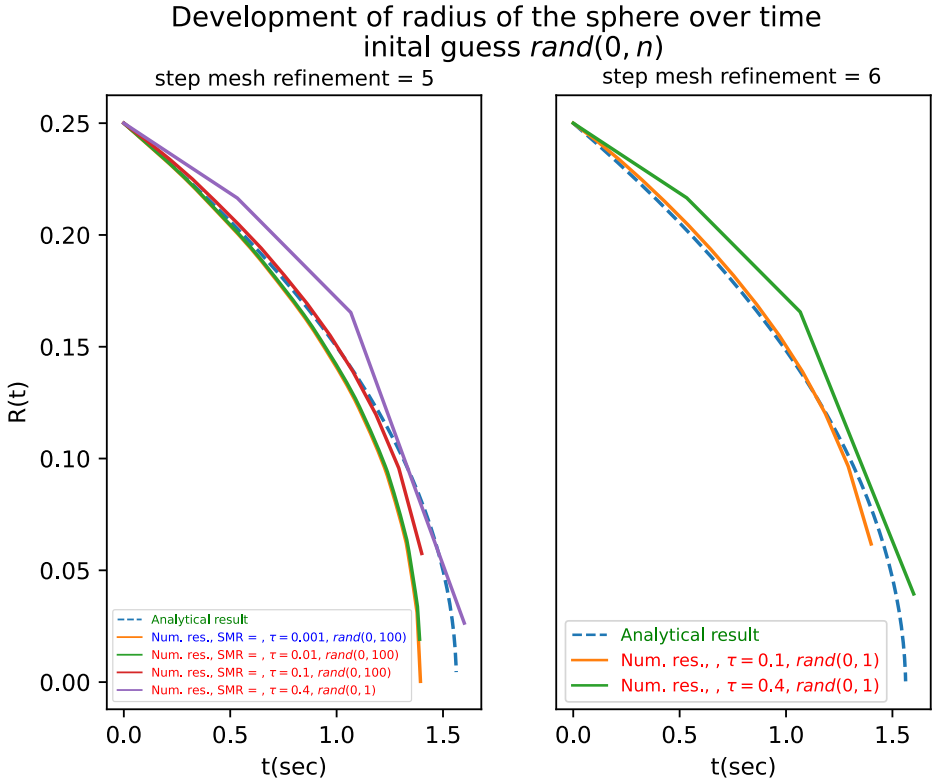


Figure 5.9.: development of the radius of sphere over time ; (blue) convex energy functional, (red) non-convex energy functional $E_{IM}(.,.)$, in $\tau = 0.4$, random number is $rand(0,1)$ for the rest $rand(0,100)$; left with 5 steps mesh refinement, right 6 steps mesh refinement

6. Conclusion

In this thesis, a numerical algorithm was developed to solve the Allen-Cahn equation as a nonlinear parabolic PDE. The algorithms used to solve the equation are two forms of subspace correction methods, full approximation scheme, a successive subspace correction method, as the main solver and nonlinear restricted additive Schwarz method, a parallel successive subspace correction method, as a smoother for FAS. The first challenge in the algorithm is whether the NRAS method is a proper smoother for FAS, i.e. whether it satisfies the smoothing condition in the nonlinear multigrid method. This question was answered at the end of Chapter (2) and it is proved that NRASM is a proper smoother for FAS. As mentioned, the main goal of the thesis is to solve the Allen-Cahn equation numerically on GPU. In this regard, in the second chapter, numerical methods (FAS and NRASM) were explained in more detail from implementation point of view, and also local nonlinear solver, globally convergent inexact Newton's method, and linear solver, GMRES were discussed. In GPU having a matrix-free implementation is more desirable; therefore, these methods were explained for having matrix-free form on GPU. The result of applying the algorithm to the Allen-Cahn equation, in Chapter (4), shows that using Strategy 2, in Figure (4.2), the number of iterations of V-Cycle FAS is independent of the size of the problem to solve the Allen-Cahn equation. This also demonstrates that the NRAS method is a proper smoother for FAS and complements the other component of FAS, i.e., the coarse grid correction operator. All the results in Chapter (4) are based on the fact that FAS works on a convex problem; therefore, the discretized Allen-Cahn equation should be convex. This fact imposes constraints on size of time step in both backward Euler and Crank-Nicolson, which are used in the current project as time schemes.

Looking at FAS as a multilevel optimization method provides a new perspective to analyze the nonlinear multigrid. In this respect, in addition to the convexity condition in the discretized problem, the nonlinearity of the equation is also another critical factor that should be considered when FAS is used as the main numerical solver. Combining FAS and NRASM gives another potency to relax the necessary of convexity of the problem solved by FAS and also the problem can have higher nonlinearity. Using NRASM as a preconditioner-smoother in the algorithm NE-FAS-NRAM, the problem with higher nonlinearity and also non-convexity can be solved iteratively, this is what was shown, analytically and numerically, in Chapter (5). Therefore, the size of the time step in solving the Allen-Cahn equation can be much larger (≈ 400 larger than the case of a convex discretized Genzberg-Landau energy functional). And also by using random numbers with as an initial guess, it can be observed that the algorithm NE-FAS-NRAM can handle higher nonlinearity. Although in solving the Allen-Cahn

equation, the NE-FAS-NRAM algorithm fails to converge for very poor initial guesses, such as random numbers $rand(0, 1000)$, it is very robust for some moderate range of random numbers but for large time steps.

Bibliography

[AC79a] Samuel M Allen and John W Cahn. "A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening". In: *Acta metallurgica* 27.6 (1979), pp. 1085–1095.

[AC79b] Samuel M Allen and John W Cahn. "A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening". In: *Acta metallurgica* 27.6 (1979), pp. 1085–1095.

[BBH09] Peter W Bates, Sarah Brown, and Jianlong Han. "Numerical analysis for a nonlocal Allen-Cahn equation". In: *Int. J. Numer. Anal. Model* 6.1 (2009), pp. 33–49.

[BD85] Randolph E Bank and Craig C Douglas. "Sharp estimates for multigrid rates of convergence with general smoothing and acceleration". In: *SIAM journal on numerical analysis* 22.4 (1985), pp. 617–633.

[BGN70] Billy L Buzbee, Gene H Golub, and Clair W Nielson. "On direct methods for solving Poisson's equations". In: *SIAM Journal on Numerical analysis* 7.4 (1970), pp. 627–656.

[BH83] Dietrich Braess and Wolfgang Hackbusch. "A new convergence proof for the multigrid method including the V-cycle". In: *SIAM journal on numerical analysis* 20.5 (1983), pp. 967–975.

[BH86] Peter N Brown and Alan C Hindmarsh. "Matrix-free methods for stiff systems of ODE's". In: *SIAM Journal on Numerical Analysis* 23.3 (1986), pp. 610–638.

[BHM00] William L Briggs, Van Emden Henson, and Steve F McCormick. *A multigrid tutorial*. SIAM, 2000.

[BP87] James H Bramble and Joseph E Pasciak. "New convergence estimates for multigrid algorithms". In: *Mathematics of computation* 49.180 (1987), pp. 311–329.

[Bra77] Achi Brandt. "Multi-level adaptive solutions to boundary-value problems". In: *Mathematics of computation* 31.138 (1977), pp. 333–390.

[Bra97] Dietrich Braess. *Finite elemente: Theorie, schnelle löser und anwendungen in der elastizitätstheorie*. Springer, 1997.

[Bro+84] Richard C Brower et al. "Geometrical models of interface evolution". In: *Physical Review A* 29.3 (1984), p. 1335.

[Bro87] Peter N Brown. "A local convergence theory for combined inexact-Newton/finite-difference projection methods". In: *SIAM Journal on Numerical Analysis* 24.2 (1987), pp. 407–434.

[BS90] Peter N Brown and Youcef Saad. "Hybrid Krylov methods for nonlinear systems of equations". In: *SIAM Journal on Scientific and Statistical Computing* 11.3 (1990), pp. 450–481.

[Cai+94] X-C Cai et al. "Newton-Krylov-Schwarz methods in CFD". In: *Numerical methods for the Navier-Stokes equations: Proceedings of the International Workshop Held at Heidelberg, October 25–28, 1993*. Springer. 1994, pp. 17–30.

[Che+98] Xinfu Chen et al. "Convergence of numerical solutions to the Allen-Cahn equation". In: *Applicable analysis* 69.1-2 (1998), pp. 47–56.

[Cia13] Philippe G Ciarlet. *Linear and nonlinear functional analysis with applications*. SIAM, 2013.

[CJ84] Tony F Chan and Kenneth R Jackson. "Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms". In: *SIAM Journal on scientific and statistical computing* 5.3 (1984), pp. 533–542.

[CL11] Xiao-Chuan Cai and Xuefeng Li. "Inexact Newton methods with restricted additive Schwarz based nonlinear elimination for problems with high local nonlinearity". In: *Siam journal on scientific computing* 33.2 (2011), pp. 746–762.

[CM94] Tony F Chan and Tarek P Mathew. "Domain decomposition algorithms". In: *Acta numerica* 3 (1994), pp. 61–143.

[CS99] Xiao-Chuan Cai and Marcus Sarkis. "A restricted additive Schwarz preconditioner for general sparse linear systems". In: *Siam journal on scientific computing* 21.2 (1999), pp. 792–797.

[DGG09] Victorita Dolean, Martin J Gander, and Luca Gerardo-Giorda. "Optimized Schwarz methods for Maxwell's equations". In: *SIAM Journal on Scientific Computing* 31.3 (2009), pp. 2193–2213.

[DH97] Maksymilian Dryja and Wolfgang Hackbusch. "On the nonlinear domain decomposition method". In: *BIT Numerical Mathematics* 37 (1997), pp. 296–311.

[DS83] Ron S Dembo and Trond Steihaug. "Truncated-Newton algorithms for large-scale unconstrained optimization". In: *Mathematical Programming* 26.2 (1983), pp. 190–212.

[DS96] John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, 1996.

[DW90] Maksymilian Dryja and Olof B Widlund. "Some domain decomposition algorithms for elliptic problems". In: *Iterative methods for large linear systems*. Elsevier, 1990, pp. 273–291.

[Eva22] Lawrence C Evans. *Partial differential equations*. Vol. 19. American Mathematical Society, 2022.

[EW94] Stanley C Eisenstat and Homer F Walker. "Globally convergent inexact Newton methods". In: *SIAM Journal on Optimization* 4.2 (1994), pp. 393–422.

[EW96] Stanley C Eisenstat and Homer F Walker. "Choosing the forcing terms in an inexact Newton method". In: *SIAM Journal on Scientific Computing* 17.1 (1996), pp. 16–32.

[Fix82] George J Fix. "Phase field methods for free boundary problems". In: (1982).

[FL15] Xiaobing Feng and Yukun Li. "Analysis of symmetric interior penalty discontinuous Galerkin methods for the Allen–Cahn equation and the mean curvature flow". In: *IMA Journal of Numerical Analysis* 35.4 (2015), pp. 1622–1651.

[Fre93] Roland W Freund. "A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems". In: *SIAM journal on scientific computing* 14.2 (1993), pp. 470–482.

[FS95] Jouni Freund and Rolf Stenberg. "On weakly imposed boundary conditions for second order problems". In: *Proceedings of the Ninth Int. Conf. Finite Elements in Fluids*. Venice. 1995, pp. 327–336.

[Gan+08] Martin J Gander et al. "Schwarz methods over the course of time". In: *Electron. Trans. Numer. Anal* 31.5 (2008), pp. 228–255.

[GL50] V. L. Ginzburg and L. D. Landau. "On the theory of superconductivity". English. In: *Zh. Èksper. Teor. Fiz.* 20 (1950), pp. 1064–1082. ISSN: 0044-4510.

[GLV11] Emmanuil H Georgoulis, Omar Lakkis, and Juha M Virtanen. "A posteriori error control for discontinuous Galerkin methods for parabolic problems". In: *SIAM journal on numerical analysis* 49.2 (2011), pp. 427–458.

[GM83] Gene H Golub and D Mayers. "The use of preconditioning over irregular regions". In: *system* 2 (1983), p. 2.

[GMN02] Martin J Gander, Frédéric Magoules, and Frédéric Nataf. "Optimized Schwarz methods without overlap for the Helmholtz equation". In: *SIAM Journal on Scientific Computing* 24.1 (2002), pp. 38–60.

[GRS07] Christian Grossmann, Hans-Gorg Roos, and Martin Stynes. *Numerical Treatment of Partial Differential Equations: Translated and revised by Martin Stynes*. Springer, 2007.

[GS83] Charles William Gear and Youcef Saad. "Iterative solution of linear equations in ODE codes". In: *SIAM journal on scientific and statistical computing* 4.4 (1983), pp. 583–601.

[HP17] Mark Harris and Kyrylo Perelygin. *Cooperative Groups: Flexible CUDA Thread Programming*. 2017. URL: https://developer.nvidia.com/blog/cooperative-groups/.

[HR89] Wolfgang Hackbusch and Arnold Reusken. "Analysis of a damped nonlinear multilevel method". In: *Numerische Mathematik* 55 (1989), pp. 225–246.

[Ise09] Arieh Iserles. *A first course in the numerical analysis of differential equations*. 44. Cambridge university press, 2009.

[Jeg13] Hansgeorg Jeggle. *Nichtlineare Funktionalanalysis: Existenz von Lösungen nichtlinearer Gleichungen*. Springer-Verlag, 2013.

[Kel03] Carl T Kelley. *Solving nonlinear equations with Newton's method*. SIAM, 2003.

[Kel95] Carl T Kelley. *Iterative methods for linear and nonlinear equations*. SIAM, 1995.

[KK04] Dana A Knoll and David E Keyes. "Jacobian-free Newton–Krylov methods: a survey of approaches and applications". In: *Journal of Computational Physics* 193.2 (2004), pp. 357–397.

[KKL84] David A Kessler, Joel Koplik, and Herbert Levine. "Geometrical models of interface evolution. II. Numerical simulation". In: *Physical Review A* 30.6 (1984), p. 3161.

[KKL85] David A Kessler, Joel Koplik, and Herbert Levine. "Geometrical models of interface evolution. III. Theory of dendritic growth". In: *Physical Review A* 31.3 (1985), p. 1712.

[Kre91] Erwin Kreyszig. *Introductory functional analysis with applications*. John Wiley & Sons, 1991.

[Lio+88] Pierre-Louis Lions et al. "On the Schwarz alternating method. I". In: *First international symposium on domain decomposition methods for partial differential equations*. Vol. 1. Paris, France. 1988, p. 42.

[Liu+22] Lulu Liu et al. "A nonlinear elimination preconditioned inexact Newton algorithm". In: *SIAM Journal on Scientific Computing* 44.3 (2022), A1579–A1605.

[LKK18] Lulu Liu, David E Keyes, and Rolf Krause. "A note on adaptive nonlinear preconditioning techniques". In: *SIAM Journal on Scientific Computing* 40.2 (2018), A1171–A1186.

[LM06] Omar Lakkis and Charalambos Makridakis. "Elliptic reconstruction and a posteriori error estimates for fully discrete linear parabolic problems". In: *Mathematics of computation* 75.256 (2006), pp. 1627–1658.

[LMP] Omar Lakkis, Charalambos Makridakis, and Tristan Pryer. "A comparison of duality and energy a posteriori estimates for L8 (0, T; L2 (O)) in parabolic problems". In: ().

[LU68] O. A. Ladyzhenskaya and N. N. Ural'tseva. *Linear and quasilinear elliptic equations*. English. Vol. 46. Math. Sci. Eng. Elsevier, Amsterdam, 1968.

[MK94] Paul R McHugh and Dana A Knoll. "Comparison of standard and matrix-free implementations of several Newton-Krylov solvers". In: *AIAA journal* 32.12 (1994), pp. 2394–2400.

[Nas00] Stephen G Nash. "A multigrid approach to discretized optimization problems". In: *Optimization Methods and Software* 14.1-2 (2000), pp. 99–116.

[Nat07] Frédéric Nataf. "Recent developments on optimized Schwarz methods". In: *Domain Decomposition Methods in Science and Engineering XVI* (2007), pp. 115–125.

[Nit71] Joachim Nitsche. "Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind". In: *Abhandlungen aus dem mathematischen Seminar der Universität Hamburg*. Vol. 36. 1. Springer. 1971, pp. 9–15.

[NL11] Stephen G Nash and Robert Michael Lewis. "Assessing the performance of an optimization-based multilevel method". In: *Optimization Methods and Software* 26.4-5 (2011), pp. 693–717.

[NW06] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 2006.

[OR00] James M Ortega and Werner C Rheinboldt. *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.

[Pip57] Alfred Brian Pippard. "An experimental determination of the Fermi surface in copper". In: *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences* 250.979 (1957), pp. 325–357. URL: `http://rsta.royalsocietypublishing.org/content/250/979/325.short`.

[PW98] Michael Pernice and Homer F Walker. "NITSOL: A Newton iterative solver for nonlinear systems". In: *SIAM Journal on Scientific Computing* 19.1 (1998), pp. 302–318.

[QV99] Alfio Quarteroni and Alberto Valli. *Domain decomposition methods for partial differential equations*. Oxford University Press, 1999.

[Ray64] L. J. W. S. Rayleigh. "Scientific Papers". In: Six volumes bound as three. Vol. 3: 1887-1892. Vol. 4: 1892-1901 (1964).

[Reu88] Arnold Reusken. "Convergence of the multilevel full approximation scheme including the V-cycle". In: *Numerische Mathematik* 53 (1988), pp. 663–686.

[Riv08] Béatrice Rivière. *Discontinuous Galerkin methods for solving elliptic and parabolic equations: theory and implementation*. SIAM, 2008.

[RR06] Michael Renardy and Robert C Rogers. *An introduction to partial differential equations*. Vol. 13. Springer Science & Business Media, 2006.

[SBG98] Barry Smith, Petter E Bjørstad, and William Gropp. "Domain decomposition: parallel multilevel methods for elliptic partial differential equations". In: *(No Title)* (1998).

[Sch70] Hermann Amandus Schwarz. *Ueber einen Grenzübergang durch alternirendes Verfahren*. Zürcher u. Furrer Berlin, 1870.

[SS17] Huailing Song and Chi-Wang Shu. "Unconditional energy stability analysis of a second order implicit–explicit local discontinuous Galerkin method for the Cahn–Hilliard equation". In: *Journal of Scientific Computing* 73.2 (2017), pp. 1178–1203.

[SS86] Youcef Saad and Martin H Schultz. "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems". In: *SIAM Journal on scientific and statistical computing* 7.3 (1986), pp. 856–869.

[TOS01] Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. *Multigrid methods*. Academic press, 2001.

[TW04] Andrea Toselli and Olof Widlund. *Domain decomposition methods-algorithms and theory*. Vol. 34. Springer Science & Business Media, 2004.

[TX02] Xue-Cheng Tai and Jinchao Xu. "Global and uniform convergence of subspace correction methods for some convex optimization problems". In: *Mathematics of Computation* 71.237 (2002), pp. 105–124.

[Van79] Johannes Diederik Van der Waals. "The thermodynamic theory of capillarity under the hypothesis of a continuous variation of density". In: *Journal of Statistical Physics* 20.2 (1979), pp. 200–244.

[Van92] Henk A Van der Vorst. "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems". In: *SIAM Journal on scientific and Statistical Computing* 13.2 (1992), pp. 631–644.

[Xie96] Dexuan Xie. "New nonlinear multigrid analysis". In: *NASA Conference Publication*. Citeseer. 1996, pp. 793–808.

[XT06] Chuanju Xu and Tao Tang. "Stability analysis of large time-stepping methods for epitaxial growth models". In: *SIAM Journal on Numerical Analysis* 44.4 (2006), pp. 1759–1779.

[Xu+19] Jinchao Xu et al. "On the stability and accuracy of partially and fully implicit schemes for phase field modeling". In: *Computer Methods in Applied Mechanics and Engineering* 345 (2019), pp. 826–853.

[Xu92] Jinchao Xu. "Iterative methods by space decomposition and subspace correction". In: *SIAM review* 34.4 (1992), pp. 581–613.

[YD06] Irad Yavneh and Gregory Dardyk. “A multilevel nonlinear method”. In: *SIAM journal on scientific computing* 28.1 (2006), pp. 24–46.

[Yse93] Harry Yserentant. “Old and new convergence proofs for multigrid methods”. In: *Acta numerica* 2 (1993), pp. 285–326.

[Zha+24] Xinyue Evelyn Zhao et al. “Bifurcation analysis reveals solution structures of phase field models”. In: *Communications on Applied Mathematics and Computation* 6.1 (2024), pp. 64–89.

A. Appendix - Brief on functional analysis

Definition A.1 *We denote by $L^p(\Omega)$ the class of all measurable functions ϕ, defined on $\Omega \subset \mathbb{R}^n$, such that*

$$\int_\Omega |\phi|^p ds < \infty$$

defined by the norm

$$\|\phi\|_{L^p(\Omega)} := \left(\int_\Omega |\phi|^p ds \right)^{\frac{1}{p}}$$

for $1 \le p \le \infty$. $L^2(\Omega)$ is a separable Hilbert space with inner product as

$$\langle \phi, \psi \rangle_{L^2(\Omega)} := \int_\Omega \phi\psi ds$$

Definition A.2 *Sobolev spaces are vector subspaces of $L^p(\Omega)$. For $m \in \mathbb{N}$ and $1 \le p \le \infty$, we can define a function $\|.\|_{W^{m,p}(\Omega)}$ such as*

$$\|\phi\|_{W^{m,p}(\Omega)} := \left(\int_\Omega \sum_{0 \le |a| \le m} |D^a \phi|^p \right)^{\frac{1}{p}}$$

where

$$D^a \phi := \frac{\partial^{|a|}}{\partial^{a_1} x_1 \dots \partial^{a_n} x_n} \phi$$

for $a := (a_1, a_2, \dots a_n)$.

Definition A.3 *Assume X and Y are two vector space and a mapping $T : X \to Y$ is a linear operator from X into Y, or a linear functional or linear form if $Y = \mathbb{R}$, if*

$$T(\alpha x + \beta y) = \alpha T(x) + \beta T(y) \ \ \forall x, y \in X \text{ and } \alpha, \beta \in \mathbb{R}$$

Assume X is a real, reflexive Banach space and X' is a dual space of X. And let $T : X \to X'$ is a mapping.

Definition A.4 *A mapping $T : X \to X'$ is **continuous** if $\forall u, \nu \in X$ such that*

$$\|T(u) - T(\nu)\|_{X'} \to 0 \text{ as } \|u - \nu\|_X \to 0.$$

Definition A.5 *We say that a mapping $T : X \to X'$ is **bounded** if it maps bounded sets in X to bounded set in X'.*

Definition A.6 *A mapping $T : X \to X'$ is **monotone** if*

$$\langle T(u) - T(\nu), u - \nu \rangle \geq 0 \quad \forall u, \nu \in X$$

Definition A.7 *Let X and Y be two normed vector spaces, $\mathcal{L}(X;Y)$ be a vector spaces formed by all continuous linear mappings from X into Y and equipped with the norm defined by*

$$\|A\|_{\mathcal{L}(X;Y)} := \sup_{x \in X \setminus \{0\}} \frac{\|Ax\|_Y}{\|x\|_X} \quad \forall A \in \mathcal{L}(X;Y)$$

Definition A.8 *if Ω in an open subset of the space X, the mapping $g : \Omega \subset X \to Y$ is differentiable at a point $u \in \Omega \subset X$ if there exists an element A in the space $\mathcal{L}(X;Y)$ such that*

$$\lim_{\|h\|_X \to 0} \frac{\|g(u+h) - g(u) - A(h)\|_Y}{\|h\|_X} \to 0$$

if $A \in \mathcal{L}(X;Y)$ is unique and the mapping $g : \Omega \subset X \to Y$ is continuous .i.e., if $r_0 < 0$ and $B(a; r_0) \subset \Omega$

$$\|g(u+h) - g(u)\| \leq \|A(h) + \delta(h)\| \, \|h\| \quad \forall \|h\| < r < r_0$$

*the linear map $A \in \mathcal{L}(X;Y)$ defined in this fashion is called **Fréchet derivative** of mapping f **at the point u** and $A(h)$ is called the **Fréchet differential** denoted by $Dg(u) \in \mathcal{L}(X;Y)$ of $g(u)$ at point u [Cia13].*

Definition A.9 *The mapping $g : \Omega \subset X \to Y$ at point $u \in \Omega$ and a vector nonzero $v \in X$ and $I(\theta) \subset \mathbb{R}$ is an open interval, assume the function $\theta \in I(\theta) \subset \mathbb{R} \to g(u + \theta v) \in Y$ is differentiable at $\theta = 0$. Then g is said to have **Gâteaux derivative** at $u \in \Omega$ in direction of v and defined by*

$$\partial_G g := \lim_{\substack{\theta \to 0 \\ \theta \neq 0}} \frac{g(u + \theta v) - g(u)}{\theta}$$

Remark A.1 *It should be emphasized that Gâteaux differentiability does not imply differentiability, but Fréchet differentiability implies Gâteaux differentiability in all directions $h \in X$ but does not imply the converse.*