Asynchronous domain decomposition method in structure mechanics – case of the global/local coupling

Ahmed El Kerim

université PARIS-SACLAY

Asynchronous domain decomposition method in structural mechanics – the case of the global/local coupling

Méthode de décomposition de domaine asynchrone en calcul des structures – cas du couplage global/local

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°579 : sciences mécaniques et énergétiques, matériaux et géosciences (SMEMaG)
Spécialité de doctorat: Mécanique des solides
Graduate School : Sciences de l'ingénierie et des systèmes
Référent : ENS Paris-Saclay

Thèse préparée dans les unités de recherche: LMPS - Laboratoire de Mécanique Paris-Saclay (Université Paris-Saclay, CentraleSupélec, ENS Paris-Saclay, CNRS) & MICS - Laboratoire Mathématiques et informatique pour la complexité des systèmes (CentraleSupélec, Université Paris-Saclay)
Sous la direction de :
Pierre GOSSELET, Directeur de recherche, CNRS - Université de Lille
Frédéric MAGOULÈS, Professeur des univeristés, Ecole CentraleSupélec - Université Paris-Saclay

Thèse soutenue à Paris-Saclay, le 12 mai 2023, par

Ahmed EL KERIM

Composition du jury

Membres du jury avec voix délibérative

Julien YVONNET Professeur des universités, Université Gustave Eiffel	Président
Pierre SPITERI Professeur émérite, Institut national polytechnique de Toulouse	Rapporteur
Daniel B. SZYLD Full professor, Temple University of Philadelphia	Rapporteur
Pierre-Alain BOUCARD Professeur des universités, ENS Paris-Saclay - Université Paris-Saclay	Examinateur
Nicole SPILLANE Chargée de recherche CNRS, Centre de Mathématiques Appliquées - École polytechnique	Examinatrice
Robin BOUCLIER Maître de conférences HDR, Institut National des Sciences Appliquées Toulouse	Examinateur

THESE DE DOCTORAT

NNT : 2023UPAST064

Acknowledgments

After three rewarding and exciting years, a chapter is ending. It has been an incredible and transformative journey, brimming with countless challenges, personal growth, and enriching experiences. Reflecting upon this beautiful time, I will treasure these memories with a profound sense of pride and gratitude, for they have shaped me profoundly and left an indelible mark on my life.

First of all, I would like to express my sincere gratitude to Pierre Spiteri and Daniel B. Zsyld, two recognized experts in the field of asynchronous computing research, for the privilege of having them as reviewers of my thesis and for their insightful comments during the detailed examination of the manuscript.

More generally, I express my sincere gratitude to Julien Yovennet, Pierre-Alain Boucard, Nicole Spillane, and Robin Bouclier for graciously agreeing to be members of my jury. Your valuable comments and sustained attention to my work are much appreciated. I would also like to thank Augstin Parret-Freaud, who participated as a guest member of my jury.

I am deeply grateful to my thesis supervisors, Pierre Gosselet and Frédéric Magoulès. Pierre, your constant support, pertinent ideas, availability, exceptional pedagogical qualities, and continued good humor have been of invaluable help to me along the way. I sincerely thank you for all the times you welcomed me into your home in Lille. Frédéric, thank you for your unfailing confidence in me, sound advice, and countless instructive discussions. Your kindness and guidance broadened my horizons during my thesis. I want to express my sincere gratitude to both of you for trusting me from the beginning to the end of this project.

I would also like to thank all my colleagues in the Scientific Computing team at the MICS laboratory. Guillaume, even though you were not officially part of the supervisory team, I am immensely grateful for your guidance, unlimited assistance, constant availability, and boundless kindness. The countless exchanges we had over these three years have been invaluable. I would also like to thank Quimeng for welcoming me when I first arrived and Abbal for his kindness, availability, and discussions on setting up the cluster in the sb109 room.

A special mention goes to my colleagues at the LMPS laboratory. I am incredibly grateful to my friends with whom I have shared most of my time. My neighbor Paul-William, thank you for your kindness, good humor, and companionship during conferences. Myriam, thank you for always being there and for your help in preparing for my defense. Matthew, thank you for the coffee break anecdotes and for assisting me in reading a part of my manuscript in preparation for my defense. Roxane, thank you for welcoming me from day one and

discussing football throughout these three years. I would also like to thank my office mates, Perla and Achraff, for their good humor and kindness. Thanks also to Mihaja, Corentin, Takwa, Cédric, Afsal, Fatima, Sofiane, Zakaria, Louis, Mathieu and Antoine. I extend my gratitude to all those who participated in my thesis pre-defenses.

Finally, I want to express my deepest gratitude to my family. To my parents, thank you for your unwavering support since the beginning of my education and especially for standing by me since my arrival in France to pursue my studies. Also, thanks To my sister, two brothers, nephew, and niece. This thesis is dedicated to my entire family, whose unwavering belief in me has driven my accomplishments.

Contents

1 - Introduction

1.1 . Background & Objectives

Structural mechanics is an essential field of engineering that focuses on the analysis and design of structures such as buildings, bridges, and aircraft. The objective is to understand how these structures behave and respond to different loads and environmental conditions while ensuring their safety, efficiency, and cost-effectiveness.

However, analyzing and designing complex structures can be time-consuming and computationally intensive, especially for large-scale problems. To address these challenges, parallel computing has emerged as a powerful tool in structural mechanics. Parallel computing involves breaking down a computational task into smaller, independent parts that can be executed simultaneously on multiple processing units, such as CPU cores or GPUs.

Parallel computing offers several advantages in structural mechanics:

- Speed and Efficiency: By harnessing the processing power of multiple cores or GPUs, parallel computing significantly reduces computation time for large-scale problems. It can expedite structural analysis and design processes.
- Scalability: Parallel computing allows computational resources to scale according to the problem's complexity. This is crucial in structural mechanics, where problems can vary from simple to highly complex.
- Accuracy: Parallel computing can enhance result accuracy by enabling the use of finer mesh resolution or more precise numerical methods. It facilitates the utilization of larger and more complex models, leading to more accurate outcomes compared to smaller models.

One of the most commonly used methods in structural mechanics for parallel computing is the parallel domain decomposition approach, typically based on finite element analysis. This method involves dividing a large computational domain into smaller, independent subdomains that can be solved in parallel. It offers significant computational advantages, particularly for complex problems involving linear or nonlinear, static or dynamic scenarios.

However, parallelizing domain decomposition methods presents certain challenges:

- Data Communication: For problems with highly interdependent subdomains, communication overhead can become a bottleneck in the computation process.

- Load Balancing: Distributing the computational load evenly across multiple processing units can be challenging. This imbalance can lead to certain subdomains taking significantly longer to solve, while others remain idle.

Asynchronous iteration is a parallel computing method that has gained popularity in recent years. It allows for the concurrent execution of multiple parallel tasks, where each task operates independently without waiting for others to finish. Instead of blocking the execution flow until all tasks are completed, the program continues executing tasks asynchronously in the background. This approach optimizes system resources utilization and overall execution time, especially for tasks with longer durations. Asynchronous programming concepts, such as asynchronous functions and threads, are commonly employed in high-performance distributed computing systems.

Asynchronous domain decomposition offers several advantages:

- Flexibility: Computation can proceed at different speeds, allowing each processing unit to operate at its maximum performance. This results in faster overall computation times, particularly for highly parallelizable problems.
- Robustness: Computation can proceed without synchronizing across processing units, even in the presence of communication failures. This makes it suitable for solving problems in large-scale distributed computing environments.
- Load Balancing: Computational load can be dynamically balanced based on the processing capabilities of each unit, leading to more efficient resource utilization and faster overall computation times.

However, one significant challenge in asynchronous iterations is maintaining data consistency, as computations can proceed at different speeds.

The main objective of this Ph.D. thesis is to advance the understanding and development of asynchronous domain decomposition methods. The work aims to address the challenges faced by conventional domain decomposition techniques in parallel computing, particularly in terms of load balancing and communication overhead.

Through a detailed analysis of a specific domain decomposition method in the field of structural mechanics, this thesis aims to address the current limitations and gaps in the literature. It is worth noting that the chosen domain decomposition method is particularly popular in industry as it fulfills their requirements in a non-intrusive manner.

The research questions addressed in this thesis include:

- How can asynchronous domain decomposition methods be designed to achieve better load balancing and communication efficiency in parallel computing environments?

- What are the trade-offs between communication overhead, load balancing, and accuracy in asynchronous domain decomposition methods?

This work focuses on the development and analysis of new asynchronous domain decomposition algorithms that can overcome these challenges. The algorithms will be evaluated and compared with existing synchronous methods in terms of efficiency, scalability, and robustness.

1.2 . Outline

Following this introductory chapter, this thesis is organized as follows:

- Chapter 2: This chapter begins with a review of relevant academic literature related to the development of domain decomposition methods. It provides an overview of both early and recent methods and explains the theory and algorithms underlying primal methods in detail. Additionally, it introduces the current state of non-intrusive global/local coupling methods and discusses their performance limitations due to the alternation between two computation steps. Furthermore, the chapter introduces the concept of asynchronous parallel computation, highlighting its potential improvement for domain decomposition methods in general, and specifically in the case of non-intrusive global/local coupling.
- Chapter 3: This chapter presents a mathematical overview of non-intrusive global/local coupling. It explores the method's application to linear elliptic and nonlinear problems. Additionally, it interprets the non-intrusive global/local method as a right-preconditioned primal domain decomposition method, offering a novel approach.
- Chapter 4: This chapter presents an asynchronous version of the non-intrusive global/local computation method for linear and nonlinear elliptic problems, building upon the new interpretation from the previous chapter. It establishes a proof of convergence for the discretized system using paracontractions techniques.
- Chapter 5: This chapter provides an implementation of an asynchronous code for non-intrusive global/local coupling using the Remote Memory Access (RMA) technique with the Message Passing Interface (MPI). RMA-MPI enables processes to directly access the memory of other processes in a distributed system, resulting in faster communication and increased efficiency compared to traditional data-copying methods.
- Chapter 6: This chapter presents numerical results obtained from studies conducted on academic and industrial cases. The objective is to compare the

asynchronous version of the non-intrusive global/local coupling with the accelerated synchronous version, demonstrating the advantages of the asynchronous approach.

- Chapter 7: This concluding chapter summarizes the main findings, highlights the contributions of the study, and provides recommendations for future research.

1.3 . Scientific Contributions

1.3.1 . Articles

- Ahmed El Kerim, Pierre Gosselet, Frederic Magoules. Asynchronous Global-Local Non-Invasive Coupling for Linear Elliptic Problems. *Computer Methods in Applied Mechanics and Engineering*, 2023, 406 (115910), ⟨10.1016/j.cma.2023.115910⟩.

  This article presents the first asynchronous version of the non-invasive global-local coupling method, capable of effectively handling multiple and potentially adjacent patches. We provide a new interpretation of the coupling as a primal domain decomposition method and prove the convergence of the relaxed asynchronous iteration. The asynchronous paradigm overcomes several limitations of the performance of the global-local coupling. We illustrate the method with various linear elliptic problems encountered in thermal and elasticity studies.

1.3.2 . Proceeding

- Ahmed El Kerim, Pierre Gosselet, Frédéric Magoulès. Couplage Global-Local en asynchrone pour des problèmes linèaires. 15ème colloque national en calcul des structures, Université Polytechnique Hauts-de-France [UPHF], May 2022, 83400 Hyères-les-Palmiers, France.

- Ahmed EL KERIM, Pierre Gosselet, Frederic Magoules. Asynchronous scalable version of the Global-Local non-invasive coupling. 9th European Conference for Aeronautics and Space Sciences, Jun 2022, Lille, France. ⟨10.13009/EUCASS2022-4830⟩.

1.3.3 . Abstracts at conferences

- 17th International Miklos Ivanyi PhD, DLA Symposium, 25 - 26 Octobre 2021, Pecs, Hungary.

- SIAM Conference on Parallel Processing for Scientific Computing (PP22) 23 - 26 fevrier 2022 2022, Seattle, Washington, U.S.

- 15ème colloque national en calcul des structures, 16 – 20 mai 2022, Giens, France.

- EUCASS-3AF 2022, 9th European Conference for Aerospace Sciences, 27 Juin - 1 Juillet Lille, 2022, France.
- 27th International Domain Decomposition Conference, 25 - 29 Juillet 2022, Prague, Czech Republic.

2 - Literature review

As mentioned in the introduction, this work is interested in the parallel simulation of structural problems discretized by the finite element method leading to massive nonlinear systems. Using domain decomposition methods is a natural strategy to distribute the computation over many computational units. So in this chapter, a literature review is presented on the history of the domain decomposition methods, emphasizing the most used methods in structural mechanics. After, the existing works on the global/local coupling techniques are reviewed and their advantages and inconveniences compared to the classical domain decomposition methods are specified. Then the parallel aspect of all these methods are discussed by identifying the limit due to the high cost of communications when using large calculation tools.

2.1 . Introduction

The numerical simulation of mechanical problems requires the consideration of behaviors and geometries that can vary from simple linear academic problems to nonlinear industrial problems. Finite element methods provide a powerful framework for convergence toward the solution sought.

However, a suitable choice of finite elements and a well-refined mesh are required. This last condition can lead to massive systems whose resolution is costly in computation time and memory storage.

One strategy is the domain decomposition method, which divides the structure under study into several sub-structures. A strong point of this technique is that it is well adapted for parallel computing since a sub-problem can be defined on each sub-structure to be solved in parallel.

An iterative process is then established to obtain the solution. The idea is to exchange data between the different sub-domains at each iteration, corresponding to a communication between the processors.

2.2 . Domain decomposition

2.2.1 . Motivation

First introduced by H.A. Schwarz 1870 [96], the motivation behind domain decomposition methods was to find the solution to the Poisson PDE (equation 2.1) defined on a complex geometry as shown in Figure 2.1:

$$\text{Find } u \text{ such that: } \begin{cases} -\Delta u = f \text{ in } \Omega \\ u = 0 \text{ on } \partial\Omega \end{cases} \tag{2.1}$$

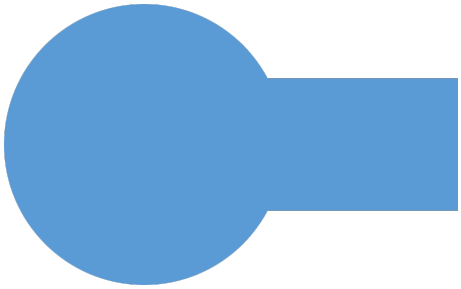
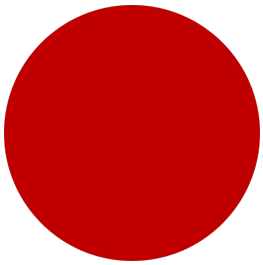


Figure 2.1: Schwarz first domain decomposition problem

with $\partial\Omega$ the boundary of the domain Ω.

Schwarz considered that the domain Ω of Figure 2.1 was the union of a disk and a rectangle. The suggested solution was to divide the domain Ω, into two subdomains corresponding to two simple geometries, where one can compute the solution of the problem for any given boundary conditions,



(a) Circular subdomain Ω_1

(b) Rectangular subdomain Ω_2

Figure 2.2: Domain decomposition

Figure 2.2 shows the decomposition of the domain Ω into two subdomains: the circular subdomain Ω_1 in Figure 2.2a and the rectangular subdomain Ω_2 in Figure 2.2b.

2.2.2 . Schwarz alternating method

Schwarz then established an iterative approach, known as the overlapping alternating Schwarz method, to compute the solution of the problem on the whole domain Ω, by alternating between a resolution in Domain Ω_1 and another one in Domain Ω_2. The idea is to retrieve the solution on the shared boundary within the blue overlap area $\alpha = \Omega_1 \cap \Omega_2$ in Figure 2.3, calculated when solving one subdomain and impose it on the other subdomain.

So the system at the iteration $n+1$ can be written as:

Find u_1^{n+1} such that:

$$\begin{cases} -\Delta u_1^{n+1} = f \text{ in } \Omega_1 \\ u_1^{n+1} = u^n \text{ on } \partial\Omega_1 \cap \Omega_2 \\ u_1^{n+1} = 0 \text{ on } \partial\Omega_1 \cap \partial\Omega \end{cases}$$

Find u_2^{n+1} such that:

$$\begin{cases} -\Delta u_2^{n+1} = f \text{ in } \Omega_2 \\ u_2^{n+1} = u_1^{n+1} \text{ on } \Omega_2 \cap \partial\Omega_1 \\ u_2^{n+1} = 0 \text{ on } \partial\Omega_2 \cap \partial\Omega \end{cases}$$

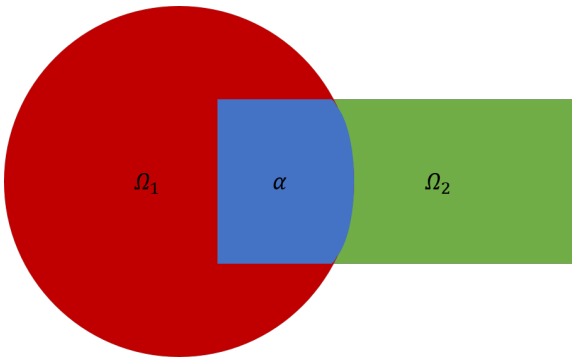


Figure 2.3: Schwarz alternating method with overlap

This method is not parallel due to the alternating nature of the computations between the subdomains, which imposes that the calculation in Ω_1 is finished before launching the other computation on the domain Ω_2. When more than two subdomains are involved, it is possible to color the subdomains and solve the subdomains of the same color in parallel.

Remark 1. Note that the convergence of these methods depends strongly on the overlap size; the larger the overlap width, the faster the convergence.

2.2.3 . Schwarz parallel method

To raise the issue of parallelization, the work presented in [69] introduced the first approach to parallelize the previous Schwarz algorithm, Iteration n+1 can be written as:

Find u_1^{n+1} such that:

$$\begin{cases} -\Delta u_1^{n+1} = f \text{ in } \Omega_1 \\ u_1^{n+1} = u^n \text{ on } \partial\Omega_1 \cap \Omega_2 \\ u_1^{n+1} = 0 \text{ on } \partial\Omega_1 \cap \partial\Omega \end{cases}$$

Find u_2^{n+1} such that:

$$\begin{cases} -\Delta u_2^{n+1} = f \text{ in } \Omega_2 \\ u_2^{n+1} = u_1^n \text{ on } \partial\Omega_2 \cap \Omega_1 \\ u_2^{n+1} = 0 \text{ on } \partial\Omega_2 \cap \partial\Omega \end{cases}$$

The idea is to eliminate the alternating state of the algorithm where the $(n+1)^{th}$ computation on Ω_1 to compute the $(n+1)^{th}$ solution on Ω_2 must be used, by calculating both subdomains simultaneously using the boundary condition from the n^{th} iteration of the neighboring subdomain.

So in both algorithms, the idea was to check each problem in its subdomain and, after that, by ensuring via an iterative process, the algorithm's convergence while guaranteeing the equality of the field of displacement in the area of the overlap.

This approach was then generalized for the case with several subdomains using parallel computing and more general geometries and applications. One can find more details about the convergence theory and the implementation in [36, 98, 110]. The references [69, 39] relate the history of development of domain decomposition methods.

2.2.4 . Non overlapping Schwarz method

Other than their adaptation to parallel computation by allowing the distribution of the calculation of each subdomain on a computational unit, the domain decomposition methods allow to couple several problems of different natures (fluid-structure, magneto-mechanical, ...etc.). For this reason the methods with overlap may seem inadequate and therefore a non-overlapping variant of Schwarz's methods was later introduced in the work of Lions [70].

It is thus supposed that the two subdomains Ω_1 and Ω_2 are such that $\bar{\Omega} = \bar{\Omega}_1 \cup \bar{\Omega}_2$ and $\Omega_1 \cap \Omega_2 = \varnothing$, and their interface is introduced as $\Gamma = \partial\Omega_1 \cap \partial\Omega_2$, see Figure 2.4.

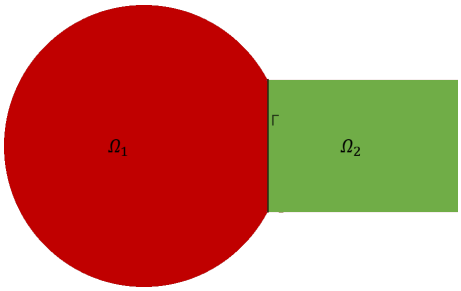


Figure 2.4: Two non overlapping subdomains

The non-overlapping Schwarz domain decomposition method requires introducing two parameters, named q_1 and q_2, which act as "interface stiffness" or impedance and which allow for the introduction of the Robin or Fourier boundary conditions on the interface. These parameters offer the possibility to find their best configuration, hence the name Optimized Schwarz Method (OSM) is often used for this method. ∂_n corresponds to the normal derivative operator, the parallel non-overlapping Schwarz iteration can be written as:

Find u_1^{n+1} such that:

$$\begin{cases} -\Delta u_1^{n+1} = f \text{ in } \Omega_1 \\ (\partial_n + q_1) u_1^{n+1} = (-\partial_n + q_1) u_2^n \text{ on } \Gamma \\ u_1^{n+1} = 0 \text{ on } \partial\Omega_1 \cap \partial\Omega \end{cases}$$

Find u_2^{n+1} such that:

$$\begin{cases} -\Delta u_2^{n+1} = f \text{ in } \Omega_2 \\ (\partial_n + q_2) u_2^{n+1} = (-\partial_n + q_2) u_1^n \text{ on } \Gamma \\ u_2^{n+1} = 0 \text{ on } \partial\Omega_2 \cap \partial\Omega \end{cases}$$

Note that it is also possible to derive an alternating version, and that optimized (i.e. Robin) interface conditions can also be used in the presence of an overlap.

In structural mechanics a well-known version of OSM is the Latin approach [63] which combines such interface conditions with a nonlinear solver, making it possible to solve complex elastoviscoplastic problems with friction at the interface.

2.2.5 . Non stationary methods

The Schwarz methods rely on a fixed point iteration with a contraction property. It is possible to derive other approaches which do not naturally express as

the search for a fixed point, and when turned into a fixed point (e.g. using a preconditioner) do not have a contraction property, making it necessary to use enhanced solvers like Krylov in the linear case or Newton in the nonlinear case.

These methods are often referred to as sub-structuring methods, and one of the first was [90], developed by mechanical engineers for the finite element analysis of complex structures. A full historical review can be found in [40].

To simplify the introduction of the method, the case of Figure 2.4 is considered with two subdomains Ω_1 and Ω_2, $\bar{\Omega} = \bar{\Omega}_1 \cup \bar{\Omega}_2$ and $\Omega_1 \cap \Omega_2 = \varnothing$, separated by $\Gamma = \partial\Omega_1 \cap \partial\Omega_2$.

If independent fields u_1 and u_2 are considered in the subdomains, beside the Poisson equation inside the subdomains, they must satisfy two interface conditions in order to be the restriction of the solution of the problem set on the whole domain Ω. The first condition is the continuity $u_{1|\Gamma} = u_{2|\Gamma}$, and the second is the balance of the fluxes $\partial_n u_{1|\Gamma} + \partial_n u_{2|\Gamma} = 0$.

This leads to the classical approaches:

- The **primal** approach:

$$\begin{aligned}&\text{Find the Dirichlet condition } u_\Gamma\\&\text{such that } \partial_n u_{1|\Gamma} + \partial_n u_{2|\Gamma} = 0, \text{ where}\\&\text{for } s = 1,2, \quad u_s \text{ solves } \begin{cases} -\Delta u_s = f \text{ in } \Omega_s \\ u_s = u_\Gamma \text{ on } \Gamma \\ u_s = 0 \text{ on } \partial\Omega_s \cap \partial\Omega \end{cases}\end{aligned}$$

- The **dual** approach:

$$\begin{aligned}&\text{Find the Neumann condition } \lambda_\Gamma\\&\text{such that } u_{1|\Gamma} = u_{2|\Gamma}, \text{ where}\\&\text{for } s = 1,2, \quad u_s \text{ solves } \begin{cases} -\Delta u_s = f \text{ in } \Omega_s \\ \partial_n u_s = (-1)^s \lambda_\Gamma \text{ on } \Gamma \\ u_s = 0 \text{ on } \partial\Omega_s \cap \partial\Omega \end{cases}\end{aligned}$$

After discretization, all these methods enter the framework of linear systems where the unknown is a boundary condition and the residual corresponds to the non-satisfaction of the remaining interface conditions. Due to the different physical nature of the unknown and of the residual, it is clear that stationary iteration is not possible. Anyhow, powerful preconditioners are available which can be used in conjunction with a Krylov solver, leading to efficient methods like the balancing domain decomposition (BDD) [77] or the finite element tearing and interconnecting (FETI) method [35] and their recent variants [60, 62].

The literature regarding these methods is very rich, with important progress being recently accomplished regarding automated coarse problems [99] or multipreconditioned solvers [17].

2.2.6 . Primal approach

In the next chapter, a new interpretation of the non invasive global/local method, studied in this thesis, is proposed as a primal domain decomposition method with a preconditioner. This subsection aims at providing enough details on the standard primal approach. The presentation of the primal approach presented here is inspired from [85]. For simplicity it is made in a discrete setting.

2.2.6.1 . Setting of the method

A classical quasi-static mechanical (or thermal) problem set on a domain Ω and discretized with the finite element method is considered. If nonlinearity is involved, an incremental approach is considered and this study is restricted to one increment. The system to be solved, called reference system here, can be written as:

$$\text{Find } \mathbf{u} \text{ so that } \mathbf{f}_{int}(\mathbf{u}) + \mathbf{f}_{ext} = 0 \tag{2.2}$$

Using the language of mechanics, $\mathbf{f}_{ext}$ stands for the (generalized) external forces, $\mathbf{f}_{int}$ is the vector of internal forces and $\mathbf{u}$ is the vector of unknown displacements (Dirichlet conditions are assumed to have been eliminated).

Note that in the case of linear problems, the internal forces take the form of a linear application:

$$\mathbf{f}_{int}(\mathbf{u}) = -\mathbf{K}\mathbf{u} \tag{2.3}$$

where $\mathbf{K}$ is the sparse symmetric definite positive stiffness matrix.

The conforming partition of Ω into N non-overlapping subdomains Ω^s is considered, so that each element belongs to exactly one subdomain. Superscript (s) will refer to data attached to domain Ω^s. Let Γ^s denote the interface of subdomain Ω^s. The classical notation is used to distinguish between interface (boundary) degrees of freedom, with a b subscript, and internal degrees of freedom, with a subscript i. Let $\Gamma = \cup\Gamma^s$ be the set of all interface degrees of freedom. The trace operators are defined as $\mathbf{T}^s : \Omega^s \to \Gamma^s$. Also, the assembly operators are defined as $\mathbf{A}^s : \Gamma^s \to \Gamma$ which connect subdomains together. The subscript Γ is used for quantities defined on the global interface.

In order to write the equilibrium of subdomain Ω^s, The introduction of the vector of nodal traction $\boldsymbol{\lambda}^s$ that is imposed by its neighboring subdomains on its interface is required:

$$\mathbf{f}^s_{int}(\mathbf{u}^s) + \mathbf{f}^s_{ext} + \mathbf{T}^{s^T}\boldsymbol{\lambda}^s = 0 \tag{2.4}$$

Where $\mathbf{T}^{s^T}$ can be viewed as an extension-by-zero operator inside the subdomain. Note that the b subscript is omitted for $\boldsymbol{\lambda}^s$ because by nature this quantity only exists on the boundary of the subdomain.

In order to make the subdomains' equilibrium equivalent to the global equilibrium (2.2), the addition of two interface conditions is needed:

$$\begin{aligned} &\text{Continuity of displacement, } \exists \mathbf{u}_\Gamma \text{ such that } \forall s,\ \mathbf{u}_b^s = {\mathbf{A}^s}^T \mathbf{u}_\Gamma, \\ &\text{Balance of interface traction, } \sum_s \mathbf{A}^s \boldsymbol{\lambda}^s = 0 \end{aligned} \tag{2.5}$$

It was chosen to express the continuity of the displacement by saying that local displacements should be the restriction of a common interface displacement. Another possibility would be to require the nullity of the jump of displacement using a signed assembly operator. This would be the starting point of dual approaches which need not be detailed here.

Assuming the well-posedness of local Dirichlet problems, it is possible to reword the global problem in terms of the unknown interface displacement $\mathbf{u}_\Gamma$:

$$\begin{aligned} &\text{Find } \mathbf{u}_\Gamma \text{ such that } \sum_s \mathbf{A}^s \boldsymbol{\lambda}^s = 0 \\ &\text{where } \forall s,\ (\boldsymbol{\lambda}^s, \mathbf{u}^s) \text{ are solutions to local Dirichlet problems:} \\ &\begin{cases} \mathbf{f}_{int}^s(\mathbf{u}^s) + \mathbf{f}_{ext}^s + {\mathbf{T}^s}^T \boldsymbol{\lambda}^s = 0 \\ \mathbf{T}^s \mathbf{u}^s = {\mathbf{A}^s}^T \mathbf{u}_\Gamma \end{cases} \end{aligned} \tag{2.6}$$

Note that the local Dirichlet problems correspond to solving:

$$\begin{aligned} &\text{Dirichlet condition: } \mathbf{u}_b^s = {\mathbf{A}^s}^T \mathbf{u}_\Gamma \\ &\text{Internal nonlinear problem: Find } \mathbf{u}_i^s,\ \mathbf{f}_{int,i}^s \begin{pmatrix} \mathbf{u}_i^s \\ \mathbf{u}_b^s \end{pmatrix} + \mathbf{f}_{ext,i}^s = 0 \\ &\text{Interface post-processing: } \boldsymbol{\lambda}^s = -\mathbf{f}_{int,b}^s \begin{pmatrix} \mathbf{u}_i^s \\ \mathbf{u}_b^s \end{pmatrix} - \mathbf{f}_{ext,b}^s \end{aligned} \tag{2.7}$$

2.2.6.2 . Condensation

As subdomains' internal quantities are always obtained from the solution of local problems with given interface conditions, the convergence of non-overlapping domain decomposition methods is governed by the convergence of interface unknowns. It is thus convenient to only think in terms of interface quantities and introduce condensed formulation where the internal equilibrium is implicitly always satisfied. In particular, the subdomains' Dirichlet-to-Neumann operator is defined as $\mathbf{s}^s$:

$$\boldsymbol{\lambda}^s = \mathbf{s}^s(\mathbf{u}_b^s; \mathbf{f}_{ext}^s) \text{ as defined in (2.7)} \tag{2.8}$$

Note that in the case of linear systems, the explicit expression of the (affine) Dirichlet-to-Neumann operator can be written as:

$$\mathbf{s}_{lin}^s(\mathbf{u}_b^s; \mathbf{f}_{ext}^s) = \underbrace{\left(\mathbf{K}_{bb}^s - \mathbf{K}_{bi}^s {\mathbf{K}_{ii}^s}^{-1} \mathbf{K}_{ib}^s\right)}_{\mathbf{S}^s} \mathbf{u}_b^s - \underbrace{\left(\mathbf{f}_{ext,b}^s - \mathbf{K}_{bi}^s {\mathbf{K}_{ii}^s}^{-1} \mathbf{f}_{ext,i}^s\right)}_{\mathbf{b}^s} \tag{2.9}$$

$\mathbf{S}^s$ is the well-known Schur complement matrix and $\mathbf{b}^s$ is the condensed right-hand side. The Schur complement inherits important properties from Matrix $\mathbf{K}^s$, in particular symmetry and (semi)definiteness positivity.

It is thus possible to rewrite the reference system as:

$$\text{Find } \mathbf{u}_\Gamma \text{ such that } \sum \mathbf{A}^s \mathbf{s}^s(\mathbf{A}^{s^T}\mathbf{u}_\Gamma; \mathbf{f}^s_{ext}) = 0 \tag{2.10}$$

In [85], it is proposed to solve the system with a Newton-Raphson solver, which involves solving tangent systems which have the structure of a linear primal domain decomposition problem. In the following section, the well established ingredients used to efficiently solve such systems are detailed.

2.2.6.3 . Solution strategy in the linear case

A focus is placed on the solution of the linearized system (2.10):

$$\underbrace{\left(\sum \mathbf{A}^s \mathbf{S}^s \mathbf{A}^{s^T}\right)}_{\mathbf{S}_\Gamma} \mathbf{u}_\Gamma = \underbrace{\sum \mathbf{A}^s \mathbf{b}^s}_{\mathbf{b}_\Gamma} \tag{2.11}$$

where the different operators are given in (2.9). In order to avoid exchanging dense matrices between processors, it is recommended to use a Krylov solver. Moreover, it is not necessary to actually form the Schur complement, one only needs to compute the solution to Dirichlet problems (using a factorization of $\mathbf{K}^s_{ii}$).

For the Krylov solver to converge efficiently, it is necessary to use a preconditioner. The one which naturally arises is the Neumann-Neumann:

$$\left(\sum \mathbf{A}^s \mathbf{S}^s \mathbf{A}^{s^T}\right)^{-1} \simeq \left(\sum \tilde{\mathbf{A}}^s \mathbf{S}^{s^\dagger} \tilde{\mathbf{A}}^{s^T}\right) \tag{2.12}$$

where the $(\tilde{\mathbf{A}}^s)$ are scaled assembly operators such that $\sum \mathbf{A}^s \tilde{\mathbf{A}}^{s^T} = \mathbf{I}_\Gamma$, and $\mathbf{S}^{s^\dagger}$ is a pseudo-inverse of $\mathbf{S}^s$. Note that it can be computed as $\mathbf{S}^{s^\dagger} = \mathbf{T}^s \mathbf{K}^{s^\dagger} \mathbf{T}^{s^T}$. The preconditioner thus corresponds to the parallel solution of Neumann problems set on the subdomains.

For the use of the pseudo-inverse to be well-defined, it is necessary to ensure that it is applied to a vector belonging to the image of the operator. This constraint is usually implemented using a projector/initialization procedure. Let $\mathbf{R}^s$ be a matrix whose columns form a basis of kernel of $\mathbf{K}^s$, i.e. the rigid body motions of the subdomain, then $\mathbf{R}^s_b = \mathbf{T}^s \mathbf{R}^s$ is a basis of the kernel of $\mathbf{S}^s$. The following quantities are introduced:

$$\begin{aligned} \tilde{\mathbf{G}} &= \left(\dots \tilde{\mathbf{A}}^s \mathbf{R}^s_b \dots\right), \\ \mathbf{P} &= \mathbf{I} - \tilde{\mathbf{G}}\left(\tilde{\mathbf{G}}^T \mathbf{S}_\Gamma \tilde{\mathbf{G}}\right)^{-1} \tilde{\mathbf{G}}^T \mathbf{S}_\Gamma \\ \mathbf{u}_{\Gamma_0} &= \tilde{\mathbf{G}}\left(\tilde{\mathbf{G}}^T \mathbf{S}_\Gamma \tilde{\mathbf{G}}\right)^{-1} \tilde{\mathbf{G}}^T \mathbf{b}_\Gamma \end{aligned} \tag{2.13}$$

Since the subdomains' rigid body motions are linearly independent and $\mathbf{S}_\Gamma$ is a SPD matrix, the matrix $(\tilde{\mathbf{G}}\mathbf{S}_\Gamma\tilde{\mathbf{G}})$ is also SPD and thus invertible.

$\mathbf{P}$ is a projector such that $\mathbf{P}\tilde{\mathbf{G}} = 0$ and $\mathbf{P}^T\mathbf{S}_\Gamma\tilde{\mathbf{G}} = 0$. The matrix $\left(\tilde{\mathbf{G}}^T\mathbf{S}_\Gamma\tilde{\mathbf{G}}\right)$ is called the coarse problem by analogy with multigrid approaches and it plays a central role in the convergence of the method. It is also a technical difficulty when implementing the method, and it can become a bottleneck for large number of subdomains.

The interface displacement is searched for under the form $\mathbf{u}_\Gamma = \mathbf{u}_{\Gamma_0} + \mathbf{P}\hat{\mathbf{u}}_\Gamma$, and the unknown $\hat{\mathbf{u}}_\Gamma$ is the solution to the following system:

$$\mathbf{S}_\Gamma\mathbf{P}\hat{\mathbf{u}}_\Gamma = \mathbf{b}_\Gamma - \mathbf{S}_\Gamma\mathbf{u}_{\Gamma_0} = \mathbf{P}^T\mathbf{b}_\Gamma \tag{2.14}$$

Note that $\mathbf{S}_\Gamma\mathbf{P} = \mathbf{P}^T\mathbf{S}_\Gamma\mathbf{P}$ so that the system is symmetric semi-definite, and it can be preconditioned by the Neumann-Neumann scaled operator $\left(\sum \tilde{\mathbf{A}}^s\mathbf{S}^{s^\dagger}\tilde{\mathbf{A}}^{s^T}\right)$.

As equipped, the method takes the name of Balancing Domain Decomposition (BDD) [77, 67] and it is proved to be scalable for a large class of problems. Anyhow, some situations are known to cause problems, like corners in shell models, or jagged interfaces, or heterogeneity near the interface. In theses cases it is proved that the coarse problem should be enriched by more modes than the simple rigid body motions which can be computed for each subdomain by a generalized eigenvalue problem, named GENEO [99].

Also, it is worth mentioning the more recent BDDC approach where the coarse problem is implemented by directly imposing some continuity between subdomains during the preconditioning step [62]. This version presents some algorithmic advantages, and it avoids the factorization of the pseudo-inverse which can be numerically difficult.

2.2.7 . Multi-scale methods

The Balancing Domain Decomposition method is equipped with a coarse problem that ensures the well-posedness of local Neumann problems during the preconditioning step.

From the mechanical point of view, the coarse problem ensures the equilibrium over the whole structure of the resultant and the moment (also known as the torsor) of the forces applied to the subdomains. According to the Saint-Venant principle, the remaining part to be computed is localized. This means that thanks to the coarse problem there should be no need to transfer long-range information and the solution could be found by local exchanges between neighboring subdomains. As the number of neighbors remains bounded even when the number of subdomains increases, the coarse problem should ensure the scalability of the method.

It appears that the coarse problem is a key ingredient for domain decomposition methods to achieve high performance. It plays the same role as the coarse grid in multigrid methods [107] and it bears strong similarity with algebraic multigrid methods.

In methods like BDDC or FETI-DP [61], the coarse problem corresponds to ensuring the primal continuity of some well-chosen interface degrees of freedom.

The coarse problem can also be viewed as a long-range simplification of the problem, what is sometime referred to as a numerical multi-scale method. For instance, the micro-macro Latin method [64] can be viewed as a parallel non-overlapping Schwarz method equipped with a coarse problem based on the Saint-Venant principle. An attempt to optimize the macro problem based on GENEO concept was proposed in [87].

2.3 . The non-invasive global/local coupling

2.3.1 . Introduction

Nested models, as presented in Figure 2.5, are ubiquitous in industry to give a multi-scale description of structures. They call for the use of multi-scale domain decomposition methods to avoid a prohibitive cost of meshing and remeshing of the structure that can subsequently lead to systems whose resolution and storage are cost prohibitive.

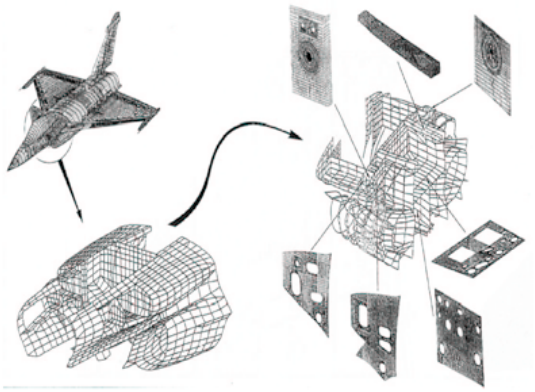


Figure 2.5: Nested models, courtesy Dassault Aviation

The motivation of industry to use non-intrusive multiscale coupling strategies is to be able to provide methods that can be integrated into legacy industrial software codes. To do so, they must respect several constraints, including the limitation to use the standard outputs and inputs of the software, which limits the choice of computational methods in mechanics, because these methods are generally developed in homemade codes but are not made to be used outside the academic domain.

The idea of the global/local coupling is to replace a global model with more precise models in some local parts of the structure, solved with dedicated software. An iterative approach is implemented to achieve the strong coupling. Of course, it comes with a convergence threshold and the need to tolerate a controlled error compared to the reference monolithic computation.

2.3.2 . Illustration

Generally, when studying the design of a part with a specific material, complex behaviors tend to develop in some local zones. This can be explained by the exposure of these areas to more critical physical phenomena such as higher heat, higher load or cracking, etc. Very particular and complex geometries can also distinguish these areas from the rest of the structure.

To illustrate the type of study performed, an approximation of a 2D turbine blade as in Figure 2.6 is considered. In this model, two zones of interest, shown in yellow and green, with specific complex geometries are considered. Another blue global zone exists where no important behaviors or complex geometries are distinguished. This model is defined on several scales, illustrated in the following

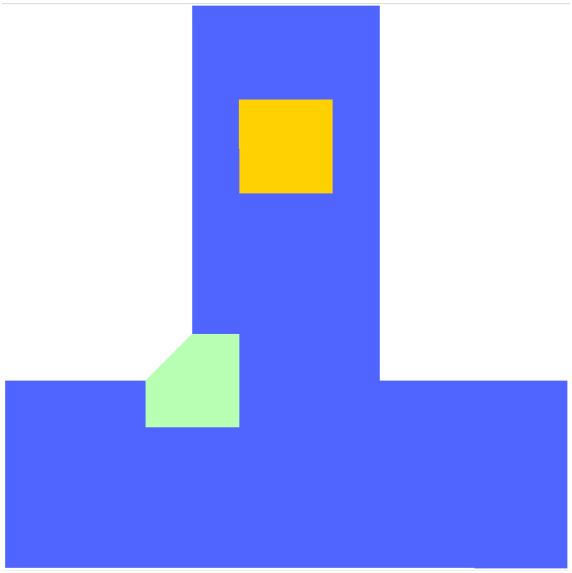


Figure 2.6: 2D turbine blade

Figures.

Figure 2.7 corresponds to a coarse scale representation of the 2D turbine blade as in Figure 2.6 also known as the global model, where its coarse mesh cannot consider what happens in the zones of interest. A highly refined mesh is required to simulate it well. These zones of interest can be much larger than the mesh of the global problem. $\Omega^{1,G}$ and $\Omega^{2,G}$ are the two zones of interest and $\Omega^{0,G}$ the complementary domain. Γ_A is the interface between these zones of interest and this complementary domain.

As in Figure 2.8, geometries or behaviors can exist that are reproduced in some regions of interest (local models) of the global structure (global model). The geometrical modification can correspond to removing a part of the geometry of the global structure and introducing holes. It can also correspond to strong nonlinear behaviors, like plasticity, viscoplasticity, fatigue, and cracking. $\Omega^{1,F}$ and $\Omega^{2,F}$

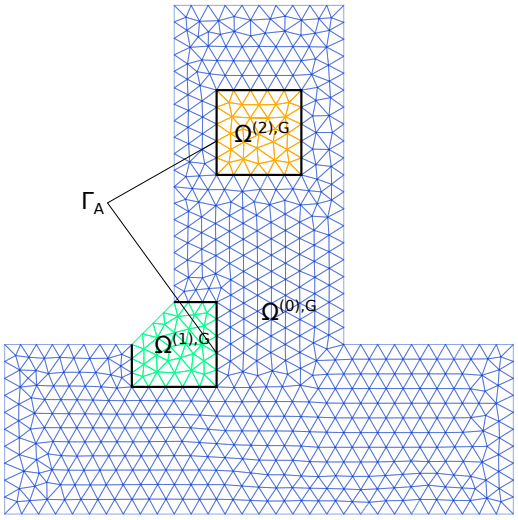


Figure 2.7: Global problem

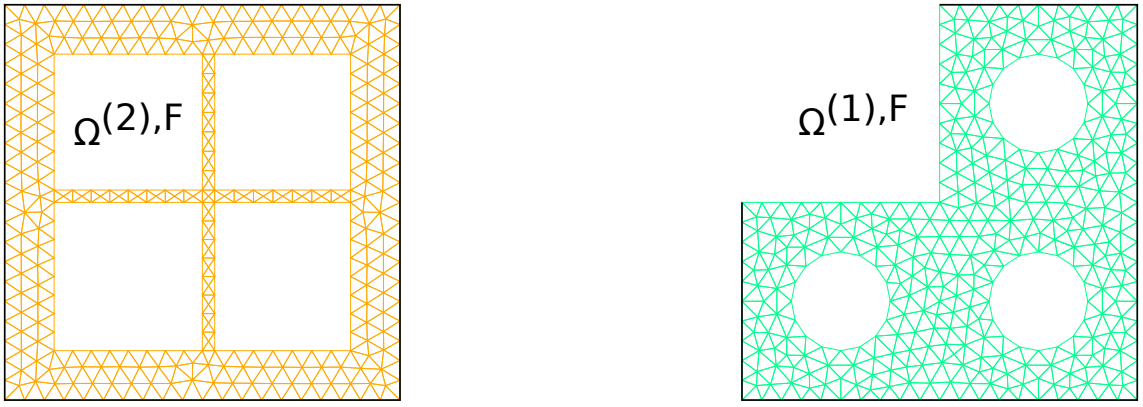


Figure 2.8: Refined zones of interest

correspond to the refined zones of interest.

The assembly of the local models on the zones of interest with the global model on the rest of the structure corresponds to what is known in the literature as the reference problem (Figure 2.9), and this problem is generally costly to build and solve. Indeed, in spite of the recent progress in meshing software, obtaining a good mesh for a complex part is still the most time-consuming step of an industrial simulation, before computation itself.

The multiscale coupling approaches used to find the solution of the reference problem without going through its resolution proceeds by two steps. First, an evaluation of the global coarse model on the whole structure, which allows estimating the large flows of efforts in the domain. Then, a second evaluation of the zones of interest with the local model on refined meshes to substitute the global model in these zones.

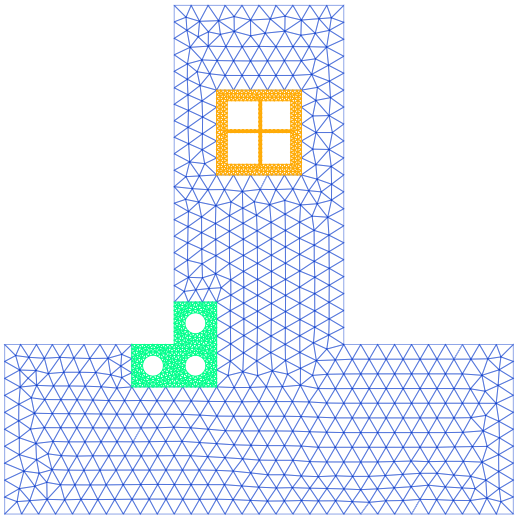


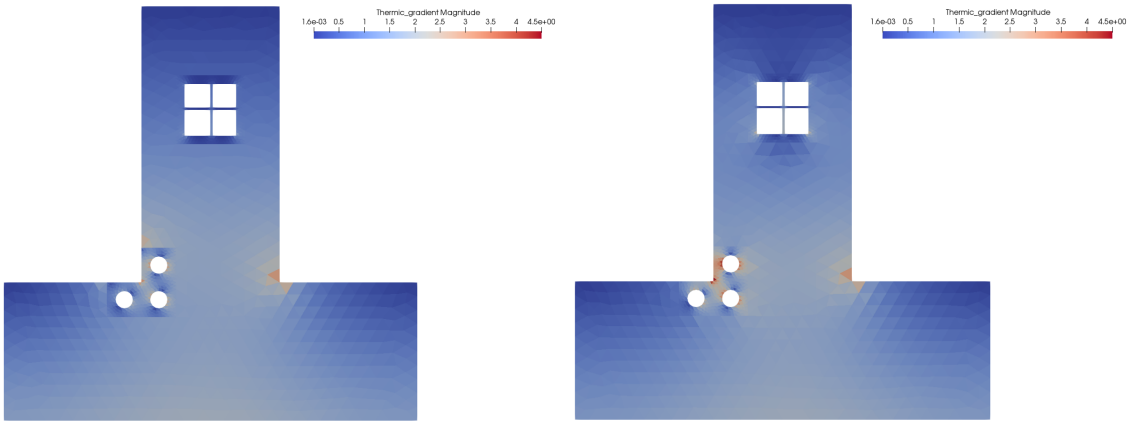
Figure 2.9: Reference problem

2.3.3 . Submodeling

An approach generally used by the industry is the well-known submodeling technique [58, 91, 22]. This coupling technique is often already integrated into commercial software. It consists in solving the discretized finite element global problem on the coarse scale of the structure and extracting the solution obtained on the interface of the zones of interest to send it as a Dirichlet conditions to the local models. Then the discretized finite element problem is solved on the local models by considering all the specificities of these zones of interest and using the condition at the boundary sent by the global problem. Finally, the solution obtained locally replaces the solution of the global model in these zones.

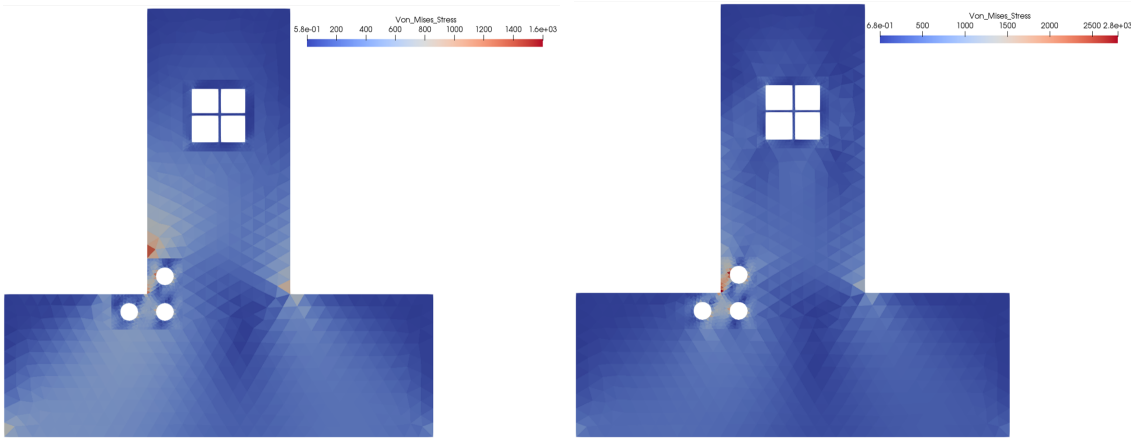
However, despite the advantages of this approach, it leads to large errors that can be materialized by the imbalance on the interface of the patches in the reference model, because the effects of local fine problems are not sent back to the global model, and interactions between local problems are thus impossible to account for.

To illustrate the limits of this method, the example of the 2D turbine blade as in figure 2.6 presented in the previous section is considered with thermal or elasticity linear problems. Then, significant quantities are considered for the mechanical analysis, like the Von-Mises and the thermal gradient.



(a) Submodeling solution (b) Reference solution

Figure 2.10: Comparison of the thermal gradient for the submodeling and reference approaches



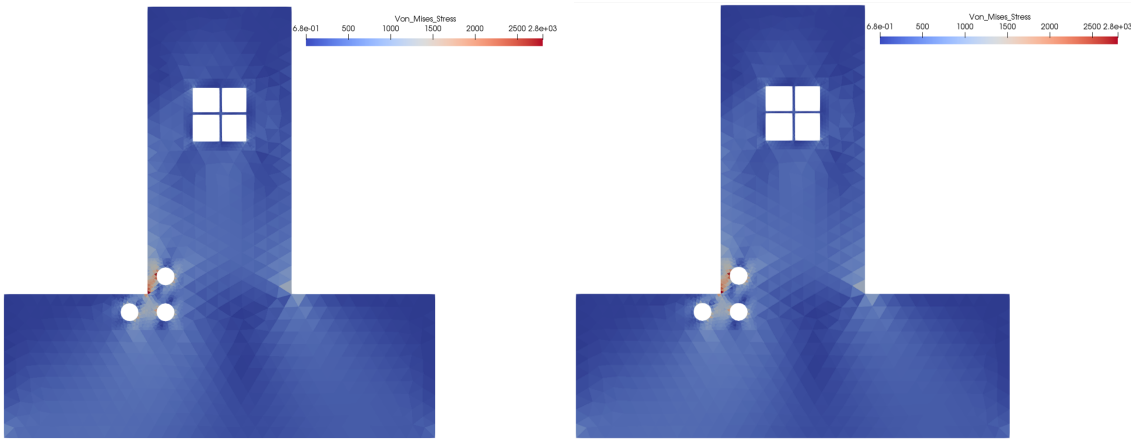
(a) Submodeling solution (b) Reference solution

Figure 2.11: Comparison of the von Mises stress for the submodeling and reference approaches

Figures 2.10 and 2.11 present the von Mises stress obtained from the resolution of a linear elasticity problem and the gradient of the resolution of a linear thermal problem. They show the quantities obtained in submodeling compared to the ones obtained in the reference problem. In both situations, a discontinuity corresponding to the disequilibrium at the interfaces can be observed for the quantities calculated by the submodeling method. The error observed in these results is due to not considering the effect of local problems on the global problem.

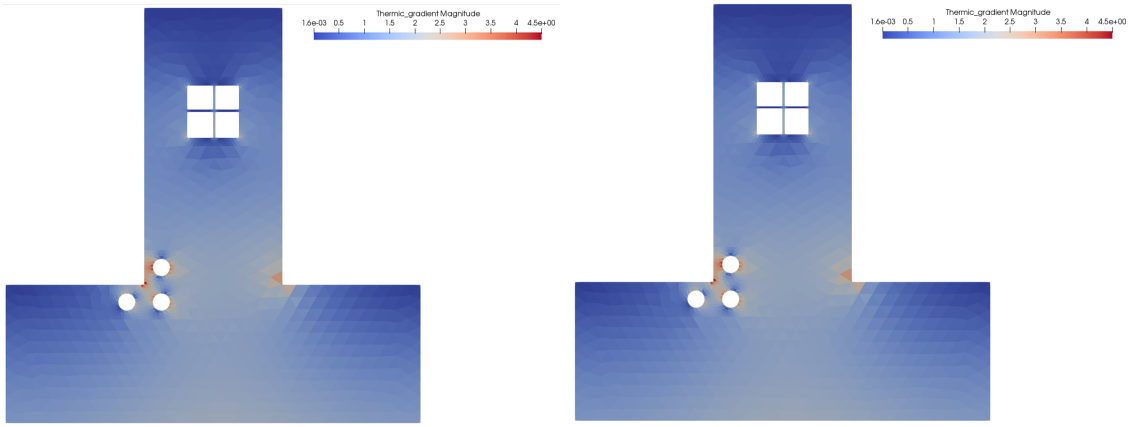
2.3.4 . Motivation

The global/local coupling has been introduced to correct the errors induced by the submodeling method while keeping the non-intrusive aspect. The idea is to put in place an iterative approach that sets up data exchange between the global model and the local models. This exchange takes into account the effect of each on the other, allowing an accurate evaluation of the reference solution.



(a) Global Local solution (b) Reference solution

Figure 2.12: Comparison of the von Mises stress for the Global local coupling and reference approaches



(a) Global Local solution (b) Reference solution

Figure 2.13: Comparison of the thermal gradient for the Global Local coupling and reference approaches

The method is non-invasive in the sense that it is adapted to coupling commercial (closed) and research software. Its first implementation in Abaqus was

proposed in [47]. Thus, it was implemented to couple research codes and legacy commercial software like Abaqus [15], Code_Aster [28], or Z-set [106].

The global/local coupling is strongly related to many reanalysis techniques [57, 108, 109], domain decomposition methods [54, 50], and multiscale methods [64].

The main idea is to start from a simplified global model and then allow local alterations (geometry, material, load, and mesh) to be inserted and their effect to be evaluated without heavy intervention on the initial model (see [1] for a pedagogic presentation). It was successfully applied in many contexts like the introduction of local plasticity and geometrical refinements [47], the computation of the propagation of cracks in a sound model [28], the evaluation of stochastic effects with deterministic computations [20, 86], the taking into account of the exact geometry of connectors in an assembly of plates [52], in the NURBS context in order to simplify the modeling of local behavior within a NURBS patch [16], in the analysis of composite structures with elastic shell representation at global scale and solid modeling at local scale [53]. In [66], the method was used for the coupling of a global model described with an IsoGeometric Analysis (IGA) and a local models descritized using the standard finite elements methods. In [28] the method was used in order to implement a nonlinear domain decomposition method [59, 23, 55, 85] in a non-invasive manner in Code_Aster. Extension of the approach to explicit dynamics was proposed in [12], improved in [13], and applied to the prediction of delamination under impact loading in [14].

2.3.5 . Principle of the method

The classical scenario is illustrated in Figure 2.14. A linear global coarse model is used to describe a large structure. After the initial computation (Figure 2.7), some zones of interest $\Omega^{s,G}$ $(s > 0)$ are selected because some criterion has been exceeded or because it was known from the beginning that some details were missing in the Global model. This is the case for the presented illustration where geometrical details and adapted meshes are introduced in the fine modeling of the zones of interest $\Omega^{s,F}$. Material laws could also be modified. Fine computations are run in parallel on the patches using the global solution as Dirichlet boundary condition (for $s > 0$, the fine and global subdomains may differ, but their interface Γ^s must be the same $\Gamma^s = \Omega \cap \partial\Omega^{s,G} = \Omega \cap \partial\Omega^{s,F}$).

The error can be materialized by the lack of balance of the fluxes between the global zone not covered by patches, denoted by Ω^0 and the Fine models. As can be seen in Figure 2.11a, which shows the von Mises stress and where the fine models overwrite the global ones. There is a discontinuity at the interface which does not exist in the reference computation where all interactions are taken into account. See Figure 2.11b which corresponds to a direct computation of the reference model where the zones of interest are described with the fine models, see Figure 2.9.

With the residual being the lack of balance at the interface, it can be reinjected into the global model as an immersed Neumann condition on the interface. The

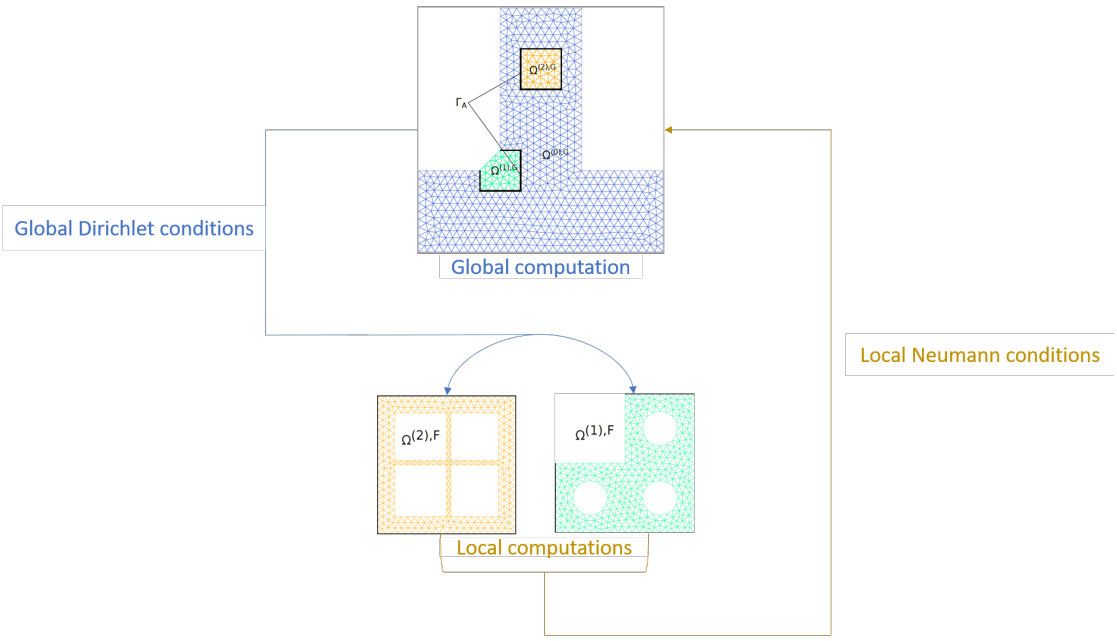


Figure 2.14: Global Local non-invasive coupling iteration

global model corrected by this interface load can be solved, followed by local Dirichlet problems and the iteration can go on. So at this point the global/local coupling is a simple iterative technique (a Richardson iteration for its simpler version) aiming at obtaining the Reference solution from computations carried on the Global and Fine models (that is to say without the potentially cumbersome creation of the Reference model) with minimal intervention on the models and software. One notes that if the calculation is stopped after the first iteration and the nodal reactions are not sent back to the global model a Submodeling method is performed.

2.3.6 . Parallel global/local coupling

Despite its robustness and non-intrusiveness, the parallelization of the global/local coupling remains a significant problem. As the literature explains, several case studies are found involving patches and then the parallelization of these patches using MPI. The coupling performance remains limited, even if the parallelization of these local patches can allow for a gain in performance. The synchronization at the end of each local computation imposes all the patches to wait for the end of the last calculation of the heaviest patches in terms of computation time, which raises the central problem of load balance. A second problem that this method suffers from, like all the two-step methods, is that the evaluation of the global problem is performed sequentially with the local problems. What this means is that even if the patches are well-balanced and parallelized, there will be a step of inactivity while the global one is performing its computation, which

makes the method not scalable using this classical synchronous parallelization techniques.

All the cited applications were developed in a synchronous framework that has been taken advantage of by accelerators (Aitken, quasi-Newton, Krylov). See [50], where the method is proved to be an implementation of an alternating Dirichlet-Robin approach where the Robin parameter corresponds to the condensation of the coarse domain covered by the patch. One can also see [28] where a parallel version of the method was presented with several adjacent patches similar and parallelized like a domain decomposition method. However, due to the alternating nature of the method, where the global model is performed alternating the local ones, its computational performance is inherently limited, with processors computing the local problems being idle while the global model is computing and the same for the global when the locals are computing.

To show the actual parallelization technique used, the case in Figure 2.6 is considered, with two zones of interest. The idea is to parallelize the two local problems and to keep an evaluation of the global problem on another processor.

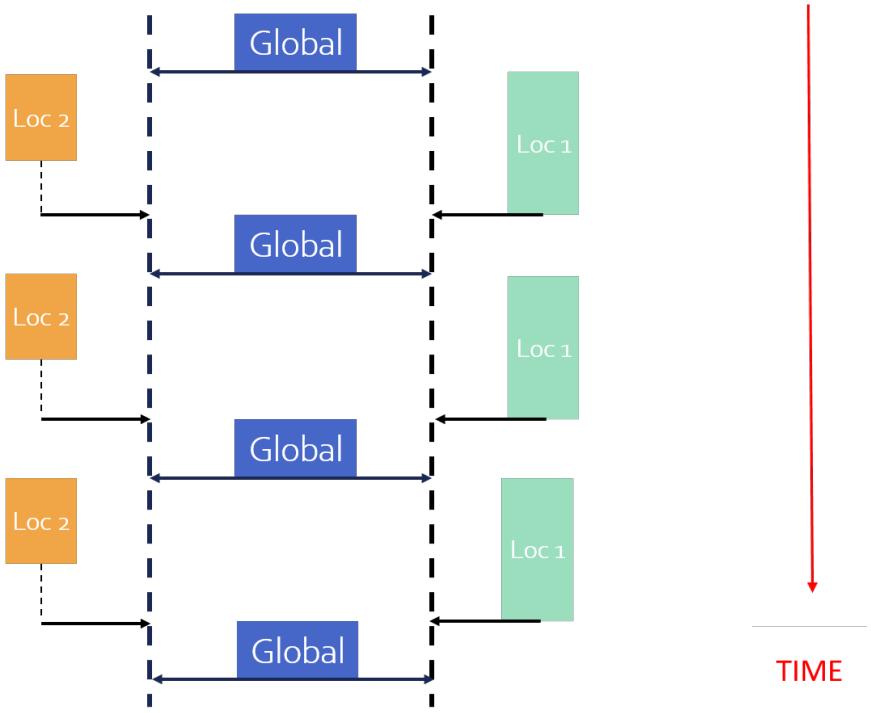


Figure 2.15: Synchronous iteration

Figure 2.15 shows a sequence of iterations of the global/local coupling parallelized with classical synchronous techniques. The global problem is presented in blue, and the locals are presented in orange and green. At first the influence of the nature of the alternating method can be seen during the global evaluation, where the local problems are waiting for an update from the global. In the

same way, it can be seen that the evaluation of the locals makes the global one wait. Also, the importance of these periods of inactivity can be seen, as well as the degradation that they bring to the method's performance. A second point that can be observed is the load imbalance between the two local problems, which forces problem 1 to wait for the end of the calculation of problem 2 to synchronize and send the necessary information to the global problem.

It can be imagined that the performance will decrease in the case of several patches with more important load imbalance or also in the case when the global problem has a significant size which will increase the duration of the sequential part.

In the next section the asynchronous parallel technique which correspond to the suppression of the synchronization step is presented. This enables the elimination of waiting periods.

2.4 . Asynchronous iterations

Asynchronous parallel computing techniques have been applied to domain decomposition methods to improve their performance and scalability. First, several works have targeted classical alternating Schwarz methods with overlap by proving its theoretical convergence with some numerical illustration. In [81] a new theoretical study of asynchronous algorithms with flexible communication is presented for the Schwarz alternating method, an application of this approach in linear and nonlinear cases can be found in [103, 102, 7]. In [37] an asynchronous version of weighted additive Schwarz method is investigated, its convergence is shown, and some numerical results show a significant improvement compared to the synchronous version. [18] presents an application of the asynchronous alternating Schwarz method to a large structural mechanic problem using the supercomputer Grid5000. In [65] an asynchronous additive Schwarz was investigated theoretically and numerically to solve nonlinear problem with finite difference scheme. In [105] the rate of convergence of asynchronous domain decomposition methods is studied in the context of convex optimization. Recently, [48] presents a novel scalable asynchronous two-level Schwarz method.

Other works have been interested in asynchronous domain decomposition methods without overlap. As mentioned in the introduction, those methods are well suited to the problem coming from mechanics, which interests this study. One of the first works realized in this framework is presented in [71] where a first convergence proof of the classical sub-structuring is presented with interesting numerical results for a 3D Poisson problem. Later, [43] found an improvement of the method and proposed a Gauss–Seidel scheme to alternate between resolution on the interface and on the local subdomains. In [46] a coarse space correction is added allowing a good scalability of the method. Other studies were interested in the asynchronous optimized Schwarz method [76] and one can find

several applications of it in [41, 30, 112]. In [45] the convergence of primal Schur domain decomposition has been established under suitable relaxation and in [111] an asynchronous multigrid method was presented within a shared memory systems.

2.4.1 . Idea

To introduce the basic notions of asynchronous parallel computing, a PDE problem is supposed with a finite element discretization which leads to a large linear system to be solved:

$$Ax = b \tag{2.15}$$

with $A \in R^{N\times N}$ matrix, $x \in R^N$ solution vector and $b \in R^N$ right-hand side vector.

2.4.1.1 . Classic iterative solver

Compared to direct solvers, iterative solvers evaluate the solution by successive approximation, they use much less memory and are more suited to parallelism than their direct counterparts at the price of an uncertain number of operations to achieve convergence.

The idea is to build a sequence x_k with $k \in \mathbb{N}$ such that :

$$\lim_{k\to\infty} x_k = A^{-1}b$$

The algorithm is stopped after reaching a certain chosen precision.

Generally the linear iterative algorithm can be written as :

$$\begin{cases} \text{Initial chosen: } x_0 \\ \text{Compute at the step k+1: } x_{k+1} = Tx_k + c \end{cases} \tag{2.16}$$

The iteration matrix T corresponds to a splitting of the matrix A, for example: $A = M - N$, with M a non singular matrix and then, the system in (2.15) can be written as:

$$\begin{aligned} (M-N)x &= b \\ Mx &= Nx + b \text{ (Fixed point equation)} \end{aligned} \tag{2.17}$$

From (2.17) the following linear iterative algorithm can be deduced:

$$x_{k+1} = M^{-1}Nx_k + M^{-1}b, \text{ For a given } x_0 \tag{2.18}$$

Thus $T = M^{-1}N$. The convergence of this algorithm is guaranteed [5] if $\rho(T) < 1$

2.4.1.2 . Parallel computing

The use of these iterative methods is usually accompanied by the implementation of a parallel process that allows the distribution of the computation task in several subtasks and calls upon a certain number of machines to take care of these subtasks. The idea is to reproduce these calculations simultaneously in

parallel. A significant problem in parallel computing is the communication management between these different machines, which generally corresponds to an assembly of the global solution from the solutions computed in parallel or the diffusion of global information to all the machines as a message of convergence, for example.

The idea is to split the vector x_{k+1} in the system (2.16) into L parts and to use L processors to compute each part in parallel

$$\begin{pmatrix} x_{k+1}^1 \\ \cdots \\ x_{k+1}^L \end{pmatrix} = \begin{pmatrix} f^1(x_k^1, \cdots, x_k^L) \\ \cdots \\ f^L(x_k^1, \cdots, x_k^L) \end{pmatrix}$$

The calculation of block j of the vector x^j with $1 \leq j \leq L$ at the iteration $k+1$ depends on all the other L blocks calculated in the previous iteration k. Therefore, an exchange of information is mandatory between the different blocks before passing from one iteration to the other.

A classical parallelization model that exists is the model with synchronized communications and calculations. The idea is that at each iteration, a step called synchronization is set up, where all the processors are blocked until the end of the calculation of the last processor to exchange the information and launch a new iteration.

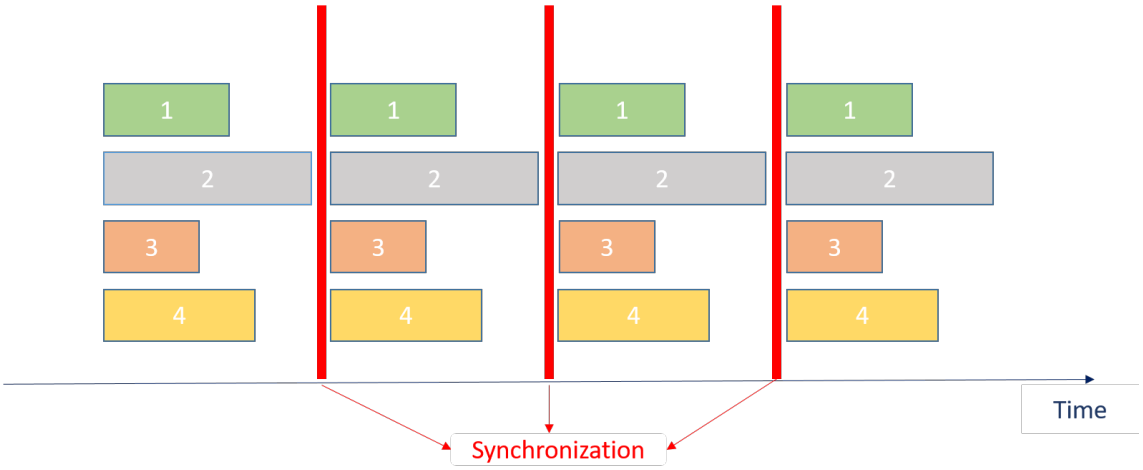


Figure 2.16: Synchronous model with L=4

Figure 2.16 presents a graph of synchronous iterations; with $L = 4$, there are four processors, and a part of the calculation takes place on each of the processors. At the end of each iteration, processors who finished early, in this case, processors 1,3 and 4, are obliged to wait for processor 2, which takes more time for calculation; these waiting times are modeled by the empty spaces between the moment when the calculation is finished and the moment when the exchange of data takes place.

The computations start at each iteration at the same time. The waiting times are considered a period of inactivity which can, with their repetition, degrade the performance of the parallel algorithms considerably

Several parameters can affect the performance of the used parallel approach:

- The architecture of the used machines (heterogeneity of the processors, number of processors, organization of the memory, speed of the interconnection network between the machines),
- The method of computation because there is no universal method that performs in all cases,
- The implementation of the stopping criterion for the iterative methods,
- The weight of the sequential irreducible part of the parallel method.

2.4.1.3 . Illustration

Asynchronous parallel computing, first introduced in [93], was then studied numerically in [19] under the chaotic relaxation technique to solve large linear systems ensuring its convergence with contraction properties. It has subsequently been the subject of several convergence studies. [79] generalized the study in [19] to nonlinear problems. The work in [8] allowed the first implementation of asynchronous methods on multiprocessor architectures with unbounded communication delays between processors. In [32] general convergence results for the asynchronous iterations based on the notion of classical contraction is presented. Recent work in [21] shows interesting theoretical and practical results for the Richardson iterations from the asynchronous point of view.

In [101, 38, 3], one can find a global review of asynchronous iterations from both theoretical and implementation points of view. Also in [100] one can find a simple and didactic presentation of the asynchronous iterations.

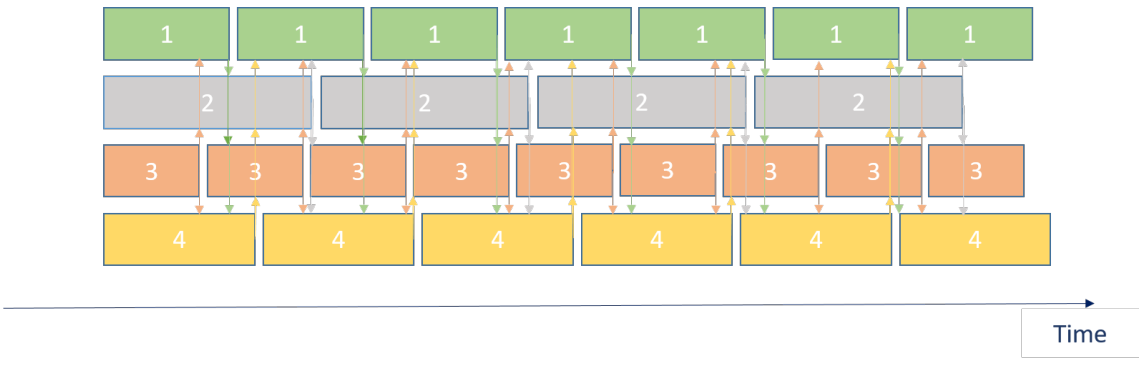


Figure 2.17: Asynchronous model with L=4

Parallel asynchronous iterations correspond to a fixed point without any synchronization step, i.e., all the processors start simultaneously at the beginning of

the calculation but stop only after the end of the calculation, meaning the algorithm's convergence. The data exchange (sending or receiving) is systematically done by each processor independent of the others once its calculation is finished.

In the most common case where there is an unbalanced load distribution or processors that do not have the same computation speed, the asynchronous model is very advantageous compared to the synchronous model. A drastic decrease in communication time is observed with the suppression of this synchronization step.

2.4.2 . Mathematical model

The parallel asynchronous model corresponding to equation 2.4.1.2 for solving the linear system 2.15 is written in the following form:

$$\begin{cases} x_0 = (x_0^1, \cdots, x_0^L) \\ x_{k+1}^j = \begin{cases} f^j(x_{s_1(k)}^1, \ldots\ldots\ldots\ldots x_{s_L(k)}^L) \text{ if } j \in H_k \\ x_k^j \text{ if } \notin H_k \end{cases} \\ j = 1, \cdots L \\ k = 0, 1, \cdots \end{cases} \tag{2.19}$$

The algorithm describes the behavior of an iterative process parallelized asynchronously using L processors. At each iteration $k+1$, the processor j computes x_{k+1}^j by using the data available by the other processors. If there is no new data, the value of the previous iteration x_k^j is kept.

- H_k is a non-empty subset corresponding to the updated components at the iteration k
- $r_j(k) = k - s_j(k)$ corresponds to the potential delay function of the processor j when computing the block j at iteration $k+1$.

2.4.3 . Convergence

As presented before, the asynchronous paradigm allows some processors to go faster than others by removing the synchronization barrier, which leads to more information being exchanged for some processors than others.

However, a local prediction cannot be made for the number of iterations. The advantages, as mentioned before, are numerous. However, the passage in asynchronous can be complicated for certain numerical methods by leading to their divergence because the conditions required to make them converge in synchronous, which are the same ones as that of the algorithm in sequential, are not sufficient to guarantee the convergence of the method in asynchronous. Thus, theoretical studies are essential for the proof of the convergence in the asynchronous case.

Since their introduction, several theoretical studies have been interested in converging asynchronous computation methods.

The first works of Chazan and Miranker have established sufficient but not necessary proof of convergence:

Theorem 1. *If $\rho(|T|) < 1$, with $|T| = (|T_{i,j}|)$ and ρ the spectral radius of the matrix, then the asynchronous iterations converge to the solution x^* whatever the initialization.*

However, if in the other case $\rho(|T|) > 1$, there exists a set of delays and an initial solution x_0 allowing the convergence of the asynchronous model. Several other techniques of analysis of the convergence of the asynchronous iterations are found, which are based on different approaches and with a different level of asynchronism in each case of study. First, we find the technique of partial ordering in [79, 80]. Other studies use the same techniques with the discrete maximum principle for nonlinear problem using the notion of M-Function [29], for non-singular linear and nonlinear systems

2.4.4 . Stopping Asynchronous iterations

The detection of convergence in asynchronous, or rather the stopping of the algorithm when the solution vector approaches the desired solution, is a major problem in the case of asynchronous iterations. Generally, whether in the sequential or parallel case in synchronous, the stopping or convergence criterion corresponds to the evaluation of a residual formed from a reduction operation applied to the other residuals calculated locally. This operation is thus realized with the help of synchronization. However, the asynchronous parallelization model is nondeterministic, and the notion of iterations is not the same as in the synchronous, where a certain number of iterations are performed by all processors before converging. In asynchronous, each one performs its iterations locally without taking into account the progress of the other and therefore establishes a protocol that allows evaluating the progress of the algorithm to know if the criterion of convergence is reached or not. While remaining in an asynchronous framework is a highly complex task either from a numerical analysis point of view or from a computer science point of view, i.e., programming, especially with multiprocessor architectures and distributed memories.

The most used asynchronous iteration-stopping techniques are based on the observation of the local state of the solution in each processor to detect if the convergence is reached locally and then send the information to a processor, which manages the iteration stopping. In the literature several approaches can be found. In [11, 10, 95] a special processor is designed to check the convergence locally on each processor and then use a specific protocol to check the global state of the algorithm convergence. In [6] a non-centralized approach to global convergence detection based only on local information required from the processors has been established, as well as a theoretical proof. The performance of the approach has been improved in [4]. In [82] a new criterion is considered for parallel linear fixed point methods using macro-iterations. In [75], a good global

evaluation of the residual using non-blocking communication techniques was introduced. This study was based on the use of a protocol based on the snapshot algorithm to assemble the global vector, an improvement of the performance of this approach is presented in [72]. The works in [44, 42] allow to implement a protocol-free distributed convergence detection technique. The idea is to diffuse the global residual using non-blocking global reduction techniques, enabling the calculation to stop once the global convergence is reached.

As, presented asynchronous convergence detection usually requires a specific detection protocol, which can sometimes be complex. In the studied case of global/local coupling the residual is assembled on the global model and it is always available. Therefore, the stopping criterion is the same as in synchronous because it is based on the calculation of the same residual at each iteration.

3 - Derivation of the non-invasive Global/local coupling

In this chapter, a new derivation of the global/local coupling is proposed. Indeed, there are many ways to derive the global/local coupling in the literature, hence the many names which were given to the method (Dirichlet-Robin, one variant of localized multigrid, semi-Schwarz Lagrange...). The most advanced theoretical framework was probably the one developed by [86] in a synchronous context with non-adjacent patches (which are equivalent to one non-connected patch).

This new interpretation makes it easy to treat as many patches as wanted, possibly adjacent patches, with analysis similar to what was proposed in [28], and also makes it possible to compare the method with the primal domain decomposition method. For simplicity, the continuous problem is not considered, and the focus will be on the properties of the discretized system. This chapter sets up the method. The convergence of the asynchronous iteration is the subject of the next chapter.

The method is presented for a quasi-static mechanical problem, but other cases like thermal problems can easily be deduced. A structure occupying a domain Ω is considered, submitted to given load and boundary conditions. One load increment is studied.

The system to be solved can be written as:

$$\begin{aligned} \operatorname{div}(\sigma) + f &= 0 \text{ in } \Omega \\ \sigma \cdot n &= g \text{ on } \partial_n \Omega \\ u &= u_d \text{ on } \partial_d \Omega \\ \sigma = \sigma(u) &= H(\nabla_s u) \text{ in } \Omega \end{aligned} \tag{3.1}$$

Where u is the displacement field, σ the Cauchy stress tensor, u_d is the given displacement on the Dirichlet part of the boundary $\partial_d\Omega$, g is the traction imposed on the Neumann part of the boundary $\partial_n\Omega$, f is the body load. For simplicity, infinitesimal strain is assumed, $\nabla_s u$ is the symmetric part of the gradient of displacement. The mechanical behavior is symbolized by the operator H. In the linear elasticity case, H is the Hooke tensor. More general behaviors can be considered like elasto(visco)plasticity, in which case internal variables should be added to the system but for simplicity they are not written.

The weak form reads: find $u \in H^1_d(\Omega)$ (i.e. satisfying Dirichlet boundary condition):

$$\int_\Omega \sigma(u) : \nabla_s v \, \mathrm{dx} = \int_\Omega f \cdot v \, \mathrm{dx} + \int_{\partial_n \Omega} g \cdot v \, \mathrm{ds} \qquad \forall v \in H^1_0(\Omega) \tag{3.2}$$

The term of the left-hand side is the opposite of the internal mechanical work, the term on the right-hand side is the mechanical work of external forces.

Using a variational formulation form of this problem, the convergence conditions of the Global/local coupling will be presented. A discrete form will be presented, and finally, after the deduction of the condensed form, the new derivation of the Global/local coupling is introduced.

3.1 . Variational formulation framework

The domain Ω is partitioned into $N+1$ non-overlapping subdomains (Ω^s). $\Gamma^s = \partial\Omega^s \cap \Omega$ is the interface of the subdomain Ω^s, and $\Gamma = \cup\Gamma^s$ the total interface. The equation (3.2) can be written under the form :

$$\text{Find } u \in V(\Omega), a(u,v) = l(v), \forall v \in V_0(\Omega) \tag{3.3}$$

with :

$$a(u,v) := \sum_{s=0}^{N} a^s(u_{|\Omega^s}, v_{|\Omega^s})$$

$$l(v) := \sum_{s=0}^{N} l^s(v_{|\Omega^s})$$

where $V(\Omega) \subset H^1(\Omega)$ is the subset of H^1-fields satisfying Dirichlet conditions and $V^0(\Omega)$ is the associated vector space. In the following, the Fine and Global modeling will be distinguished when needed. The following hypotheses are assumed:

- The patches are regular enough for trace operators to be well defined.
- The loads (boundary conditions and body load) lead to a linear H^1-continuous forms $l^{s,G}$ and $l^{s,F}$. Dirichlet conditions are regular enough to be continued in the domains.
- For $s \geq 0$, the Global patches are characterized by symmetric bilinear forms $a^{s,G}$ which are H^1-continuous, and coercive in the H^1 semi-norm; at least one patch is H^1-coercive (that is to say with enough Dirichlet conditions).
- If it exists, the patch 0 (complementary domain) is the same in the Fine and Global modelings.
- For $s > 0$, Fine patches are characterized by forms $a^{s,F}$ which are the sum of two terms:

  $$a^{s,F} = a_b^{s,F} + a_m^{s,F}$$

  where :

  - $a_b^{s,F}$: Bilinear H^1-continuous and coercive in the H^1 semi-norm (full H^1 norm for at least one patch $s > 0$)

- $a_m^{s,F}$: Linear continuous in the second variable, radially continuous and monotonous in the first variable.

Note that the subdomain $\Omega^{s,F}$ needs not coincide with its Global counterpart $\Omega^{s,G}$ as long their interface Γ^s is identical (it is possible to add holes in $\Omega^{s,F}$).

These hypotheses lead to the following properties:

- The Global model is linear and it can be derived from a quadratic energy, leading to a variational formulation with a symmetric coercive bilinear form.
- The patches-problem with Dirichlet condition on the interface are associated with continuous coercive strongly monotone formulations which lead to the uniqueness, existence and continuity of the solution with respect to the given load and to the Dirichlet condition in $W(\Gamma^s)$, the trace space of $V(\Omega^s)$.

The total trace space is written $W(\Gamma)$, we assume it is identical for the fine and global models $V(\Omega^G)$ and $V(\Omega^F)$.

Following [86], these properties make it possible to define well-posed continuous Dirichlet-to-Neumann maps on the subdomains. The assembled Global model can be used to define a norm on the interface fields. The strong monotonicity introduces a bounding from below (similar to the coercivity of bilinear forms) with a positive constant which is useful to prove the existence of a converging relaxed iteration.

These properties are preserved by a classical finite element formulation.

In the following subsections, the concepts needed to derive the Global/local coupling are defined.

3.1.1 . Reference problem

The reference problem, generally indexed by R, corresponds to the exact representation of the structure in terms of geometry and behavior. It is, therefore, the combination of the fine representation of the zones of interest $(\Omega^s)_{s>0}$ with the coarse representation of the rest of the structure (Ω^0) where no complex behavior or geometry are distinguished. The last one is generally called a complementary problem. In some situations, it does not exist, and the reference problem corresponds to the assembly of the local fine problems.

After a finite element discretization, this reference problem is written in the following variational form:

$$\text{Find } u \in V^R, \text{ such that } a^R(u,v) = l^R(v), \forall v \in V_0^R \tag{3.4}$$

where V^R is the subspace of $H^1(\Omega^R)^d$ of fields satisfying the Dirichlet conditions. V_0^R is the associated vector subspace. It inherits the classical H^1 Hilbert structure. $\|u\|$ is the H^1 norm and $|u|$ is the H^1 seminorm.

As mentioned earlier, Ω^R is partitioned into N +1 sufficiently regular non-overlapping subdomains $\Omega^{s,F}, (0 \leq s \leq N)$. It has been assumed that the forms in (3.4) have an additive structure with respect to the domain: $\forall (u,v) \in V^R \times V_0^R$,

$$a^R(u,v) := \sum_{s=0}^{N} a^{s,F}(u_{|\Omega^{s,F}}, v_{|\Omega^{s,F}})$$
$$l^R(v) := \sum_{s=0}^{N} l^{s,F}(v_{|\Omega^{s,F}})$$

For $s \geq 0$, $V^{s,F}$ is defined as, the space of functions of V^R restricted to $\Omega^{s,F}$: $u^s \in V^{s,F} \Leftrightarrow \exists u \in V^R$ such that $u^s = u_{|\Omega^{s,F}}$.

The following assumptions are made and the necessary constants for our analysis are introduced:

1. For $s \geq 0$, $l^{s,F}$ is a linear continuous form on $V_0^{s,F}$. Its norm is introduced as:

$$\forall v^{s,F} \in V_0^{s,F}, |l^{s,F}(v^{s,F})| \leq \|l^{s,F}\|^{\star}_{V^{s,F}} \|v^{s,F}\|_{V^{s,F}} \tag{3.5}$$

2. Subdomain $s = 0$ can be non-existent. If it exists, $a^{0,F}$ is a symmetric bilinear continuous semi-coercive form:

$$\begin{aligned} &\exists M^{0,F} > 0, \forall (u^{0,F}, v^{0,F}) \in (V_0^{0,F})^2, |a^{0,F}(u^{0,F}, v^{0,F})| \leq M^{0,F} \|v^{0,F}\|_{V^{0,F}} \|u^{0,F}\|_{V^{0,F}} \\ &\exists C^{0,F} \geqslant 0, \forall u^{0,F} \in (V_0^{0,F}), a^{0,F}(u^{0,F}, u^{0,F}) \geq C^{0,F} \|u^{0,F}\|^2_{V^{0,F}} \end{aligned} \tag{3.6}$$

3. For $s > 0$, $a^{s,F}$ is semi-continuous, semi-coercive, linear in the second variable, and strongly monotone in the first variable:

$$\begin{aligned} &\exists M^{s,F} > 0, C^{s,F} \geq 0, \forall (u_1^{s,F}, u_2^{s,F}, v^{s,F}) \in (V^{s,F})^2 \times V_0^{s,F} \\ &|a^{s,F}(u_1^{s,F}, v^{s,F}) - a^{s,F}(u_2^{s,F}, v^{s,F})| \leq M^{s,F} \|v^{s,F}\|_{V^{s,F}} \|u_1^{s,F} - u_2^{s,F}\|_{V^{s,F}} \\ &a^{s,F}(u_1^{s,F}, u_1^{s,F} - u_2^{s,F}) - a^{s,F}(u_2^{s,F}, u_1^{s,F} - u_2^{s,F}) \geq C^{s,F} \|u_1^{s,F} - u_2^{s,F}\|^2_{V^{s,F}} \end{aligned} \tag{3.7}$$

4. At least one of the forms $a^{s,F}$ for $s \geq 0$, is (strictly) coercive, which corresponds to the Reference problem containing sufficiently many Dirichlet conditions.

Under these hypotheses, the reference problem admits a unique solution.

3.1.2 . Global problem

The global problem (superscript G) is a classical discrete finite element model corresponding to a simplification of the discrete reference problem, suited for fast computation and capable of giving a correct representation of the long range

fluxes. It does not need to be locally accurate. For instance, for a slender structure, a shell finite element model, known to correctly transfer generalized forces, can be used as a global model [51]. Also, the material constitutive law can be simplified, as well as the topology and the geometry. In general, the global problem is sufficiently simple to be solved with a sequential solver.

The same notation as in the previous section is used, and for $s \geq 0$ a simplified version of the problem is introduced, written with superscript G, for global.

The following hypotheses are assumed:

- For $s \geq 0$, all $a^{s,G}$ must be symmetric bilinear continuous semi-coercive forms (with at least one strictly coercive).
- Subdomain 0 is identical in the Global and Reference formulations, $a^{0,G} = a^{0,F}$ and $l^{0,G} = l^{0,F}$.

The Global forms can be built from a simplified geometry of the subdomain, $\Omega^{s,G}$, as long as the Global and Fine interfaces match: $\partial\Omega^{s,G} \cap \Omega = \partial\Omega^{s,F} \cap \Omega = \Gamma^s$. For instance, in Figure 2.7, the Global subdomains do not possess holes and a corner is omitted.

Thus V^G and $(V^{s,G})$ can be defined independently of the Fine model, as long as the trace spaces W and (W^s) are the same.

The Global problem corresponds then to the "linearized and simplified" version of the Reference problem.

The variational formulation reads:

$$\text{Find } u \in V^G, \text{ such that } a^G(u,v) = l^G(v), \ \forall v \in V_0^G \tag{3.8}$$

with : $\forall (u,v) \in V^G \times V_0^G$,

$$a^G(u,v) := \sum_{s=0}^{N} a^{s,G}(u_{|\Omega^{s,G}}, v_{|\Omega^{s,G}}), \quad l^G(v) := \sum_{s=0}^{N} l^{s,G}(v_{|\Omega^{s,G}}) \tag{3.9}$$

3.1.3 . Global/local coupling

The Global/local coupling consists in using the Global problem with an extra ingredient: the interface load $p \in W^\star$.

The global problem variational formulation reads:

$$\text{Find } u \in V^G, \text{ such that } a^G(u,v) = l^G(v) + \langle p, T^G v \rangle, \forall v \in V_0^G \tag{3.10}$$

where $T^G : V^G \to W$ is the trace operator, and the duality bracket is in (W^*, W).

The resolution of the global problem allows obtaining the unknown global displacement extracted on the total interface Γ: $u_\Gamma^G = T^G u^G$ which serves as Dirichlet condition for the Fine problems, imposed by Lagrange multipliers $(\lambda^{s,F})_{s\geq 0}$. The

local trace operator $T^{s,F} : V^{s,F} \longrightarrow W^s$ is used.

$$\text{For given } u_\Gamma^{s,G} \in W^s, \text{ find } (u^{s,F}, \lambda^{s,F}) \in V^{s,F} \times W^{s,\star}, \text{s.t}:$$
$$\begin{cases} a^{s,F}(u^{s,F}, v^{s,F}) - \langle \lambda^{s,F}, T^{s,F} v^{s,F} \rangle = l^{s,F}(v^{s,F}), \ \forall v^{s,F} \in V_0^{s,F} \\ \langle \mu, T^{s,F} u^{s,F} - u_\Gamma^{s,G} \rangle = 0, \forall \mu^s \in W^{s^\star} \end{cases} \tag{3.11}$$

Under the chosen hypotheses, these problems are well-posed, they have a unique solution which depends continuously on the inputs [97]. The signs are chosen such that $\lambda^{s,F}$ has the physical meaning of an imposed flux on the boundary of the subdomain.

The residual can then be computed as:

$$r = -\sum_{s \geq 0} A^s \lambda^{s,F} \tag{3.12}$$

where $A^s : W^{s,*} \to W^*$ is the injection operator. The coupling iteration can be written as:

$$p = p + \omega r \tag{3.13}$$

where $\omega > 0$ is a relaxation parameter.

3.2 . Discrete formulation framework

This section describes the global/local coupling method in a finite element framework. In order to be more practical, the presentation is ordered in agreement with the unfolding of the iteration. More, the general case of a nonlinear global problem is considered, which is not covered by the current convergence theory.

3.2.1 . Global problem

3.2.1.1 . Initial global problem

After classical finite element discretization, the (uncorrected) global problem to be solved can be written as:

$$\mathbf{f}_{int}^G(\mathbf{u}) + \mathbf{f}_{ext}^G = 0 \tag{3.14}$$

where $\mathbf{f}_{int}^G$ is the vector of internal forces and $\mathbf{f}_{ext}^G$ the vector of external forces. Their m^{th} component reads:

$$\begin{aligned} (\mathbf{f}_{int}(\mathbf{u}))_m &= -\int_\Omega \sigma_h(\mathbf{u}) : \mathrm{grad}_s(\phi_m)\,\mathrm{dx} \\ (\mathbf{f}_{ext})_m &= \int_\Omega f \cdot \phi_m\,\mathrm{dx} + \int_{\partial_n\Omega} g \cdot \phi_m\,\mathrm{ds} \end{aligned} \tag{3.15}$$

where ϕ_m is the finite element shape function associated with the m^{th} degree of freedom.

We introduce a decomposition of the *global* domain into $N+1$ non-overlapping subdomains (Ω^s) with $s \in [0..N]$. Subdomains are supposed to be sets of connected elements so that the decomposition is matching at the interface.

The interface is defined as the boundary degrees of freedom of the patches. For each subdomain $\Gamma^{s,G} = \bigcup_j (\partial\Omega^{s,G} \cap \partial\Omega^{j,G}) \setminus \partial_d\Omega^G$, the interface is constituted by degrees of freedom shared with other subdomains, excluded Dirichlet degrees of freedom. Globally, $\Gamma^G = \bigcup_s \Gamma^{s,G}$.

The boundary (interface) degrees of freedom (index Γ) can be separated, from internal degrees of freedom (index i). The trace operators which extract the boundary degrees of freedom of a vector can be defined as $\mathbf{T}^{s,G} : \Omega^{s,G} \to \Gamma^{s,G}$ and $\mathbf{T}^G : \Omega^G \to \Gamma^G$, for instance $\mathbf{T}^{s,G}\mathbf{u}^{s,G} = \mathbf{u}^{s,G}_\Gamma$. The transpose is an extension-by-zero operator.

Let $\mathbf{A}^s$ be the interface injection operator $\Gamma^{s,G} \to \Gamma^G$ like in primal domain decomposition methods as presented in the Section 2.2.6

For a given $\mathbf{u}^G$ solution to (3.14), the nodal reaction $\lambda^{s,G}$ are defined at the boundary of the subdomains. It can be computed with different approaches:

- Algebraic post-processing, using the subdomain injection operator $\bar{\mathbf{A}}^s : \Omega^{s,G} \to \Omega^G$:
$$\mathbf{f}^{s,G}_{int}(\bar{\mathbf{A}}^s\mathbf{u}^G) + \mathbf{f}^{s,G}_{ext} + \mathbf{T}^{s^T}\boldsymbol{\lambda}^{s,G} = 0 \tag{3.16}$$
- Integration:
$$\boldsymbol{\lambda}^{s,G}_m = \int_{\Omega^{s,G}} \sigma_h(\mathbf{u}^{s,G}) : \mathrm{grad}_s(\phi_m)\,\mathrm{dx} - \int_{\Omega^{s,G}} f\cdot\phi_m\,\mathrm{dx} - \int_{\partial_n\Omega^{s,G}} g\cdot\phi_m\,\mathrm{ds} \tag{3.17}$$
where m is an interface degree of freedom of subdomain $\Omega^{s,G}$.
- Solution of a Dirichlet problem on the subdomain:
$$\text{Find } (\mathbf{u}^{s,G}, \boldsymbol{\lambda}^{s,G}) \text{ solution to:}$$
$$\begin{cases} \mathbf{T}^{s,G}\mathbf{u}^{s,G} = \mathbf{A}^{s^T}\mathbf{T}^G\mathbf{u}^G \\ \mathbf{f}^{s,G}_{int}(\mathbf{u}^{s,G}) + \mathbf{f}^{s,G}_{ext} + \mathbf{T}^{s^T}\boldsymbol{\lambda}^{s,G} = 0 \end{cases} \tag{3.18}$$

Note that formula (3.17) or (3.16) are not always implemented in commercial software, in which case the computation of $\boldsymbol{\lambda}^{s,G}$ must resort to (3.18).

Remark 2. In the linear case, $\mathbf{f}^{s,G}_{int}(\mathbf{u}^{s,G}) = -\mathbf{K}^{s,G}\mathbf{u}^{s,G}$ where $\mathbf{K}^{s,G}$ is the stiffness matrix of the subdomain, $\mathbf{f}^{s,G}_{ext} = \mathbf{f}^{G,s}$ the vector of generalized forces, and:

$$\begin{pmatrix} \mathbf{K}^{s,G}_{ii} & \mathbf{K}^{s,G}_{i\Gamma} \\ \mathbf{K}^{s,G}_{\Gamma i} & \mathbf{K}^{s,G}_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} \mathbf{u}^{s,G}_i \\ \mathbf{u}^{s,G}_\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{f}^{G,s}_i \\ \mathbf{f}^{G,s}_\Gamma + \boldsymbol{\lambda}^{s,G} \end{pmatrix} \tag{3.19}$$

3.2.1.2 . Corrected global problem

During the coupling iterations, the following modified global problem is used:

$$\mathbf{f}_{int}^{G}(\mathbf{u}) + \mathbf{f}_{ext}^{G} + \mathbf{T}^{G^T}\mathbf{p}_\Gamma = 0 \tag{3.20}$$

$\mathbf{p}_\Gamma$ is an interface load, it introduces a lack of balance between nodal reactions where

$$\sum \mathbf{A}^s \boldsymbol{\lambda}^{s,G} = \mathbf{p}_\Gamma \tag{3.21}$$

3.2.2 . Fine problems

The Fine problems are set on the refined subdomains $\Omega^{s,F}$. For a good matching of the models, the interface is assumed to be geometrically identical in the global and fine models. It coincides with the edges of finite elements in all models. Anyhow, non-matching interpolations are tolerated, and the Global-to-Fine transfer matrices ($\mathbf{J}^s$) are introduced, enabling the definition of Fine Dirichlet problems with boundary conditions coming from the Global model. The transpose of these matrices enables to transfer of nodal reactions from the local models to the global one.

The Fine version of the local interfaces is written $\Gamma^{s,F}$ and the local fine trace operators are $\mathbf{T}^{s,F} : \Omega^{s,F} \to \Gamma^{s,F}$. $\mathbf{J}^s : \Gamma^{s,G} \to \Gamma^{s,F}$ is the interpolation operator between the meshes.

The fine problems can be written as:

$$\begin{cases} \text{For given } \mathbf{u}_\Gamma^G \text{ on } \Gamma, \text{for all } s > 0, \text{ find } \mathbf{u}^{s,F} \text{ in } \Omega^{s,F} \text{ and } \boldsymbol{\lambda}^{s,F} \text{ on } \Gamma^s \text{ s.t.} \\ -\mathbf{f}_{int}^{s,F}(\mathbf{u}^{s,F}) = \mathbf{f}^{s,F} + \mathbf{T}^{s,F^T}\boldsymbol{\lambda}^{s,F} \\ \mathbf{T}^{s,F}\mathbf{u}^{s,F} = \mathbf{J}^s\mathbf{A}^{s^T}\mathbf{u}_\Gamma^G \end{cases} \tag{3.22}$$

Thanks to the chosen hypothesis, these problems are well-posed.

The strong monotonicity property resulting from the previous assumptions translates into:

$$-\left(\mathbf{f}_{int}^{s,F}(\mathbf{u}^{s,F}) - \mathbf{f}_{int}^{s,F}(\mathbf{v}^{s,F})\right)^T\left(\mathbf{u}^{s,F} - \mathbf{v}^{s,F}\right) \geqslant \gamma^{s,F}\left\|\mathbf{u}^{s,F} - \mathbf{v}^{s,F}\right\|^2 \tag{3.23}$$

For simplicity, the discrete Euclidean norm is used, which makes $\gamma^{s,F} > 0$ dependent on the mesh.

3.2.3 . Reference problem

The Reference problem as presented before is the collection of Fine problems connected to the same interface displacement $\mathbf{u}_\Gamma^G$ and such that the nodal reactions $\lambda^{s,F}$ are in balance once projected back on the Global interface.

First, the clarification of the role played by Subdomain 0, which might be non-existent is needed. It is a subdomain, sometimes called Complement domain in

the global/Local literature, where the Fine and Global model coincide (same geometry Ω^0, same properties, same load, same approximation).

In particular, it is associated with a linear system and simply serves to process the nodal reaction $\boldsymbol{\lambda}^0$ using (3.18), (3.17) or (3.16). In some implementations, it is proposed to be computed as:

$$\mathbf{A}^0\boldsymbol{\lambda}^0 = \mathbf{p}_\Gamma - \sum_{s>0} \mathbf{A}^s\boldsymbol{\lambda}^{s,G} \tag{3.24}$$

because it may be more convenient to compute $(\lambda^{s,G})$ from the global subproblems for $s > 0$.

Now, the Reference problem can be formulated as:

$$\begin{cases} \text{Find } \mathbf{u}_\Gamma^G \text{ on } \Gamma \text{ s.t} \\ \mathbf{r}_\Gamma := -\left(\boldsymbol{\lambda}^0 + \sum_{s=1}^{N} \mathbf{A}^s\mathbf{J}^{s^T}\boldsymbol{\lambda}^{s,F}\right) = 0 \\ \text{where the reactions are obtained from (3.22) and (3.24).} \end{cases} \tag{3.25}$$

The minus sign is meant to make $\mathbf{r}$ agree with the classical definition of the residual when the system is linear.

3.2.4 . Global/local coupling

The aim of the coupling is to achieve (3.25) using (3.20),(3.22),(3.18) or ((3.17)) or ((3.16)) or (3.24). To do so, a simple modified Richardson iteration is used.

Starting from $\mathbf{p}_\Gamma = 0$, $\mathbf{u}^G$ is computed in ((3.20)), then $\mathbf{u}_\Gamma^G$ is used as a Dirichlet condition to compute the Fine reactions $\boldsymbol{\lambda}^{s,F}$ using (3.22) and (3.24), finally the residual $\mathbf{r}_\Gamma$ is the lack of balance between the nodal reactions as in (3.25).

If the residual is not small enough, the interface load is updated as $\mathbf{p}_\Gamma = \mathbf{p}_\Gamma + \omega\mathbf{r}_\Gamma$.

It can be proved that under the chosen hypothesis, there exist $0 < \omega_{\max}$ such that the iteration converge for all $0 < \omega < \omega_{\max}$. In practice, dynamic relaxation through Aitken's δ^2 technique gives an excellent performance.

3.3 . Condensation at the interface

Because all manipulated data are associated with mechanical problems in balance, and thanks to the well-posedness of the local Dirichlet problems induced by the chosen hypotheses, the convergence is driven by the interface, and it is convenient to formally eliminate internal degrees of freedom and to condense all problems at the interface.

To do so, Dirichlet-to-Neumann (DtN) maps is introduced, and the nodal reactions $\boldsymbol{\lambda}^s$ on the subdomain's interface are written as a function of the imposed

Dirichlet condition $\mathbf{u}_\Gamma^s$ (the following formulas apply to both Global and Fine subdomains):

$$\boldsymbol{\lambda}^s = \mathbf{s}^s(\mathbf{u}_\Gamma^s; \mathbf{f}_{ext}^s) \quad \text{means} \quad \begin{cases} \exists \mathbf{v}^s \text{ such that } \begin{cases} \mathbf{T}^s \mathbf{v}^s = \mathbf{u}_\Gamma^s \\ \left(\mathbf{f}_{int}^s(\mathbf{v}^{s,G}) + \mathbf{f}_{ext}^s\right)_i = 0 \end{cases} \\ \boldsymbol{\lambda}^s := -\left(\mathbf{f}_{int}^s(\mathbf{v}^{s,G}) + \mathbf{f}_{ext}^s\right)_\Gamma \end{cases} \tag{3.26}$$

Remark 3. In the linear case, the DtN operator is affine:

$$\boldsymbol{\lambda}^s = \mathbf{S}^s \mathbf{u}_\Gamma^s - \mathbf{b}^s \quad \text{with} \quad \begin{cases} \mathbf{S}^s = \mathbf{K}_{\Gamma\Gamma}^s - \mathbf{K}_{\Gamma i}^s (\mathbf{K}_{ii}^s)^{-1} \mathbf{K}_{i\Gamma}^s \\ \mathbf{b}^s = \mathbf{f}_{ext\,\Gamma}^s - \mathbf{K}_{\Gamma i}^s (\mathbf{K}_{ii}^s)^{-1} \mathbf{f}_{ext,i}^s \end{cases} \tag{3.27}$$

$\mathbf{S}^s$ is the Schur complement matrix and $\mathbf{b}^s$ the condensed right-hand side.

If the monotonicity property is applied to the two fields obtained from the computation of Dirichlet problems, a monotonicity property for the Dirichlet to Neuman map is obtained:

$$\left(\mathbf{s}^{s,F}(\mathbf{u}_\Gamma^{s,F}; \mathbf{f}^{s,F}) - \mathbf{s}^{s,F}(\mathbf{v}_\Gamma^{s,F}; \mathbf{f}^{s,F})\right)^T \left(\mathbf{u}_\Gamma^{s,F} - \mathbf{v}_\Gamma^{s,F}\right) \geqslant \gamma^{s,F} \left\|\mathbf{u}^{s,F} - \mathbf{v}^{s,F}\right\|^2 \geqslant \gamma^{s,F} \left\|\mathbf{u}_\Gamma^{s,F} - \mathbf{v}_\Gamma^{s,F}\right\|^2 \tag{3.28}$$

Similarly, the continuity of the Dirichlet problem with respect to the boundary condition leads to a bounding of the form:

$$\left\|\mathbf{s}^{s,F}(\mathbf{u}_\Gamma^{s,F}) - \mathbf{s}^{s,F}(\mathbf{v}_\Gamma^{s,F})\right\| \leqslant M^{s,F} \left\|\mathbf{u}_\Gamma^{s,F} - \mathbf{v}_\Gamma^{s,F}\right\| \tag{3.29}$$

3.3.1 . Global problem

The corrected global problem (3.20) can be written in condensed form, for given interface load $\mathbf{p}_\Gamma$, find interface displacement $\mathbf{u}_\Gamma^G$ such that:

$$\sum_{s=0}^{N} \mathbf{A}^s \mathbf{s}^{s,G}(\mathbf{A}^{s^T} \mathbf{u}_\Gamma^G; \mathbf{f}^{s,G}) = \mathbf{p}_\Gamma \tag{3.30}$$

i.e.

$$\mathbf{u}_\Gamma^G = \mathcal{S}^{G^{-1}}(\mathbf{f}^G; \mathbf{p}_\Gamma^G) \tag{3.31}$$

In the case of a linear global problem and in order to connect subdomain together, the assembly operators $\mathbf{A}^s$ that map the local interface on the Global interface are used. Thanks to these operators, the Global problem can be rephrased as:

$$\underbrace{\left(\sum_{s=0}^{N} \mathbf{A}^s \mathbf{S}^{s,G} \mathbf{A}^{s^T}\right)}_{\mathbf{S}^G} \mathbf{u}_\Gamma^G = \underbrace{\left(\sum_{s=0}^{N} \mathbf{A}^s \mathbf{b}^{s,G}\right)}_{\mathbf{b}^G} + \mathbf{p}_\Gamma \tag{3.32}$$

One can recognize a primal domain decomposition [68] with the original extra load $\mathbf{p}_\Gamma$. $0 < \gamma^G < M^G$ the bounds of the spectrum of $\mathbf{S}^G$.

3.3.2 . Reference problem

Similarly, the condensed Reference problem is obtained by assembling all the Fine subproblems together on the interface coarse grid.The fine displacement at the interface is forced to follow the equation $\mathbf{u}_\Gamma^{s,F} = \mathbf{J}^s \mathbf{u}_\Gamma^{s,G}$ and the balance of reactions is evaluated on the global interface. This leads to the condensed formulation of the *reference* problem where *fine* models are used (except on subdomain Ω^0), find reference interface displacement $\mathbf{u}_\Gamma^R$ such that:

$$\sum_{s=0}^{N} \mathbf{A}^s \mathbf{J}^{s^T} \mathbf{s}^{s,F}(\mathbf{J}^s \mathbf{A}^{s^T} \mathbf{u}_\Gamma^R; \mathbf{f}^{s,F}) = 0 \tag{3.33}$$

i.e.:

$$\mathcal{S}^F(\mathbf{u}_A^R; \mathbf{f}^F) = 0 \tag{3.34}$$

Up to the mesh incompatibility which is not totally standard, this system is often referred to as the starting point of the primal nonlinear domain decomposition of [85], where it is proposed to apply a Newton-Raphson linearization and use classical Neumann-Neumann preconditioner for the tangent system together with balancing coarse space to ensure scalability. It is also the starting point of HPC iterative solvers like BDD [78] or BDDC [27] were efficient parallel multilevel preconditioners are designed in linear, and relocalization strategies are possible in the nonlinear case [24, 85, 61].

3.3.3 . Global/local coupling

The global/local coupling as presented before is a simple technique to iteratively find the solution to the reference problem using only simpler computations, by alterning beetwen the global computation and the local one.

The global/local coupling can be formulated as:

$$\begin{aligned} &\text{Find extra load } \mathbf{p}_\Gamma^G \text{ such that} \\ &\mathcal{S}^F(\mathcal{S}^{G^{-1}}(\mathbf{f}^G; \mathbf{p}_\Gamma^G); \mathbf{f}^F) = 0 \end{aligned} \tag{3.35}$$

If the following notation is used:

$$\mathcal{S}^F = (\mathcal{S}^F - \mathcal{S}^G) + \mathcal{S}^G$$

The previous equation can be rewritten as: Find extra load $\mathbf{p}_\Gamma^G$ such that:

$$\mathbf{r} := -\left(\mathbf{p}_\Gamma^G - (\mathcal{S}^G - \mathcal{S}^F)(\mathcal{S}^{G^{-1}}(\mathbf{f}^G; \mathbf{p}_\Gamma^G); \mathbf{f}^F)\right) = 0 \tag{3.36}$$

Note that the sign is adjusted so that the residual is consistent with the convention $\mathbf{r} := \mathbf{b} - \mathbf{M}\mathbf{x}$ when solving $\mathbf{M}\mathbf{x} = \mathbf{b}$. Mechanically speaking, the residual is the opposite of the lack of balance between subdomains in the *reference* problem.

Equation(3.36) suggests using a stationary iteration as described in Algorithm 1. One recognizes a modified Richardson iteration. The fundamental result, which

Algorithm 1: Sequential stationary iterations

Initialization $\mathbf{p}_\Gamma = 0$, ω sufficiently small
while $\|\mathbf{r}\|$ *is too large* **do**
  Resolution of the Global system (3.32), $\mathbf{u}_\Gamma^G = \mathbf{S}^{G^{-1}}(\mathbf{p}_\Gamma + \mathbf{b}^G)$
  if Ω^0 *exists* **then**
    Post-processing (3.24), $\mathbf{q}^0 := \boldsymbol{\lambda}^0 = \mathbf{S}^0 \mathbf{A}^{0^T} \mathbf{u}_\Gamma^G - \mathbf{b}^{0,G}$
  end
  for $s > 0$ **do**
    Fine solution (3.22), $\boldsymbol{\lambda}^{s,F} = \mathbf{s}^{s,F}(\mathbf{J}^s \mathbf{A}^{s^T} \mathbf{u}_\Gamma^G; \mathbf{f}^{s,F})$
  end
  Compute residual $\mathbf{r} = -(\mathbf{A}^0 \boldsymbol{\lambda}^0 + \sum_{s \geq 0} \mathbf{A}^s \mathbf{J}^{s^T} \boldsymbol{\lambda}^{s,F})$
  Update $\mathbf{p}_\Gamma = \mathbf{p}_\Gamma + \omega \mathbf{r}$
end

can be obtained from the general theory of Schwarz domain decomposition methods [2] or operator splitting techniques [94], is that under monotonicity hypothesis of the semilinear form in (3.2), the iteration converges for sufficiently small relaxation parameter. Simple acceleration procedures are possible like Aitken, quasi-Newton or Krylov, see [50].

3.4 . Global/local coupling as a primal domain decomposition method

Equation (3.35) makes it possible to interpret the global/local coupling as a right-preconditioner to the primal reference system (3.33). Note that other interpretations exist, including as a special multigrid method, or an implementation of a non-overlapping alternating Dirichlet-Robin Schwarz domain decomposition method; see [50] for a list.

Contrarily to the recommended (scalable) strategy to solve this system, briefly described in Section 2.2.6 this preconditioner does not possess an additive (i.e. parallel) structure, it is thus not expected to scale up to very large number of subdomains. Anyhow, if the *global* problem is simple enough to be solved efficiently, it generally provides excellent information so that convergence can be fast and interesting performance can be achieved. In particular, It can not be expected from it require multiscale information as provided by BDD's coarse problem made out of zero energy modes (rigid body motions) or well-chosen GENEO-modes [99] in case of poorly-conditioned problem.

Moreover the global/local coupling works in a nonlinear context. Note that the Global problem is an affine preconditioner, meaning that not only it affects the spectrum of the iteration operator but it also embeds a good initialization. Contrarily to the GENEO-BDD approach where Krylov solver is mandatory, the glob-

al/local coupling supports a stationary iteration. Furthermore, the right-preconditioning does not modify the nature of the residual of the system to be solved, allowing flexibility, and in our context, asynchronism.

4 - Asynchronous global/local non invasive coupling

In this chapter, the asynchronous version of the global/local coupling is presented. First, the asynchronous algorithm with an explanation of the sequence of iterations is presented. Then a theoretical study of the convergence of the asynchronous iterations under a certain relaxation coefficient is presented. This study covers the cases with a linear global problem and linear locals as well as the nonlinear case with monotone locals.

4.1 . Asynchronous algorithm

The previous chapter presented the global/local non-intrusive coupling from an equation point of view. It can be written as a classical synchronous as in algorithm 2. In practice, it is recommended to use dynamic relaxation with Aitken's formula to find the relaxation parameter ω.

Algorithm 2: Synchronous stationary iterations

Initialization $\mathbf{p}_\Gamma = 0$, ω sufficiently small
while $\|\mathbf{r}\|$ *is too large* **do**
  Resolution of the Global system (3.30) or (3.32), $\mathbf{u}_\Gamma^G = \mathbf{S}^{G^{-1}}(\mathbf{p}_\Gamma + \mathbf{b}^G)$
  if Ω^0 *exists* **then**
    Post-processing (3.24), $\mathbf{q}^0 := \boldsymbol{\lambda}^0 = \mathbf{S}^0\mathbf{u}_\Gamma^{0,G} - \mathbf{b}^{0,G}$
  end
  Global *scatters* $\mathbf{A}^{s^T}\mathbf{u}_\Gamma^G$ to subdmains $s > 0$
  for $s > 0$ **do**
    Patch *receives* $\mathbf{A}^{s^T}\mathbf{u}_\Gamma^G$
    Local solution (3.22), $\boldsymbol{\lambda}^{s,F} = \mathbf{s}^{s,F}(\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{u}_\Gamma^G; \mathbf{f}^{s,F})$
    Patch *sends* of $\mathbf{q}^s := \mathbf{J}^{s^T}\boldsymbol{\lambda}^{s,F}$ to the Global
  end
  Global *gathers* all $\mathbf{q}^s$
  Global computes residual $\mathbf{r} = -\sum_s \mathbf{A}^s\mathbf{q}^s$
  Global updates $\mathbf{p}_\Gamma = \mathbf{p}_\Gamma + \omega\mathbf{r}$
end

Now an asynchronous parallel version of this algorithm is established. The idea is that each processor updates its calculation as soon as one new piece of information is available from one of the other processors, without having to wait for all the other processors to synchronize. To illustrate this technique, the same situation as presented in the synchronous case of Figure 2.16 is considered.

Figure 4.1 presents the asynchronous method in the case where the global problem is updated as soon as new data is obtained from one of the local prob-

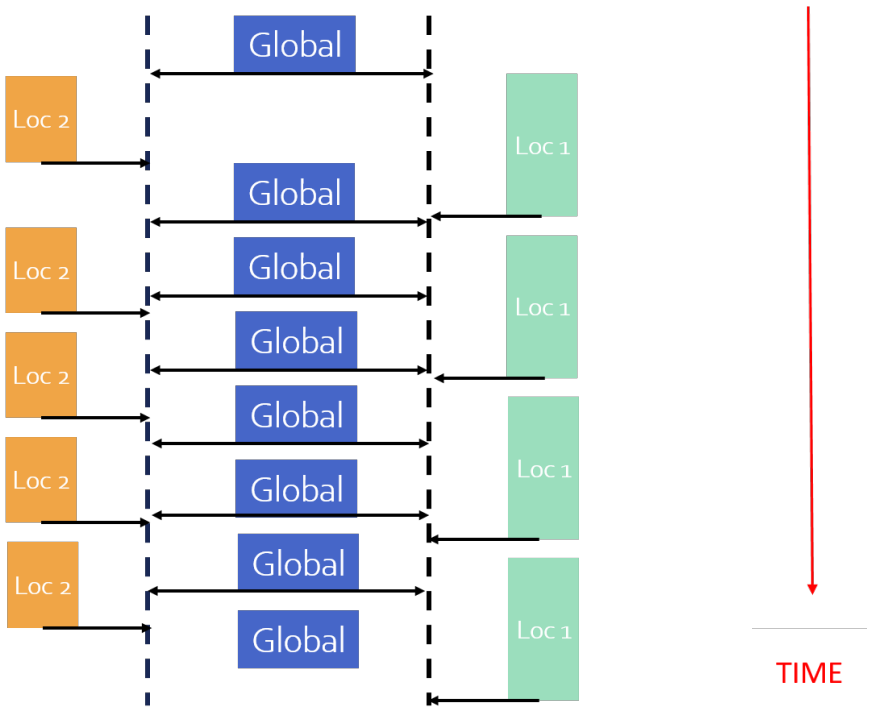


Figure 4.1: Asynchronous iterations

lems. Otherwise, it waits without doing any calculation. From the patches' point of view, the fine models wait for the global problem to send new information, otherwise, they idle. The advantage of this model is both to move forward as soon as further information is present and to avoid redoing calculations with the same data as before. One can also imagine the case where the asynchronous model makes calculations without stopping with or without new data. Thus, calculations already made will have to be redone, whether for the global or the local.

Thus, Based on the figure 4.1, Algorithm **??**, presents an asynchronous version of the Algorithm 2.

Remark 4. One crucial point is that the reaction of the complement subdomain $\boldsymbol{\lambda}^{0,G}$ must be kept synchronized with the *global* iteration. It seems thus more practical to use a software able to post-process $\boldsymbol{\lambda}^{0,G}$ after the global solve, and avoid the methods that involves solving Dirichlet problems on the global version of the patches.

4.2 . Convergence proof of the asynchronous iteration

Proving convergence of asynchronous iteration can be tedious. In the studied case, the Global domain has the advantages of playing a special role such that it can be used to cadence the solver.

Algorithm 3: Asynchronous iterations using RDMA

while $\|\mathbf{r}\|$ *is too large* **do**
  switch *Rank* s **do**
    case $s == 0$ *(global domain)* **do**
      Global *gathers* all $\mathbf{q}^s$
      Global computes residual $\mathbf{r} = -\sum_s \mathbf{A}^s\mathbf{q}^s$
      Global updates $\mathbf{p}_\Gamma = \mathbf{p}_\Gamma + \omega\mathbf{r}$
      Global solve system (3.32), $\mathbf{u}_\Gamma^G = \mathbf{S}^{G^{-1}}(\mathbf{p}_\Gamma + \mathbf{b}^G)$
      if Ω^0 *exists* **then**
        Post-processing (3.24), $\mathbf{q}^0 := \boldsymbol{\lambda}^0 = \mathbf{S}^0\mathbf{u}_\Gamma^{0,G} - \mathbf{b}^{0,G}$
      end
      Global *scatters* $\mathbf{A}^{s^T}\mathbf{u}_\Gamma^G$ to subdmains $s > 0$
    end
    case $s > 0$ *(local patch)* **do**
      Patch *receives* $\mathbf{A}^{s^T}\mathbf{u}_\Gamma^G$
      Local solves (3.22), $\boldsymbol{\lambda}^{s,F} = \mathbf{S}^{s,F}\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{u}_\Gamma^G - \mathbf{b}^{s,F}$
      Local *sends* of $\mathbf{q}^s := \mathbf{J}^{s^T}\boldsymbol{\lambda}^{s,F}$ to the Global
    end
  end
end

As presented before, several techniques exist to study the convergence of an asynchronous method. The difficulty is added in the studied case because the studied problems are without discrete maximal principle, so the matrices do not possess the favorable M-property [9], and the non-intrusive objective makes it impossible to recover such properties by invasive manipulations.

A recent study [21] proved the convergence of Richardson iterations in asynchronous but with a delay bounded at 2 and for linear problems. To generalize that study and go to the case with larger delay, the paracontraction techniques introduced in [33] are used. The idea is to formulate the method as a succession of contractive operators (for a well-chosen relaxation) sharing a common fixed point. We can find other applications of this approach for linear and nonlinear problems in [89, 34, 104].

For now, the study requires the linearity of the global problem, which is then characterized by a symmetric positive definite Schur complement, For now, our study requires the linearity of the global problem which thus takes the form of equation (3.19). Note that the stiffness matrix of (3.19) being SPD, so is its Shcur complement, and as presented before, $0 < \gamma^G < M^G$ the bounds of its spectrum.

4.2.1 . Paracontactions

Let (T_m) be a finite family of paracontractions with a common fixed point $\hat{x}$ in some Hilbert space E. In other words:

- $\forall x \in E,\ \|T_m(x) - \hat{x}\| < \|x - \hat{x}\|$ or $T_m(x) = x$,
- $\forall m,\ T_m(\hat{x}) = \hat{x}$.

Then a sequence of the form:

$$x_{j+1} = T_{m(j)}(x_j) \tag{4.1}$$

converges to $\hat{x}$, assuming that all the paracontractions (T_m) are sufficiently frequently activated [33].

4.2.2 . Asynchronous formulation

Before studying the convergence of the asynchronous model, a rewriting of the problem is introduced, considering the delays affecting the patches.

Referring to Algorithm 4, during the step from iteration j to $j+1$ it is considered that, some patches provide new pieces of information in order to evaluate the residual, anyhow these pieces of information may be related to old configurations $\mathbf{p}_{\Gamma j-\sigma(s,j)}$.

Remark 5. $\sigma(s,j) \geqslant 0$: are delay functions, modeling the delay of the subdomain *s* at iteration *j* of the global problem.

So that the asynchronous iteration can be modeled as:

$$\begin{cases} \mathbf{u}^G_{\Gamma,j} = \mathbf{S}^{G^{-1}}(\mathbf{b}^G + \mathbf{p}_{\Gamma j}) \\ \text{If } s = 0 : \mathbf{q}^0_j = \mathbf{S}^0(\mathbf{A}^{0^T}\mathbf{u}^G_{\Gamma,j} - \mathbf{b}^{0,G}) \\ \text{If } s > 0 : \mathbf{q}^s_j = \begin{cases} \mathbf{J}^{s^T}\mathbf{s}^{s,F}(\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{u}^G_{\Gamma,j-\sigma(s,j)}; \mathbf{f}^{s,F}) \text{ Updated} \\ \mathbf{q}^s_{j-1}, \text{ Not updated} \end{cases} \\ \mathbf{r}_j = -\left(\mathbf{A}^0\mathbf{q}^0_j + \sum_{s>0}\mathbf{A}^s\mathbf{q}^s_j\right) \\ \mathbf{p}_{\Gamma j+1} = \mathbf{p}_{\Gamma j} + \omega\mathbf{r}_j \end{cases} \tag{4.2}$$

For subdomains not updated: $\sigma(s,j) = \sigma(s,j-1) + 1$.

It is crucial to note that if it exists, subdomain 0 always contributes to the evaluation of the residual. Since computing $\mathbf{q}^0_{j+1}$ is only a post-processing of the Global solution, this constraint is not really a problem. In order to unify notations, $\sigma(0,j) = 0$ is introduced, $\forall j$, And thus, the residual at a global iteration j is written in the asynchronous form:

$$\mathbf{r}_j = \sum_{s=0}^{N} \mathbf{A}^s\mathbf{J}^{s^T}\mathbf{s}^{s,F}(\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}(\mathbf{p}_{\Gamma j-\sigma(s,j)} + \mathbf{b}^G); \mathbf{f}^{s,F}) \tag{4.3}$$

and then:

$$\mathbf{p}_{\Gamma j+1} = \mathbf{p}_{\Gamma j} - \omega \sum_{s=0}^{N} \mathbf{A}^s \mathbf{J}^{s^T} \mathbf{s}^{s,F}(\mathbf{J}^s \mathbf{A}^{s^T} \mathbf{S}^{G^{-1}}(\mathbf{p}_{\Gamma j-\sigma(s,j)} + \mathbf{b}^G); \mathbf{f}^{s,F}) \tag{4.4}$$

Note that this expression is valid only after all local patches have at least contributed once to the estimation of the residual.

In order to ensure that at some point all patches provide new information, it is assumed that:

$$\exists D \geqslant 0 \text{ such that } \forall (s,j),\ \sigma(s,j) \leqslant D \tag{4.5}$$

For a given delay $0 \leqslant k \leqslant D$, $\varpi(k,j)$ the set of subdomains (s) such that $\sigma(s,j) = k$.

Let $\hat{\mathbf{p}}_\Gamma$ be the solution to the coupling problem:

$$\sum_{s=0}^{N} \mathbf{A}^s \mathbf{J}^{s^T} \mathbf{s}^{s,F}(\mathbf{J}^s \mathbf{A}^{s^T} \mathbf{S}^{G^{-1}}(\hat{\mathbf{p}}_\Gamma + \mathbf{b}^G); \mathbf{f}^{s,F}) = 0 \tag{4.6}$$

Then, if at some point $\forall k \in [0, D],\ \mathbf{p}_{\Gamma j-k} = \hat{\mathbf{p}}_\Gamma$, $\mathbf{p}_{\Gamma j+1} = \hat{\mathbf{p}}_\Gamma$ whatever the distribution of delays among the subdomains, which makes it a common fixed point for any situation in (4.4).

Note that there may be some (unlikely) situations where the iteration (4.4) stalls. For instance, $\forall s\ \mathbf{f}^{s,F} = 0$ and $\mathbf{A}^{s^T} \mathbf{S}^{G^{-1}}(\mathbf{p}_{\Gamma j-\sigma(s,j)} + \mathbf{b}^G) = 0$, which corresponds to the delayed load always being applied far from the concerned subdomains. This is covered by the theory ($T_m(x) = x$ case). So the assumption (4.5) is strengthened, and the blocking is required to last at most D iterations.

What remains to be proved is the contractive nature of the non-stalling iteration. More precisely the objective is to prove that there exists some non-empty interval of relaxation parameter which makes any series of D iterations a contraction. First, the linear case is studied where the contraction can be characterized by the spectral radius of a linear operator. Then the case of nonlinear monotone fine patches is considered where the contraction is characterized by the decrease of a well-chosen norm of the error.

4.2.3 . Analysis of the linear case

First, the case of linear local problems is considered. In that case the local Dirichlet to Neuman operator takes the form of an affine operator.

The asynchronous global/local coupling iteration can be rewritten as:

$$\begin{aligned} \mathbf{p}_{\Gamma j+1} &= \mathbf{p}_{\Gamma j} - \omega \left(\sum_{s=0}^{N} \hat{\mathbf{S}}^{s,F} \mathbf{S}^{G^{-1}} \mathbf{p}_{\Gamma j-\sigma(s,j)} + \hat{\mathbf{b}} \right) \\ \mathbf{p}_{\Gamma j+1} &= \mathbf{p}_{\Gamma j} - \omega \left(\sum_{k=0}^{D} \left(\sum_{s \in \varpi(k,j)} \hat{\mathbf{S}}^{s,F} \right) \mathbf{S}^{G^{-1}} \mathbf{p}_{\Gamma j-k} + \hat{\mathbf{b}} \right) \end{aligned} \tag{4.7}$$

with:

$$\begin{cases} \hat{\mathbf{S}}^{s,F} &= \mathbf{A}^s \mathbf{J}^{s^T} \mathbf{S}^{s,F} \mathbf{J}^s \mathbf{A}^{s^T} \\ \hat{\mathbf{b}} &= \sum_{s=0}^{N} \mathbf{A}^s \mathbf{J}^{s^T} (\mathbf{S}^{s,F} \mathbf{J}^s \mathbf{A}^{s^T} \mathbf{S}^{G^{-1}} \mathbf{b}^G - \mathbf{b}^{s,F}) \end{cases} \tag{4.8}$$

In order to make appear the paracontractions, a non-zero delay $D > 0$, is assumed and, in the "history space" obtained by concatenating the last $(D+1)$ values of $\mathbf{p}_{\Gamma_j}$ is the work space.

The history at iteration $j+1$ can be rewritten as:

$$\begin{pmatrix} \mathbf{p}_{\Gamma_{j+1}} \\ \mathbf{p}_{\Gamma_j} \\ \vdots \\ \mathbf{p}_{\Gamma_{j-D+1}} \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{I} - \omega \mathbf{X}_{j,0} & -\omega \mathbf{X}_{j,1} & \dots & -\omega \mathbf{X}_{j,D} \\ \mathbf{I} & 0 & \dots & 0 \\ 0 & \mathbf{I} & 0 & \dots \\ \dots & 0 & \mathbf{I} & 0 \end{pmatrix}}_{\mathbf{B}_j} \begin{pmatrix} \mathbf{p}_{\Gamma_j} \\ \mathbf{p}_{\Gamma_{j-1}} \\ \vdots \\ \mathbf{p}_{\Gamma_{j-D}} \end{pmatrix} - \begin{pmatrix} \omega \hat{\mathbf{b}} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{4.9}$$

$$\text{with } \mathbf{X}_{j,k} = \left(\sum_{s \in \varpi(k,j)} \hat{\mathbf{S}}^{s,F} \right) \mathbf{S}^{G^{-1}}$$

As noted earlier, since $\forall j,\ \sum_k \mathbf{X}_{j,k} \hat{\mathbf{p}}_\Gamma + \tilde{\mathbf{b}} = 0$, the vector obtained by repeating the solution $\hat{\mathbf{p}}_\Gamma$ is a fixed point for the above iteration.

In order to prove the paracontracting nature of the iteration, it suffices to prove that any matrix $\mathbf{B}_j$ of (4.9) can be turned into contraction by correctly selecting the relaxation $\omega > 0$. Since $\mathbf{B}_j$ is a block companion matrix, it seems natural to study its spectrum and prove that it can be bounded by 1.

The eigenvalues (λ) of $\mathbf{B}_j$ are the roots of the polynomial:

$$\det \left((1-\lambda)\lambda^D \mathbf{I} - \omega \sum_{k=0}^{D} \lambda^{D-k} \mathbf{X}_{j,k} \right) = 0 \tag{4.10}$$

This is the determinant of a real momic matrix polynomial [49]. In order to benefit from the underlying symmetry, the Cholesky factorization of $\mathbf{S}^G = \mathbf{L}\mathbf{L}^T$ is introduced, left-multiply the polynomial by $\mathbf{L}^{-1}$ and right-multiply it by $\mathbf{L}$, the roots of (4.10) are also the root of the polynomial $P_{j,\omega}(\lambda)$:

$$P_{j,\omega}(\lambda) = \det \left((1-\lambda)\lambda^D \mathbf{I} - \omega \sum_{k=0}^{D} \lambda^{D-k} \hat{\mathbf{X}}_{j,k} \right) = 0 \tag{4.11}$$

where $\hat{\mathbf{X}}_{j,k} = \mathbf{L}^{-1} \mathbf{X}_{j,k} \mathbf{L} = \mathbf{L}^{-1} \left(\sum_{s \in \varpi(k,j)} \hat{\mathbf{S}}^{s,F} \right) \mathbf{L}^{-T}$.

Using the absolute continuity of the roots of a polynomial with respect to its coefficients (see [56, 88] for instance), it has been seen that for a small enough ω, the eigenvalues tend to concentrate around the roots of $P_{j,0}(\lambda) = \det((1-\lambda)\lambda^D \mathbf{I})$, that is to say around 0 and 1.

Let $\tilde{\lambda}_{j,\omega}$ be one of the roots of $P_{j,\omega}$, and $\varepsilon = \min\left(\sin\left(\frac{\pi}{3D}\right), \frac{1}{2}\right)$, ω_0 can be found such that $\omega < \omega_0 \Rightarrow |\tilde{\lambda}_{j,\omega} - \tilde{\lambda}_{j,0}| < \varepsilon$. With $\omega_0 = min(\omega_{0,j})$ over all the possible

configuration of j. At that point, the roots that tend to zero have all modulus less than $\varepsilon < 1$, only the roots that tend to 1 could pose a problem.

In what follows, $\tilde{\lambda}_{j,\omega}$ is such a root that tends to 1, its modulus and argument can be bounded, see Figure 4.2.

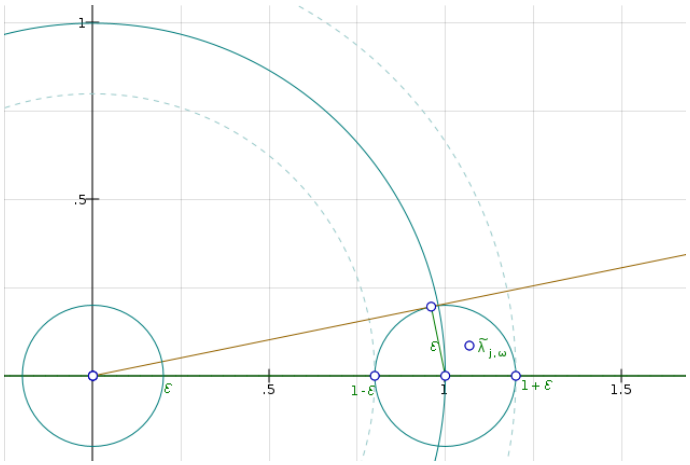


Figure 4.2: Constraining roots near 1

$|\tilde{\lambda}_{j,\omega} - 1| < \varepsilon$ implies that:

$$\begin{aligned} 1 - \varepsilon < |\tilde{\lambda}_{j,\omega}| < 1 + \varepsilon \\ |\sin(\arg(\tilde{\lambda}_{j,\omega}))| < \varepsilon \end{aligned} \tag{4.12}$$

For $\varepsilon = \sin\frac{\pi}{3D}$ and $0 \leqslant k \leqslant D$, the modulus and the reel part are bounded on (real part symbol $\Re$):

$$\begin{aligned} (1-\varepsilon)^D < |\tilde{\lambda}_{j,\omega}|^k < (1+\varepsilon)^D \\ \Re(\tilde{\lambda}_{j,\omega}^k) = |\tilde{\lambda}_{j,\omega}|^k \cos(k \arg(\tilde{\lambda}_{j,\omega})) > \frac{(1-\varepsilon)^D}{2} \end{aligned} \tag{4.13}$$

Let $\tilde{\mathbf{v}}_{j,\omega}$ be an eigenvector of the matrix polynomial associated with $\tilde{\lambda}_{j,\omega}$:

$$(1 - \tilde{\lambda}_{j,\omega})\tilde{\lambda}_{j,\omega}^D \tilde{\mathbf{v}}_{j,\omega} - \omega \sum_{k=0}^{D} \tilde{\lambda}_{j,\omega}^{D-k} \hat{\mathbf{X}}_{j,k} \tilde{\mathbf{v}}_{j,\omega} = 0 \tag{4.14}$$

The expression can be left-multiplied by the Hermitian transpose $\tilde{\mathbf{v}}_{j,\omega}^H$:

$$(1 - \tilde{\lambda}_{j,\omega})\tilde{\lambda}_{j,\omega}^D \tilde{\mathbf{v}}_{j,\omega}^H \tilde{\mathbf{v}}_{j,\omega} - \omega \sum_{k=0}^{D} \tilde{\lambda}_{j,\omega}^{D-k} \tilde{\mathbf{v}}_{j,\omega}^H \hat{\mathbf{X}}_{j,k} \tilde{\mathbf{v}}_{j,\omega} = 0 \tag{4.15}$$

To simplify, $\tilde{\mathbf{v}}_{j,\omega}$ can be chosen of unit Euclidean norm. For $\omega < \omega_0$, $\tilde{\lambda}_{j,\omega} \neq 0$, then:

$$\begin{aligned}\tilde{\lambda}_{j,\omega} &= 1 - \omega \sum_{k=0}^{D} \frac{\|\tilde{\mathbf{v}}_{j,\omega}\|^2_{\hat{\mathbf{X}}_{j,k}}}{\tilde{\lambda}^k_{j,\omega}} \\ |\tilde{\lambda}_{j,\omega}|^2 &= 1 - 2\omega \sum_{k=0}^{D} \frac{\Re(\tilde{\lambda}^k_{j,\omega})\|\tilde{\mathbf{v}}\|^2_{\hat{\mathbf{X}}_{j,k}}}{|\tilde{\lambda}_{j,\omega}|^{2k}} + \omega^2 \left| \sum_{k=0}^{D} \frac{\|\tilde{\mathbf{v}}\|^2_{\hat{\mathbf{X}}_{j,k}}}{\tilde{\lambda}^k_{j,\omega}} \right|^2\end{aligned} \tag{4.16}$$

Using (4.13), we have:

$$|\tilde{\lambda}_{j,\omega}|^2 < 1 - \omega \frac{\left(\sum_{k=0}^{D} \|\tilde{\mathbf{v}}\|^2_{\hat{\mathbf{X}}_{j,k}}\right)}{(1+\varepsilon)^D} + \omega^2 \frac{\left(\sum_{k=0}^{D} \|\tilde{\mathbf{v}}\|^2_{\hat{\mathbf{X}}_{j,k}}\right)^2}{(1-\varepsilon)^{2D}} \tag{4.17}$$

Indeed by the definition of the delay D and $\varpi, \cup_k \varpi(k,j) = 1 : N$ and for $k \neq k'$ $\varpi(k,j) \cap \varpi(k',j) = \varnothing$, so the sum of norms is simplified because each subdomain appears only once:

$$\begin{aligned}\sum_{k=0}^{D} \|\tilde{\mathbf{v}}\|^2_{\hat{\mathbf{X}}_{j,k}} &= \sum_{k=0}^{D} \tilde{\mathbf{v}}^H_{j,\omega} \hat{\mathbf{X}}_{j,k} \tilde{\mathbf{v}}_{j,\omega} = \tilde{\mathbf{v}}^H_{j,\omega} \mathbf{L}^{-1} \left(\sum_{k=0}^{D} \sum_{s \in \varpi(k,j)} \hat{\mathbf{S}}^{s,F} \right) \mathbf{L}^{-T} \tilde{\mathbf{v}}_{j,\omega} \\ &= \tilde{\mathbf{v}}^H_{j,\omega} \mathbf{L}^{-1} \left(\sum_{s=0}^{N} \hat{\mathbf{S}}^{s,F} \right) \mathbf{L}^{-T} \tilde{\mathbf{v}}_{j,\omega}\end{aligned} \tag{4.18}$$

Since $\tilde{\mathbf{v}}_{j,\omega}$ is of unit Euclidean norm, the term above can directly be bounded by the extremal eigenvalues of $\mathbf{L}^{-1} \left(\sum_{s=0}^{N} \hat{\mathbf{S}}^{s,F} \right) \mathbf{L}^{-T}$ which coincide to the generalized eigenvalues of the system:

$$\begin{aligned}\alpha_{\min} \leqslant \sum_{k=0}^{D} \|\tilde{\mathbf{v}}\|^2_{\hat{\mathbf{X}}_{j,k}} \leqslant \alpha_{\max} \\ \text{where } (\alpha) \text{ solve } \det\left(\left(\sum_{s=0}^{N} \hat{\mathbf{S}}^{s,F} \right) + \alpha \mathbf{S}^G \right) = 0\end{aligned} \tag{4.19}$$

Thus the upper bound is obtained:

$$|\tilde{\lambda}_{j,\omega}|^2 \leqslant 1 - \omega \frac{\alpha_{\min}}{(1+\varepsilon)^D} + \omega^2 \frac{\alpha^2_{\max}}{(1-\varepsilon)^{2D}}, \qquad \forall 0 < \omega < \omega_0 \tag{4.20}$$

This is a bound of the form $|\tilde{\lambda}_{j,\omega}|^2 \leqslant 1 - A\omega + B\omega^2$ (with $0 < A < B$) which is a second degree polynomial in ω, and which is less than 1 for $0 < \omega < A/B$. As a consequence:

$$|\tilde{\lambda}_{j,\omega}| < 1 \text{ for } 0 < \omega < \omega_{\text{async}} = \min\left(\omega_0, \frac{(1-\varepsilon)^D \alpha_{\min}}{(1+\varepsilon)^{2D} \alpha^2_{\max}} \right) \tag{4.21}$$

This is probably an extremely crude bound, but it has the advantage to only depend on D and not on the configuration of the iteration (index j). Thus, such a relaxation makes any $\mathbf{B}_j$ a paracontraction, and the asynchronous iteration converge.

Remark 6. For the synchronous iteration, the bound can be derived from (4.10) with $D = 0$, it is $0 < \omega < \omega_{\text{sync}} = \frac{2}{\alpha_{\max}}$.

4.2.4 . Analysis of the nonlinear case with monotone local models

First let us introduce the notation:

$$\mathbf{x}^s(\mathbf{p}_\Gamma) = \mathbf{A}^s \mathbf{J}^{s^T} \mathbf{s}^{s,F}(\mathbf{J}^s \mathbf{A}^{s^T} \mathbf{S}^{G^{-1}}(\mathbf{p}_\Gamma + \mathbf{b}^G); \mathbf{f}^{s,F}) \tag{4.22}$$

The iteration writes:

$$\mathbf{p}_{\Gamma j+1} = \mathbf{p}_{\Gamma j} - \omega \sum_{k=0}^{D} \sum_{s \in \varpi(k,j)} \mathbf{x}^s(\mathbf{p}_{\Gamma j-k}) \tag{4.23}$$

Again, the reference solution $\hat{\mathbf{p}}_\Gamma$ satisfies:

$$\sum_{s=0}^{N} \mathbf{x}^s(\hat{\mathbf{p}}_\Gamma) = 0$$

and $\forall s,\ \mathbf{p}_{\Gamma j-\sigma(s,j)} = \hat{\mathbf{p}}_\Gamma$ is a fixed point.

The following notations are considered:

- $\mathbf{e}_j = \mathbf{p}_{\Gamma j} - \hat{\mathbf{p}}_\Gamma$ the error at iteration j,
- $\mathbf{r}_j^s = \mathbf{x}^s(\mathbf{p}_{\Gamma j}) - \mathbf{x}^s(\hat{\mathbf{p}}_\Gamma)$ the contribution to the residual such that $\forall j,\ \sum_{s=0}^{N} \mathbf{r}_j^s = \mathbf{r}_j$.

Then:

$$\begin{aligned} \mathbf{p}_{\Gamma j+1} - \hat{\mathbf{p}}_\Gamma &= \mathbf{p}_{\Gamma j} - \hat{\mathbf{p}}_\Gamma - \omega \sum_{k=0}^{D} \sum_{s \in \varpi(k,j)} \left(\mathbf{x}^s(\mathbf{p}_{\Gamma j-k}) - \mathbf{x}^s(\hat{\mathbf{p}}_\Gamma) \right) \\ \mathbf{e}_{j+1} &= \mathbf{e}_j - \omega \sum_{k=0}^{D} \sum_{s \in \varpi(k,j)} \mathbf{r}_{j-k}^s \end{aligned} \tag{4.24}$$

The aim is to obtain a bound of the form:

$$e_{j+1}^2 \leqslant e_j^2 \left(1 - 2\omega A + \omega^2 B\right) \qquad \text{with } A > 0. \tag{4.25}$$

where e_j is some measure of the error.

Indeed, in that case, for $0 < \omega < 2A/B$, $(1 - 2\omega A + \omega^2 B) < 1$, the iteration is a contraction, and it converges to the common fixed point. Note that the best convergence ratio is obtained for $\omega = A/B$, it is worth $(1 - A^2/B)$. Moreover, since the iteration is always well-defined, we necessarily have $B \geqslant A^2$.

The assumptions on the local models lead to $\mathbf{p}_\Gamma \mapsto \mathbf{x}(\mathbf{p}_\Gamma)^s$ being strongly monotone and continuous, the associated constants were given in (3.29) and (3.28). The assumptions on the global model enable to exploit a Euclidean structure for the convergence analysis. Indeed, the $\mathbf{S}^{G^{-1}}$ is used as an inner product for interface reactions, with the notation:

$$\langle \mathbf{a}, \mathbf{b} \rangle_G = \mathbf{a}^T \mathbf{S}^{G^{-1}} \mathbf{b} \qquad ; \qquad \|\!| \mathbf{a} |\!\| = \langle \mathbf{a}, \mathbf{a} \rangle_G^{1/2}. \tag{4.26}$$

Due to Equation (4.24):

$$\|\!| \mathbf{e}_{j+1} |\!\|^2 = \|\!| \mathbf{e}_j |\!\|^2 - 2\omega \sum_{k=0}^{D} \sum_{s \in \varpi(k,j)} \langle \mathbf{e}_j, \mathbf{r}^s_{j-k} \rangle_G + \omega^2 \|\!| \sum_{k=0}^{D} \sum_{s \in \varpi(k,j)} \mathbf{r}^s_{j-k} |\!\|^2 \quad . \tag{4.27}$$

The first degree term needs to be further analyzed in order to make appear terms with the same delay. The recursion is obtained with:

$$\begin{aligned} \mathbf{e}_j &= \mathbf{e}_{j-1} - \omega \left(\sum_{q=0}^{D} \sum_{s \in \varpi(q,j-1)} \mathbf{r}^s_{j-1-q} \right) \\ &= \mathbf{e}_{j-k} - \omega \left(\sum_{K=1}^{k} \sum_{q=0}^{D} \sum_{t \in \varpi(q,j-K)} \mathbf{r}^t_{j-K-q} \right). \end{aligned} \tag{4.28}$$

Hence using (4.28):

$$\sum_{k=0}^{D} \sum_{s \in \varpi(k,j)} \langle \mathbf{e}_j, \mathbf{r}^s_{j-k} \rangle_G = \sum_{k=0}^{D} \sum_{s \in \varpi(k,j)} \langle \mathbf{e}_{j-k}, \mathbf{r}^s_{j-k} \rangle_G - \omega \sum_{k=1}^{D} \sum_{s \in \varpi(k,j)} \sum_{K=1}^{k} \sum_{q=0}^{D} \sum_{t \in \varpi(q,j-K)} \langle \mathbf{r}^t_{j-K-q}, \mathbf{r}^s_{j-k} \rangle_G. \tag{4.29}$$

Finally, using (4.29), the equation (4.27) can be written as:

$$\begin{aligned} \|\!| \mathbf{e}_{j+1} |\!\|^2 = \|\!| \mathbf{e}_j |\!\|^2 - 2\omega \underbrace{\sum_{k=0}^{D} \sum_{s \in \varpi(k,j)} \langle \mathbf{e}_{j-k}, \mathbf{r}^s_{j-k} \rangle_G}_{a} \\ + \omega^2 \underbrace{\left(\|\!| \sum_{k=0}^{D} \sum_{s \in \varpi(k,j)} \mathbf{r}^s_{j-k} |\!\|^2 + 2 \sum_{k=1}^{D} \sum_{s \in \varpi(k,j)} \sum_{K=1}^{k} \sum_{q=0}^{D} \sum_{t \in \varpi(q,j-K)} \langle \mathbf{r}^t_{j-K-q}, \mathbf{r}^s_{j-k} \rangle_G \right)}_{b}. \end{aligned} \tag{4.30}$$

In order to recover the situation of (4.25), find a bounding from below of the first degree coefficient a and a bounding from above for the second degree coefficient b is needed.

Using convexity and Cauchy-Schwarz inequalities:

$$b \leqslant \sum_{k=0}^{D} \sum_{s \in \varpi(k,j)} N \|\!| \mathbf{r}^s_{j-k} |\!\|^2 + 2 \sum_{K=1}^{D} \sum_{k=1}^{D} \sum_{s \in \varpi(k,j)} \sum_{q=0}^{D} \sum_{t \in \varpi(q,j-K)} \|\!| \mathbf{r}^t_{j-K-q} |\!\| \, \|\!| \mathbf{r}^s_{j-k} |\!\| \tag{4.31}$$

The bounding from above is a classical consequence of the continuity of the fine problems.

$$\begin{aligned}\left\|\left\|\mathbf{r}^s_{j-k}\right\|\right\| &= \left\|\left\|\mathbf{A}^s\mathbf{J}^{s^T}\left(\mathbf{s}^{s,F}(\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}(\mathbf{p}_{\Gamma\,j-k}+\mathbf{b}^G);\mathbf{f}^{s,F})-\mathbf{s}^{s,F}(\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}(\hat{\mathbf{p}}_\Gamma+\mathbf{b}^G);\mathbf{f}^{s,F})\right)\right\|\right\| \\ &\leqslant \alpha^s M^{s,F}\left\|\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}\mathbf{e}_{j-k}\right\| \leqslant \alpha^s M^{s,F}\left\|\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}\right\|\left\|\mathbf{e}_{j-k}\right\| \\ &\leqslant \alpha^s M^{s,F}\left\|\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}\right\|\frac{M^G}{\gamma^G}\left\|\left\|\mathbf{e}_{j-k}\right\|\right\|\end{aligned} \tag{4.32}$$

where α^s is the square root of the largest eigenvalue of the SPD matrix $(\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}\mathbf{A}^s\mathbf{J}^{s^T})$. The coarse constants are used to switch from the Euclidean norm to the $\mathbf{S}^{G^{-1}}$-norm.

The bounding from above of the individual terms is thus not complicated, the difficulty lies in the handling of the many delayed terms, with delays ranging from 0 to $2D$.

Regarding the first degree term, the bounding from below is a consequence of the monotonicity:

$$\begin{aligned}a &= \sum_{k=0}^{D}\sum_{s\in\varpi(k,j)}((\mathbf{p}_{\Gamma\,j-k}-\mathbf{b}^G)-(\hat{\mathbf{p}}_\Gamma-\mathbf{b}^G))^T\mathbf{S}^{G^{-1}}\mathbf{A}^s\mathbf{J}^{s^T} \\ &\qquad\left(\mathbf{s}^{s,F}(\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}(\mathbf{p}_{\Gamma\,j-k}+\mathbf{b}^G);\mathbf{f}^{s,F})-\mathbf{s}^{s,F}(\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}(\hat{\mathbf{p}}_\Gamma+\mathbf{b}^G);\mathbf{f}^{s,F})\right) \\ &\geqslant \sum_{k=0}^{D}\sum_{s\in\varpi(k,j)}\gamma^{s,F}\left\|\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}\mathbf{e}_{j-k}\right\|^2.\end{aligned} \tag{4.33}$$

To complete the bounding, considering more specific cases are needed. In the following, we study two non-exclusive favorable situations that are made explicit.

4.2.4.1 . Max norm on the history vector

Following [33], the history unknown is introduced:

$$\hat{\mathbf{e}}_j^T = \begin{pmatrix}\mathbf{e}_j^T & \mathbf{e}_{j-1}^T & \dots & \mathbf{e}_{j-2D}^T\end{pmatrix},$$

equipped with the sup norm

$$\left\|\hat{\mathbf{e}}_j\right\|_\infty = \max_{0\leqslant k\leqslant 2D}\left\|\mathbf{e}_{j-k}\right\|$$

$\hat{k}_j$ the delay for which the maximum is reached at iteration j.

This norm makes it trivial to bound the second degree term b (4.31) from above

using (4.32):

$$b \leqslant \underbrace{\|\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}\|^2\left(\frac{M^G}{\gamma^G}\right)^2\left(N\sum_{k=0}^{D}\sum_{s\in\varpi(k,j)}\left(\alpha^s M^{s,F}\right)^2+2\sum_{\substack{K=1\\k=1\\q=0}}^{D}\sum_{\substack{s\in\varpi(k,j)\\t\in\varpi(q,j-K)}}\alpha^s\alpha^t M^{t,F}M^{s,F}\right)}_{B_j}\|\hat{\mathbf{e}}_j\|_\infty \tag{4.34}$$

To bound a from below:

$$a \geqslant \sum_{k=0}^{D}\sum_{s\in\varpi(k,j)}\gamma^{s,F}\|\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}\mathbf{e}_{j-k}\|^2 \geqslant \sum_{s\in\varpi(\hat{k}_j,j)}\gamma^{s,F}\|\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}\mathbf{e}_{j-\hat{k}_j}\|^2, \tag{4.35}$$

To further strengthen hypothesis (4.5): It is needed to assume that sufficiently many subdomains are activated for the $\hat{k}_j$ contribution so that the following bound holds:

$$\sum_{s\in\varpi(\hat{k}_j,j)}\gamma^{s,F}\|\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}\mathbf{e}_{j-\hat{k}_j}\|^2 \geqslant \frac{\theta_{\hat{k}_j,j}}{\kappa}\|\mathbf{e}_{j-\hat{k}_j}\|^2 \geqslant \underbrace{\frac{\theta_{\hat{k}_j,j}}{\kappa}\left(\frac{\gamma^G}{M^G}\right)^2}_{A_j}|||\mathbf{e}_{j-\hat{k}_j}|||^2 \tag{4.36}$$

where $\theta_{\hat{k}_j,j}$ is the minimal non-zero eigenvalue of the matrix:

$$\left(\sum_{s\in\varpi(\hat{k}_j,j)}\mathbf{S}^{G^{-1}}\mathbf{A}^s\mathbf{J}^{s^T}\gamma^{s,F}\mathbf{J}^s\mathbf{A}^{s^T}\mathbf{S}^{G^{-1}}\right)$$

And where $\kappa \geqslant 1$ is some constant. Two cases can be distinguished:

- $\kappa = 1$ would be suited to $\mathbf{e}_{j-\hat{k}_j}$ being non-zero only on the boundary of $\bigcup_{s\in\varpi(\hat{k}_j,j)}\Omega^s$.
- $\kappa = N$ would correspond to one subdomain being activated with the $\hat{k}_j$ and the error $\mathbf{e}_{j-\hat{k}_j}$ being smoothly distributed on the domain.

In practice κ can be influenced by the load balancing between patches, and hardware properties like the speed of the network.

Introducing $A = \min A_j > 0$ and $B = \max B_j$ taken among all the potential distribution of delays in the subdomains:

$$|||\mathbf{e}_{j+1}|||^2 \leqslant |||\hat{\mathbf{e}}_j|||_\infty^2\left(1-2\omega A+B\omega^2\right), \tag{4.37}$$

which implies that:

$$|||\hat{\mathbf{e}}_{j+2D}|||_\infty^2 \leqslant |||\hat{\mathbf{e}}_j|||_\infty^2\left(1-2\omega A+B\omega^2\right), \tag{4.38}$$

and the iteration converges for $0 < \omega 2A/B$.

4.2.4.2 . Monotone convergence for non-adjacent patches

In the case of non-adjacent patches, the subdomain 0 plays a particular role as it is always synchronous and it is in contact with the whole interface (the interface degrees of freedom can be ordered such that $\mathbf{A}^0 = \mathbf{I}$ and $\mathbf{J}^0 = \mathbf{I}$). Thus, if the subdomain 0 exists, the bounding from below is simple to obtain:

$$\sum_{k=0}^{D} \sum_{s\in\varpi(k,j)} \langle \mathbf{e}_{j-k}, \mathbf{r}^s_{j-k} \rangle_G \geqslant \gamma^0 \|\mathbf{S}^{G^{-1}} \mathbf{e}_j\|^2 \geqslant \underbrace{\frac{\gamma^0 \gamma^G}{M^{G^2}}}_{A} |||\mathbf{e}_j|||^2 \tag{4.39}$$

The bounding from above of the b term is a bit more complex, because of the many delayed error terms.

A proof that there exists some interval $(\omega_{\min}, \omega_{\max})$ is proposed for which the relaxed iteration is strictly decreasing, in the sense that there exists $1 > c > 0$ which depends on ω such that, $\forall j$:

$$|||\mathbf{e}_{j+1}|||^2 \leqslant c |||\mathbf{e}_j|||^2$$

Assuming such c exists:

$$\begin{aligned}
b &\leqslant \|\mathbf{J}^s \mathbf{A}^{s^T} \mathbf{S}^{G^{-1}}\|^2 \left(\frac{M^G}{\gamma^G}\right)^2 \dots \\
&\dots \left(N \sum_{k=0}^{D} \sum_{s\in\varpi(k,j)} \left(\alpha^s M^{s,F}\right)^2 c^{-2k} + 2 \sum_{\substack{K=1 \\ k=1 \\ q=0}}^{D} \sum_{\substack{s\in\varpi(k,j) \\ t\in\varpi(q,j-K)}} \alpha^s \alpha^t M^{t,F} M^{s,F} c^{-(q+K+k)} \right) |||\mathbf{e}_j|||^2 \\
&\leqslant \underbrace{\|\mathbf{J}^s \mathbf{A}^{s^T} \mathbf{S}^{G^{-1}}\|^2 \left(\frac{M^G}{\gamma^G}\right)^2 \left(N \sum_{k=0}^{D} \sum_{s\in\varpi(k,j)} \left(\alpha^s M^{s,F}\right)^2 + 2 \sum_{\substack{K=1 \\ k=1 \\ q=0}}^{D} \sum_{\substack{s\in\varpi(k,j) \\ t\in\varpi(q,j-K)}} \alpha^s \alpha^t M^{t,F} M^{s,F} \right)}_{\tilde{B}_j} c^{-3D} |||\mathbf{e}_j|||^2
\end{aligned} \tag{4.40}$$

Introducing $\tilde{B} = \max \tilde{B}_j$, taken among all the potential distributions of delays in the subdomains:

$$|||\mathbf{e}_{j+1}|||^2 \leqslant |||\mathbf{e}_j|||^2 \left(1 - 2\omega A + \omega^2 \tilde{B} c^{-3D}\right) \tag{4.41}$$

The minimal rate of convergence is attained for $\omega_{opt} = Ac^{3D}/\tilde{B}$, and it is worth $r_{opt} = (1 - \frac{A^2 c^{3D}}{\tilde{B}})$. Figure 4.3 illustrates the existence of a domain $(c_0, 1)$ where $c \in (c_0, 1) \Rightarrow c \geqslant r_{opt}$. Note that $c_0 > 0$ depends only on A, $\tilde{B}$ and D.

For a given $c \in (c_0, 1)$, let $\delta = A^2 - (1 - c)\tilde{B} c^{-3D} > 0$, any $\omega \in (\frac{A - \sqrt{\delta}}{\tilde{B} c^{-3D}}, \frac{A + \sqrt{\delta}}{\tilde{B} c^{-3D}})$ leads to a strict decrease of the error at each iteration.

Contrarily to the synchronous case, or to previous analysis with the history vector, the difficulty is that the relaxation can not be too small.

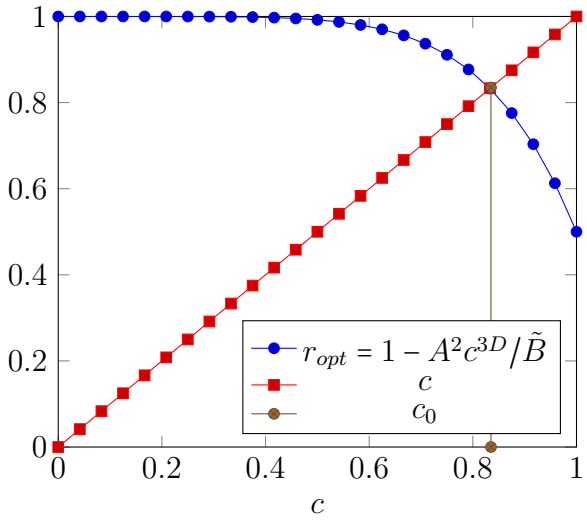


Figure 4.3: Typical dependence of rate of convergence vs c

5 - Implementation details

In this chapter, first some works are cited in which a sample implementation of the asynchronous domain decomposition methods has been realized . Then RMA-MPI parallelization techniques are detailed with some illustrative examples. Finally, the asynchronous and synchronous version of the code that was developed during the Ph.D. thesis are presented by detailing the implementation and the tools used.

5.1 . Introduction

Implementing an asynchronous communication protocol based on the classical message passing paradigm MPI has been the subject of several research works. In [75, 73], an efficient library is proposed for asynchronous domain decomposition solvers, based on classical two-sided communications. In [112, 48] the use of one-sided communications, also known as MPI-RDMA (Remote Direct Memory Access), is considered. The idea of this approach is indeed well adapted to the asynchronous calculation because there is no need to stop computing to send or receive operations.

5.2 . RMA-MPI

The classical message passing paradigm allows the transfer of data from one memory address to another, classical two-sided communications involve at least two ranks and allow data transfer from one memory address to another using explicit communication commands. Still, it does not control the arrival of the information if the synchronization is not explicit. Whatever is the kind of communication, blocking **Send**, **recv** or non-blocking **Isend**, **Irecev** coupled with command like **MPI.test()** or **MPI.wait()**, to check the status of the transfer and to synchronize it. Several transfers coupled with synchronizations can negatively influence the simulation performance.

Figure 5.1 corresponds to sending from processor 1 with the send command to processor 0, which receives with the recv command.

The MPI solution to this problem is one-sided communication, it was proposed first in the MPI2 protocol [83], allowing to decouple the data transfer from the synchronization, so that several transfers can be performed with only one synchronization barrier at the end. In [84], MPI 3 proposed a new version of one-sided communication with more efficient techniques allowing complete asynchronous communications. The one-sided communication is then an optimization for performance to reduce management overhead, also known as remote memory ac-

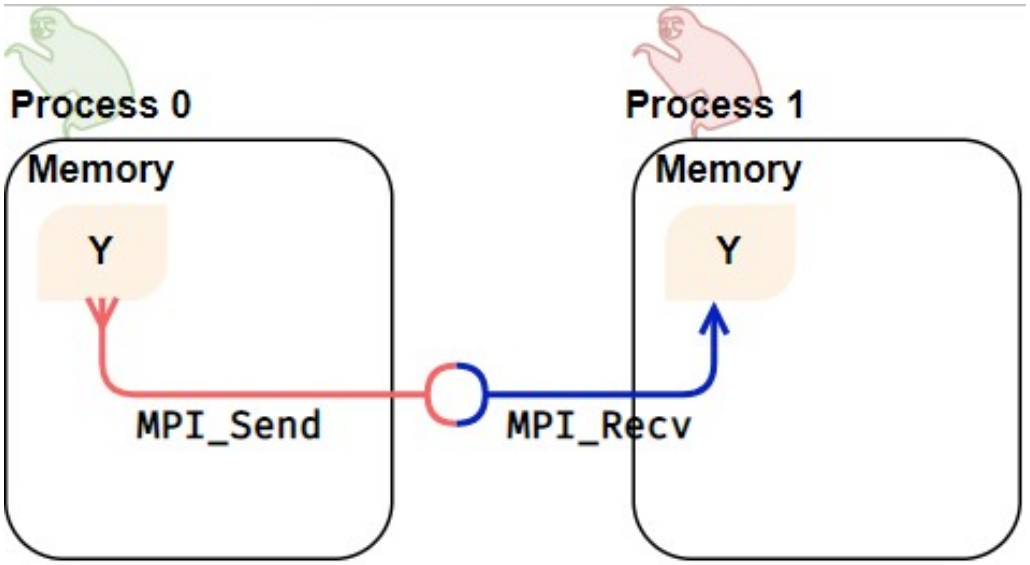


Figure 5.1: Send + Recv (Ref : Two sided communication concepts)

cess **RMA**. It can be viewed as an emulation of shared memory in a distributed context.

Remark 7. Note that the performance of the RMA strongly depends on the MPI implementation and network hardware.

The basic idea is that each rank exposes a so-called **window** of its local memory and allows other ranks either write or read from it. Ranks, in this case, are no longer identified as sender or receiver but as **origin** rank who initiates the operation and **target** rank.The latter does not participate in the data exchange.

The following sections will present how it works in detail.

5.2.1 . One sided communication workflow

The RMA-MPI workflow is based the following on five ordered steps:

1. **Window allocation**: The local memory buffer that other MPI ranks can access need to be allocated first.

2. **Open epoch**: The epoch, in MPI designation, corresponds to the time duration during which the window is open, and the other ranks can access the data. It is also seen as a synchronization call that informs the target that other ranks are ready to access data.

3. **Data accessing**: After that, each rank can access the window as it wants in order to put data in it or to get data from it. Data access does not necessarily require any action for the receiving side or the target rank that exposes

the memory. The origin ranks can just put and get data from the exposing memory without involving the target. When this target rank closes the window, it checks the data states after all the ranks end their operations. But what happened in the meantime does not matter to it.

4. **Close epoch**: Once the data operations are done, the epoch is closed, corresponding to synchronization. The target rank closes the window, it ensures that all accesses have completed (local synchronization). At this point, the target rank can read and process the data.

5. **Window free**: The window is deallocated and the memory buffer freed.

Note that during one epoch, data can be accessed as much as desired, reopen other epochs, and close as many as wanted.

5.2.2 . Window creation

As explained before, the first step is creating a memory window allowing other processors to access the variables stored in it with RMA operations. There are four so called windows models wich are different way to create windows:

1. **Window_create:** Creates a new memory window from an already existing buffer, which is well allocated, the buffer's pointer is passed as an argument of this function with the variable type, and the window is created.

2. **Window_allocate:** Allocates a new memory window instead of using an existing buffer like **Window_create**. The buffer is created when needing to use it, which allows MPI to decide where to allocate the memory. It is sometimes faster but not coherent for some MPI implementation and hardware architecture.

3. **Window_create_dynamic:** Exposes a specific buffer that is not available yet, and it is just attached later on with MPI_win_attach and MPI_win_detach, which is a sort of performance optimization because it allows the dynamic management of the window creation.

4. **Window_allocate_shared:** It is a specific case for Window_allocate, which means that buffers are allocated and used in the shared memory within one node, which only works in **MPI_comm_shared**. It allows using high-speed operations because it uses only copy-and-write memory operations.

All the routines used for window creation are called collectively by all the processors. However, the allocation of these windows and the call of dynamic routines are locally done by each processor separately.

5.2.3 . Free window

When **origin** processors end their sending operations, the windows must be freed after closing the access epoch, using the command **Window_free()**. It is called collectively by all the processors associated with the window. Processors can only reach this command after completing all operations within the window. It also blocks and returns nothing until all processors have executed it to ensure that no processor continues to access the window afterward.

5.2.4 . Communication operations

The remote memory access (RMA) proposes three communication functions that allow, once access to the window is acquired, to read data with the command **Get**, to write directly in the window with **Put**, and to update a variable in this window with reduction operations using **Accumulate**.

All these functions correspond to non-blocking communications performed within an epoch as specified before.

1. **Put:** It is the equivalent of the **Send** operation in the case of two-sided communication which requires a **Recv** operation to ensure reception on the receiver processor. However, with the one-sided communication, this function realizes the whole task guaranteeing the completion of the data writing in the **target** processor window without involving it in the transfer operation.

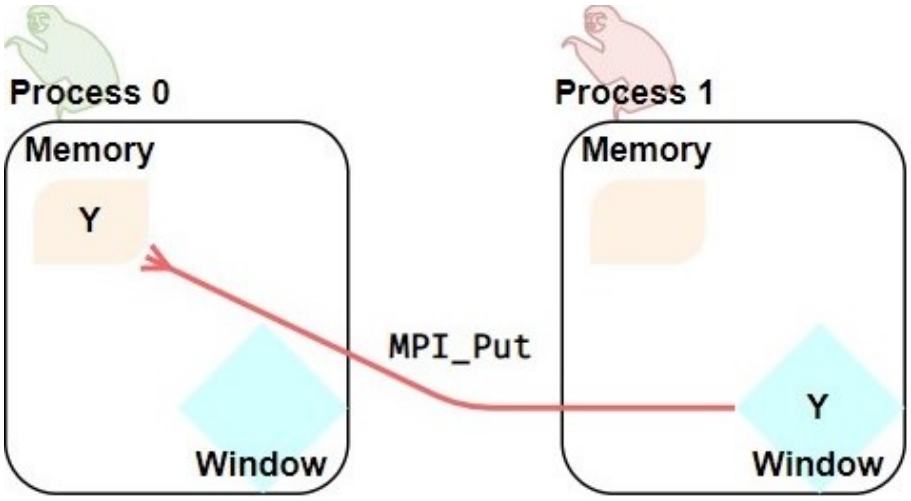


Figure 5.2: Put (Ref : ENCSS One sided communication concepts)

The figure 5.2 corresponds to a Put communication where the **origin** processor 0 copies a piece of data Y in the window whose access was allocated by the **target** processor 1.

2. **Get:** This operation is similar to **Put**, except that the communication order

is reversed. The **origin** processor reads the data in the **target** processor window and copies the value into its memory.

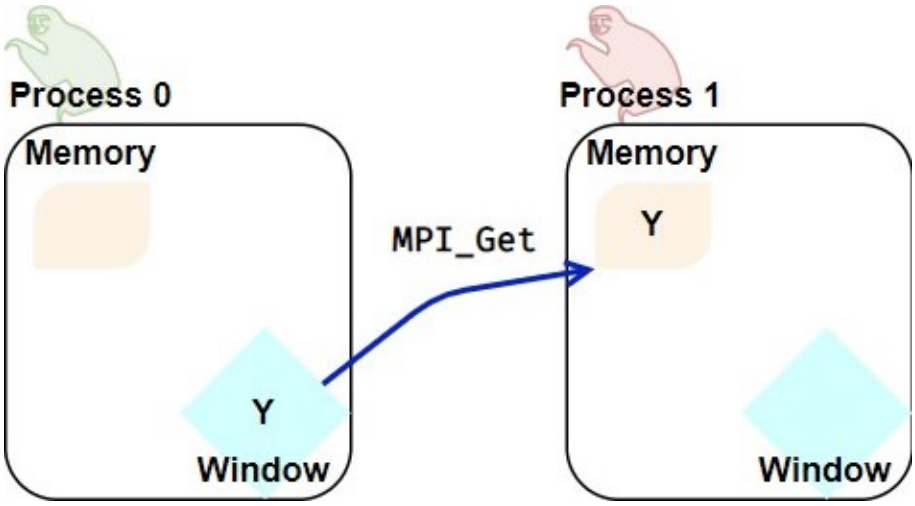


Figure 5.3: Get (Ref : ENCSS One sided communication concepts)

The figure 5.3 corresponds to a Get communication, where **origin** processor 1 recovers a data Y in the window whose access was allocated to it by **target** processor 0.

3. **Accumulate:** This operation is also similar to the **Put** operation, except that instead of overwriting the value in the **target** processor window with the new value proposed by the **origin** processor, operations are applied with the **Op** argument as in the **MPI_Reduce** function, such as **MPI_SUM**, to sum all the values coming from the **origin** processors.

5.2.5 . Synchronization

A synchronization stage generally follows these communications operations. Two types of synchronization in RMA can be distinguished: active and passive.

5.2.5.1 . Active synchronization

The target participates in the synchronization. It is similar to the classic message passing paradigm, but an amount of data can be sent without synchronization and still have to synchronize at the end, including the target ranks. Two ways are considered active synchronization which correspond to the epoch's opening and closing:

- **Window_Fence**: **MPI.Win.Fence()** is a collective operation which starts an epoch at the beginning and closes it at the end. Both the **target** and **origin** call it. Between the two calls, data access can be done as much as wanted.

Since it is a collective call, they will synchronize. So there is no need to specify the target or the origin because they are all involved, and all the ranks are starting the epoch simultaneously and closing it simultaneously, the second **MPI.Win.Fence()** call closes the epoch and enforces the synchronization.

Code Listing 5.1: Example of RMA-MPI with MPI.win.Fence()

```
# Import python package for MPI
from mpi4py import MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
# Number of processors
nprocs = comm.Get_size()
# Size of the buffer used to create the window
N=10
# Creation of the Window for the origin rank = 0
if rank == 0:
window = MPI.Win.Create(None, comm=MPI.COMM_WORLD)
# Creation of the Window for the target ranks != 0
if rank != 0:
window = MPI.Win.Create(np.zeros(N), comm=MPI.COMM_WORLD)
# Fill the send vector with
if rank == 0:
U = np.ones(N)
# Fisrt MPI.win.Fence() collective call to start the epoch
window.Fence()
# The origin rank = 0 put data in the window of target ranks
if rank == 0 :
for i in range(1, nprocs):
window.Put([uG, MPI.DOUBLE], i)
# The final MPI.win.Fence() collective call to close the
                                epoch and synchronize all
                                ranks
window.Fence()
# Window deallocation
window.free()
```

Code 5.1 illustrates MPI-RMA case with an active synchronization triggered by **MPI.Win.Fence()**. It corresponds to sending a vector of size N from processor 0 (origin) to the other processors (targets). Thus, this command's collective call can be seen at the beginning to initiate the epoch and close it in all the ranks at the end and synchronize the data simultaneously in all the processors.

- **Post/Start/Complete/Wait**: Unlike **MPI.Win.Fence()**, which is called collectively by all processors, this approach is used where only a certain number of processors are involved in communication within a specific window.

  The idea is that a **target** processor allows access to its window to a certain number of processors, which is specified by the **MPI.Win.Post** com-

mand. Then the **origin** processors execute the **MPI.Win.Start** command and pass to it the group of **target** processors they want to access their windows. Then, at the end of the transfer data operations, those **origin** processors execute the **MPI.Win.complete** command to ensure the completion of the data transfer. The **target** processors then call the **MPI.Win.Wait** function to ensure the completion of the communication at the **target** processors. Access to the **target** windows is not possible before the **MPI.Win.Post** function's call. The function **MPI.Win.Wait** cannot be executed until the **MPI.Win.complete** operation is done for all **origin** processors.

5.2.5.2 . Passive synchronization

In the passive synchronization, the target does not even participate in the synchronization, It is similar to shared memory, the idea is to access data without involving the target. The origin has the full control.

- **Open the epoch**: To open the access to the epoch, the origin processors call the command **MPI.win.Lock(Target rank)**, this command allows origin processors to access the window that the target has made accessible in an exclusive (only one processor has access at any time) or a shared way (several processors can access it at the same time). Origin processors can also call the command **MPI.win.Lock_all()** that allows controlling the window access in shared way for several associated processors.

- **Complete transfer operations**: There are several commands to ensure the completion of all RMA operations after the epoch is opened with the **MPI.win.Lock_all()** or **MPI.win.Lock(Target rank)** commands:

  1. **MPI_Win_flush(Target rank)**: Ensure that all local and remote transfer operations at the target processor on the specified window, initiated by the origin processor, are completed before continuing program execution. It allows for the optimization of program performance by overlapping communication and computation. For example, a process may initiate a remote memory access operation, then perform some local calculations before calling MPI_Win_flush to ensure the operation is completed before continuing.
  2. **MPI_Win_flush_all()**: Completes the data transfer operations remotely at all target processors on the specified window, initiated by the origin processor calling the function.
  3. **MPI_Win_flush_local(Target rank)**: Completes the data transfer operations locally at the target processor on the specified window, initiated by the origin processor calling the function.
  4. **MPI_Win_flush_local_all()**: Completes the data transfer operations locally at all target processors on the specified window, initiated by the

origin processor calling the function.

After completing the operations with these commands, the data can be read or written in the window.

- **Close the epoch**: At the end of the RMA operations, the origin processor calls the command **MPI.win.Unlock(Target rank)**, which closes the access to the window. After having been reassured that all operations have been completed. Or they call **MPI.win.Unlock_all()** if the epoch was opened with the command **MPI.win.Lock_all()**.

Code Listing 5.2: Example of RMA-MPI with Passive synchronization()

```
# Import python package for MPI
from mpi4py import MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
# Number of processors
nprocs = comm.Get_size()
# Size of the buffer used to create the window
N=10
# Creation of the Window for the origin rank = 0
if rank == 0:
window = MPI.Win.Create(None, comm=MPI.COMM_WORLD)
# Creation of the Window for the target ranks != 0
if rank != 0:
window = MPI.Win.Create(np.zeros(N), comm=MPI.COMM_WORLD)

if rank == 0:
# The origin rank = 0 put data in the window of target ranks
uG = np.ones(N)
window.Lock_all()
for i in range(1, nprocs):
window.Put([uG, MPI.DOUBLE], i)
window.Flush_all()
window.Unlock_all()
# Window deallocation
window.free()
```

Code 5.2 shown a version of the code Code 5.1 with a Passive synchronization. **MPI.Win.Lock_all()** is used to open an epoch on all the target processors and close it with **MPI.Win.Unlock_all()**, also **MPI.Win.Flush_all()** is used to ensure the completion of the **Put** operations on all the target processors. So as it can be seen, this operation involves only the processor of rank 0 which sends a vector of size N to the other processors (targets).

5.3 . Illustration

To illustrate what has been presented in the previous sections, the following ping-pong algorithm is used:

- Rank = 0 : Sends a buffer to rank 1
- Rank = 1 : Modifies the buffer and sends it back to the rank 0

This algorithm is implemented in 3 different parallel versions:

- A non-blocking version with two-sided communication (Isend, Irecv)
- A version with the one-sided communication using active synchronization (blocking)
- A version with the one-sided communication using passive synchronization (asynchronous)

The idea is to use two processors and to increase the size of the sent buffer to analyze the impact of its length on the performance of those techniques. This study was conducted using two processors within one node, the size vector is scaled from 1 to 100.000 The computation time is the average of the computation times required by the 2 processors after several simulations. In Figure 5.4, the curves giving the computation time can be seen according to the size of the vector sent.

The passive synchronization is ten times faster with small vectors and two times faster for the largest vectors. However, the active synchronization model is slightly faster than the non-blocking two-sided communication even if this synchronization using Fence is expensive by blocking each time to complete the communications before continuing the calculations. This quick study on two processors with a simple algorithm like this one, where the computation load is not very important, allows to show the clear improvement RDMA can bring compared to the classical two-sided communication model.

5.4 . Code details

A homemade code is implemented for the non-intrusive global/local coupling method during the Ph.D Thesis. This code was realized in python. The non-intrusive aspect of the global/local coupling allows the use of industrial finite element codes. But in our case, the (more research-oriented) finite element library Getfem [92] is used, which allowed to manage all the construction parts of the finite element model, the interpolation operators for the coupling, and the resolution part using the solvers already linked by this library (like the famous MUMPS solver). The choice was particularly motivated by the availability of postprocessing methods to obtain the nodal reaction of the complement domain $\boldsymbol{\lambda}^0$, which is an operation not always simple to implement in legacy industrial software.

For the construction of the finite element model, Getfem proposes two ways:

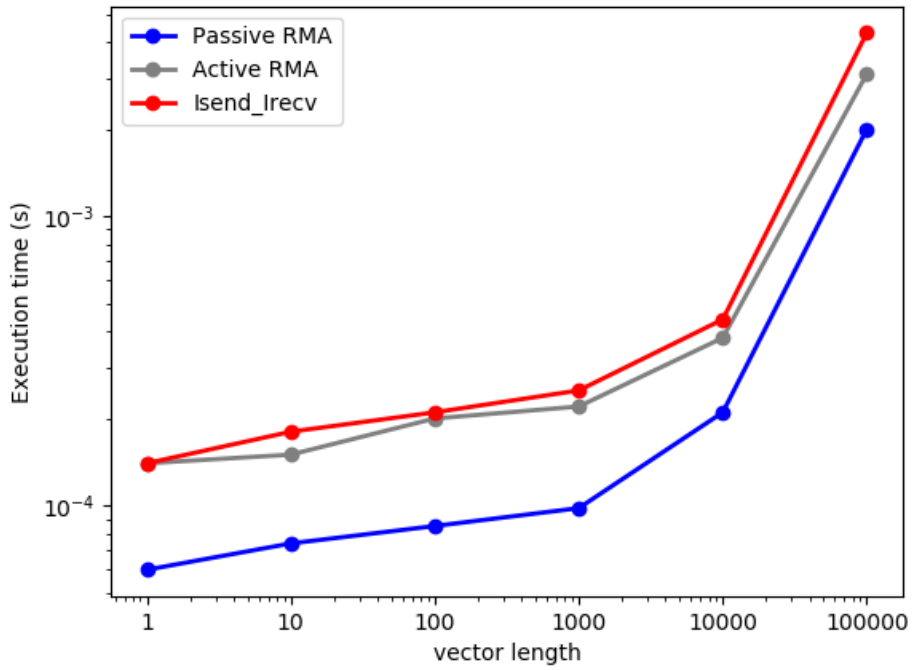


Figure 5.4: RMA vs Two sided

- The construction of the linear system and its exportation in matrix-vector format in order to solve it externally with chosen algebraic solvers.

- The use of bricks where no need to build the linear system explicitly. Getfem then appears like a black box to which the input data are specified like the problem the presented problem, the boundary conditions, and the right-hand side.

This last option is the one selected during the thesis. It allows for solving several types of linear problems in thermal or linear elasticity, non-linear elliptic problems, and problems of elastoplasticity type.

The developed code allows solving these problems with a global/local coupling parallelized in synchronous or asynchronous way. MPI [25] is used for the parallel processing part and specially MPI-RMA techniques.

5.4.1 . Code architecture

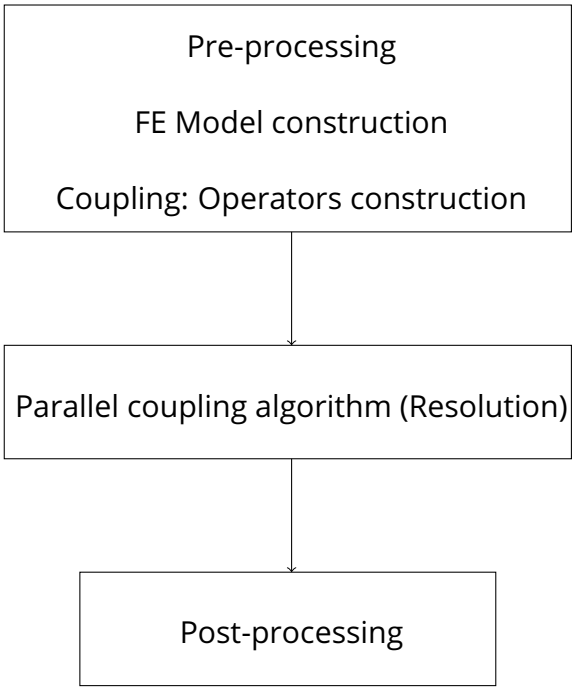
The code is structured in several modules:

- **Preprocessing**: This module contains the **interface_identification** function, which is in charge of reading the data from the provided mesh, de-

tecting the number of patches and identifying the interfaces of each patch, seeing if there are common interfaces with other patches.

- Several problems have been treated, and several modules have been defined: **linear_elasticity**, **laplace_gl** (for thermal problems in linear and nonlinear), and then a **plasticity** model. Each of these models contains those two functions:
  - **FE_GLOBAL_MODEL**: defines the global problem on the whole structure with its boundary conditions and right-hand side.
  - **FE_LOCAL_MODEL**: function called in parallel by the patches to build the local problems and impose the correct boundary conditions if it is a patch on the Dirichlet boundary.
- **Coupling**: This module contains two functions:
  - **Global_to_local_coupling**: This allows to build of the interpolation operator, which helps to pass the global coarse mesh to the local fine mesh. This operator is computed in parallel for each patch.
  - **Add_lagrange_multipliers**: this function allows defining and initializing for each local problem its Lagrange multipliers on its interface.
- **Resolution**: This module contains three functions:
  - **Global_resolution**: It solves the global problem on the whole structure. This function allows the resolution with the Aitken accelerator as well as with a fixed relaxation coefficient.
  - **Local_resolution**: It is called in parallel and allows solving the local problem for each patch and then post-processing to calculate the nodal reaction or the Lagrange multiplier on the interface.
  - **Residual_computation**: It is called at each iteration to evaluate the residual. It recovers the nodal reactions calculated by the local problems. It post-processes the global problem in case the complementary domain exists to calculate the nodal reactions on the interface from a global point of view.
- **Post_processing** : Depending on the problem studied, this module contains several functions to calculate the gradient in the case of elliptic problems or the von Mises stress in the case of mechanical problems. This module also allows plotting the solution (displacement or temperature).

The architecture and calling order of the modules presented in the code are illustrated in the following diagram:



From this diagram it can be seen that the central part is the coupling algorithm part where the parallelization is done. This part only uses the **Resolution** module. In the following two sections, the details of the different parallelization techniques used are presented. The associated pieces of code are given in appendix.

5.4.2 . Window creation

The non-intrusive global/local coupling algorithm is characterized by sending the Dirichlet conditions from the global to the local patches and sending the nodal reactions from the local patches to the global. One last communication is set up for the management of the convergence detection, the global sends a message to the locals to signal the algorithm's convergence. Each of these three communications is assigned a window.

So in conclusion parallel computing scheme master-slave is the one used.

- To send the Dirichlet conditions from the global problem to local problems. The processor of rank 0, in charge of the global problem computation, sends to each processor in charge of a local calculation the part of the displacement corresponding to its interface. Each of these processors must allocate a memory corresponding to the window in which the global problem will come to put the information.

  The part of the code B.1 corresponds to the allocation of these windows

- Sending nodal reaction from the processors in charge of the local problems to the processor of rank 0 requires creating a window in the memory of

processor 0 for each of these local problems. So a list of vectors of adequate size is created and a list of windows associated with each of the vectors.

The part of the code B.2 corresponds to the allocation of these windows.

- After detecting the convergence, the global problem sends a message to the local problems, which stop the calculations on all the processors. For that, a boolean with an initial value of 0 is used, and if the algorithm converges, it has the value 1. So a window is created in each local processor so that the processor of rank 0 can write the information.

  The part of the code B.3 corresponds to the allocation of these windows.

At the end of the algorithm and after convergence, the memory allocated to these windows is freed as in the code B.4.

5.4.3 . Synchronous version

In this subsection, the focus is on the synchronous parallel implementation of the coupling algorithm, which can be summarized in 3 blocks:

- Global displacement window synchronization
- if rank $= 0$:
  - Global computation.
  - Put global displacement in local windows.
- Global displacement window synchronization

As explained before, this first block is wrapped between two synchronizations, which correspond to the opening and closing of an epoch. During this epoch, processor 0 performs a resolution of the global problem and then sends the Dirichlet boundary conditions to the local problems.

- Local nodal reaction window synchronization
- if rank $\neq 0$:
  - Local computation.
  - Put local nodal reaction in the global window.
- Local nodal reaction window synchronization

Again, this second block is wrapped too between two synchronizations, corresponding to the opening and the closing of an epoch. During this epoch, All the other processors except rank 0 solve their local problems. They then post-process the results to compute the nodal reactions on the interface and put them in the global window.

- Convergence boolean window synchronization
- if rank $= 0$:
  - Residual computation.
  - Put convergence boolean to local windows.
- Convergence boolean window synchronization

The last block has the same two synchronizations, corresponding to an epoch's opening and closing. During this epoch, After the evaluation of the residual, the processor of rank 0 checks the convergence status and then sends an update value of the boolean to the other processors. Algorithm 4 shows the pseudo-code corresponding to the synchronous version.

Algorithm 4: Synchronous stationary iterations using RDMA

```
Window creation + Initialization p_Γ = 0, ω sufficiently small
while ‖r‖ is too large do
    MPI.Fence()(For the global displacement window)
    if rank == 0 then
        Resolution of the Global system (3.30) or (3.32), u_Γ^G = S^{G^{-1}}(p_Γ + b^G)
        if Ω^0 exists then
            Post-processing (3.24), q^0 := λ^0 = S^0 u_Γ^{0,G} − b^{0,G}
        end
        Put A^{s^T} u_Γ^G in subdmains s > 0 windows ;
    end
    MPI.Fence()(For the global displacement window)
    MPI.Fence()(For the local nodal reaction)
    if rank != 0 then
        Fine solution (3.22), λ^{s,F} = S^{s,F} J^s A^{s^T} u_Γ^G − b^{s,F}
        Patch Put q^s := J^{s^T} λ^{s,F} in the rank 0 window
    end
    MPI.Fence()(For the local nodal reaction)
    MPI.Fence()(For the convergence detection window)
    if rank == 0 then
        Global computes residual r = − Σ_s A^s q^s
        Global updates p_Γ = p_Γ + ωr
    end
    MPI.Fence()(For the convergence detection window)
end
Window free
```

Code B.5 shows the implementation of the synchronous version in our code. Each part is commented on and allows making the distinction between each of the three blocks presented before

5.4.4 . Asynchronous version

For the implementation of the asynchronous algorithm, MPI-RMA passive synchronization presented above is used, two implementations are considered:

- The first implementation proposes an asynchronous model that updates itself as soon as new information arrives. This algorithm can be divided into three blocks:
  - Rank $= 0$:
    1. The global problem is solved by using a specific relaxation coefficient and updating it with a Neuman condition based on an asynchronously computed residual. The global displacement is stored in a buffer whose last component contains a variable indicating the current global iteration.
    2. To put the displacement as a boundary condition for the other processors, an epoch is open for each one using passive synchronization. The putting is realized, and the completion of this sending is assured before closing this epoch.
  - Rank $\neq 0$:
    1. To check if a new Dirichlet condition is available, each processor launches a loop that opens an epoch in its window and checks if the variable corresponding to the global problem iteration has been updated.
    2. Once sure to have new information, each processor solves its local problem, then performs post-processing to calculate the nodal reactions.
    3. Each processor opens an epoch on its allocated window on the target rank 0, puts the nodal reaction, and then closes the epoch. The nodal reactions calculated by each processor are stored in a buffer whose last component contains the corresponding local iteration.
  - Rank $= 0$:
    1. The rank 0 launches a loop on all the windows in which the other processors put their nodal reactions and checks with the variable corresponding to the number of local iterations of each one if new data is accessible.
    2. Rank 0 evaluates the residual, computes its norm, and checks if the algorithm has converged or not.

3. If the algorithm has converged, rank 0 opens an epoch in the window allocated by each of the other processors, puts the boolean with the new value that corresponds to the convergence, and then closes the epoch.

The pseudo-code in Algorithm 5 presents a version based on MPI-RDMA techniques with passive synchronization.

Algorithm 5: Asynchronous iterations using RDMA

```
Window creation + Initialization p_Γ = 0, ω sufficiently small
MPI.Lock(target) (For all the window by specifying the specific target of
 each one) while ‖r‖ is too large do
    switch Rank s do
        case s == 0 (global domain) do
            Global reads all (q^s) in its windows
            Global computes residual r = − Σ_s A^s q^s
            Global updates p_Γ = p_Γ + ωr
            Global solve system (3.32), u_Γ^G = S^{G^{-1}}(p_Γ + b^G)
            if Ω^0 exists then
                Post-processing (3.24), q^0 := λ^0 = S^0 u_Γ^{0,G} − b^{0,G}
            end
            Global puts A^{s^T} u_Γ^G in all subdomains s > 0 windows
            Flush(subdomains s window)
        end
        case s > 0 (local patch) do
            Local reads new A^{s^T} u_Γ^G in its window
            Local solves (3.22), λ^{s,F} = S^{s,F} J^s A^{s^T} u_Γ^G − b^{s,F}
            Local puts q^s := J^{s^T} λ^{s,F} in Rank o's window
            Flush(o)
        end
    end
end
MPI.UnLock(target) (For all the window by specifying the specific target of
 each one)
```

The implementation of this algorithm is in code B.6.

This implementation is well recommended if the number of patches is not very significant. In the chapter of applications it will be shown that because of the network used, the choice of this implementation is restricted to few cases

- The second implementation corresponds to the case where the global and the local computation is done all the time, whether with new information or just reusing the information they have in the memory from previous iterations.

  This implementation can be summarized in the following five blocks:

  - Contrary to the first implementation, the epochs are opened and closed at each communication, in this implementation, the epochs are opened before the launching of the algorithm, which will allow to do a series of communications during these epochs until the convergence of the algorithm, and then to close the epochs at the end.
    1. rank $= 0$: Rank 0 opens access to all the open windows allocated by the other target processors to send the Dirichlet conditions and also the boolean corresponding to the stopping criterion of the algorithm after the convergence
    2. rank $\neq 0$: Each processor opens the access to the epoch allocated by rank 0 to receive the nodal reactions.
  - Rank $= 0$:
    1. The global problem is solved by using a specific relaxation coefficient and updating it with a Neuman condition based on an asynchronously computed residual.
    2. The displacement is sent as a boundary condition for the other processors, and the completion of this sending is assured on all the target processors.
  - Rank $\neq 0$:
    1. Each processor solves its local problem with the available boundary Dirichlet condition, then performs post-processing to calculate the nodal reactions.
    2. Each processor puts the nodal reaction on the window allocated by the rank 0 and ensures its completion.
  - Rank $= 0$:
    1. Rank 0 evaluates the residual with the available nodal reactions, computes its norm, and checks if the algorithm has converged or not.
    2. If the algorithm has converged, rank 0 puts the boolean with the new value corresponding to the convergence in the window allocated to it by each of the other processors and ensures its completion on all the target processors.
  - After the convergence of the algorithm:

1. rank $= 0$: Rank 0 closes all the opened accesses to the windows allocated by the other target processors.
2. rank $\neq 0$: Each processor closes the access to the windows allocated by rank 0 to receive the nodal reactions.

The implementation of this algorithm is in code B.7. It is recommended in the case of a very high number of patches, where the probability of an update from one of these patches to the global problem is very high.

6 - Applications

This last chapter, interest in confirming the theoretical results presented previously and examining the performance of the implemented asynchronous algorithm compared to the synchronous one. A variety of linear and nonlinear problems have been studied. For this purpose, the study moved from simple 2D academic cases to more complicated 3D academic and industrial cases. The idea was to compare the asynchronous and synchronous algorithms on different global/local coupling situations considering few or many patches. Weak scalability studies as well as tests that focus on the load imbalance that is regularly faced in this type of problems are considered.

6.1 . Setup

6.1.1 . Methods

During all these studies, the first idea is to compare the synchronous and the asynchronous models without relaxation. A second goal is, to compare the best performance attainable. In the synchronous case, the powerful Aitken accelerator is used (which can be viewed as an efficient way to find a good dynamic relaxation), in the asynchronous, an optimal relaxation coefficient obtained empirically by trial-and-error is used.

Note that the concept of optimal relaxation is a bit ill-posed for the asynchronous case, as the relaxation depends on the frequency of the updates which is hardware dependent.

Note that asynchronous acceleration techniques have been investigated during the thesis to improved more performance to the asynchronous model, following the works in [26] where the idea was to accelerate the convergence of asynchronous iterations with Aitken's acceleration technique using a low-rank approximation based on SVD. The study presents several examples in the case of the RAS method. The idea is based on the fact that the error operator depends on the iterations and the use of the SVD for a low-rank approximation of this operator allows accelerating the convergence based on an SVD applied to the iterated solutions. After several attempts with this technique, It can be conclude that it is still slower than an empirically relaxed asynchronous model. In the rest of the study, this approach will not be considered.

For the rest of this chapter:

- *Synch*: synchronous iteration without relaxation($\omega = 1$),
- *Aitken*: Aitken-accelerated (synchronous) iteration,

- *Async*: asynchronous iteration without relaxation ($\omega = 1$),
- *Relax async*: asynchronous iteration with optimized relaxation.

In order to evaluate the performance, the number of iterations performed by the global problem and the number of maximum and minimum iterations performed by the local problems is presented. Of course, for the synchronous version, the number of iterations performed is the same for the global and local problems.

We also present wallclock time measurements. In order to smooth the variability of asynchronous computations, the average over 3 executions is given with identical setup.

6.1.2 . Cluster

The studies were carried out with the cluster of the LMPS simulation center using several workstations with an ethernet network. These machines are quite heterogeneous with 4 different generation of CPUs:

- Intel(R) Xeon(R) CPU E5-1660 v3 (Haswell) @ 3.00GHz (8 cores)
- Intel(R) Xeon(R) CPU E5-2630 v4 (Broadwell) @ 2.20GHz (10 cores)
- Intel(R) Xeon(R) Silver 4116 CPU (Skylake) @ 2.10GHz (12 cores)
- Intel(R) Xeon(R) W-2255 CPU (Cascade Lake) @ 3.70GHz (10 cores)

Besides the heterogeneity, another characteristic is that the cluster can be used by several users simultaneously, there is no queuing system.

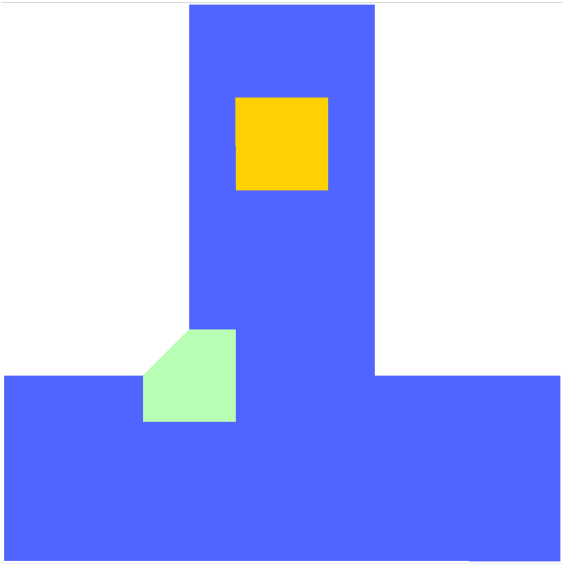
We use one MPI process for the global problem and as many processes as need to distribute the local problems according to the study. As much as possible the MPI processes are allocated to the cores of the same CPUs.

6.1.3 . Academic cases

6.1.3.1 . 2D test-case

The first test-case, Figure 6.1, is supposed to be a 2D approximation of a 3D turbine blade, it is inspired from [47]. This case comprises two zones of interest in which complex geometrical details are added. In the green zone of interest, the addition of 3 circular holes and a modification at the boundary of the global structure can been seen. The yellow patch inside the structure is characterized by the addition of four square voids.

For this case, the subdomain 0 (complementary zone) exists, it is represented in blue in the global model. Null Dirichlet conditions are imposed on the bottom side, null Neumann conditions are imposed elsewhere. A constant body load is given. These two areas of interest have a refined mesh compared to the complementary area.



(a) Global problem

(b) Zones of interest
Ce cas contient une zone complémentaire

Figure 6.1: 2D academic test-case

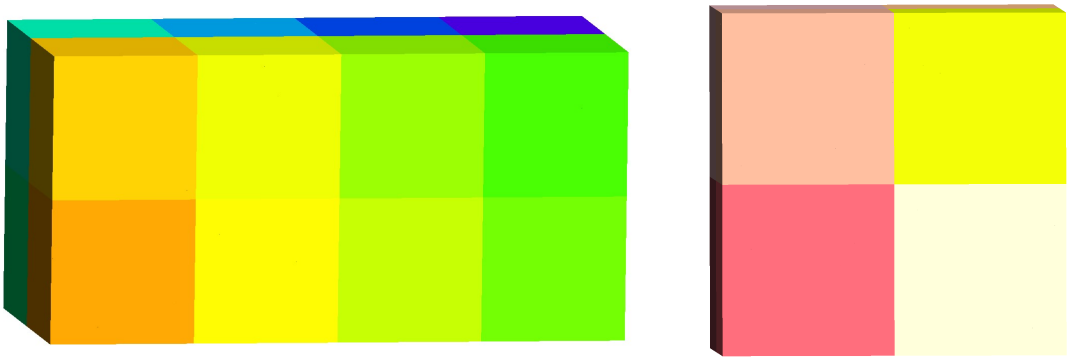
The idea of the study will be to consider linear or nonlinear behaviors and to see locally what the geometrical details bring. This case composed of 2 patches is too simple to allow us to conclude on the performance of the asynchronous version. Still, it allows simple illustrations, and it enables us to close the loop initiated by our presentation of the global/local coupling of Section 2.3.

6.1.3.2 . 3D test-case

For the second academic study, the idea is being able to generate as many patches as desired, in particular non-overlapping contiguous patches that cover the global model. Thus a cuboid domain covered by cuboid patches is considered, see for instance the 8-patch case on Figure 6.2b and the 16-patch case on Figure 6.2a.

Two representations of the cubes are considered. The cubes in the global model are homogeneous, and their mesh is coarse, see Figure6.3a. The cubes in the local models have refined meshes (see Figure 6.3b), they are heterogeneous, with a spherical inclusion inside each cube (by default the sphere is centered radius is a quarter of the side of the cube, but these parameters vary in some studies, see Figure 6.3c). Note that the fine meshes are built independently on the patches so that they are not constrained to match at the interface. On the contrary, the global mesh is conforming at the interface.

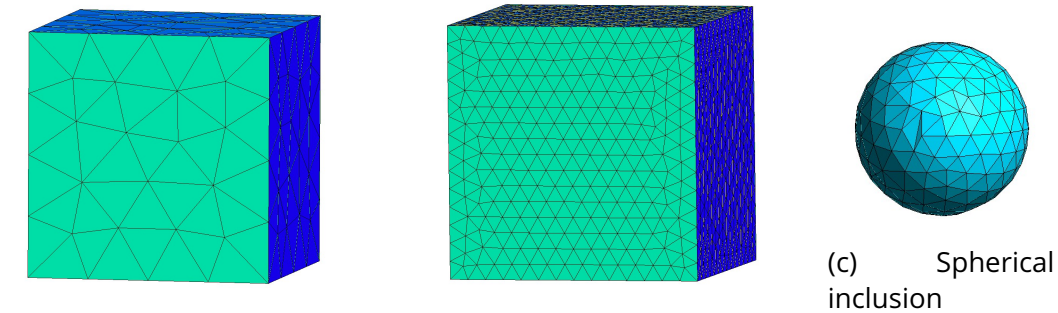
The contrast between the material properties of the sphere and of the cube is a parameter of the studies.



(a) 16 patches (b) 8 patches

Figure 6.2: fig:3Dacad

Null Dirichlet boundary conditions are imposed on one face of the domain, null Neumann conditions are imposed elsewhere. A constant body load is given.



(a) Coarse representation (b) Fine representation (c) Spherical inclusion

Figure 6.3: Example of one unit cube with different representations

6.2 . Linear cases

The asynchronous global/local coupling is assessed on two academic examples: the simple 2D case of Figures 6.1a and 6.1b, and the 3D case involving many patches like in the Figure 6.2b.

For this two linear problems are considered:

- The Poisson equation, which models thermal problems:

$$\begin{aligned} &\text{Find } u : \Omega \subset \mathbb{R}^d \to \mathbb{R} \\ -\operatorname{div}(a \operatorname{grad}(u)) &= f \text{ in } \Omega \\ u &= 0 \text{ on } \partial_d \Omega \\ \frac{\partial u}{\partial n} &= 0 \text{ on } \partial\Omega \setminus \partial_d \Omega \end{aligned} \tag{6.1}$$

  For simplicity, homogeneous boundary conditions are used. In some cases a contrast of conductivity coefficient a is used. The source term is simply equal to 1.

- The linear elasticity equation:

$$\begin{aligned} &\text{Find } u : \Omega \subset \mathbb{R}^d \to \mathbb{R}^d \\ \operatorname{div}(\sigma) + f &= 0 \text{ in } \Omega \\ u &= 0 \text{ on } \partial_d \Omega \\ \sigma \cdot n &= 0 \text{ on } \partial\Omega \setminus \partial_d \Omega \\ \sigma &= \frac{E}{1+\nu}\left(\varepsilon(u) + \frac{\nu}{1-2\nu}\operatorname{tr}(\varepsilon(u))I\right) \\ \varepsilon(u) &= \frac{1}{2}(\nabla u + (\nabla u)^T) \end{aligned} \tag{6.2}$$

  E is Young's modulus, and $\nu = 0.3$ is Poisson's coefficient. In some cases, a contrast of Young's modulus is used. The source term is simply the vector with all components equal to 1.

To avoid redundancy in the results, the results in linear thermal and others in linear elasticity are either considered both cases are presented only when important distinctions appear.

6.2.1 . Simple 2D test-case

To begin with the illustrations, the test-case of Figures 6.1a and 6.1b is considered where the patches only introduce geometric alterations. The patches and the global model are treated on three different cores.

As shown in Table 6.1, the problem is of very small dimension, and the patches are well-balanced, which is in favor of synchronous algorithms.

Problem	**Global**	1^{st} **Zone of interest**	2^{nd} **Zone of interest**
Nodes	701	381	379

Table 6.1: Size of the domains or the 2D test case.

Tables 6.2 and 6.3 present the performance in terms of time and number of iterations (the numbers in the brackets correspond to the number of solves in the patches). For these small cases, Aitken remains unbeatable. The interest of finding a good relaxation for the asynchronous iteration is observed to perform better than the raw synchronous iteration.

To explain the choice of optimal relaxation coefficient, in Figure 6.4 the computation times obtained for different relaxation coefficients is presented. The plot tends to show a fairly well-marked minimum. Note that the same approach to choosing the relaxation coefficient is applied to all the examples that will follow. However, the value of the coefficient is insignificant as it depends not only on the mechanical problem but also on the hardware configuration, so we did not find it useful to systematically write it.

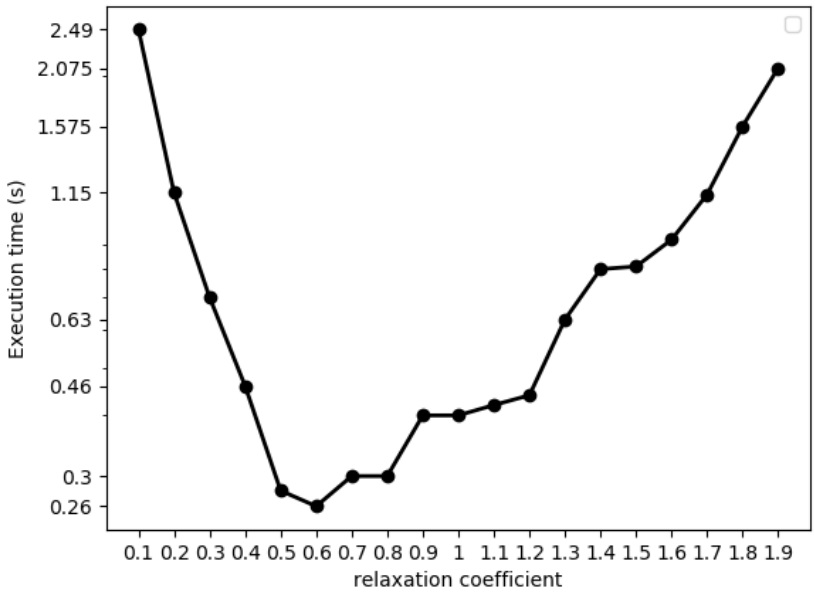


Figure 6.4: Relaxation coefficient study for thermal problem

What is more interesting to observe is the large amount of computation that can be done by the asynchronous solver thanks to the removal of waiting time.

6.2.2 . Preliminary study

- Linear elasticity problem
- 16 subdomains 6.2a
- Heterogeneity: Young's modulus in the spherical inclusion is 100 times lower than in the rest of the cube, the global model uses the stiffer modulus.

Variant	**Sync.** $\omega = 1$	**Sync.** Aitken	**Async.** $\omega = 1$	**Async.** ω_{opt}
Time (s)	0.31	0.17	0.4	0.26
#iter. glob.	23	12	43	35
#loc. sol. [min, max]	·	·	[95, 106]	[82, 85]

Table 6.2: 2D test-case: performance for thermal problem.

Variant	**Sync.** $\omega = 1$	**Sync.** Aitken	**Async.** $\omega = 1$	**Async.** ω_{opt}
Time (s)	0.67	0.3	0.6	0.52
#iter. glob.	43	16	53	48
#loc. sol. [min, max]	·	·	[112,119]	[100,107]

Table 6.3: 2D test-case: performance for the elasticity problem.

- Convergence criterion: global residual $< 10^{-7}$

Problem	**Global**	**Local (One subdomain)**
Numbre of nodes	2675	1249

Table 6.4: Mesh data for subsection 6.2.2

Considering these data, several machine configurations are tested. In the different situations, comparing the two synchronous models (with optimal relaxation) presented in the previous chapter (with and without waiting for new information) with the synchronous Aitken.

- **1** 16-core machine oversubscribed with 17 MPI processes. In this case study, the local ethernet network is not used. The case of a shared memory configuration within a single machine is considered.

Aitken #iter #time (s)	**Asynchronous with wait** #iter. glob. [[min, max] #loc.] #time (s)	**Asynchronous w/o wait** #iter. glob. [[min, max] #loc.] #time (s)
21 & 37.65s	540[62, 92] & **14.18s**	649[71, 106] & **15.83s**

- **2** 10-core CPUs non-oversubscribed for 17 MPI processes. The idea it to force the use of the local ethernet network between the machines. However, In this configuration the machines are in fully utilized.

Aitken #iter #time (s)	**Asynchronous with wait** #iter. glob. [[min, max] #loc.] #time (s)	**Asynchronous w/o wait** #iter. glob.[[min, max] #loc.] #time (s)
21 & 7.36s	64[64, 65] & **6.67s**	458[60, 106] & **6.65s**

- **17** CPUs using 1 core per machine. In this situation the CPUs are not much solicited but all the communications pass through the network.

Aitken #iter #time (s)	**Asynchronous with wait** #iter. glob. [[min, max] #loc.] #time (s)	**Asynchronous w/o wait** #iter. glob. [[min, max] #loc.] #time (s)
21 & **4.79s**	67[67 - 68] & 8.77s	749[65, 171] & 8.81s

Comparing the computation times, it can be seen that the two asynchronous models are much faster than the Aitken in the first case with an oversubscribed machine. the difference decreases in the second case, but the two asynchronous models remain faster. In the third case, where machines have a light computational load, as the load balance is almost perfect, the processes are naturally progressing synchronously, and the Aitken technique is faster than the asynchronous models.

In a second analysis, the number of iterations is compared. As expected, Aitken is a deterministic method which requires the same number of iterations in the 3 cases.

However,the asynchronous model without wait has a very variable number of iterations depending on the configuration, with a significant number of global iterations compared to the local ones, due to the repetition of the global computations without new piece of information.

For the asynchronous model with wait, It can be seen that in the first case with a single machine, the number of iterations varies between global and local problems. However, the passage to several machines linked by the local ethernet network leads to a similar number of iterations between the local and the global. In fact our Ethernet network does not support RDMA communication by default, and it generates implicit synchronizations when **MPI.win.Lock** and **MPI.win.Unlock** commands are used to check if new data is available in the target processor.

Thus, to remain in a purely asynchronous mode, the following studies the asynchronous mode without waiting is considered for the analysis of the study introducing several calculation machines. However, the asynchronous model with waiting will be used for the cases with few patches where the calculation can be done within a single machine. Note that other infrastructures, like infiniband, allow RDMA networking.

6.2.3 . Rate of communication study

In order to better grasp the impact of synchronization on communication and waiting time, a preliminary study is proposed based on the thermal problem set on the well-balanced 64-subdomain case (with a heterogeneity ratio of 100, the thermal diffusion coefficient in the spherical inclusion being 100 times lower than in the rest of the cube). The dimensions of the problems are recalled in Table 6.5. Different numbers of MPI processes are used. This means that for less that 65

processes, one MPI rank has to handle several subdomains. The percentage of communication time in the total simulation time is presented, for a convergence tolerance of 10^{-7} for the norm of the residual.

Problem	**Global**	**Local (One subdomain)**
Number of nodes	1449	1858

Table 6.5: Mesh data

First, the synchronous and asynchronous iterations are compared without relaxation.

#ranks	**Synchronous** #iter #time (s) [% communication time]	**Asynchronous** #iter. glob.[#loc. sol. [min, max]] #time (s) [% communication time]
9	– & ≥1h	7683[980, 1109] & 455s [8%]
17	– & ≥1h	7682[1804, 3856] & 469s [17%]
33	– & ≥1h	7692[3828, 7364] & 481s [35%]
65	7707 & 5362.83 [94.8%]	7692[3249, 7737] & 489.81s [63%]

Table 6.6: Analysis of the time spent in communication (64-subdomain thermal case) without relaxation.

In table 6.6, the number of iterations, the calculation time, and the percentage that the communication time represents of this calculation time are summarized.

For the synchronous version, the cases studied could not finish their calculation after one hour, which corresponds to the time allocated on the machines, except the last case with 65 CPUs, where a very high percentage (95%) of time spent in communication are seen. This result can be explained by several factors, the most important of which is the sequential side of the method, which imposes synchronization when sending data from global to local and local to global. One can also note that the computational load is very light, so most of the time is spent managing communications.

In the asynchronous case, It can be seen that the computation time is quite close in all the cases and more than ten time faster than the synchronous version, with a global number of iterations almost constant in all cases. However, the number of local iterations changes from one case to another because the computation load per CPU decreases, which leads to the increase of the number of iterations performed, one can also see that the percentage of communication increases with the rise of the number of processes, it remains around 65% in the case involving 65 processes.

Now, the focus is on the case where the comparison is between the Aitken accelerator for the synchronous iteration and optimal relaxation for the asynchronous iteration.

#ranks	**Aitken** #iter #time (s) [% communication time]	**Asynchronous** #iter. glob.[#loc. sol. [min, max]] #time (s) [% communication time]
9	25 & 11.72s [30%]	334[49, 54] & 22.4s[10%]
17	25 & 8.08s [80%]	182[56, 77] & 13.25s[10%]
33	25 & 4.53s [71%]	104[65, 124]& 8.13s[16%]
65	25 & **8.57s [97%]**	105[81, 160] & **8.40s[46%]**

Table 6.7: Analysis of the time spent in communication (64-subdomain thermal case).

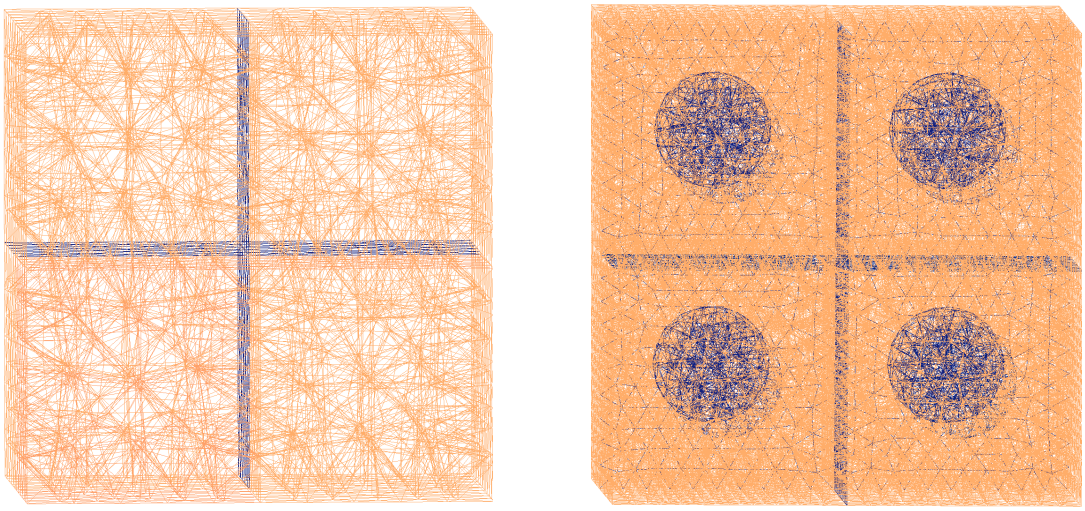
The same study as in the previous one is considered. On Table 6.7 it can be observed that, in this case, the asynchronous approach is globally slower than the accelerated synchronous one. However, the proportion of time spent in communication increases strongly in the synchronous case (up to 97%) and much more moderately in the asynchronous case (never more than 50%), which leads to the asynchronous approach being faster in the 65-process case. In particular the transition between one node (9 subdomains) computation and two nodes (17 subdomains) leads to a strong increase of the time spent in communication in the synchronous case whereas it is unmoved in the asynchronous case.

6.2.4 . Weak scalability 3D test-case

Now the academic 3D case is considered. The idea is to realize a study of weak scalability. A cubic geometry as in Figure 6.5 is preserved while adding patches. The cases made out of n^3 ($n = 2 \dots 7$) cube patches are treated. As classically done for weak scalability assessment of domain decomposition methods, the size of the domain increases with the number of subdomains. Note that the whole domain is covered with patches ($\Omega^0 = \varnothing$). The Global model is homogeneous, whereas the Local models contain one softer spherical inclusion, see Figures 6.5a and 6.5b. One side of the Global model is submitted to Dirichlet conditions. In the case of thermal problems, the inclusions have a diffusion coefficient 10 times lower than the rest of the domain, whereas in the elasticity case the Young's modulus in the inclusions is 100 times lower than in the rest of the domain.

Even if their meshes are not identical, the patches are well-balanced in terms of degrees of freedom and numerical complexity (since the problem is linear). Of course, the Global model grows along the study, from 8 times smaller than one patch to 3.7 times larger. Table 6.8 sums up the number of nodes for each case.

Figures 6.6 and 6.7 compare the performance in wallclock time of the relaxed



(a) Global model (b) Reference model

Figure 6.5: Weak scalability test-case: $2 \times 2 \times 2$ subdomains

# of subdomains	**8**	**27**	**64**	**125**	**216**	**343**
Global	233	667	1449	2681	4465	6903
Local (1 subdomain)	1858	1858	1858	1858	1858	1858

Table 6.8: Number of nodes in the meshes for the weak scalability study.

asynchronous iteration (with hand-tuned relaxation) and the synchronous iteration with Aitken's dynamic relaxation. The good performance of the asynchronous version is observed despite the good load-balance.

For small test-cases (8 and 27 subdomains), the size of the global problem is negligible compared to the locals'. This means that the sequential phase of the synchronous coupling is realized very quickly and this leads to the Aitken accelerator being faster than the asynchronous solver. However, for 64 subdomains and more, this step becomes heavier and takes more synchronous time. For the asynchronous method, the Global solve is realized simultaneously as the locals'. Thus, the execution time increases very slightly from one case to another and remains 2 to 3 times lower than for Aitken.

#patches	**8**	**27**	**64**	**125**	**216**	**343**
Aitken #iter.	11	13	12	11	11	11
Async. #iter. glob.	255	256	87	65	69	71
Async. #loc. sol. [min, max]	[32,39]	[43,74]	[49,153]	[84,207]	[276,694]	[407,2902]

Table 6.9: Weak scalability: Number of iterations in the thermal case.

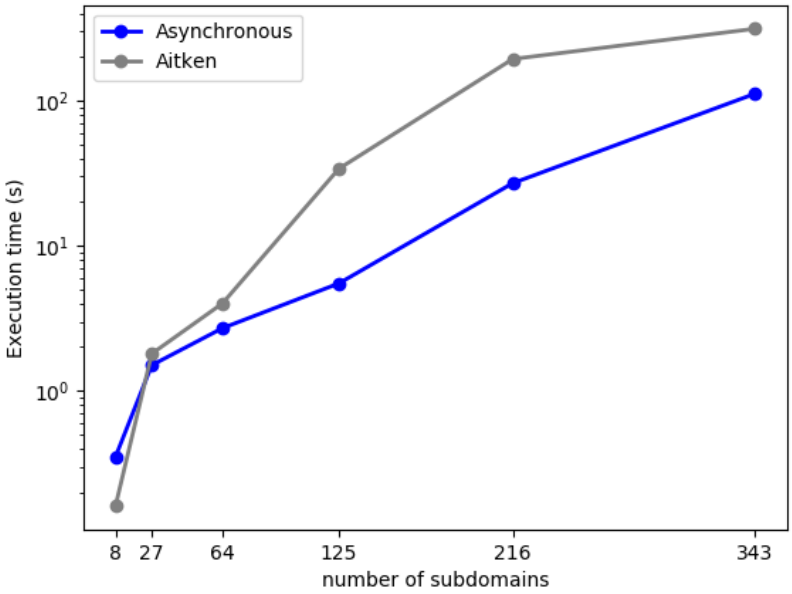


Figure 6.6: Time performance in the weak scalability study for linear thermal problem

#patches	**8**	**27**	**64**	**125**	**216**	**343**
Aitken #iter.	22	21	25	25	26	29
Async. #iter. glob. Async. #loc. sol. [min, max]	2065 [78,240]	1349 [102,237]	372 [128,475]	296 [157,517]	295 [147,514]	209 [175,407]

Table 6.10: Weak scalability: Number of iterations in the elasticity case.

Tables 6.9 and 6.10 gather the number of iterations for each case. In the asynchronous case, the number of Global solves is given as well as the minimum and maximum numbers of patches' solves.

The number of iterations barely varies in the synchronous experiments (in particular for the thermal problem) for all studied cases. For the asynchronous solver, it can be seen that in the 8 and 27 subdomains where the global problem is very light, many more solves are performed by the global model than by the local models. Because of the non-waiting asynchronous model, the global problem repeats several times the same calculation without having new input from the locals. However when the size of this problem increases for the case with 64 and 125 subdomains, the locals make more repetitive iterations while waiting for the update of the global problem, this last one performs only a few iterations.

Note the performance achieved in the elasticity case (2 times faster) despite the tremendous number of iterations (7 times more).

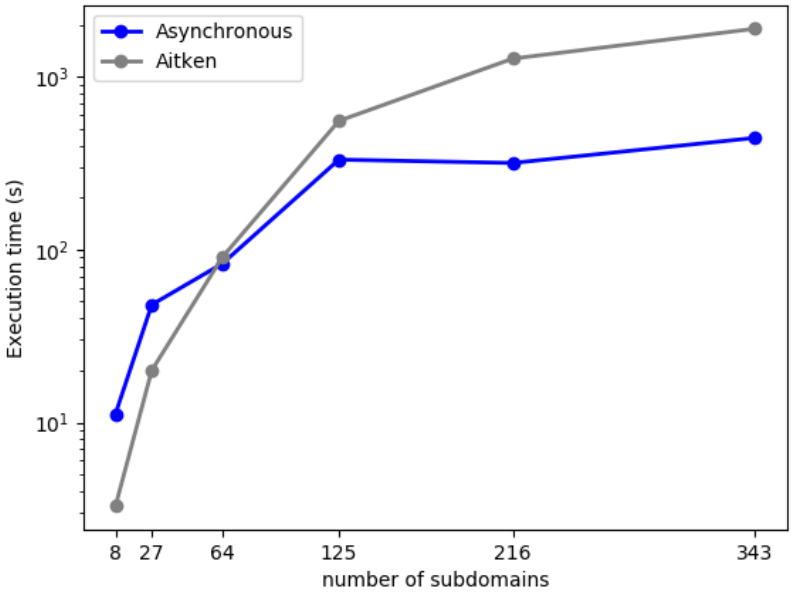


Figure 6.7: Time performance in the weak scalability study for linear elasticity problem

6.2.5 . Load imbalance study

The previous studies considered an almost perfect load balance. In this section, the case of load imbalance is studied, which is very interesting to show the effect of synchronization and the advantages that the asynchronous model can bring.

This study has been carried out on the 128-patch geometry (Figure 6.8) with dimensions given in Table 6.11. The Young's modulus in the spherical inclusion is 100 times lower than in the rest of the cube, with a global residual norm aimed to be lower than 10^{-7}

Problem	**Global**	**Local (One subdomain)**
Numbre of nodes	2769	1254

Table 6.11: Mesh data

The idea, in this case, is to assign a different computational load to each core, assuming having a limited number of cores at disposal; in this case, just 65 cores. The distribution of tasks being random, the following two cases are considered:

1. Access to all the cluster's machines, but only allocate a certain number of cores per machine. So the machines work less. However, exchanging information is larger because it involves many machines.

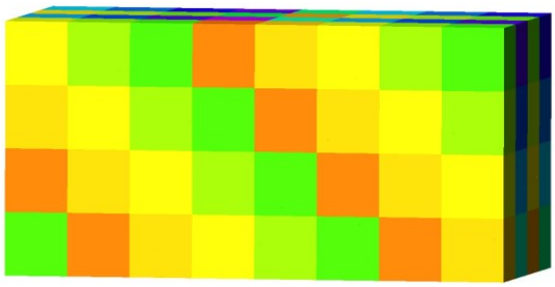


Figure 6.8: 128 subdomains with $\Omega^0 = \varnothing$

Model	**Aitken**	**Relaxed asynchronous**
Time(s)	150	130.07
Async. #iter. glob.	24	239
Async. #loc. sol. [min, max]	-	[70 - 338]

In the asynchronous case, despite the very large number of iterations realized in global and local it is still faster. The significant difference between the minimum and the maximum number of iterations due to the load imbalance can be seen. This situation, which in some way disadvantages the synchronous, allows to see that the asynchronous is more adapted to a load imbalance situation.

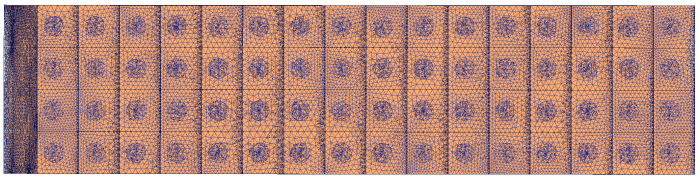
2. Consider a limited number of machines that allows it to reach 65 cores. Intensively used machines and a less solicited exchange network

Model	**Aitken**	**Relaxed asynchronous**
Time(s)	286.9	177.27s
Async. #iter. glob.	24	353
Async. #loc. sol. [min, max]	-	[69 - 417]

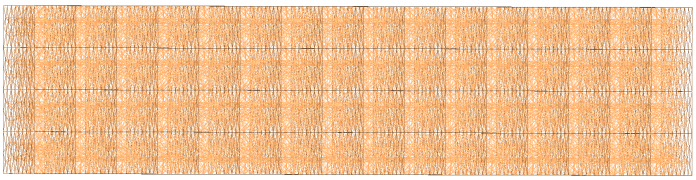
Globally the computation times are larger than in the previous situation, the strong solicitation of the machines penalizes the synchronous, and the asynchronous algorithm is significantly faster.

In a second study, the idea is to evaluate the influence of a significant disequilibrium in the number of nodes to be handled by processors. It starts from a geometry formed with a 16x4x4 repetition of cubes with spherical inclusion. Figure 6.9b corresponds to the global problem on the structure with (quasi-identical) homogeneous subdomains. Each Fine subdomain has a randomly chosen number of nodes compared to the other subdomains, allowing to have very refined subdomains and others slightly refined, see Figure 6.9a.

The table 6.12 summarizes the number of nodes for the global problem and the smallest and largest number of nodes among the 256 Fine subdomains. The most refined subdomain is ten times larger than the least refined:



(a) Reference problem with unbalanced patches



(b) Global representation

Figure 6.9: Load imbalance case with 256 patches

# of subdomains	**Global**	**Smallest local**	**Biggest local**
Number of nodes	5490	534	4698

Table 6.12: Mesh details

First, consider a linear thermal problem with a thermal diffusion coefficient in the spherical inclusion ten times lower than in the rest of the cube (moderate heterogeneity). As usual, convergence is achieved for a residual norm of less than 10^{-7}. The performance is given in Table 6.13.

Variant	**Sync.** Aitken	**Async.** ω_{opt}
Time (s)	881.55	79.44
#iter. glob.	36	506
#loc. sol. [min, max]	·	[348, 6788]

Table 6.13: Poor load balancing case: Iterations & Time (thermal problem)

Second, consider a linear elasticity problem, with a Young modulus in the spherical inclusion 10^4 times lower than in the rest of the cube (strong heterogeneity). As usual, convergence is achieved for a residual of less than 10^{-7}. The performance is given in Table 6.14.

This case study has been performed using 257 cores, one for the global problem and one processor for each one of the 256 local problems.

The number of iteration is very large in the asynchronous case, but the CPU time is much reduced: 10 times in the thermal case and 2 times for the elasticity case. Again, this highlights the prohibitive cost of synchronization.

Variant	**Sync.** Aitken	**Async.** ω_{opt}
Time (s)	3509.6	1904.34
#iter. glob.	113	2354
#loc. sol. [min, max]	·	[818, 2951]

Table 6.14: Poor load balancing case: Iterations & Time (linear elasticity problem)

6.3 . Nonlinear cases

In this section, the nonlinear cases covered by the theoretical study is considered, that is to say the global problem is linear and the local problems are monotonic. This case study allows introducing another type of load imbalance, associated with the unevenly distributed nonlinear intensity among the patches due to structure effects. Two types of nonlinear problems are considered:

- **Scalar nonlinear elliptic problem**, inspired by [86]:

$$\begin{aligned} &\text{Find } u : \Omega \subset \mathbb{R}^d \to \mathbb{R} \\ -\operatorname{div}(a \operatorname{grad}(u)) + u^3 &= f \text{ in } \Omega \\ u &= 0 \text{ on } \partial_d \Omega \\ \frac{\partial u}{\partial n} &= 0 \text{ on } \partial\Omega \setminus \partial_d \Omega \end{aligned} \tag{6.3}$$

  with $f = 1$ and u^3 the nonlinear term.

- **Associated elastoplasticity with linear kinematic & isotropic hardening**

$$\begin{aligned} &\text{Find } u : \Omega \subset \mathbb{R}^d \to \mathbb{R}^d \\ \operatorname{div}(\sigma) + f &= 0 \text{ in } \Omega \\ u &= 0 \text{ on } \partial_d \Omega \\ \sigma \cdot n &= 0 \text{ on } \partial\Omega \setminus \partial_d \Omega \\ \varepsilon(u) &= \varepsilon^e + \varepsilon^p \\ \sigma &= \frac{E}{1+\nu}\left(\varepsilon^e + \frac{\nu}{1-2\nu}\operatorname{tr}(\varepsilon^e) I\right) \\ \text{Yield function } f(\sigma, \alpha) &= \| \operatorname{Dev}(\sigma - \frac{2}{3} H_k \varepsilon^p) \| - \frac{2}{3}\sqrt{\sigma_{y0} + H_i \alpha} \leqslant 0 \end{aligned} \tag{6.4}$$

  where ε^e is the elastic part of the strain tensor and ε^p the plastic part. H_i and H_k are the positive scalar isotropic and kinematic hardening modulus. A classical θ-scheme is used for the (pseudo)-time integration. This model and solution technique are readily available in GetFem.

Only one load increment is considered in the following. In the respective works, the asynchronous for the parallelization in time will be considered based on [15] in synchronous, in which several cyclic loadings are considered.

6.3.1 . Preliminary study

The focus will be on cases where the load imbalance is due to the intensity of the nonlinearity. First study the 128-subdomain case of Table 6.11. the spherical inclusions are considered non-linear whereas the rest of the structure evolves linearly. Subdomains near the Dirichlet conditions are submitted to a stronger nonlinearity than others.

Variant	**Sync.** Aitken	**Async.** ω_{opt}
Time(s)	31.5	**15.84**
#iter. glob.	30	197
#loc. sol. [min, max]	·	[242-1315]

Table 6.15: Iterations & Time (Nonlinear scalar elliptic inclusion)

Variant	**Sync.** Aitken	**Async.** ω_{opt}
Time(s)	508	**155**
#iter. glob.	25	84
#loc. sol. [min, max]	·	[119 - 428]

Table 6.16: Iterations & Time (Nonlinear plastic inclusion)

Tables 6.15 and 6.16 show the results obtained in the case of scalar and elasto-plastic problems. Like in the linear case, the number of iterations is much more important with the asynchronous approach but the time is much lower (up to 3 times for plasticity).

the same studies are repeated this time with the case of 256 subdomains with:

Problem	**Global**	**Local (One subdomain)**
Number of nodes	5490	2308

Variant	**Sync.** Aitken	**Async.** ω_{opt}
Time(s)	304.33	**87.6**
#iter. glob.	30	240
#loc. sol. [min, max]	·	[412 - 7425]

Table 6.17: Iterations & Time (Nonlinear scalar elliptic inclusion)

Variant	**Sync.** Aitken	**Async.** ω_{opt}
Time(s)	703	**403**
#iter. glob.	30	95
#loc. sol. [min, max]	·	[181 - 594]

Table 6.18: Iterations & Time (Nonlinear plastic inclusion)

Tables 6.17 and 6.18 show the obtained results. Similar conclusions can been drawn. In a hard-to-predict manner, the gain is improved in the scalar case and reduced in the plastic one.

6.3.2 . Weak scalability

the same study as Subsection 6.2.4 is conducted but with nonlinear inclusions.

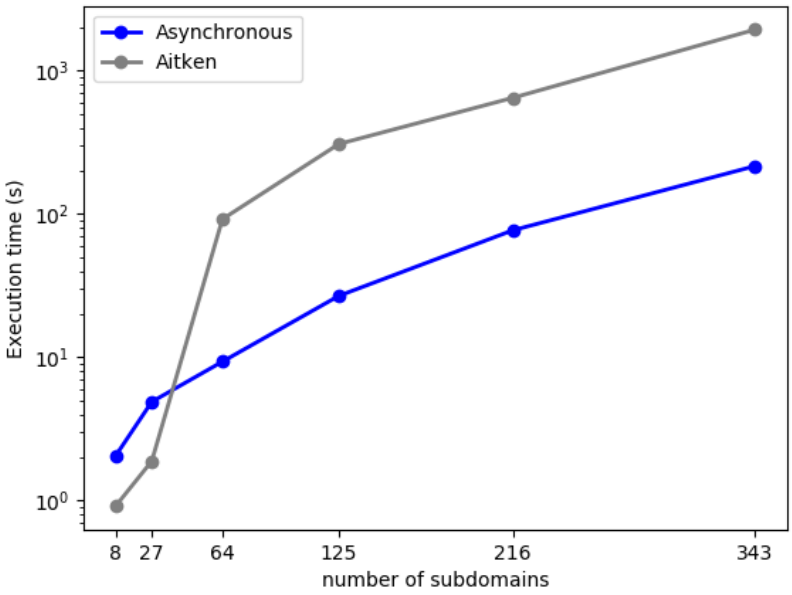


Figure 6.10: Weak scalability: time in the nonlinear thermal case.

#patches	**8**	**27**	**64**	**125**	**216**	**343**
Aitken #iter.	5	10	14	20	24	30
Async. #iter. glob.	720	41	65	151	207	229
Async. #loc. sol. [min, max]	[11 - 27]	[40 - 41]	[45 - 54]	[103 - 177]	[216 - 398]	[306 - 729]

Table 6.19: Weak scalability: Number of iterations in the nonlinear thermal case.

Figure 6.10 and Table 6.19 present the performance of the methods in the scalar case. As soon as enough patches are involved, the asynchronous version behaves much better than its synchronous counterpart, with a ten-fold reduction in time.

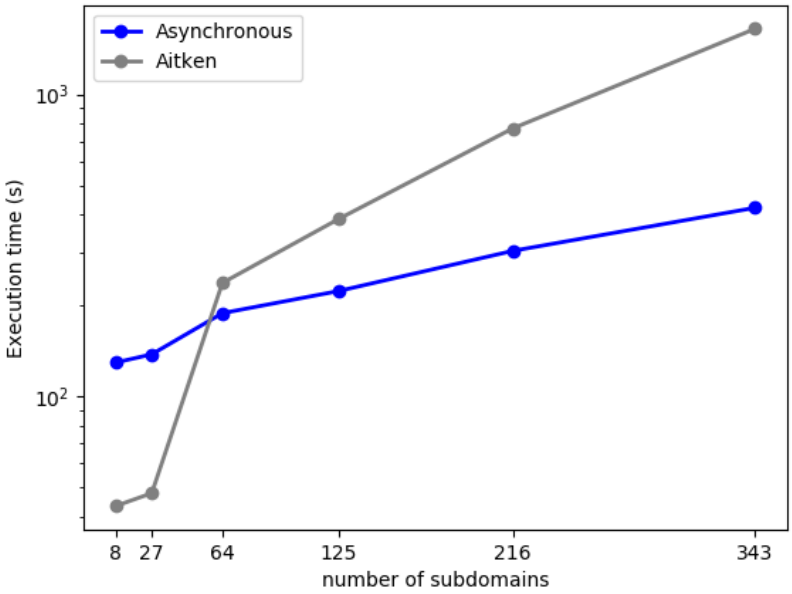


Figure 6.11: Weak scalability: time in the plasticity case.

#patches	**8**	**27**	**64**	**125**	**216**	**343**
Aitken #iter.	25	23	23	25	26	27
Async. #iter. glob.	1083	72	77	81	99	96
Async. #loc. sol. [min, max]	[68 - 191]	[84 - 183]	[106 - 241]	[126 - 490]	[152 - 410]	[178 - 518]

Table 6.20: Weak scalability: Number of iterations in the plasticity case.

Figure 6.11 and Table 6.20 present the performance of the methods in the elastoplastic case. Here, the time vs patches curve is much flatter in the asynchronous case, which corresponds to a much more scalable method.

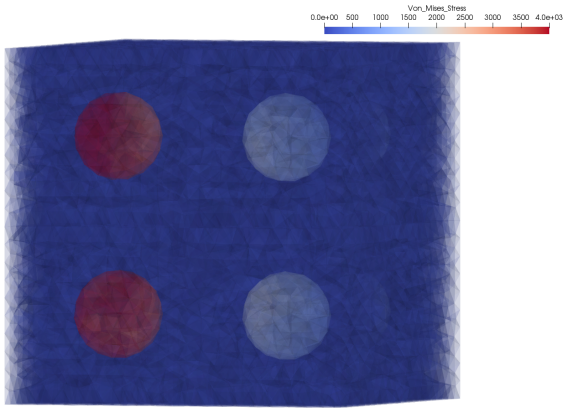


Figure 6.12: Von Mises stress in the case of 8 subdomains

Figure 6.12 shows the von Mises stress in the case with eight subdomains, in order to illustrate the fact that some spherical inclusions could reach their plastic limit = 3750, but others remained in an elastic regime.

6.4 . 3D industrial problem

In this part, a test case inspired by an industrial problem is considered. The geometry corresponds to the turbine blade of an aircraft engine. The Global model makes use of a simplified geometry which omits cooling micro-perforations. The two zones of interest are two critical regions of the domain where the precise geometry (with the perforations) is taken into account, see Figures 6.13 and 6.14.

The number of nodes of the meshes are given in Table 6.21, it can be see that one zone of interest is about two times larger than the other one which is roughly of the same size as the Global model.

Problem	**Global**	1^{st} **Zone of interest**	2^{nd} **Zone of interest**
Nodes	3.10^5	3.10^5	6.10^5

Table 6.21: Mesh data for the industrial test-case

The parallel analysis is conducted using three cores: one for the global problem and the other two for each zone of interest. Note that it appears that some difficulty in configuring the linear solver has appeared embedded in GetFem and the solution time of the mechanical systems was unexpectedly long compared to what can be observed with industrial software dealing with similar problems.

First, a linear thermal problem is considered with a constant source term.

Table 6.22 presents the computation times of the relaxed asynchronous, the synchronous without relaxation, and the Aitken model. The simplicity of the linear behavior allows a fast convergence with Aitken, and the asynchronous com-

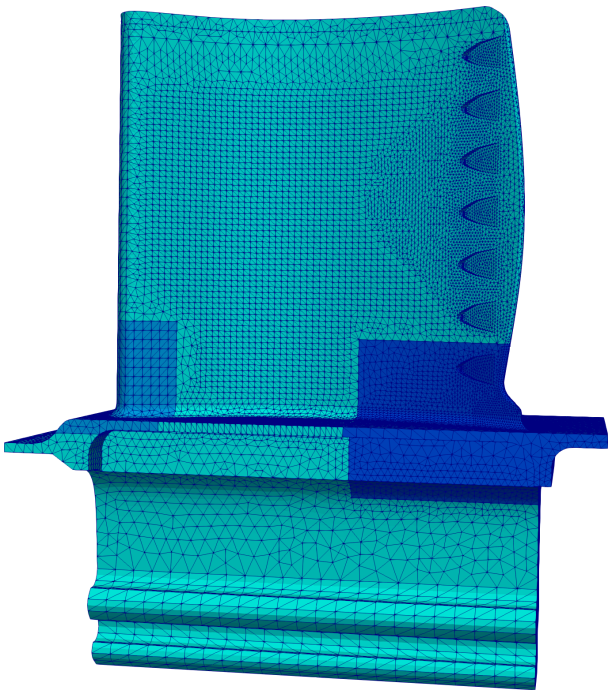


Figure 6.13: Global problem

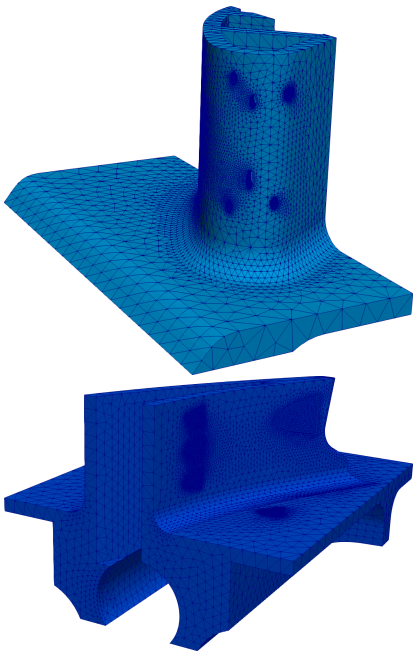


Figure 6.14: Zones of interest

Variant	**Sync.** $\omega = 1$	**Sync.** Aitken	**Async.** ω_{opt}
Time (s) [% communication time]	1447s[76.7%]	836.43s [30.46%]	1444.2s[2%]
#iter. glob.	21	12	90
#loc. sol. [min, max]	·	·	[91 - 32]

Table 6.22: 3D test-case: performance for thermal problem.

putation with optimized relaxation does barely better than the raw synchronous iteration. Comparing these two configurations, the communication time is negligible for the asynchronous approach (2%) whereas it is predominant in the synchronous case (76%). A better direct linear solver in the subdomains would have lead to better performance for the asynchronous iteration.

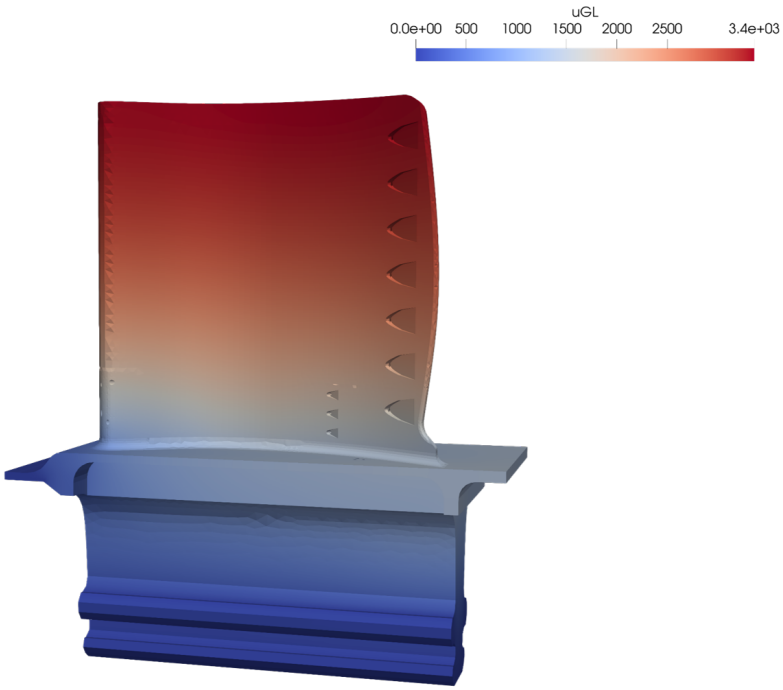


Figure 6.15: Temperature by the global local solution

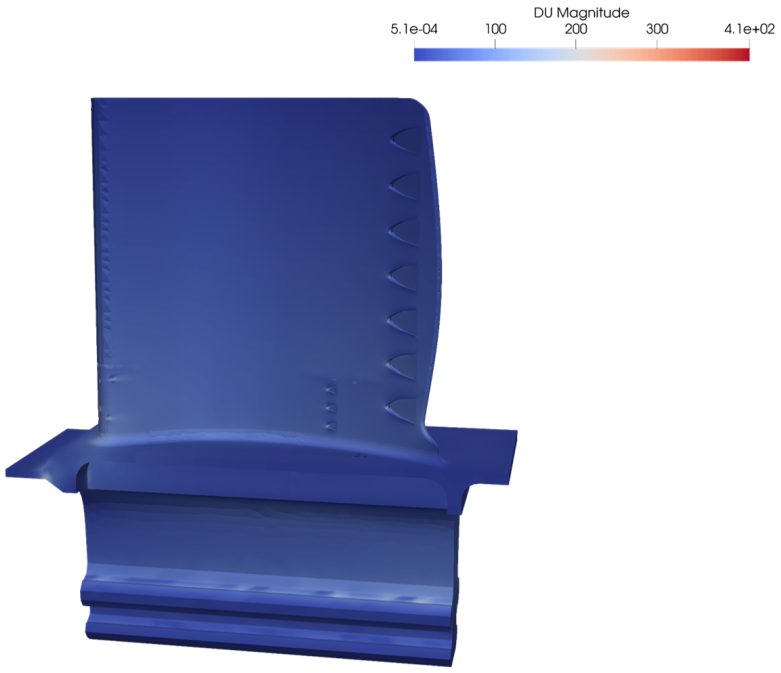


Figure 6.16: Gradient obtained by the global local solution

Figures 6.15 and 6.16 show the evolution of the temperature and its gradient

inside the turbine blade. As expected, it can be seen that the gradient is significant in the towers of the holes added in the zones of interest.

In a second time, the study of this problem is considered in the case where the global problem, as well as the local ones, are nonlinear. This study is a step ahead of our theory because the proof of convergence in the totally nonlinear case is not yet established. The performance is summed up in Table 6.23

Variant	**Sync.** Aitken	**Async.** ω_{opt}
Time (s)	3351	2055
#iter. glob.	13	19
#loc. sol. [min, max]	·	[6-16]

Table 6.23: 3D test-case: performance for the nonlinear case.

The asynchronous iteration is about 30% faster than the accelerated (Aitken) synchronous iteration. The largest subdomain needs fewer iterations in the asynchronous case (6) than in the synchronous case (13); in fact what mattered was having the Global and other subdomain sufficiently converged.

7 - Conclusion

In this work, a new method of asynchronous parallelization of the non-invasive global local coupling has been proposed. This new method has significantly improved the method's performance by limiting the degradation due to the alternating nature of the method's calculations. This work focused on three main steps:

- The formulation of the method in the form of a primal domain decomposition method: a first contribution of this work has been to review the state of the art on the global local coupling method and its history. The method has been studied from a theoretical point of view, starting from the basic variational formulation, to the finite element condensed system on the interface, which allowed the interpretation of the coupling as a primal domain decomposition method of the reference problem, right-preconditioned by the global problem. This preconditioner is less capable of HPC than the traditional BDD, but it embeds two-level mechanical information which permits more flexibility and, especially in this case, allows the use of fixed point iteration paving the way to asynchronous iterations.

- Theoretical study of the convergence in asynchronous: after the analysis of the method in its current form, it was possible to take a step forward by setting up an original theoretical study of convergence in the asynchronous case, based on paracontraction techniques. Two cases were distinguished according to the behavior of our local problems while considering the possibility of adjacent and non-adjacent patches, which corresponds to the presence or not of a complementary domain. The global problem was linear in all the studies. First, linear local patches were considered, and thanks to this property, the contraction could be rephrased in terms of the spectral radius of a matrix applied to the "history vector", i.e. vector with delayed components, which allowed for the deduction of the convergence for a sufficiently small relaxation. Second, the nonlinear case with monotonic local patches was considered, which corresponds to a large class of problems in solid mechanics. In that case, the linear global system provided a favorable Hilbert space where it was possible to directly bound the error between the exact solution and the iteration. This permitted the explanation of various phenomena like the sawtooth convergence with decreasing errors peaks for sufficiently small relaxation, and the monotone convergence for smaller (but not too small) relaxation.

- Implementations and numerical results: for the implementation part of the coupling, the non-intrusive aspect of the method allowed the use of a particular finite element library as a black box and to manage the communications

between the different sub-domains using an advanced technique in asynchronous parallel computing called RDMA. The asynchronous non-invasive global local coupling algorithm developed in this thesis proved to be more efficient, scalable, and robust than the synchronous one coupled with the Aitken accelerator through a series of numerical experiments and comparisons. Important insights into the trade-offs between communication overhead, load balancing, and accuracy in asynchronous domain decomposition methods were provided.

The application of asynchronous iterations can be extended to more complex computational problems, such as plasticity and nonlinear elliptic problems, demonstrating the potential of asynchronous methods to tackle a wide range of scientific and engineering problems. Overall, the findings of this work have the potential to impact the design and development of parallel algorithms for a wide range of applications. The results provide a solid foundation for future work and pave the way for developing even more advanced and efficient parallel algorithms. There are several promising directions for future research in this area, including:

- Finding an efficient protocol to estimate the optimal relaxation parameter, and even better adapting the relaxation to the data available at a given iteration. More generally trying to derive algorithmic asynchronous accelerators.
- Using hardware accelerators, such as graphics processing units (GPUs), to explore hybrid techniques that combine the benefits of multiple accelerator technologies.
- Setting up a theoretical study to extend the proofs obtained in this work to the case where the global problem is also nonlinear (monotone), since convergence has been observed in practice.
- Implementing time asynchronism to deal with complex history of loads, with a particular interest in the determination of stabilized elastoviscoplastic cycles [15].

A - Résumé

A.1 . Le couplage global/local

Le couplage global/local est une méthode puissante permettant d'améliorer la précision des résultats mécaniques en prenant en compte les effets locaux dans un modèle global. Cette méthode combine les avantages d'un modèle grossier (efficacité de calcul, représentation générale du système) avec les détails fournis par des modèles plus fins.

L'approche du couplage global/local peut être mise en œuvre en utilisant des méthodes de traitement et des procédures d'entrée/sortie standard dans les codes éléments finis. Cela facilite son intégration dans les flux de travail existants, sans nécessiter de modifications majeures du code ou d'interfaces personnalisées. Des codes commerciaux tels que Abaqus ou Code_Aster prennent en charge cette méthode, ce qui permet aux ingénieurs de l'industrie de l'adopter facilement.

Sur le plan conceptuel, le couplage global/local peut être considéré comme une méthode de décomposition de domaine. Cela signifie que le domaine global est divisé en sous-domaines, où des modèles plus fins sont utilisés pour résoudre les problèmes locaux. Ces sous-domaines sont ensuite couplés au modèle global en appliquant des conditions aux limites appropriées. L'interaction entre les modèles globaux et locaux permet de tenir compte des effets locaux sur le comportement global du système.

Malgré sa robustesse et sa non-intrusivité, la parallélisation du couplage global/local reste un problème significatif. Comme l'explique la littérature, plusieurs études de cas impliquent l'utilisation de patchs et la parallélisation de ces patchs à l'aide de MPI. Les performances du couplage restent limitées, même si la parallélisation de ces patchs locaux peut permettre un gain de performance. La synchronisation à la fin de chaque calcul local oblige tous les patchs à attendre la fin du dernier calcul des patchs les plus lourds en termes de temps de calcul, ce qui pose le problème central de l'équilibrage de charge. Un deuxième problème que cette méthode rencontre, comme toutes les méthodes en deux étapes, est que l'évaluation du problème global est effectuée séquentiellement avec les problèmes locaux. Cela signifie que même si les patchs sont bien équilibrés et parallélisés, il y aura une période d'inactivité pendant que le problème global effectue ses calculs, ce qui rend la méthode non évolutive en utilisant des techniques de parallélisation synchrones classiques.

Pour résoudre ces problèmes, des approches asynchrones ont été proposées. Les itérations asynchrones permettent aux processeurs de travailler de manière indépendante dès qu'ils ont des données à traiter, sans attendre les autres. Cela réduit le temps d'attente inutile et améliore l'utilisation des ressources de calcul. De plus, les itérations asynchrones offrent une meilleure tolérance aux retards de

réseau, aux déséquilibres de charge et aux architectures fortement hétérogènes.

A.2 . Le couplage global/local asynchrone

A.2.1 . Algorithme asynchrone

L'idée fondamentale de l'algorithme asynchrone est de tirer parti des moments d'inactivité des processeurs pour effectuer des calculs. Dès qu'une nouvelle donnée est disponible, le système détecte un processeur libre et lance immédiatement le calcul correspondant. Cette approche asynchrone permet d'optimiser l'utilisation des ressources de calcul en évitant les temps d'attente inutiles.

La séquence temporelle asynchrone, illustrée dans la Figure 4.1, met en évidence le fait que l'élimination de la synchronisation accroît l'intensité du calcul. En évitant une coordination stricte entre les processeurs, chaque processeur peut travailler de manière indépendante dès qu'il dispose de données à traiter. Cela accélère le processus global de calcul.

L'algorithme asynchrone est détaillé dans l'Algorithme **??**. Il est important de noter que, dans chaque itération, le modèle global effectue toujours une opération d'assemblage pour construire un résidu. Cette caractéristique est avantageuse, car elle facilite la détection de la convergence. La présence d'un résidu permet de mesurer la différence entre les résultats actuels et les résultats attendus, et de déterminer si le processus de calcul a atteint un état de convergence satisfaisant. Dans d'autres méthodes, cette étape peut poser des problèmes, mais dans notre approche, elle est simplifiée grâce à la construction régulière du résidu.

En conclusion, en adoptant une approche asynchrone et en exploitant les temps d'inactivité des processeurs, notre méthode permet d'optimiser l'utilisation des ressources de calcul et d'accélérer le processus global de calcul. La détection de la convergence est facilitée grâce à l'assemblage régulier d'un résidu, ce qui améliore l'efficacité et la fiabilité de notre méthode par rapport à d'autres approches existantes [74, 79].

Une preuve théorique de convergence de l'algorithme asynchrone global/local peut être établie en utilisant le cadre des paracontractions [31]. Le résultat principal indique que pour une certaine valeur donnée du paramètre de relaxation, l'itération asynchrone converge avec des pacthes locaux lineaire et non lineaires et un probleme global lineaire.

Cependant, un inconvénient de l'algorithme asynchrone est qu'actuellement, aucune stratégie d'accélération n'est disponible. Néanmoins, il convient de souligner que l'algorithme asynchrone présente des avantages importants en termes d'utilisation efficace des ressources de calcul disponibles. Il permet d'exploiter les temps d'inactivité des processeurs et d'effectuer des calculs dès que de nouvelles données sont disponibles.

A.2.2 . Implémentation

La mise en œuvre d'un protocole de communication asynchrone a fait l'objet de plusieurs travaux de recherche, généralement basés sur MPI, comme décrit dans l'article [75], où l'idée est d'utiliser une communication classique bidirectionnelle. Cependant, de nouvelles recherches basées sur une communication unidirectionnelle [112, 48], également connue sous le nom de MPI-RDMA, ont prouvé l'efficacité de ces techniques et leur adaptation à la communication asynchrone.

Les performances de ces techniques dépendent de la version de MPI utilisée et du réseau. Nous avons réalisé des tests sur plusieurs configurations : OPENMPI, INTELMPI, MPICH, et différentes architectures réseau telles que l'Ethernet classique, l'Infiniband plus avancé ou Intel OPA. L'influence de ces choix a été observée dans les communications asynchrones, parfois avec une synchronisation implicite imposée par le réseau ou des opérations de communication moins performantes selon la version de MPI.

L'idée générale de RDMA (Remote Direct Memory Access) est de permettre l'accès aux données sur d'autres machines sans impliquer la machine cible. Nous créons une partie de la mémoire appelée fenêtre dans laquelle nous plaçons les données recherchées. Les autres machines peuvent effectuer des opérations de type PUT ou GET pour mettre à jour ces informations dans la fenêtre ou les récupérer et les utiliser par la suite. Cette idée est donc bien adaptée au calcul asynchrone car nous sommes dans une procédure où nous n'avons pas besoin d'interrompre le calcul pour effectuer des opérations d'envoi ou de réception.

Les figures 5.2 et 5.3 correspondent à une communication entre un processeur 0 et un processeur 1, où ce dernier effectue deux opérations de communication PUT et GET sur le processeur 0 pour recevoir la valeur X et envoyer la valeur Y. Ces deux opérations, comme mentionné précédemment, sont effectuées sans impliquer le processeur 0.

Ces communications sont généralement suivies d'une étape de synchronisation. On distingue deux types de synchronisation dans RDMA : la synchronisation active, où nous effectuons une opération collective pour mettre à jour tout le monde avant de passer d'une itération à une autre avec la commande **MPI_WIN_Fence()**, et la synchronisation passive, utilisée dans le mode asynchrone. Cette technique consiste à synchroniser chaque processeur sans nécessairement effectuer une synchronisation globale. Chaque processeur ouvre une époque avec **MPI_Win_lock** et effectue ces opérations PUT et GET dans cette époque avant de la fermer avec **MPI_Win_Unlock**. Les opérations d'achèvement d'envoi **MPI_Win_Flush** suivent ces opérations à l'intérieur de cette époque pour garantir que l'envoi est terminé.

A.3 . Résultats numériques

Dans le cadre de nos résultats numériques, nous avons utilisé notre code implémenté en Python, qui est soutenu par divers outils et logiciels. Nous avons utilisé GMSH pour générer les géométries et les maillages des cas étudiés, tandis

que la bibliothèque Getfem a été utilisée pour l'approximation par éléments finis. Pour la partie parallèle, nous avons fait appel à la bibliothèque mpi4py.

Les expériences ont été réalisées sur le cluster du centre de simulation LMPS, en utilisant plusieurs postes de travail connectés par un réseau Ethernet. Il convient de noter que ces machines présentent une certaine hétérogénéité, avec différentes générations de processeurs Intel, notamment Intel(R) Xeon(R) CPU E5-1660 v3 (Haswell) @ 3.00GHz, Intel(R) Xeon(R) CPU E5-2630 v4 (Broadwell) @ 2.20GHz, Intel(R) Xeon(R) Silver 4116 CPU (Skylake) @ 2.10GHz, et Intel(R) Xeon(R) W-2255 CPU (Cascade Lake) @ 3.70GHz.

Nous avons effectué des comparaisons entre les versions synchrone et asynchrone, en tenant compte de la présence ou de l'absence de relaxation. Dans le cas synchrone, nous avons utilisé l'accélération d'Aitken pour adapter dynamiquement la relaxation. En revanche, pour le cas asynchrone, nous avons procédé à des essais et erreurs afin de trouver le meilleur coefficient de relaxation pour obtenir les résultats optimaux.

Nos résultats numériques visaient à confirmer les résultats théoriques présentés précédemment et à évaluer les performances de l'algorithme asynchrone par rapport à la version synchrone. Pour ce faire, nous avons étudié une variété de problèmes, qu'ils soient linéaires ou non linéaires. Nous avons commencé par des cas académiques relativement simples en 2D, puis nous avons progressivement augmenté la complexité en passant à des cas académiques et industriels plus avancés en 3D.

L'objectif était de comparer les performances des algorithmes asynchrone et synchrone dans différentes situations de couplage global/local, en prenant en compte le nombre de patchs impliqués. Nous avons également examiné la scalabilité faible de notre approche, ainsi que les problèmes de déséquilibre de charge auxquels nous sommes régulièrement confrontés dans ce type de simulations.

Ces résultats numériques nous permettent de valider nos conclusions théoriques, d'évaluer l'efficacité de l'algorithme asynchrone par rapport à la version synchrone et de mieux comprendre les performances de notre approche dans différentes configurations de problèmes.

A.4 . Conclusion & Perspectives

Ce travail a introduit une nouvelle méthode asynchrone de couplage global-local non invasif, qui a considérablement amélioré les performances par rapport à la méthode synchrone. Les principales étapes de cette méthode comprennent la formulation du couplage sous la forme d'une décomposition de domaine primale, l'étude théorique de la convergence en mode asynchrone, ainsi que la mise en œuvre et les résultats numériques qui ont confirmé l'efficacité et la robustesse de l'algorithme asynchrone par rapport à sa version synchrone.

Les itérations asynchrones ont démontré leur capacité à être étendues à des

problèmes plus complexes tels que la plasticité et les problèmes elliptiques non linéaires, ce qui témoigne de leur potentiel pour résoudre une large gamme de problèmes scientifiques et d'ingénierie. Les résultats de cette étude ont des implications significatives pour la conception d'algorithmes parallèles dans divers domaines d'application.

Plusieurs pistes prometteuses pour la recherche future ont été identifiées. Tout d'abord, il est essentiel de trouver des méthodes efficaces pour estimer le paramètre de relaxation optimal, voire développer des accélérateurs algorithmiques asynchrones. De plus, l'utilisation d'accélérateurs matériels tels que les GPU ou l'exploration de techniques hybrides combinant différentes technologies d'accélération ouvrent de nouvelles perspectives. Une autre direction de recherche importante consisterait à étendre les preuves théoriques obtenues dans ce travail aux problèmes globaux non linéaires, qui sont souvent rencontrés dans de nombreux domaines de la mécanique des solides. Enfin, l'application de l'asynchronisme temporel pour gérer les charges complexes, en particulier pour la détermination des cycles élastoviscoplastiques stabilisés, représente un domaine de recherche intéressant à explorer.

Dans l'ensemble, cette étude a fourni une base solide pour les travaux futurs dans le domaine du couplage global-local asynchrone. Les résultats obtenus ont démontré l'efficacité et la pertinence de l'algorithme asynchrone par rapport à la méthode synchrone, tout en ouvrant la voie au développement d'algorithmes parallèles plus avancés et efficaces pour résoudre une grande variété de problèmes scientifiques et d'ingénierie.

B - Appendices

B.1 . Windows free and allocation

Code Listing B.1: Creation of the Global displacement window

```
#The Global Displacement vector
uG = np.zeros(Gnbd + 1, dtype=float)
#The rank 0 does not have to allowed any memory
if rank == 0:
win_uG = MPI.Win.Create(None, comm=MPI.COMM_WORLD)
# The other processor allowing a memory equal to the size of the
                                Global vector displacement at the
                                 interface
if rank != 0:
win_uG = MPI.Win.Create(uG, comm=MPI.COMM_WORLD)
```

Code Listing B.2: Creation of the local nodal reaction windows

```
# Buffers list
rG_win_s = np.zeros((nprocs - 1, Gnbd + nprocs - 1))
# Windows list
win_rG_s = {}
#The rank 0 creates a number of windows equal to the number of
                                patches
if rank == 0:
for s in range(0, nprocs - 1):
win_rG_s[s] = MPI.Win.Create(rG_win_s[s, :], comm=MPI.COMM_WORLD)
# The other does not have to allow any memory
if rank != 0:
for s in range(0, nprocs - 1):
win_rG_s[s] = MPI.Win.Create(None, comm=MPI.COMM_WORLD)
```

Code Listing B.3: Creation of the windows for the convergence detection boolean

```
# Boolean variable
converged = np.arange(1, dtype='i')
# Initialization
converged[0] = 0
# The processor of rank 0 does not have to allow any window
if rank == 0:
win_conv = MPI.Win.Create(None, comm=MPI.COMM_WORLD)
# Allowing a window of the size of the boolean on each other
                                processor
if rank != 0:
win_conv = MPI.Win.Create(converged, comm=MPI.COMM_WORLD)
```

Code Listing B.4: Free the allocated windows

```
win_uG.Free()
win_conv.Free()
for s in range(0, nprocs - 1):
win_rG_s[s].Free()
```

B.2 . Synchronous code

Code Listing B.5: Implementation of the synchronous global/local algorithm

```
# While the boolean variable equal to 0 we do
while ((converged[0] == 0)):

# Fence collective call for the synchronization (opening an epoch
                                access for the rank 0 on the
                               other rank windows)
win_uG.Fence()
# Rank 0 in charge of the global computation
if rank == 0:
# Call of the function global resolution to resolve with a chosen
                                solver (Aitken, or classic) the
                               Global Problem, by pG imposing as
                                an immersed Neuman condition
# pG is updated with the call of this function using the residual
                                rG_j
uG, rG_j, rG_j_1, omega_new = \
Resolution_gl.Global_resolution(solveur, rG_j_1, rG_j, ci,
                               omega_old, omega_new, pG,
                               pG_index, rG, Gmd)
# Rank 0 excute a Put operation, to send the Global displacement
                               uG to all of the other target
                               Rank
for i in range(1, nprocs):
win_uG.Put([uG, MPI.DOUBLE], i)
loc_iter += 1
# Fence collective call for the synchronization (closing the
                               epoch access)
win_uG.Fence()

# Fence collective call for the synchronization (opening an epoch
                                access for all the processor (
                               origin) to processor of rank 0 (
                               target) window)
for s in range(0, nprocs - 1):
win_rG_s[s].Fence()
# Each processor except Rank 0 is in charge of a local problem
                               computation.
if rank != 0:
# Initialization of local residual computed for each local
                               problem
```

```
rG_loc[0:Gnbd] = 0
# Call of the function local resolution to resolve the local
                                Problem, by imposing the
                                Dirichlet condition uG
for i in range(0, size):
(uF[i], LFonG[Gbound[i]]) = \
Resolution_gl.Local_resolution(Fmd[i], Interpo[i], uG[0:Gnbd],
                                Gbound[i], new_region_100_fin[i],
                                 uFd[i],C[i], F_index[i], FM[i])
rG_loc[Gbound[i]] += LFonG[Gbound[i]]
loc_iter += 1
# Orgin processors excute a Put operation, to send the nodal
                                reaction stored on the rG_loc to
                                the target processor of rank 0

win_rG_s[rank - 1].Put([rG_loc, MPI.DOUBLE], 0)
# Fence collective call for the synchronization (closing the
                                epoch access)
for s in range(0, nprocs - 1):
win_rG_s[s].Fence()

# Fence collective call for the synchronization (opening an epoch
                                 access for the rank 0 on the
                                other rank windows) to stop the
                                algorithm after convergence
win_conv.Fence()
if rank == 0:
# The processor of rank 0 compute the global residual norm  using
                                 the receiving nodal reaction
                                rG_win_s from the local problems.
norm_rG, rG = Resolution_gl.Residu_computation(Gnbd, nprocs,
                                rG_win_s)
print("it", ci, "norm r", norm_rG, "rank:", " ", rank)
ci += 1
# Check if the algorithm converges
if norm_rG < 1.e-6:
# If convergence, updating the boolean value converged and set it
                                 to 1
converged[0] = 1
# Processor of rank 0 sends the value of this boolean to the all
                                the other target processor
for i in range(1, nprocs):
win_conv.Put([converged, MPI.INT], i)
# Fence collective call for the synchronization (closing the
                                epoch access)
win_conv.Fence()
```

B.3 . Asynchronous code

Code Listing B.6: Implementation of asynchronous global/local coupling with waiting

```
while converged[0] == 0:
# Rank 0 solves the global problem with a classic solver and a
                                chosen relaxation coefficient
                                omega_new
if rank == 0:
uG[0:Gnbd], rG_j, rG_j_1, omega_new = \
Resolution_gl.Global_resolution(solveur, rG_j_1, rG_j, ci,
                                omega_old,omega_new, pG, pG_index
                                , rG, Gmd)
# Updating the number of iteration of the global problem
loc_iter += 1
uG[Gnbd] = loc_iter
# The rank 0 open access epoch on each of the other processors
                                using lock using passive
                                synchronization then put data and
                                 ensure its completion on the
                                target processor then close the
                                access on this target processor.
for i in range(1, nprocs):
win_uG.Lock(i)
win_uG.Put([uG, MPI.DOUBLE], i)
win_uG.Flush(i)
win_uG.Unlock(i)

# All processors except 0 open an access epoch on a passive
                                synchronization way, checking if
                                new Global displacement is
                                available during this epoch.
                                After reviewing this status, the
                                epoch is closed.
if rank != 0:
while loc_G == loc_G_prec and converged[0] == 0:
win_uG.Lock(rank)
loc_G = np.copy(uG[Gnbd])
win_uG.Unlock(rank)
loc_G_prec = np.copy(loc_G)
rG_loc[0:Gnbd] = 0
# All processors asynchronously solve the local problems, using
                                different Global displacements
                                from various iterations
for i in range(0, size):
(uF[i], LFonG[Gbound[i]]) = \
Resolution_gl.Local_resolution(Fmd[i], Interpo[i], uG[0:Gnbd],
                                Gbound[i], new_region_100_fin[i],
                                 uFd[i], C[i], F_index[i], FM[i])
rG_loc[Gbound[i]] += LFonG[Gbound[i]]
```

```
  loc_iter += 1
  rG_loc[Gnbd + rank - 1] = loc_iter
  # All processors open access passively on the rank 0,
                                  asynchronously Put nodal reaction
                                   on these windows, ensure its
                                  completion and close the opened
                                  epoch.
  win_rG_s[rank - 1].Lock(0)
  win_rG_s[rank - 1].Put([rG_loc, MPI.DOUBLE], 0)
  win_rG_s[rank - 1].Flush(0)
  win_rG_s[rank - 1].Unlock(0)

  # Processor 0 opens an access epoch on a passive synchronization
                                  way, checking if new nodal
                                  reactions is available from other
                                   processors during this epoch.
                                  After reviewing this status, the
                                  epoch is closed.
  if rank == 0:
  while np.array_equal(Glob_c, Glob_c_prec) and converged[0] == 0:
  for i in range(0, nprocs - 1):
  win_rG_s[i].Lock(0)
  Glob_c[i] = rG_win_s[i, Gnbd + i]
  win_rG_s[i].Unlock(0)
  Glob_c_prec = Glob_c.copy()
  # The residual is evaluated and its norm is computed
  norm_rG, rG = Resolution_gl.Residu_computation(Gnbd, nprocs,
                                  rG_win_s)
  print("it", ci, "norm r", norm_rG, "rank:", " ", rank)
  ci += 1
  # If the algorithm converged, open an access epoch on a passive
                                  synchronization way, send the
                                  stopping criterion to all the
                                  other processors to ensure its
                                  completion, and then close these
                                  epochs.
  if norm_rG < 1.e-6:
  converged[0] = 1
  for i in range(1, nprocs):
  win_conv.Lock(i)
  win_conv.Put([converged, MPI.INT], i)
  win_conv.Flush(i)
  win_conv.Unlock(i)
```

Code Listing B.7: Implementation of asynchronous global/local coupling without waiting

```
%\begin{mypython}[caption={Implementation of asynchronous Global/
                                  Local coupling without waiting},
                                  label=without]
  # Rank 0 opens access using passive synchronization to all the
```

```
                                target processor windows for
                                global displacement and boolean
                                variable for stopping convergence
                                .
if rank == 0:
win_uG.Lock_all()
win_conv.Lock_all()
# All processors except ranks 0 open access on the target rank 0
                                using passive synchronization,
                                for the nodal reaction sends
if rank != 0:
win_rG_s[rank - 1].Lock(0)
while converged[0] == 0:
# Rank 0 solves the global problem with a relaxation coefficient
                                omega_new
if rank == 0:
uG[0:Gnbd], rG_j, rG_j_1, omega_new = \
Resolution_gl.Global_resolution(solveur, rG_j_1, rG_j, ci,
                                omega_old, omega_new, pG,
                                pG_index, rG, Gmd)
# Update the number of iteration of the global problem
loc_iter += 1
uG[Gnbd] = loc_iter
# Put the global displacement to all the target processors and
                                ensure the completion of the
                                operation to all these processors
                                 using Flush_all()
for i in range(1, nprocs):
win_uG.Put([uG, MPI.DOUBLE], i)
win_uG.Flush_all()

# All processors asynchronously solve the local problems, using
                                different Global displacements
                                from various iterations.
if rank != 0:
rG_loc[0:Gnbd] = 0
for i in range(0, size):
(uF[i], LFonG[Gbound[i]]) = \
Resolution_gl.Local_resolution(Fmd[i], Interpo[i], uG[0:Gnbd],
                                  Gbound[i],new_region_100_fin[i]
                                , uFd[i], C[i],F_index[i], FM[i])
rG_loc[Gbound[i]] += LFonG[Gbound[i]]
loc_iter += 1
rG_loc[Gnbd + rank - 1] = loc_iter
# All processors asynchronously Put nodal reaction on the rank 0
                                windows and ensure its completion
                                 on the target processor 0
win_rG_s[rank - 1].Put([rG_loc, MPI.DOUBLE], 0)
win_rG_s[rank - 1].Flush(0)

# Rank 0 Compute the residual and send a boolean to the other
```

```
                                    processors and ensure  its
                                    completion to stop the algorithm
                                    if it is converged.
if rank == 0:
norm_rG, rG = Resolution_gl.Residu_computation(Gnbd, nprocs,
                                    rG_win_s)
print("it", ci, "norm r", norm_rG, "rank:", " ", rank)
ci += 1
if norm_rG < 1.e-6 and ci > 10:
converged[0] = 1
for i in range(1, nprocs):
win_conv.Put([converged, MPI.INT], i)
win_conv.Flush(0)
# Rank 0 closes all the opening epoch on the other processors
if rank == 0:
win_uG.Unlock_all()
win_conv.Unlock_all()
# All processors close the opening access epoch on the rank 0
                                    windows
if rank != 0:
win_rG_s[rank - 1].Unlock(0)
```

Bibliography

[1] Olivier Allix and Pierre Gosselet. Non intrusive global/local coupling techniques in solid mechanics: An introduction to different coupling strategies and acceleration techniques. In L. De Lorenzis and A. Düster, editors, *Modeling in Engineering Using Innovative Numerical Methods for Solids and Fluids*, volume 599 of *CISM International Centre for Mechanical Sciences – Courses and Lectures*, pages 203–220. Springer Nature Switzerland AG, 2020.

[2] Lori Badea. On the Schwarz alternating method with more than two subdomains for nonlinear monotone problems. *SIAM Journal on Numerical Analysis*, 28(1):179–204, 1991.

[3] Jacques M. Bahi, Sylvain Contassot-Vivier, and Raphaël Couturier. *Parallel iterative algorithms: from sequential to grid computing*. Chapman and Hall/CRC, 2007.

[4] Jacques M. Bahi, Sylvain Contassot-Vivier, and Raphaël Couturier. An efficient and robust decentralized algorithm for detecting the global convergence in asynchronous iterative algorithms. In *International Conference on High Performance Computing for Computational Science*, pages 240–254. Springer, 2008.

[5] Jacques M. Bahi, Sylvain Contassot-Vivier, and Raphaël Couturier. *Parallel iterative algorithms from sequentiel to grid computing. Numerical analysis and scientific computing*. Chapman & Hall/CRC, 2008.

[6] Jacques M. Bahi, Sylvain Contassot-Vivier, Raphaël Couturier, and Flavien Vernier. A decentralized convergence detection algorithm for asynchronous parallel iterative algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 16(1):4–13, 2005.

[7] Jacques M. Bahi, Jean-Claude Miellou, and Karim Rhofir. Asynchronous multisplitting methods for nonlinear fixed point problems. *Numerical Algorithms*, 15:315–345, 1997.

[8] Gerard M. Baudet. Asynchronous iterative methods for multiprocessors. *Journal of the Association for Computing Machinery*, 25(2), 1978.

[9] Abraham Berman and Robert J. Plemmons. *Nonnegative matrices in the mathematical sciences*. SIAM, 2 edition, 1994.

[10] Dimitri P. Bertsekas and John N. Tsitsiklis. Convergence rate and termination of asynchronous iterative algorithms. In *Proceedings of the 3rd International Conference on Supercomputing*, pages 461–470, 1989.

[11] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena scientific, Belmont (MA), 1989.

[12] Omar Bettinotti, Olivier Allix, and Benoît Malherbe. A coupling strategy for adaptive local refinement in space and time with a fixed global model in explicit dynamics. *Computational Mechanics*, pages 1–14, 2013.

[13] Omar Bettinotti, Olivier Allix, Umberto Perego, Victor Oncea, and Benoît Malherbe. A fast weakly intrusive multiscale method in explicit dynamics. *International Journal for Numerical Methods in Engineering*, 100(8):577–595, 2014.

[14] Omar Bettinotti, Olivier Allix, Umberto Perego, Victor Oncea, and Benoît Malherbe. Simulation of delamination under impact using a global local method in explicit dynamics. *Finite Elements in Analysis and Design*, 125:1–13, 2017.

[15] Maxime Blanchard, Olivier Allix, Pierre Gosselet, and Geoffrey Desmeure. Space/time global/local noninvasive coupling strategy: Application to viscoplastic structures. *Finite Elements in Analysis and Design*, 156:1–12, 2019.

[16] Robin Bouclier, Jean-Charles Passieux, and Michel Salaün. Local enrichment of NURBS patches using a non-intrusive coupling strategy: Geometric details, local refinement, inclusion, fracture. *Computer Methods in Applied Mechanics and Engineering*, 300:1–26, 2016.

[17] Christophe Bovet, Augustin Parret-Fréaud, and Pierre Gosselet. Two-level adaptation for adaptive multipreconditioned FETI. *Advances in Engineering Software*, 152, 2021.

[18] Ming Chau, Thierry Garcia, and Pierre Spitéri. Asynchronous schwarz methods applied to constrained mechanical structures in grid environment. *Advances in Engineering Software*, 74:1–15, 2014.

[19] Dan Chazan and Willard L. Miranker. Chaotic relaxation. *Linear Algebra and Its Application*, 2:199–222, 1969.

[20] Mathilde Chevreuil, Anthony Nouy, and Elias Safatly. A multiscale method with patch for the solution of stochastic partial differential equations with localized uncertainties. *Computer Methods in Applied Mechanics and Engineering*, 255(0):255 – 274, 2013.

[21] Edmond Chow, Andreas Frommer, and Daniel B. Szyld. Asynchronous Richardson iterations: theory and practice. *Numerical Algorithms*, 87(4):1635–1651, 2021.

[22] Nathan Cormier, Brian S. Smallwood, Gleen B. Sinclair, and G. Meda. Aggressive submodelling of stress concentrations. *International Journal for Numerical Methods in Engineering*, 46(6):889–909, 1999.

[23] Philippe Cresta, Olivier Allix, Christian Rey, and Stéphane Guinard. Nonlinear localization strategies for domain decomposition methods: application to post-buckling analyses. *Computer Methods in Applied Mechanics and Engineering*, 196(8):1436–1446, 2007.

[24] Philippe Cresta, Olivier Allix, Christian Rey, and Stéphane Guinard. Nonlinear localization strategies for domain decomposition methods: Application to post-buckling analyses. *Computer Methods in Applied Mechanics and Engineering*, 196(8):1436–1446, 2007.

[25] Lisandro Dalcin and Yao-Lung L. Fang. mpi4py: Status update after 12 years of development,. *Computing in Science & Engineering*, 23(4):47–54, 2021.

[26] D. Tromeur Dervout. Acceleration of the convergence of the asynchronous ras method. In Susanne C. Brenner, Eric Chung, Axel Klawonn, Felix Kwok, Jinchao Xu, and Jun Zou, editors, *Domain Decomposition Methods in Science and Engineering XXVI*, pages 773–780, Cham, 2022. Springer International Publishing.

[27] Clark R. Dohrmann. A preconditionner for substructuring based on constrained energy minimization. *SIAM Journal for Scientific Computing*, 25:246, 2003.

[28] Mickaël Duval, Jean-Charles Passieux, Michel Salaün, and Stéphane Guinard. Non-intrusive coupling: recent advances and scalable nonlinear domain decomposition. *Archives of Computational Methods in Engineering*, 23(1):17–38, 2014.

[29] Didier El Baz. M-functions and parallel asynchronous algorithms. *SIAM Journal on Numerical Analysis*, 27(1):136–140, 1990.

[30] Mireille El Haddad, José C. Garay, Frédéric Magoulès, and Daniel B. Szyld. Synchronous and asynchronous optimized Schwarz methods for one-way subdivision of bounded domains. *Numerical Linear Algebra with Applications*, 27(2), 2020.

[31] Ahmed El Kerim, Pierre Gosselet, and Frédéric Magoulès. Asynchronous global–local non-invasive coupling for linear elliptic problems. *Computer Methods in Applied Mechanics and Engineering*, 406:115910, 2023.

[32] Mouhamed Nabih El Tarazi. Some convergence results for asynchronous algorithms. *Numerische Mathematik*, 39:325–340, 1984.

[33] Ludwig Elsner, Israel Koltracht, and Michael Neumann. Convergence of sequential and asynchronous nonlinear paracontractions. *Numerische Mathematik*, 62:305–319, 1992.

[34] Lei Fang and Panos J Antsaklis. Asynchronous consensus protocols using nonlinear paracontractions theory. *IEEE Transactions on Automatic Control*, 53(10):2351–2355, 2008.

[35] Charbel Farhat and Francois-Xavier Roux. The dual schur complement method with well-posed local neumann problems. *Contemporary Mathematics*, 157:193, 1994.

[36] Tony F.Chan and Tarek P.Mathew. Domain decomposition algorithms. *Acta numerica*, pages 61 – 143, 1994.

[37] Andreas Frommer, Hartmut Schwandt, and Daniel B. Szyld. Asynchronous weighted additive Schwarz methods. *Electronic Transactions on Numerical Analysis*, 5(48-61):1–5, 1997.

[38] Andreas Frommer and Daniel B. Szyld. On asynchronous iterations. *Journal of Computational and Applied Mathematics,*, 123:201–216, 2000.

[39] Martin J. Gander. Schwarz methods over the course of time. *Electronic Transactions on Numerical Analysis.*, 31:228–255, 2008.

[40] Martin J. Gander and Xuemin Tu. On the origins of iterative substructuring methods. In *Domain Decomposition Methods in Science and Engineering XXI*, volume 98 of *Lecture Notes in Computational Science and Engineering*, pages 597–605. Springer International Publishing, 2014.

[41] José C. Garay, Frédéric Magoulès, and Daniel B. Szyld. Synchronous and asynchronous optimized Schwarz methods for Poisson's equation in rectangular domains. *ETNA - Electronic Transactions on Numerical Analysis*, 2022.

[42] Guillaume Gbikpi-Benissan and Frédéric Magoulès. Protocol-free asynchronous iterations termination. *Advances in Engineering Software*, 146:102827, 2020.

[43] Guillaume Gbikpi-Benissan and Frédéric Magoulès. Asynchronous substructuring method with alternating local and global iterations. *Journal of Computational and Applied Mathematics*, 393:116–133, 2021.

[44] Guillaume Gbikpi-Benissan and Frédéric Magoulès. Distributed asynchronous convergence detection without detection protocol. *arXiv preprint arXiv:2206.15418*, 2022.

[45] Guillaume Gbikpi-Benissan and Frédéric Magoulès. Resilient asynchronous primal schur method. *Applications of Mathematics*, 2022.

[46] Guillaume Gbikpi-Benissan, Marina Rynkovskaya, and Frédéric Magoulès. Scalable asynchronous domain decomposition solvers for non-homogeneous elastic structures. *Advances in Engineering Software*, 174:103299, 2022.

[47] Lionel Gendre, Olivier Allix, Pierre Gosselet, and François Comte. Non-intrusive and exact global/local techniques for structural problems with local plasticity. *Computational Mechanics*, 44(2):233–245, 2009.

[48] Christian Glusa, Erik G. Boman, Edmond Chow, Sivasankaran Rajamanickam, and Daniel B. Szyld. Sacalable asynchronous domain decomposition solvers. *SIAM Journal on Scientific Computing*, 42(6):384–409, 2020.

[49] Israel Gohberg, Peter Lancaster, and Leiba Rodman. *Matrix Polynomials*. Society for Industrial and Applied Mathematics, 2009.

[50] Pierre Gosselet, Maxime Blanchard, Olivier Allix, and Guillaume Guguin. Non-invasive global-local coupling as a Schwarz domain decomposition method: acceleration and generalization. *Advanced Modeling and Simulation in Engineering Sciences*, 5(4), 2018.

[51] Guillaume Guguin, Olivier Allix, Pierre Gosselet, and Stéphane Guinard. Non intrusive coupling between 3d and 2d laminated composite models based on finite element 3d recovery. *International Journal for Numerical Methods in Engineering*, 98(5):324–343, 2014.

[52] Guillaume Guguin, Olivier Allix, Pierre Gosselet, and Stéphane Guinard. On the computation of plate assemblies using realistic 3d joint model: a non-intrusive approach. *Advanced Modeling and Simulation in Engineering Sciences*, 3(16), 2016.

[53] Stéphane Guinard, Robin Bouclier, Mateus Toniolli, and Jean-Charles Passieux. Multiscale analysis of complex aeronautical structures using robust non-intrusive coupling. *Advanced Modeling and Simulation in Engineering Sciences*, 5(1):1, 2018.

[54] Frédéric Hecht, Alexei Lozinski, and Olivier Pironneau. Numerical zoom and the Schwarz algorithm. In *Proceedings of the 18th conference on domain decomposition methods*, 2009.

[55] Jorge Hinojosa, Olivier Allix, Pierra-Alain Guidault, and Philippe Cresta. Domain decomposition methods with nonlinear localization for the buckling

and post-buckling analyses of large structures. *Advances in Engineering Software*, 70:13–24, 2014.

[56] Kenichi Hirose. Continuity of the roots of a polynomial. *The American Mathematical Monthly*, 127(4):359–363, 2020.

[57] C. C. Jara-Almonte and Charles E. Knight. The specified boundary stiffness/-force SBSF method for finite element subregion analysis. *International Journal for Numerical Methods in Engineering*, 26(7):1567–1578, 1988.

[58] FS Kelley. Mesh requirements for the analysis of a stress concentration by the specified boundary displacement method. In *Proceedings of the Second International Computers in Engineering Conference, ASME*, pages 39–42, 1982.

[59] David E Keyes. Aerodynamic applications of Newton-Krylov-Schwarz solvers. In *Fourteenth International Conference on Numerical Methods in Fluid Dynamics*, pages 1–20. Springer, 1995.

[60] Axel Klawonn, Martin Kühn, and Oliver Rheinbach. Adaptive coarse spaces for FETI-DP in three dimensions. *SIAM Journal on Scientific Computing*, 38(5):A2880–A2911, 2016.

[61] Axel Klawonn, Martin Lanser, and Oliver Rheinbach. Nonlinear FETI-DP and BDDC methods. *SIAM Journal on Scientific Computing*, 36(2):A737–A765, 2014.

[62] Axel Klawonn, Patrick Radtke, and Oliver Rheinbach. Adaptive coarse spaces for BDDC with a transformation of basis. In *Twenty Second International Conference on Domain Decomposition Methods*, 2014.

[63] Pierre Ladevèze. *Nonlinear computational structural mechanics: new approaches and non-incremental methods of calculation*. Springer, Berlin, 1999. translation by J. Simmonds.

[64] Pierre Ladevèze, Olivier Loiseau, and David Dureisseix. A micro-macro and parallel computational strategy for highly heterogeneous structures. *International Journal for Numerical Methods in Engineering*, 52(1-2):121–138, 2001.

[65] Erkki K. Laitinen, Alexander Lapin, and Jali Pieskä. Asynchronous domain decomposition methods for continuous casting problem. *Journal of Computational and Applied Mathematics*, 154:393–413, 2003.

[66] Evgeniia Lapina, Paul Oumaziz, Robin Bouclier, and Jean-Charles Passieux. A fully non-invasive hybrid IGA/FEM scheme for the analysis of localized nonlinear phenomena. *Computational Mechanics*, 2022.

[67] Patrick Le Tallec. Domain decomposition methods in computational mechanics. *Computational Mechanics Advances*, 1(2):121–220, 1994.

[68] Patrick Le Tallec, Y.H. De Roeck, and Marina Vidrascu. Domain decomposition methods for large linearly elliptic three-dimensional problems. *Journal of Computational and Applied Mathematics*, 34(1):93, 1991.

[69] Pierre-Louis Lions. On the schwarz alternating method. i. In *First international symposium on domain decomposition methods for partial diferential equations*, pages 1–42, Paris, France, 1988.

[70] Pierre-Louis Lions. On the schwarz alternating method. iii: a variant for nonoverlapping subdomains. In *Third international symposium on domain decomposition methods for partial diferential equations, volume 6*, Domain Decomposition Methods in Science and Engineering XXVI, page 202–223, SIAM Philadelphia, PA., 1988.

[71] Frédéric Magoulès and Cédric Venet. Asynchronous iterative substructuring methods. *Mathematics and Computers in Simulation*, 145:34–49, 2018.

[72] Frédéric Magoulès and Guillaume Gbikpi-Benissan. Distributed convergence detection based on global residual error under asynchronous iterations. *IEEE Transactions on Parallel and Distributed Systems*, 29(4):819–829, 2017.

[73] Frédéric Magoulès and Guillaume Gbikpi-Benissan. Jack: An asynchronous communication kernel library for iterative algorithms. *The Journal of Supercomputing*, 73(8):3468–3487, 2017.

[74] Frédéric Magoulès and Guillaume Gbikpi-Benissan. Distributed convergence detection based on global residual error under asynchronous iterations,. *IEEE transactions on parallel and distributed systems*, 29, 2018.

[75] Frédéric Magoulès and Guillaume Gbikpi-Benissan. Jack2: An mpi-based communication library with non-blocking synchronization for asynchronous iterations. *Advances in Engineering Software*, 119:116–133, 2018.

[76] Frédéric Magoulès, Daniel B. Szyld, and Cédric Venet. Asynchronous optimized Schwarz methods with and without overlap. *Numerische Mathematik*, 137:199–227, 2017.

[77] Jan Mandel. Balancing domain decomposition. *Communications in Numerical Methods in Engineering*, 9(3):233, 1993.

[78] Jan Mandel and Marian Brezina. *Balancing Domain Decomposition: Theory and Performance in Two and Three Dimensions*. University of Colorado at Denver, Denver, CO, USA, 1993.

[79] Jean-Claude Miellou. Algorithmes de relaxation chaotiques à retard. *ESAIM Mathematical modelling and numerical analysis*, 9:55 – 82, 1975.

[80] Jean-Claude Miellou. Asynchronous iterations and order intervals. In *Proceedings of the international workshop on Parallel algorithms & architectures*, pages 85–96, 1986.

[81] Jean-Claude Miellou, Didier El Baz, and Pierre Spitéri. A new class of asynchronous iterative algorithms with order intervals. *Mathematics of Computation*, 67(221):237–255, 1998.

[82] Jean-Clause Miellou, Pierre Spitéri, and Didier El Baz. A new stopping criterion for linear perturbed asynchronous iterations. *Journal of computational and applied mathematics*, 219(2):471–483, 2008.

[83] Forum MPI. Mpi-2: Extensions to the message-passing interface, 2003.

[84] Forum MPI. Mpi: A message-passing interface standard version 3.0, 2012.

[85] Camille Negrello, Pierre Gosselet, Christian Rey, and Julien Pebrel. Substructured formulations of nonlinear structure problems — influence of the interface condition. *International Journal for Numerical Methods in Engineering*, 107(13):1083–1105, 2016.

[86] Anthony Nouy and Florent Pled. A multiscale method for semi-linear elliptic equations with localized uncertainties and non-linearities. *ESAIM: Mathematical Modelling and Numerical Analysis*, 52(5):1763 – 1802, 2018. 39 pages.

[87] Paul Oumaziz, Pierre Gosselet, Karin Saavedra, and Nicolas Tardieu. Analysis, improvement and limits of the multiscale latin method. *Computer Methods in Applied Mechanics and Engineering*, 384:113955, 2021.

[88] Adam Parusinski and Armin Rainer. Optimal regularity of roots of polynomials. working paper or preprint, Mar 2016.

[89] Matthias Pott. On the convergence of asynchronous iteration methods for nonlinear paracontractions and consistent linear systems. *Linear Algebra and its Applications*, 283(1-3):1–33, 1998.

[90] Janusz S. Przemieniecki. Matrix structural analysis of subtructures. 1:138–147, 1963.

[91] Jonathan B Ransom, Susan L McCleary, Mohammad A Aminpour, and Norman F Knight Jr. Computational methods for global/local analysis. *NASA STI/Recon Technical Report N*, 92:33104, 1992.

[92] Yves Renard and Konstantinos Poulios. Getfem: Automated fe modeling of multiphysics problems based on a generic weak form language,. *Advances in Engineering Software*, 47:1–31, 2021.

[93] Jack L Rosenfeld. A case study in programming for parallel-processors. *Communications of the ACM*, 12(12):645–655, 1969.

[94] Ernest K. Ryu and Stephen Boyd. Primer on monotone operator methods. *Applied and Computational Mathematics*, 15(1):3–43, 2016.

[95] Serap Ayşe Savarí and Dimitri P. Bertsekas. Finite termination of asynchronous iterative algorithms. *Parallel Computing*, 22(1):39–56, 1996.

[96] Hermann A. Schwarz. ¨uber einen grenz¨ubergang durch alternierendes verfahren. *Vierteljahrsschrift der Naturforschenden Gesellschaft in Zurich*, 15:272–286, 1870.

[97] László Simon. Application of monotone type operators to nonlinear pde's. Author's edition, Budapest, 2013.

[98] Barry Smith, Petter Bjorstand, and William Gropp. *Domain decomposition parallel multilevel methods for elliptic partial differential equations*. Cambridge university press, 1996.

[99] Nicole Spillane and Daniel J. Rixen. Automatic spectral coarse spaces for robust FETI and BDD algorithms. *International Journal for Numerical Methods in Engineering*, 95(11):953–990, 2013.

[100] Pierre Spitéri. Synthetic presentation of iterative asynchronous parallel algorithms. In *6th conference on parallel, distributed, GPU and Cloud computing for engineering (PARENG)*, Pécs (Hungary), 2019.

[101] Pierre Spitéri. Parallel asynchronous algorithms: A survey. *Advances in engineering software*, 149, 2020.

[102] Pierre Spitéri, Jean-Claude Miellou, and Didier El Baz. Asynchronous Schwarz alternating methods with flexible communication for the obstacle problem. *Calculateurs parallèles, réseaux et systèmes répartis*, 13:47–66, 2001.

[103] Pierre Spitéri, Jean Claude Miellou, and Didier El Baz. Parallel asynchronous schwarz and multisplitting methods for a nonlinear diffusion problem. *Numerical Algorithms*, 33:461–474, 2003.

[104] Yangfeng Su and Amit Bhaya. Convergence of pseudocontractions and applications to two-stage and asynchronous multisplitting for singular m-matrices. *SIAM Journal on Matrix Analysis and Applications*, 22(3):948–964, 2001.

[105] Xue-Cheng Tai and Paul Tseng. Convergence rate analysis of an asynchronous space decomposition method for convex minimization. *Mathematics of Computation*, 71(239):1105–1135, 2002.

[106] Maxence Wangermez, Olivier Allix, Pierre-Alain Guidault, Oana Ciobanu, and Christian Rey. Non-intrusive global-local analysis of heterogeneous structures based on a second-order interface coupling. *Computational Mechanics*, 69:1241–1257, 2022.

[107] Pieter Wesseling. *An introduction to multigrid methods*. Edwards, 2004.

[108] Jhon D. Whitcomb. Iterative global/local finite element analysis. *Computers and structures*, 40(4):1027–1031, 1991.

[109] Jhon D. Whitcomb and Kyeongsik Woo. Application of iterative global/local finite-element analysis. part 1: linear analysis. *Communications in Numerical Methods in Engineering*, 9:745–745, 1993.

[110] Olof B. Widlund and Andrea Toselli. *Domain decomposition methods - algorithms and theory*, volume 34 of *Series in computational mechanics*. Springer, 2005.

[111] Jordi Wolfson-Pou and Edmond Chow. Asynchronous multigrid methods. *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 149, 2020.

[112] Ichitaro Yamazaki, Edmond Chow, Aurelien Bouteiller, and Jack Dongarra. Performance of asynchronous optimized schwarz with one-sided communication. *Parallel Computing*, 86:66 – 81, 2019.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences mécaniques et énergétiques, matériaux et géosciences (SMEMAG)

Titre: Méthode de décomposition de domaine asynchrone en calcul des structures – cas du couplage global/local

Mots clés: Couplage global/local non intrusif, décomposition de domaine asynchrone, méthodes de décomposition de domaine linéaires et non linéaires, techniques de paracontraction, MPI-RDMA.

L'analyse et la conception de structures complexes peuvent être chronophages et nécessiter des calculs intensifs, notamment pour les problèmes à grande échelle. Les méthodes de décomposition de domaine sont devenues un outil puissant en mécanique des structures pour relever ces défis. Elles consistent à diviser une tâche de calcul plus petites et indépendantes tâches qui peuvent être exécutées en parallèle. Des travaux récents montrent de nombreux avantages lors du couplage du calcul parallèle asynchrone avec ces méthodes, permettant de surmonter les limites des méthodes synchrones classiques et une utilisation plus efficace des ressources de calcul et un meilleur parallélisme, résultant en des temps de solution plus rapides.

Dans ce travail de recherche, nous présentons la première version asynchrone du couplage global/local non intrusif, capable de traiter efficacement plusieurs patchs éventuellement adjacents. Nous proposons une nouvelle interprétation du couplage comme une méthode de décomposition de domaine primale préconditionnée à droite. Nous démontrons également la convergence de l'itération asynchrone relaxée dans les cas linéaires et non-linéaires monotones en utilisant les techniques de paracontraction. Par la suite, nous proposons une mise en œuvre basée sur les techniques MPI-RDMA. Cette implémentation est comparée avec une méthode synchrone accélérée, nous l'illustrons sur plusieurs problèmes elliptiques linéaires, tels que ceux rencontrés dans les études thermiques et d'élasticité et sur des problèmes non-linéaires, notamment d'élastoplasticité. Nous observons que le paradigme asynchrone élimine de nombreux problèmes de performance du couplage global/local.

Title: Asynchronous domain decomposition method in structure mechanics – the case of the global/local coupling

Keywords: Non-invasive Global Local coupling, Asynchronous domain decomposition, Linear and non linear domain decomposition method, Paracontraction techniques, MPI-RDMA.

The analysis and design of complex structures can be time-consuming and computationally intensive, especially for large-scale problems. Domain decomposition methods have become a powerful tool in structural mechanics to address these challenges. They divide a computational task into smaller and independent tasks that can be executed in parallel. Recent work shows many advantages when coupling asynchronous parallel computation with these methods, overcoming the limitations of classical synchronous methods and resulting in more efficient use of computational resources and better parallelism, resulting in faster solution times.

This research work presents the first asynchronous version of non-intrusive global/local coupling, capable of efficiently processing multiple possibly adjacent patches. A new interpretation of the coupling by a primal domain decomposition method is proposed. The convergence of relaxed asynchronous iteration in the linear and nonlinear cases using paracontractions techniques is also demonstrated. Subsequently, an implementation based on MPI-RDMA techniques is proposed. This implementation is then confronted with an accelerated synchronous method. The implementation is illustrated on several linear elliptic problems, such as those encountered in thermal and elasticity studies, and on nonlinear problems, such as nonlinear elliptic and plasticity problems. The asynchronous paradigm eliminates many global/local coupling performance problems.